Advanced Continual Machine Learning



University of
South Australia

Muhammad Anwar Ma'sum

Master of Computer Science

STEM, University of South Australia

A thesis submitted for the degree of Doctor of Philosophy

May 2025

List of Publication

This thesis contains material from the following 9 papers:

1. **MA Ma'sum**, M Pratama, E Lughofer, W Ding, W Jatmiko. Assessor-guided learning for continual environments. Information Sciences 640, 119088 https://doi.org/10.1016/j.ins.2023.119088 [**IF: 6.8, CORE A**]

2. **MA Ma'sum**, M Pratama, S. Ramasamy, L Liu, H Habibullah, R Kowalczyk. Vision and Language Synergy for Rehearsal Free Continual Learning. The Thirteenth International Conference on Learning Representations (ICLR 2025). https://openreview.net/forum?id=9aZ2ixiYGd [**CORE A***]

3. **MA Ma'sum**, M Pratama, E Lughofer, L Liu, H Habibullah, R Kowalczyk. Few-shot continual learning via flat-to-wide approaches. IEEE Transactions on Neural Networks and Learning Systems. 36 (5), 8966-8978, https://doi.org/10.1109/TNNLS.2024.3415048 [**IF: 8.9, CORE A***]

4. **MA Ma'sum**, M Pratama, R Savitha, L Liu, H Habibullah, R Kowalczyk. Unsupervised Few-Shot Continual Learning for Remote Sensing Image Scene Classification. IEEE Transactions on Geoscience and Remote Sensing 62, 4707214. https://doi.org/10.1109/TGRS.2024.3445887 [**IF: 8.6, #2 in RS**]

5. **MA Ma'sum**, M Pratama, S Ramasamy, L Liu, H Habibullah, R Kowalczyk. PIP: Prototypes-Injected Prompt for Federated Class Incremental Learning. Proceedings of the 33rd ACM International Conference on Information and Management (CIKM). 2024: 1670–1679. https://doi.org/10.1145/3627673.3679794 [**CORE A**]

6. **MA Ma'sum**, M Pratama, L Liu, H Habibullah, R Kowalczyk. Federated

Few-Shot Class-Incremental Learning. The Thirteenth International Conference on Learning Representations (ICLR 2025). https://openreview.net/forum?id=ZiPoAlKf9Y **[CORE A*]**

7. **MA Ma'sum**, M. Pratama, S. Ramasamy, L. Liu, H. Habibullah, and R Kowalczyk. "PROL : Rehearsal Free Continual Learning in Streaming Data via Prompt Online Learning". Accepted at the International Conference on Computer Vision 2025. https://openreview.net/forum?id=mKvOmPHnhO **[CORE A*]**
8. **MA Ma'sum**, MD Sarkar, M Pratama, S Ramasamy, S Anavatti, L Liu, R Kowlczyk. Dynamic Long-Term Time-Series Forecasting via Meta Transformer Networks. IEEE Transactions on Artificial Intelligence 5 (8), 4258 – 4268. https://doi.org/10.1109/TAI.2024.3365775 **[Q1]**
9. **MA Ma'sum**, M Pratama, I Skrjanc. Latest Advancements Towards Catastrophic Forgetting under Data Scarcity: A Comprehensive Survey on Few-Shot Class Incremental Learning. ArXiv:2502.08181. https://doi.org/10.48550/arXiv.2502.08181

In addition, contributions were also made in the following publication. However, this paper is not considered a part of this thesis:

- N Paeedeh, M Pratama, **MA Ma'sum**, W Mayer, Z Cao, R Kowalczyk. Cross-domain few-shot learning via adaptive transformer networks. Knowledge-Based Systems 288, 111458. https://doi.org/10.1016/j.knosys.2024.111458 **[IF: 7.6, CORE B]**

Acknowledgements

I have received a great deal of support and assistance throughout the writing of this thesis.

First and foremost, I'm mostly indebted to my main supervisor, Associate Professor Mahardhika Pratama, for his guidance, encouragement, and dedication, which sharpened my knowledge and skills to a higher level.

Second, I would like to thank my co-supervisors and collaborators, i.e, Professor Lin Liu, Dr. Savitha Ramasamy, Dr. Habibullah, and Professor Ryszard Kowalczyk, for their insightful feedback and advice that nurture my knowledge into a broader view.

Third, I would like to acknowledge the University of South Australia for the full PhD scholarship, and the Tokopedia-UI research center for the GPU access, with which both of the supports I can complete my PhD study.

In addition, I would like to thank my parents, my wife, my child, and my family for their prayers, continuous love, patience, and support that always accompany me.

Finally, I thank my friends for their support, cheers, and positive vibes during my study: Furqon, Naeem, Hady, Pavel, and Siyi. I also thank the MiB and MIIAS communities that assist me in balancing my psychological and spiritual health.

Acknowledgement of Funding Support

I acknowledge the support of the Enterprise Research Scholarship (ERS) by the University of South Australia.

Declaration

I declare that this thesis presents the original research, free of plagiarized materials without acknowledgement, and has not been submitted for a higher degree to any other University or Institution.



Muhammad Anwar Ma'sum 9th May 2025

Abstract

This thesis conducts a deep investigation and develops continual learning (CL) methods to address catastrophic forgetting (CF) for dynamic environments where data gathering is conducted continuously. More specifically, the proposed methods are proposed to handle CF in various settings and constraints: rehearsal and rehearsal-free, data scarcity, stand-alone and distributed, stored and streaming data. In more challenging scenarios, the proposed method addresses catastrophic forgetting with more complex constraints, i.e., data scarcity with label scarcity, and data scarcity in the distributed system.

The existing rehearsal-based CL methods utilize constant factors for their loss coefficients, which leads to sub-optimality in overcoming catastrophic forgetting. In addition, the existing methods suffer from a class-imbalance problem since the memory size is far smaller than the currently learned data size. To overcome these problems, our work starts with the assessor-guided learning (AGLA) strategy for continual learning, where an assessor guides the learning process of a base learner by controlling the direction and pace of the learning process, thus allowing an efficient learning of new environments while protecting against the catastrophic interference problem. The assessor is trained in a meta-learning manner with a meta-objective to boost the learning process of the base learner. It performs a soft-weighting mechanism of every sample accepting positive samples while rejecting negative samples. The training objective of a base learner is to minimize a meta-weighted combination of the cross entropy loss function, the dark experience replay (DER) loss function, and the knowledge distillation loss function, whose interactions are controlled in such a way as to attain an improved performance. AGLA deployed a compensated over-sampling (COS) strategy to deal with the

class imbalance problem

Our work continues with the investigation that AGLA can not handle catastrophic forgetting in a rehearsal-free setting. Meanwhile, leveraging the generalization ability from the pre-trained model, the prompt-based methods have demonstrated their success in a rehearsal-free setting. However, it still suffers from catastrophic forgetting due to inter-task vector similarity and unfitted new components of previously learned tasks. On the other hand, the language-guided approach falls short of its full potential due to minimal utilization of knowledge and participation in the prompt tuning process. To overcome these challenges, we design language as prompt generators (LEAPGen), a new prompt generation and its learning method to handle catastrophic forgetting in a rehearsal-free setting. The key principles of our approach are (1) LEAPGen utilizes fixed language embedding as input for prompt generation instead of a task-shared learnable vector that is continuously updated. The input is selected from the catalog of language descriptors based on their similarity to the input. (2) LEAPGen utilizes task-wise generators instead of growing generators to generate a prompt component; thus, the current task generators are frozen in the upcoming tasks training. (3) LEAPGen utilizes a new soft task ID prediction to limit the search space of matching descriptors. The mechanism is different from the task ID prediction in the previous methods. (4) LEAPGen utilizes the generated language-based prompt as the auxiliary data to be appended to the input embedding.

Going deeper into a more challenging constraint, we found that AGLA and LEAPGen can not handle catastrophic forgetting under data scarcity. To overcome the challenge, we propose a Flat-to-Wide Approach (FLOWER) for few-shot continual learning problems. FLOWER is configured as a prototypical network driven by the prototypical loss. FLOWER puts forward the flat-to-wide learning idea to discover flat-wide minimum regions. It minimizes the losses of noise-perturbed parameters of the feature extractor over several runs. The flat-wide parameters are maintained in the subsequent few-shot learning tasks by constraining feature extractor parameters to prevent the catastrophic forgetting problem. Our approach differs from the previous Flat-minima method, which heavily focuses on the flat minimum regions, resulting in over-dependence on the base learning task.

In addition, no memory is imposed by FLOWER, and the wide learning concept is integrated as an attempt to drive flat regions to be wide, thus scaling well for large-scale problems. The concept of ball generator is integrated in FLOWER to prevent the over-training problem due to scarce samples in few-shot learning tasks.

Despite having proven its effectiveness in supervised few-shot continual learning, FLOWER assumes abundant labeled samples on the base task, which may not be applicable in real-world applications such as remote sensing, where the data has varying camera parameters, spectral ranges, resolutions, etc, and class labels are hardly available. To address the catastrophic forgetting under the unlabeled data scarcity, we propose a novel unsupervised flat-wide learning approach (UNISA) for unsupervised few-shot continual learning for remote sensing image scene classifications, which does not depend on any labeled samples for its model updates. UNISA is developed from the idea of prototype scattering and positive sampling for learning representations, while the catastrophic forgetting problem is tackled with the flat-wide learning approach combined with a ball generator to address the data scarcity problem. The positive sampling leverages the pseudo-label generated by the unsupervised method, i.e., k-means clustering.

Moving to the distributed system, AGLA and LEAPGen, which work excellently on a stand-alone system, can not perform adequately in a distributed system where CL is conducted by collaborative agents in a federated manner. Meanwhile, the current federated class incremental learning (FCIL) state-of-the-art (SOTA) trains and shares whole backbone parameters, resulting in a large number of parameters to train, which leads to longer training time, and a high communication cost. Second, in the current SOTAs, the clients privately share perturbed images with the central server for the aggregation process. It may violate data privacy principles since perturbation doesn't guarantee information leakage. To address the challenges, we propose a new approach named prototype-injected prompt (PIP). In our approach, a client trains and shares only a small set of parameters called a prompt, while the backbone model remains frozen and unshareable. As an advantage, we could utilize any backbone with any complexity, number of parameters, and size. Our approach utilizes deep features as a shareable prototype to handle the non-i.i.d. data between clients. As the second advantage, our approach offers

far smaller prototypes and guarantees data privacy as deep features represent a different structure and value compared to raw or perturbed samples.

In the FCIL setting, PIP assumes that each class carries abundant labeled samples that may be less realistic in real-world applications, where a client may hold few samples for a class. We emphasize the data scarcity issue leading to the prototype-bias and over-fitting problems in FCIL and define a new problem, namely Federated Few-Shot Class Incremental Learning (FFSCIL). We propose a novel method for the FFSCIL problem termed Unified Optimized Prototype Prompt (UOPP), built upon a prompt learning framework coupled with static and dynamic prototypes optimized by Neural Ordinary Differential Equation (ODE). The proposed method utilizes an adaptive dual-head to enhance its predictive accuracy. To our knowledge, our proposed method is the first prompt-based method that integrates prompt tuning, prototype rectification by a trainable network, and adaptive dual classifiers in a single pipeline.

LEAPGen works excellently on rehearsal-free continual learning, but can not overcome the catastrophic forgetting in streaming data since the data is only one-time accessible. To overcome the CF on streaming data, we develop a novel learning method that achieves two compulsory aspects in online continual learning (OCL), i.e., (1) achieving stability-plasticity and (2) being efficient both in training and inference phases. To achieve those 2 mandatory components, we develop a novel prompt online learning (PROL) method that accommodates the following components: (1) a single lightweight prompt generator for all tasks, (2) a set of scalers and shifters associated with the class-wise learnable keys, (3) pre-trained model (PTM) generalization preservation via cross-correlation matrices, and (4) hard-soft learning updates for a better parameter tuning. The generator is trained only on the first task to avoid forgetting and achieve stability, while scalers and shifters are trained on every task to achieve plasticity. To achieve high throughput, our lightweight generator has fewer parameters than a single generator and fewer than a single task-specific prompt.

The previous chapters propose methods to tackle catastrophic forgetting applied in the classification domain and image datasets. The last contribution proposes

an adaptation of the continual learning concept into a regression domain i.e., dynamic long-term time-series forecasting. A reliable long-term time-series forecaster is highly demanded in practice but comes across many challenges, such as low computational and memory footprints as well as robustness against dynamic learning environments. We propose Meta-Transformer Networks (MANTRA) to deal with the dynamic long-term time-series forecasting tasks. MANTRA relies on the concept of fast and slow learners where a collection of fast learners learns different aspects of data distributions while adapting quickly to changes. A slow learner tailors suitable representations to fast learners. Fast adaptations to dynamic environments are achieved using the universal representation transformer layers, producing task-adapted representations with a small number of parameters.

Contents

List of Publication ii

Acknoledgements iv

Acknowledgement of Funding Support v

Declaration vi

Abstract vii

Contents xii

List of Figures xxii

List of Tables xxvi

Acronyms and Symbols xxxiii

1 General Introduction 1

1.1 Background 1

1.2 Problem Statement and Motivation 3

1.3 Aim and Objectives 6

1.4 Research Limitation 8

1.5 Major Contribution 8

1.6 Organization of The Thesis 12

2 Literature Review 14

2.1 Continual Learning 14

2.1.1 Type of Setting 15

2.1.2 Continual Learning Approaches . . . 16
2.1.3 Regularization-based Approach . . . 17
2.1.4 Structure-based approach . . . 17
2.1.5 Memory-based Approach . . . 18
2.1.6 Bias Correction Approach . . . 19
2.1.7 Prompt-based Approach . . . 19
2.1.8 LoRA-based Approach . . . 20
2.1.9 Adapter-based Approach . . . 21
2.2 Few-Shot Continual Learning . . . 21
2.2.1 Types of FSCIL Setings . . . 22
2.2.2 Backbone Tuning Approach . . . 24
2.2.3 Meta-Learning Approach . . . 24
2.2.4 Prototype-Tuning Approach . . . 25
2.2.5 Dynamic Architecture Approach . . . 26
2.2.6 Parameter Efficient Fine Tuning (PEFT) Approach . . . 26
2.3 Online Continual Learning . . . 27
2.3.1 Non-PTM-based Approach . . . 27
2.3.2 PTM-based Approach . . . 28
2.4 Federated Class Incremental Learning . . . 29
2.4.1 Whole Backbone Tuning Approach . . . 29
2.4.2 Parameter-Efficient Fine-Tuning (PEFT) Approach . . . 29
2.5 Meta Learning . . . 30
2.6 Few-Shot Learning . . . 31
2.7 Federated Learning . . . 32
2.8 Data Augmentation . . . 33
2.9 Research Gap . . . 34

3 AGLA: Assessor Guided Learning for Rehearsal-Based Continual Learning 37

3.1 Introduction . . . 38
3.2 Problem Formulation . . . 41
3.3 AGLA: Assessor Guided Learning . . . 42
3.3.1 Loss Function . . . 42

3.3.2 Meta-Training Strategy . . . 44
3.3.3 Compensated Over-Sampling Approach . . . 45
3.3.4 Algorithm . . . 48
3.4 Theoretical Analysis . . . 48
3.5 Experiment Results and Analysis . . . 51
3.5.1 Datasets . . . 53
3.5.2 Baseline Algorithms . . . 54
3.5.3 Experiment Setup . . . 55
3.5.4 Numerical Results . . . 56
3.5.5 Ablation Study . . . 57
3.5.6 Memory Analysis . . . 62
3.5.7 Performance Evaluations per Task . . . 62
3.5.8 Assessor vs w/o Assessor . . . 63
3.5.9 Evolution of Meta-weights . . . 63
3.5.10 Complexity Analysis . . . 64
3.6 Summary . . . 65

4 LEAPGen: Vision and Language Synergy for Rehearsal Free Continual Learning 67
4.1 Introduction . . . 68
4.2 Preliminary . . . 70
4.2.1 Problem Formulation . . . 70
4.2.2 ViT and Prompt-based CL . . . 70
4.2.3 Types of Prompt-based Approaches . . . 71
4.2.4 Language Guided Approaches . . . 72
4.2.5 Dilemma and Drawbacks . . . 73
4.2.6 Preliminary Analysis . . . 73
4.2.7 High Similarity of Task Key Vectors . . . 74
4.2.8 High Similarity of Language Embedding . . . 75
4.3 LEAPGen: Language as Prompt Generator . . . 76
4.3.1 Overview . . . 76
4.3.2 Architecture and Prompt Generation . . . 77
4.3.2.1 Input and Learnable Parameters . . . 77

4.3.2.2 Prompt Generation . . . 78
4.3.2.3 Generator Networks and Detailed Generation . . . 78
4.3.2.4 Soft task ID Prediction . . . 79
4.3.2.5 Learning with Auxiliary Data . . . 79
4.3.3 Formulation of Learning Mechanism . . . 80
4.3.3.1 Theoretical Foundation . . . 80
4.3.3.2 Learning Objective . . . 81
4.3.3.3 Algorithm . . . 82
4.4 Theoretical Analysis . . . 82
4.5 Experiment Results and Analysis . . . 89
4.5.1 Datasets . . . 89
4.5.2 Baseline Methods . . . 89
4.5.3 Experiment Setup: . . . 90
4.5.4 Overall Performance . . . 91
4.5.5 Comparison to Non-Prompting Approaches . . . 93
4.5.6 Historical Performance . . . 94
4.5.6.1 On 10-Task Setting . . . 94
4.5.6.2 On 5-Task and 20-Task Settings . . . 95
4.5.7 LEAPGen-lite's Performance . . . 95
4.5.8 Ablation Study . . . 97
4.5.9 Sensitivity Analysis . . . 98
4.5.10 Performance on Various Generators . . . 99
4.5.11 Performance on Various Descriptors . . . 100
4.5.12 Trade-offs Between Cost and Performance . . . 101
4.5.12.1 Performance vs Parameter and Running Time . . . 103
4.5.12.2 Total Running Time . . . 104
4.5.12.3 Inference time only . . . 104
4.5.12.4 Performance vs Storage . . . 105
4.5.13 Complexity Analysis . . . 106
4.6 Summary . . . 108

5 FLOWER: Few-Shot Continual Learning via Flat-to-Wide Approaches 109

5.1 Introduction . . . 110
5.2 Problem Formulation . . . 112
5.3 The Flat-to-Wide Approach (FLOWER) . . . 114
5.3.1 Base Learning Phase ($k = 1$) . . . 114
5.3.2 Few-Shot Learning Phase ($k > 1$) . . . 117
5.3.2.1 Feature Space Data Augmentation . . . 117
5.3.2.2 Projection-based Memory Aware Synapses . . . 118
5.3.2.3 Loss Function of Few-Shot Learning Phase . . . 119
5.3.3 Algorithm . . . 120
5.4 Theoretical Analysis . . . 123
5.5 Experiments Result and Analysis . . . 126
5.5.1 Datasets . . . 126
5.5.2 Baselines . . . 126
5.5.3 Experiment Setup . . . 127
5.5.4 Numerical Results . . . 133
5.5.5 Stability-Plasticity Analysis . . . 136
5.5.6 Ablation Study . . . 138
5.5.7 Sensitivity Analysis . . . 138
5.5.8 Analysis of Data Augmentation . . . 138
5.5.9 Complexity and Running Time Analysis . . . 139
5.6 Summary . . . 140

6 UNISA: Unsupervised Few-Shot Continual Learning for Remote Sensing Image Scene Classification 142
6.1 Introduction . . . 143
6.2 Preliminaries . . . 146
6.2.1 Continual Learning in RS . . . 146
6.2.2 Problem Definition . . . 147
6.2.3 Our Solution: Unsupervised Flat-Wide Learning Approach (UNISA) . . . 148
6.3 UNISA Base Learning Session ($k = 1$) . . . 149
6.3.1 Representation Learning . . . 150
6.3.2 Flat-Wide Base Learning Phase ($k = 1$) . . . 151

6.4 UNISA Few-Shot Learning Phase ($k > 1$) . . . 152
6.4.1 Ball Data Generator . . . 153
6.4.2 Regularization-based Approach . . . 153
6.4.3 Flat-Wide Few-Shot Learning Phases $k > 1$ . . . 154
6.4.4 Algorithm . . . 154
6.5 Experiment Results and Analysis . . . 166
6.5.1 Dataset . . . 166
6.5.2 Baseline . . . 167
6.5.3 Experiment Setup . . . 167
6.5.4 Numerical Results . . . 168
6.5.5 Ablation Study . . . 169
6.5.6 The Balance of Loss Coefficients . . . 169
6.5.7 Robustness Analysis . . . 170
6.5.8 Sensitivity Analysis . . . 170
6.5.9 Complexity Analysis and Execution Times . . . 171
6.6 Limitations . . . 173
6.7 Summary . . . 174

7 PIP: Prototypes-Injected Prompt for Federated Class Incremental Learning 175
7.1 Introduction . . . 176
7.2 Preliminaries . . . 178
7.2.1 Problem Formulation . . . 178
7.2.2 Proposed Approach and Baselines: Federated Prompt-Based Continual Learning for FCIL . . . 179
7.3 Proposed Method: Prototype-Injected Prompt (PIP) . . . 181
7.3.1 Handling Local Catastrophic Forgetting via Prompt Learning 182
7.3.2 Shared Prototype Injection: What, Why and How? . . . 183
7.3.3 Handling Class Imbalance via Prototype Augmentation . . . 184
7.3.4 Server Weighted Aggregation . . . 184
7.4 Theoretical Analysis . . . 185
7.4.1 Derivation of Weighted Aggregation . . . 185
7.4.2 Theorems . . . 188

7.5 Experiment Results and Analysis . . . 196
7.5.1 Datasets . . . 196
7.5.2 Benchmark Algorithms . . . 197
7.5.3 Experiment Setup . . . 197
7.5.4 Numerical Results . . . 197
7.5.5 Forgetting Analysis . . . 200
7.5.6 Ablation Study . . . 201
7.5.7 Performance Analysis on Diferent Task Size . . . 201
7.5.8 Performance Analysis on Smaller Local Clients . . . 202
7.5.9 Performance Analysis on Smaller Global Rounds . . . 203
7.5.10 Complexity and Running Time Analysis . . . 204
7.6 Summary . . . 206

8 FFSCIL: Federated Few-Shot Class Incremental Learning 208
8.1 Introduction . . . 209
8.2 Problem Formulation . . . 211
8.2.1 Federated Few-Shot Class-Incremental Learning (FFSCIL) . 211
8.2.2 FFSCIL Objectives . . . 212
8.3 Proposed Method: Unified Optimized Prototype Prompt(UOPP) . 212
8.3.1 Overview . . . 212
8.3.2 Minimizing Task Interference via Prompt Learning . . . 214
8.3.3 Unified Static-Dynamic Prototype and Its Usability . . . 214
8.3.3.1 Static Prototype . . . 214
8.3.3.2 Dynamic Prototype . . . 215
8.3.3.3 Unified Prototype . . . 215
8.3.4 Handling Prototype-Bias via Dynamic Prototype Rectification 216
8.3.5 Adaptive Dual-Head Classifier for A Better Prediction . . . 218
8.3.6 Federated Learning and Server-Side Aggregation . . . 218
8.3.7 Wrap Up and Final Objective . . . 219
8.3.8 Algorithm . . . 219
8.4 Theoretical Analysis . . . 219
8.5 Experimental Results and Analysis . . . 229
8.5.1 Dataset . . . 229

8.5.2 Baseline . . . 229
8.5.3 Experiment Setup . . . 230
8.5.4 General Performance . . . 231
8.5.5 Stability-Plasticity Analysis . . . 233
8.5.6 Performance on Different k-Shot . . . 233
8.5.7 Performance Analysis on Different Non-i.i.d. Level . . . 234
8.5.8 Performance Analysis on Different Participating Local Clients 234
8.5.9 Performance Analysis on Different Global Rounds . . . 236
8.5.10 Ablation Study . . . 237
8.5.11 Complexity, Running Time, Parameters, and Communication Cost . . . 238
8.5.12 Memory Consumption Analysis . . . 240
8.5.13 Running Time Analysis w.r.t Data Size and Selected Local Clients . . . 241
8.5.14 Evaluation on Standalone FSCIL . . . 242
8.5.15 Limitation and Potential Solution . . . 243
8.6 Summary . . . 245

9 PROL : Rehearsal Free Continual Learning in Streaming Data via Prompt Online Learning 246
9.1 Introduction . . . 247
9.2 Preliminary . . . 249
9.2.1 Problem Formulation . . . 249
9.2.2 The Weakness of Existing OCL Methods . . . 249
9.3 Proposed Method: Prompt Online Learning (PROL) . . . 251
9.3.1 Learnable Parameters: . . . 251
9.3.1.1 Single Lightweight Generator for All . . . 251
9.3.1.2 Learnable Scalers-and-Shifters and Class-wise Key 252
9.3.2 Prompt Generation . . . 253
9.3.3 Current-to-Previous Classes Orthogonality . . . 253
9.3.4 Preserving Generalization Property . . . 253
9.3.5 Learning, Losses, and Final Objectives . . . 254
9.3.5.1 Tuning Generators and scaler-and-shifters . . . 254

9.3.5.2 Tuning Task-wise learnable key . . . 255
9.3.5.3 Tuning Generalization preserving Matrix . . . 255
9.3.5.4 Joint Loss and Final Objective . . . 255
9.3.5.5 Hard-Soft Update (HSU) Strategy . . . 256
9.3.6 Algorithm . . . 257
9.4 Experiment Results and Analysis . . . 257
9.4.1 Dataset . . . 257
9.4.2 Baseline . . . 258
9.4.3 Experiment Setup . . . 258
9.4.4 Overall Performance . . . 259
9.4.5 Historical Performance . . . 260
9.4.6 Parameter, Throughput and Execution time . . . 260
9.4.7 Robustness Analysis . . . 261
9.4.8 Ablation Study . . . 262
9.4.9 Complexity Analysis . . . 263
9.5 Summary . . . 264

10 MANTRA : Dynamic Long-Term Time-Series Forecasting via Meta Transformer Networks 266
10.1 Introduction . . . 267
10.2 Preliminary . . . 270
10.2.1 Problem Formulation . . . 270
10.2.2 Autoformer . . . 270
10.3 Meta-Transformer Networks (MANTRA) . . . 273
10.3.1 Fast and Slow Learning Approach . . . 273
10.3.2 Drift Handling Mechanism via the URT Layer . . . 276
10.3.3 Complexity Analysis . . . 278
10.4 Experiments . . . 280
10.4.1 Datasets . . . 280
10.4.2 Implementation Details . . . 280
10.4.3 Baselines . . . 281
10.4.4 Numerical Results . . . 281
10.4.5 Ablation Studies . . . 286

10.4.6 Sensitivity Analysis 286
10.5 Summary 286

11 Conclusion and Future Work **289**
11.1 Conclusion 289
11.2 Future Work 295

Appendix A: Code Repositories **317**

Appendix B: Statements of Authorship **318**

List of Figures

1.1 Illustration of Catastrophic Forgetting Problem in Continual Learning. . 4
1.2 Illustration of Continual Learning under Data Scarcity. 5
1.3 Illustration of Continual Learning under Data Privacy Constraint. . . . 7

2.1 Difference between conventional learning and continual learning. 15
2.2 Types of CL settings, i.e., task-incremental learning, class incremental learning and domain incremental learning. 16
2.3 Types of CL approaches, i.e., regularization-based approach, structure-based approach, memory-based approach, bias correction approach, prompt-based approach, LoRA-based approach, and adapter approach. . 18
2.4 Types of FSCIL settings and their properties (a) Fully-Supervised, (b) Semi-Supervised (c) Unsupervised (d) Complete (e) Cross-Domain FSCIL. 22
2.5 Visualization of FSCIL Approaches, i.e., (a) backbone tuning approach, (b) meta learning approach, (c) prototype tuning approach, (d) dynamic structure approach, and (e) parameter efficient fine tuning (PEFT) approach. 23
2.6 OCL approaches and methods. 28
2.7 Meta-learning illustration. 30
2.8 Metric Loss Illustration. 31
2.9 Federated Learning Illustration 33
2.10 Image Augmentation Taxonomy. 34

3.1 The flow-chart of the proposed method (AGLA). 49
3.2 Classification Accuracy of Consolidated Algorithms across All Datasets. 58

3.3 The trace of losses for AGLA with or w/o Assessor in the S-CIFAR-100 problem. 59
3.4 The trace of weighting coefficients and loss for AGLA in the S-CIFAR-100 problem. 60

4.1 Visualization of Pool-Based Prompting (red area), Task-Specific Prompting (blue area), Growing Component Prompting (green area), and Language Guided Approach for Continual Learning (gray area). 71
4.2 The LEAPGen architecture. Each task contains a set of generators, a task-wise learnable, and class-wise learnable parameters associated with the descriptor embedding. Given an input image, LEAPGen predicts the task ID and finds top-k descriptors matched to the input via cosine similarity afterward. The descriptors are dispatched to the generators producing the prompt components. LEAPGen produces the final prompt and an auxiliary embedding by the weighted sum of the prompt components and the descriptors, respectively. The prompt is prepended into the ViT MSA layer, while the auxiliary is appended into the input patches. 77
4.3 Historical performance plot of the consolidated methods in 3 CIL benchmark datasets. 94
4.4 Visualization of the historical performance of consolidated algorithms in CIFAR100 and ImageNet-R with 5 tasks setting. 96
4.5 Visualization of the historical performance of consolidated algorithms in CIFAR100 and ImageNet-R with 20 tasks settings. 96
4.6 FAA of the consolidated methods under different prompt lengths (left), layer depth (middle), and maximum generators per task (right). 98
4.7 FAA, CAA, FFM, and CFM of the proposed method w.r.t types of generator. 99
4.8 Visualization of the historical performance of consolidated algorithms in ImageNet-R with 10 tasks setting, in various descriptors and LLMs. . . 101

5.1 Flat Continual Learning: during the base learning phase $k = 1$, FLOWER finds the network parameters Θ^* located in the flat local optimum region. These parameters might shift to non-flat local minima regions when learning the few-shot learning tasks $k > 1$ via $\Theta^* \rightarrow \Theta_1$. Clamping is performed for Θ_1 to stay around a flat local optimum region. 115
5.2 Wide Continual Learning: conventional continual learning approaches generate narrow ellipsoids where the local minima region is sharp, leaving very small space for model candidate parameters, i.e., intersections of all ellipsoids. On the contrary, the wide continual learning generates wide ellipsoids having wide local minima regions, allowing enough space for model candidate parameters. 120
5.3 Few Shot Learning Phase of FLOWER is driven by the three loss functions, the cross entropy loss function, the ball generator loss function, and the projection-based synaptic intelligence regularizer. It combines wide and flat learning paradigms supported by the feature space data augmentation module, with a transformation module. 121
5.4 Visualization of Performances of Consolidated Algorithms across three problems under the moderate base tasks 135
5.5 Visualization of Consolidated algorithms in MiniImagenet, CIFAR-100, and CUB-2011 dataset with 20, 20, and 50 base classes, respectively. . . 135
5.6 Visualization of Consolidated algorithms in MiniImagenet, CIFAR-100, and CUB-2011 dataset with 60, 60, and 100 base classes respectively, with 1-shot setting. 136
5.7 Visualization Base Classes and Novel Class Accuracy across all tasks in MiniImagenet, CIFAR-100, and CUB-2011 dataset with 60, 60, and 100 base classes respectively. 137

6.1 Visualization of Continual Learning for Remote Sensing Scene Classification. 146
6.2 Visualization of Model Parameters in Flat Wide Learning (d) compared to Standard Learning (a), Flat Minima Learning (b), and Wide Minima Learning (d). 148

6.3 Workflow of the Proposed Method (UNISA): UNISA is trained minimizing a joint loss function of the prototype scattering loss, the positive sampling alignment loss, the wide local minimum loss, the ball generator loss, and the regularization term. The data augmentation procedure is implemented to address the data scarcity problem. 149
6.4 Visualization of UC Merced dataset. 162
6.5 Visualization of Performances of Consolidated Algorithms across 6 datasets in 1-5 shot settings. 165

7.1 Visualization of our proposed method (PIP) that includes prototypes-injected-prompt learning with $\mathcal{L}_{l+}$ to handle local catastrophic forgetting, shared prototypes to handle non-i.i.d distributions between clients, prototypes augmentation to handle class-imbalance, and server Gaussian weighted aggregation to improve global model generalizations. 181
7.2 Performance of the consolidated methods in CIFAR100, MiniImageNet and TinyImageNet with T=5 and T=20. 202
7.3 Performance of the consolidated methods in CIFAR100 and TinyImageNet with smaller local clients (L) and smaller rounds. 205

8.1 The importance of prototype rectification to handle prototype bias (a) Initial prototype per client (b) Aggregation without rectification can not handle prototype-bias (c) Aggregation with rectification overcomes prototype-bias. 209
8.2 The visualization of UOPP, includes task-wise prompt learning empowered by shared unified static-dynamic prototypes, dynamic prototypes rectification, adaptive dual heads, and weighted aggregation. The gray-colored parameters are frozen and unshareable parameters, the green-colored parameters are trainable and shareable parameters, while the blue-colored parameters are trainable but unshareable parameters. . 213
8.3 Visualization of Biased and Optimal Prototype (a), and Rectification (b). 216
8.4 Visualization of the performance of consolidated algorithms in MiniImageNet, CIFAR100 and CUB200. 231

8.5 Visualization of the performance of consolidated methods for base classes and novel classes in CIFAR100 Dataset. 233
8.6 Visualization of Novel Classes Validation Loss with vs without Rectification. 238

9.1 The proposed PROL diagram. PROL utilizes a single generator (G_K, G_V) for all classes in all tasks, learnable scalers (a_K^i, a_V^i), and shifters (b_K^i, b_V^i) parameters asocciated to the class-wise key K^i. The generator is trained only in the first task and frozen afterward, while the scalers, shifters, and class-wise keys are trained in every task. The prompt is generated by scaling and shifting $G(x)$ by the selected scalers and shifters associated with the highest similarity key. The model is updated by joining the loss function consisting of intra-task loss $\mathcal{L}_{intra}$, inter-task loss $\mathcal{L}_{intra}$, orthogonality loss $\mathcal{L}_{ort}$, input-to-key similarity loss $\mathcal{L}_{sim}$ and generalization loss $\mathcal{L}_{gen}$. 251
9.2 Historical performance plot of the consolidated methods in 4 benchmark datasets, i.e., CIFAR100, Imagnet-R, Imagnet-A, and CUB200. 258
9.3 Detailed throughput and execution time of consolidated methods in Imagnet-R dataset. 262

10.1 MANTRA is consolidated by an ensemble of fast learners and a slow learner, where the final output is aggregated by a URT layer. Fast learners and the URT layer are trained in an end-to-end fashion, minimizing the mean-squared error loss, while the slow learner is learned via the self-supervised learning loss, controlled reconstruction loss. 274
10.2 MANTRA and Autoformer validation loss on ETTm dataset. 284
10.3 Visualization of MANTRA prediction compared to actual label on ETTm dataset. 285

List of Tables

3.1 Details of Continual Learning Problems, nTasks: number of tasks, nClasses / Task: number of classes per task, CL Setting: Continual Learning Setting. 51
3.2 Hyperparameters of Consolidated Algorithms. 52
3.3 Classification result (accuracy %) for continual learning benchmarks, averaged across 5 runs. ∗ 4 tasks only due to crash. 53
3.4 Average forgetting for continual learning benchmarks, averaged across 5 runs. ∗ 4 tasks only due to crash. 53
3.5 Classification result (accuracy) of AGLA based on various configurations. 54
3.6 Forgetting rate of AGLA based on various configurations. 54
3.7 Classification result (accuracy) of the benchmarks based on memory size. 55

4.1 Preliminary results on Imagenet-R with 10 tasks setting. 74
4.2 Cosine similarity of task-key vectors after completing 10-task training on ImageNet-R by DualPrompt method. 75
4.3 Encoded Language Embedding similarity of the first 20 classes of ImageNet-R, encoded by CLIP encoder as implemented in LGCL. . 76
4.4 Summarized numerical result on split CIFAR100 and ImageNet-R in 5, 10, and 20 tasks settings, * and ∘ denote the results obtained from 1 seeded run and the first 10 tasks, respectively due to system crash. . . . 92
4.5 Summarized result on split CUB (10-tasks). 93
4.6 Comparison of the proposed method to the LoRA approach, SLCA, and GMM. All the results of the competitor methods are taken from the respective references. 93

4.7 Ablation study on the impact of LEAPGen components to its performance, run on 10 tasks Imagenet-R. 97
4.8 Types and examples of descriptors generated by popular LLMs. 102
4.9 Performance of LEAPGEn vs ConvPrompt in various descriptors. 103
4.10 Trade-offs between cost and performance on ImageNet-R dataset with 10 task settings. ConvPrompt-Large is run for 9 tasks only since it crashed on the final task. 106
4.11 Total running time including descriptor generation, training, and testing. 107
4.12 Performance vs Storage Trade-offs in ImageNet-R dataset, * denotes measurement at task-9 since the method crashed after task-9. 107

5.1 Hyperparameters setting for our experiments. 128
5.2 Classification accuracy on split MiniImageNet dataset with 60 and 20 base classes averaged across 5 times run, #B indicates the number of base classes, * indicates the results are copied from F2M, + indicates the results copied from TOPIC. 129
5.3 Classification accuracy on split CUB 2011 dataset with 100 and 50 base classes averaged across 5 runs, #B indicates the number of base classes, * indicates the results are copied from F2M, + indicates the results copied from TOPIC, ∘ indicates the method is run 1 time due to system crash. 130
5.4 Summarized performance of consolidated methods with 1-shot setting on MiniImageNet, CIFAR-100, and CUB-200 dataset averaged across 5 time runs,* denotes only run 1 time due to system crash. 130
5.5 Harmonic Mean Accuracy (HMACC) on split MiniImageNet dataset with 60 and 20 base classes averaged across 5 runs. 131
5.6 Summarized Harmonic Mean Accuracy (HMACC) on split CIFAR100 dataset with 60 and 20 base classes, and split CUB 2011 with 100 and 50 base classes averaged across 5 runs. 131
5.7 Ablation study on split MiniImagenet dataset with 60 base classes averaged across 5 runs. 132

5.8 Sensitivity analysis of the impact of bound value to classification accuracy on the MiniImageNet dataset with 60 base classes averaged across 5 time runs. 132
5.9 Sensitivity analysis of the impact of random times to classification accuracy on MiniImageNet dataset with 60 base classes averaged across 5 time runs, * denotes only run in base task due to poor performance/ . 132
5.10 Analysis of the impact of augmentation to classification accuracy on MiniImageNet dataset with 60 base classes averaged across 5-time runs. 133
5.11 Training time of the consolidated methods in CIFAR100 dataset with 60 base classes. 140

6.1 Classification accuracy on split UC Merced dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot. . . . 156
6.2 Classification accuracy on split AID dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot. . . . 157
6.3 Classification accuracy on split RSICB256 dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot. . . . 158
6.4 Classification accuracy on split RESISC45 dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot. . . . 159
6.5 Classification accuracy on split EuroSAT dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot. . . . 160
6.6 Classification accuracy on split HyRank dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot. . . . 161
6.7 Ablation study of the proposed method with various configurations on the 5-shot UC Merced dataset, FM and WM stand for Flat Minima and Wide Minima, respectively. 162
6.8 Extended ablation study of the proposed method with various loss coefficients, i.e., $\lambda_i, i \in [1-5]$ on 5-shot UC Merced dataset, * denotes the result of our default configuration. 162
6.9 Robustness analysis of the proposed method against various noises on the 5-shot UC Merced dataset. 163

6.10 Robustness analysis of the proposed method in various hyperparameter settings on the 5-shot UC Merced dataset. Min.LR and BS denote the minimum learning rate and batch size, respectively. 163
6.11 Sensitivity analysis of the proposed method with various bound (b) value on 5-shot UC Merced dataset. 163
6.12 Impact of the number of perturbation loops (M) on the proposed method performance on the 5-shot UC Merced dataset. 164
6.13 Running time of the consolidated algorithms. 164

7.1 The difference between our proposed approach and current SOTA in Federated Class Incremental Learning (FCIL), * indicates sharable parameters, while ^{x} indicates unsharable parameters. 177
7.2 Numerical results on CIFAR100 (T=10), "BL" denotes our proposed baseline, "Avg" denotes the average accuracy of all tasks, denotes performance drop, and "Imp" denotes improvement/gap of PIP-DualP compared to the respective method. 198
7.3 Numerical results on MiniImageNet (T=10), "BL" denotes our proposed baseline, "Avg" denotes the average accuracy of all tasks, "PD" denotes performance drop, and "Imp" denotes improvement/gap of PIP-DualP compared to the respective method. 198
7.4 Numerical results on TinyImageNet (T=10), "BL" denotes our proposed baseline, "Avg" denotes the average accuracy of all tasks, "PD" denotes performance drop, and "Imp" denotes improvement/gap of PIP-DualP compared to the respective method. 199
7.5 Detailed average forgetting of the consolidated methods in CIFAR100 (T=10). 199
7.6 Ablation study on CIFAR100 dataset in one-seeded run, i.e., 2021, using DualP prompt structure. "Avg" denotes the average accuracy of all tasks, "PD" denotes performance drop, and "Imp" denotes improvement/gap of PIP compared to the respective configuration. 200
7.7 Performance of the consolidated methods in CIFAR100 and TinyImageNet with smaller selected local clients (L), ^{o} and * denote the results obtained from the first 4 and 5 tasks due to system crash. 204

7.8 Training time of the consolidated algorithm in CIFARR100, MiniImagenet and TinyImageNet. 206

8.1 Numerical result of the consolidated algorithms in CIFAR100, MiniImageNet, and CUB200 dataset in 5-shot and 1-shot setting across 3 different seeded runs. 232
8.2 Harmonic mean accuracy of the consolidated algorithms in CIFAR100 dataset with 5-shot setting. 232
8.3 Numerical result of the FFSCIL methods in CIFAR100, 7-shot, 5-shot, 3-shot, and 1-shot setting, S indicates the number of k-shot, PD indicates the performance drop, and Gap indicates the gap between the respective method and UOPP (UOPP). 235
8.4 Summary of the numerical result of the consolidated algorithms in CIFAR100 dataset with the variation of Non-i.i.d. level (a), selected local clients (b), and variation of total global round (c). 236
8.5 Summary of the numerical result of our ablation study in CIFAR100 dataset, MLP, and PB and Rect. denote the MLP-based, prototypes-based, and rectification, respectively. 236
8.6 Comparison of Parameters, Communication Cost, and Training time in CIFAR100 dataset. 238
8.7 Analysis of Memory Consumptions by Client and Server. 241
8.8 Running Time w.r.t. Percentage of Available Classes Number of Selected Local Client in Each Round. 242
8.9 Numerical result of the FSCIL methods in CIFAR100, MiniImageNet, and CUB200 dataset with 5-shot setting, PD indicates the performance drop, and Gap indicates the gap between the respective method and UOPP (UOPP). 243

9.1 Summarized performance of consolidated methods in 4 benchmark datasets, i.e., CIFAR100, Imagnet-R, Imagnet-A, and CUB200. 257
9.2 Parameter and summarized throughput and execution time of consolidated methods in ImageNet-R dataset. 261

9.3 Robustness analysis of the proposed method compared to the competitor methods in ImageNet-R dataset. 263
9.4 Ablation study of the proposed method in ImageNet-R dataset. 263

10.1 Complexities of Different Methods. 280
10.2 Multivariate results with different prediction lengths O E {96, 192, 336, 720}, averaged across 3 times run. We set the input length I as 36 for ILI and 96 for the others. 281
10.3 Univariate results with different prediction lengths O E {96, 192, 336, 720}, averaged across 3 times run. We set the input length I as 36 for ILI and 96 for the others. 282
10.4 Improvement of MANTRA to competitor methods on multivariate setting with different prediction lengths $O \in \{96, 192, 336, 720\}$, except for ILI dataset $O \in \{24, 36, 48, 60\}$. 282
10.5 Improvement of MANTRA to competitor methods on univariate setting with different prediction lengths $O \in \{96, 192, 336, 720\}$, except for ILI dataset $O \in \{24, 36, 48, 60\}$. 283
10.6 T-Test with 0.1 significance level of MANTRA vs. Autoformer on multivariate and univariate setting. 283
10.7 Ablation study on ETT and ILI dataset. 287
10.8 Sensitivity analysis on ETT dataset with the different number of fast learners. 287
10.9 Sensitivity analysis on ETT dataset with the different number of URT heads. 287

Acronyms and Symbols

Acronyms

A-GEM	Averaged Gradient Episodic Memory
AGLA	Assessor-Guided Learning Approach
ApproxFSCIL	Approximation of Few-Shot Class Incremental Learning
Assr	Assessor
Aug	Augmentation
BIC	Bias Correction
CAMA	Confidence-Aware Moving Average
CDFSCIL	Cross-Domain Few-Shot Class Incremental Learning
CFSCIL	Complete Few-Shot Class Incremental Learning
C-FSCIL	Constrained Few-Shot Class Incremental Learning
CEC	Continually Evolving Classifier
CIL	Class Incremental Learning
CL	Continual Learning
CLoRA	Customized Low Rank Adaptation
CNN	Convolutional Neural Network
COS	Compensated Over-Sampling
CODA-P	Continual Decomposed Attention-based Prompting
Comp-FSCIL	Complementary Few-Shot Class Incremental Learning
ConvPrompt	Convolutional Prompting
CPrompt	Consistent Prompting
DER	Dark Experience Replay
DIL	Domain Incremental Learning

DMC	Deep Model Consolidation
EEIL	End-to-End Incremental Learning
EWC	Elastic Weight Consolidation
EvoPrompt	Evolving Prompt
FACT	Forward Compatible Training
FCIL	Federated Class Incremental Learning (problem)
FCILP	Federated Class Incremental Learning with Prompting
FedCIL	Federated Class Incremental Learning (method)
FedCPrompt	Federated Contrastive Prompt
FedWeIT	Federated Weighted Inter-client Transfer
FIM	Fisher Information Matrix
FLOWER	Flat-to-Wide Approach
FM	Flat Minima (concept)
FSCIL	Few-Shot Class Incremental Learning
FSCL	Few-Shot Continual Learning
FSL	Few-Shot Learning
FSL	Few-Shot Lifelong Learning
F2M	Flat Minima (method)
GAN	Generative Adversarial Network
GEM	Gradient Episodic Memory
GLFC	Global-Local Forgetting Compensation
GMM	Generative Multi-modal Model
HAL	Hindsight Anchor Learning
HiDe-Prompt	Hierarchical Decomposition Prompt
ICARL	Incremental Clasifier and Representation Learning
IL	Incremental Learning
InfLoRA	Interference-free Low Rank Adaptation
KLD	Kullback–Leibler divergence
LAE	Learning Accumulation-Ensemble
LANDER	Label Text Centered Data-Free Knowledge Transfer
LEAPGen	LanguagE As Prompt Generator
LGA	Local-Global Anti-forgetting

LGCL	Language-Guided Continual Learning
LSTM	Long-Short Term Memory
LWF	Learning Without ForGetting
LoRA	Low Rank Adaptation
L2P	Learning to Prompt
MAML	Model-Agnostic Meta Learning
MAS	Memory Aware Synapse
MetaFSCIL	Meta Few-Shot Class Incremental Learning
MLP	Multi-Layer Perceptron
MOS	Model Surgery
MSA	Multi-head Self Attention
MSE	Mean-Square Error
NAS	Neural Architecture Search
NC-FSCIL	Neural Collapse Few-Shot Class Incremental Learning
OCL	Online Continual Learning
OCIL	Online Class Incremental Learning
ONPRO	Online Prototype
OrCo	Orthogonality and Contrast
PGP	Prompt Gradient Projection
POCL	Pareto Optimization Continual Learning
RBF	Radial Basis Function
ResNet	Residual Network
RLN	Representation Learning Network
R.Tr	Random Transformation
PEFTZ	Parameter-Efficient Fine Tuning
PILoRA	Prototype-guided Incremental Low Rank Adaptation
PTM	Pre-Trained Model
PLN	Prediction Learning Network
PROL	Prompt Online Learning
SCN	asymmetric collaborative network
SI	Synapse Intelligence
SGD	Stochastic Gradient Descent

SLCA	Slow Learner with Classifier Alignment
SSFSCIL	Semi-Supervised Few-Shot Class Incremental Learning
S3C	Self-Supervised Stochastic Classifier
TARGET	federatTed clAss-continual leaRninG via Exemplar-free disTillation
TIL	Task Incremental Learning
UFSCIL	Unsuprvised Few-Shot Class Incremental Learning
UNISA	Unsupervised Flat-Wide Learning Approach
UOPP	Unified Optimized Prototypes Prompt
ViT	Visual Transformer
WM	Wide Minima
XDER	Extended Dark Experience Replay

Symbols

$\mathcal{T}$	the task
$\mathcal{X}$	the input
$\mathcal{Y}$	the label
$\mathcal{C}$	the class label
N	the data size
B	the number of batches
E	the number of epochs
(x_i, y_i)	a pair of data (image) and label
$\hat{y}$	predicted label
μ	mean
σ	standard deviation
z, Z	prototype
$f(.)$	feature extractor
$g(.)$	classifier
$F(.)$	cost function
L	number of layers
H	number of heads
P	the prompt

$P(.)$	probability
$G(.)$	the prompt generator
$P(a\|b)$	probability of a given b
$\mathbb{E}$	expectation
$\mathbb{R}$	real number
$\mathbb{R}^{a\times b}$	vector with the size of $a \times b$
$\mathcal{L}$	Loss function
$\mathbb{I}(.)$	binary function, 1 if the condition is true
∇	gradient
Δ	difference
$\Theta, \theta, \Phi, \phi, \omega, \psi, \psi, W$	Networks parameters/weights
i, j, k, l, m, n, t, s	pointer/sentinel numbers
ϵ, η, ζ	small constant numbers
$others$	please follow the definition

Chapter 1

General Introduction

1.1 Background

Deep learning proves to be a powerful prediction model and knowledge discovery for many applications, e.g., gene analysis [1], [2], drug discovery [3], [4], [5], medical image analysis [6], [7], [8], [9], recommender systems [10], [11], product search [10], [12], traffic characteristics prediction [13], [14], [15], automated vehicle identification [16], [17], [18], and traffic signal timing [19], [20]. In addition, it could handle extreme tasks such as 21.000 classes of image classification[21] and twelve hours of precipitation forecasting [22]. Despite its impressive performance, it suffers from a crucial drawback in adapting to a dynamic environment, where data gathering is conducted continuously. The machine tends to forget its knowledge of the previously learned tasks while gaining new knowledge on the new task. This phenomenon is called catastrophic forgetting[23]. A previous study shows that its knowledge on a previously learned task can be degraded to 0% when the machine learns several tasks afterward [24]. This evidence calls for answers on how to accommodate deep learning models to learn continuously.

Previous studies tackle the catastrophic forgetting via several types of approaches, e.g., regularization approach [25] [26] [27] [28] [29], rehearsal approach [30] [31] [32] [33], and bias correction approaches [34] [35] [36]. Rehearsal-based methods and bias correction-based methods prove to be more effective than regularization-based methods, since the methods replay the exemplar memory from previously learned

tasks. However, the replay procedure may violate the data privacy constraint, as the data from previous tasks is unavailable or not accessible in the current task. Thus, the method can not be applied in a rehearsal-free manner. The recent advancement of the foundation model drives the study in continual learning with a pre-trained model [37], [38], [39]. Leveraging the generalization property of the pre-trained model, the prompt-based approach performs effectively in the rehearsal-free setting. In addition, it needs fewer trainable parameters since the whole pre-trained backbone is frozen. However, the pre-trained model is not a one-size-fits-all generalizer that can be applied to any dataset. For instance, the methods achieve lower performance in the dataset with fine-grained classes. Thus, it still suffers from catastrophic forgetting.

Data scarcity is a common encounter in real-world applications, as labeled data is expensive. A previous study shows that data scarcity in the newly coming classes complicates the deep learning model to handle the catastrophic forgetting [40]. Achieving model plasticity for the novel classes with few samples only, leads to biased representation, since a class representation is optimal only to the observed few samples not the whole class population [41]. In addition, the overfitting to the novel classes, e.g., due to too many training iteration leads to higher forgetting of the previously learned classes. Thus, addressing catastrophic forgetting under data scarcity in few-shot continual learning (FSCL) is a more challenging yet applicable problem. In addition, most of FSCL studies assumes the first (base) task carries abundant labeled samples [42], [43], [44], [45]. Such an assumption may be inapplicable in a real-world application due to the expensive cost of annotation. Thus, the FSCL problem may escalate into a more challenging problem where the first task holds only small number of labels.

The dynamic environment also occurs in a distributed system where data gathering is continuously conducted by a set of collaborative agents rather than a single agent. The agents continuously learn the global knowledge of the environment, but without sharing their local data due to data privacy constraint. Thus the agents build a global model by training their local model, then sharing and aggregating their local model into the global model in a federated manner [46]. Each client holds a small portion of the accumulated dataset, and two different clients

may carry different samples for the same class label. In a such condition, the catastrophic forgetting problem become harder to overcome. First, the trained local model is optimal only to the respective local data, not the accumulated data. Second, simple model aggregation, such as averaging, can not handle the non-i.i.d. distributions fitted by the local models. In the previous study on federated continual learning (FCL), each client sends synthetic dataset along with its locally trained model [47], [48]. Thus, it significantly increases the local training time and communication costs. In addition, the synthetic dataset may be similar to the original local dataset, thus it could reveal its privacy to other parties. In a simplified form, FCL calls for answers for catastrophic forgetting in a distributed system while preserving data privacy, and efficient communication cost is also a consideration, along with global model performance.

Last but not least, streaming data is also an aggravating factor in handling catastrophic forgetting. Since coming in a streaming manner, the data can be seen only once; as a consequence, the model suffers from insufficient learning. Furthermore, the distribution shift makes it harder to build the general representation of the observed data [49]. In addition, in online continual learning (OCL), the model may be requested to do inference, without waiting for a task to complete [50]. Thus, the learning process must be conducted based on the currently streamed chunk, without waiting until all the streaming chunks for a task are processed. In dealing with the catastrophic forgetting in streaming data, existing OCL methods utilize exemplar memory from previous tasks to be replayed along with currently streamed chunks [51], [52], [53]. As in standard CL, the replay approach may breach the data privacy constraint, since re-accessing the previous streamed chunks may not be permissible. As an additional drawback, the memory replay also reduces the model's throughput in processing the streaming data.

1.2 Problem Statement and Motivation

Continual learning (CL) problem is defined as the problem of learning a sequence of fully supervised tasks $\{\mathcal{T}^t\}_{t=1}^{T}$, where after finishing learning a task $\mathcal{T}^t$, a model must recognize all learned tasks i.e., $\mathcal{T}^1$,$\mathcal{T}^2$,...$\mathcal{T}^{t-1}$,$\mathcal{T}^t$. Symbol T represents the

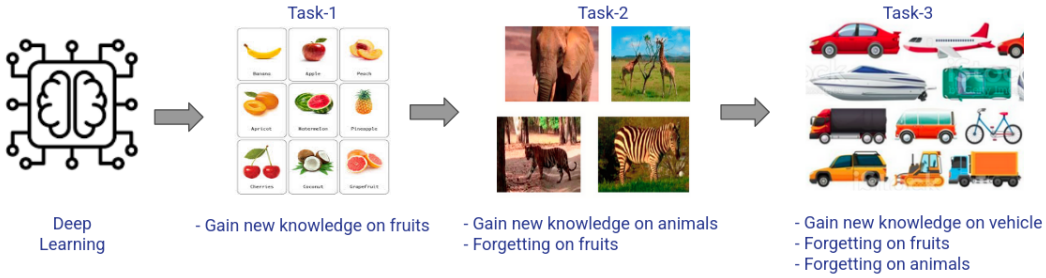


Figure 1.1: Illustration of Catastrophic Forgetting Problem in Continual Learning.

number of consecutive tasks. Each task carries pairs of training samples, i.e., $\mathcal{T}^t = \{(x_i^t, y_i^t)\}_{i=1}^{|\mathcal{T}^t|}$ where $x_i \in \mathcal{X}^t$ and $y_i \in \mathcal{Y}^t$ denotes input image and corresponding label, while $|.|$ denotes cardinality. $\mathcal{C}^t$ denotes the unique class labels in $\mathcal{Y}^t$, i.e., $\mathcal{C}^t = unique(\mathcal{Y}^t)$, and $|\mathcal{C}^t|$ denotes the number of classes in $\mathcal{T}^t$ [54], [55]. Each task t is disjoint from another task t' i.e $\forall t, t' \neq t, (\mathcal{T}^t \cap \mathcal{T}^{t'} = \emptyset)$. A memory set $\mathcal{M}^t = S(\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^{t-1})$, a small-sized sampling $(S(.))$ from previously learned tasks may be available in a rehearsal-based setting to maintain the knowledge of previously learned tasks [31].

Suppose that a deep learning model $g_\phi(f_\theta(.))$ where $f_\theta(.)$ and $g_\phi(.)$ are feature extractor and classifier respectively. On each task t, the model is trained by the currently learned task, i.e., $\mathcal{T}^t$, but it need to recognize all learned tasks, i.e., $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^{t-1}, \mathcal{T}^t$. Suppose that $\Theta^t = \{\theta^t, \phi^t\}$ is the state of the deep learning model after finishing task t, and $\Theta^1 = \{\theta^1, \phi^1\}$ is the optimal state for task-1. After finishing the first task, the model is trained with task-2, i.e., $\mathcal{T}^2$ and it state become $\Theta^2 = \{\theta^2, \phi^2\}$. The change of model state from Θ^1 into Θ^2 add its knowledge on $\mathcal{T}^2$ but it suffers from forgetting of it knowledge on $\mathcal{T}^1$ in exchange. Later on, after finishing task-3, the state become Θ^3 that carries a new knowledge on $\mathcal{T}^2$ but suffers from forgetting on $\mathcal{T}^1$ and on $\mathcal{T}^2$. Generally, after finishing task-t, Θ^t gains new knowledge on $\mathcal{T}^t$ but suffers from forgetting its knowledge on $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^{t-1}$. This phenomenon is called as catastrophic forgetting, as illustrated in figure 1.1. The memory set $\mathcal{M}^t$ may reduce the forgetting on $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^{t-1}$. but since it has a far smaller size than the current task, i.e., $|\mathcal{T}^t| >> |\mathcal{M}^t|$, the rehearsal-based method suffers from a class-imbalance problem and can not maintain all knowledge of previously learned tasks.

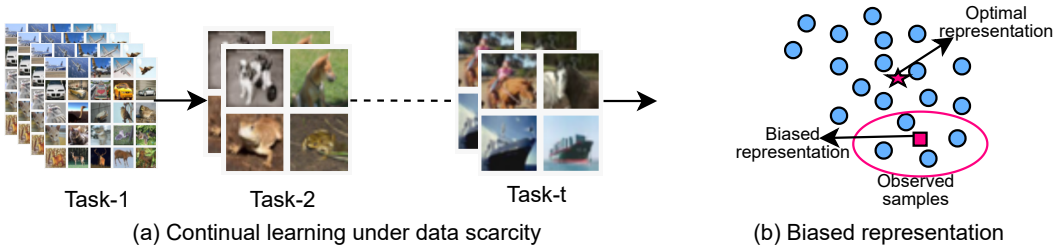


Figure 1.2: Illustration of Continual Learning under Data Scarcity.

Under a data scarcity constraint, task-t carries far smaller samples than the first task, i.e., $|\mathcal{T}^{t>1}| << |\mathcal{T}^1|$ as a class in $\mathcal{T}^{t>1}$ has only few samples, tipically 1 to 5 samples [40]. Figure 1.2 shows the illustration of biased representation from a few samples, where a representation of a class does not represent the whole sample; instead, the few observed samples only. In Addition, overfitting on $\mathcal{T}^t$ cause a higher forgetting on $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^{t-1}$ but insignificantly improve the plasticity on $\mathcal{T}^t$ [42]. Thus the previous methods limit the training epochs for $\mathcal{T}^{t>1}$ to a small number of value, e.g., 6 epochs only [42], [56]. The more challenging aspect of the FSCL problem compared to CL is that the memory based methods [31], [35], [57] can not works properly due to the biased representation and high forgetting on the previously learned tasks. In a more extreme condition where class labels is not fully available, FSCL methods suffers from a higher performance degradation [58].

As illustrated in figure 1.3, in a distributed system where continual learning is conducted by a set of collaborative clients coordinated by central server, a client l holds only a subset of current task training set, i.e., $\mathcal{T}_l^t \subset \mathcal{T}^t$. Each client-$l$ carries non-identically and distributed data (non-i.i.d) to another client l', i.e., $\forall_{l,l'} \mathcal{D}_l^t \neq \mathcal{D}_{l'}^t$ where $\mathcal{D}_l^t$ and $\mathcal{D}_{l'}^t$ are distribution of $\mathcal{T}_l^t$ and $\mathcal{T}_{l'}^t$ respectively. Non-i.i.d distribution is represented by the percentage of available classes η [47], [59]. Each task-t is learned in a federated way that is repeated in R_T rounds where in each round $r \in [1..R_T]$, a set of local clients is randomly selected from all available clients, i.e., $\{l\}_{l=1}^{L} \subset \{l\}_{l=1}^{L_{all}}$. Due to data privacy constraints, a client-l is not allowed to share any training sample $(x_i, y_i) \in \mathcal{T}_l^t$ to another client or server, but allowed to share its parameters, i.e., $\Theta_l^t = \{\theta_l^t, \phi_l^t\}$. The central server aggregates and redistributes the received local models into a global model,, i.e., $(\Theta_G, \Phi_G) =$

$Agg(\{(\Theta_G, \Phi_G)\}_{l=1}^{L})$. The main challenge of the FCL problem is that the global model can not cover all the distributed knowledge of local models with just a simple aggregation function, such as parameter averaging. Thus, it escalates the difficulty in handling catastrophic forgetting. In addition, the non-i.i.d. distribution and different numbers of samples between clients complicate the catastrophic forgetting problem in a distributed dynamic environment.

In online continual learning (OCL), i.e., continual learning for streaming data, the data comes in streaming chunks $s_1, s_2, s_3, ...$ where $s_j = \{(x_i^t, y_i^t)\}_{i=1}^{|s_j|}$ rather than in a storable $\mathcal{T}^t = \{(x_i^t, y_i^t)\}_{i=1}^{|\mathcal{T}^t|}$ manner. $|s_j|$ is relatively small, e.g., 10. The streamed data can be seen only once, as the data is expected to be unavailable after arriving. The challenge of this problem is the difficulty in acquiring the new knowledge on the currently learned task, since the model is trained insufficiently in a single epoch manner. Maintaining previously learned tasks is also difficult since the model weight $\Theta = \{\theta, \phi\}$ is continuously updated for each streamed chunk. Aside from a possibility of breaching data privacy, replay-based CL methods can not work properly on OCL since $\Theta = \{\theta, \phi\}$ can not converge to the learned task within a single epoch.

Based on the motivations above, in this thesis, we develop learning methods to address catastrophic forgetting for a dynamic environment where data gathering is conducted continuously. The proposed methods are also proposed to handle various constraints ,i.e., rehearsal and rehearsal-free, data scarcity, stand-alone and distributed, stored and streamed data. Further, we address catastrophic forgetting in more complex constraints, i.e., data scarcity with label scarcity, and data scarcity in a distributed system.

1.3 Aim and Objectives

This thesis aims to develop learning methods to address catastrophic forgetting in continual learning (CL) and its advanced sub-problems, i.e., few-shot continual learning (FSCL), federated class incremental learning (FCIL), and online continual learning (OCL). In this study, we also push forward FSCL into unsupervised few-shot continual learning (UFSCL), FCIL into federated few-shot class incremental

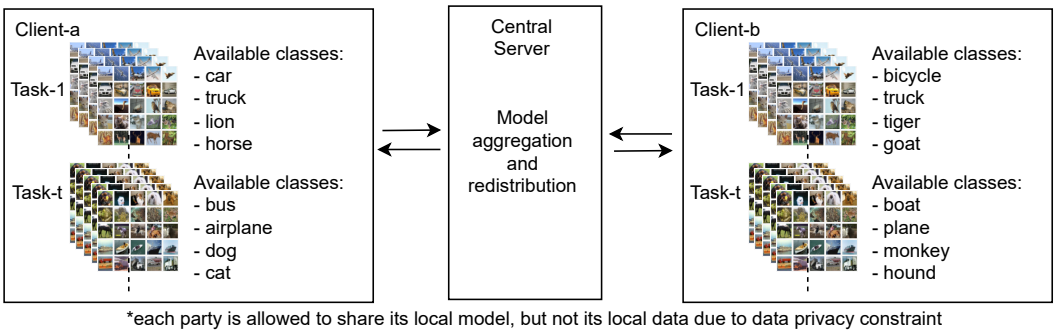


Figure 1.3: Illustration of Continual Learning under Data Privacy Constraint.

learning (FFSCIL) and develop a learning method for the proposed new problems The primary objectives are as follows:

- Develop a learning mechanism to guide the rehearsal-based CL method to adapt with the currently learned data and previously learned data.
- Develop a prompt structure and learning mechanism for rehearsal-free CL with a minimum chance of forgetting.
- Develop a customized regularization method to tackle catastrophic forgetting under data scarcity.
- Develop an unsupervised regularization method to tackle catastrophic forgetting under unlabeled data scarcity.
- Develop a communication-efficient federated class incremental learning method to mitigate catastrophic forgetting for a distributed dynamic environment while preserving data privacy.
- Develop a communication-efficient federated few-shot class incremental learning method to mitigate catastrophic forgetting under data scarcity for a distributed dynamic environment while preserving data privacy.
- Develop a customized structure and learning method to address catastrophic forgetting in streaming data that considers throughput as well as model performance.
- Develop a method for long-term time series forecasting by adopting the

concept to continual learning methods.

1.4 Research Limitation

The contribution of this thesis is considered among the pioneering studies in the field of continual learning; however, this thesis has several limitations. First, the discussed continual learning and its subproblems are mainly applied in image classification tasks. One of the chapters discussed a dynamic long-term forecasting problem applied in time-series data. This thesis does not cover the advanced vision task, such as image segmentation and other tasks that process other types of datasets, such as point cloud and text datasets.

1.5 Major Contribution

This thesis offers eight contributions as follows:

Contribution 1: The existing rehearsal-based CL methods utilize the constant factors for their loss coefficients, that leads to sub-optimality in overcoming the catastrophic forgetting [31], [35], [57]. In addition, the existing methods suffers from a class-imbalance problem since the memory size is far smaller than the currently learned data size. To overcome these problem we propose an assessor-guided learning (AGLA) strategy for continual learning where an assessor guides the learning process of a base learner by controlling the direction and pace of the learning process, thus allowing an efficient learning of new environments while protecting against the catastrophic interference problem. The assessor is trained in a meta-learning manner with a meta-objective to boost the learning process of the base learner. It performs a soft-weighting mechanism of every sample accepting positive samples while rejecting negative samples. The training objective of a base learner is to minimize a meta-weighted combination of the cross entropy loss function, the dark experience replay (DER) loss function, and the knowledge distillation loss function, whose interactions are controlled in such a way as to attain an improved performance. AGLA deployed a compensated over-sampling (COS) strategy to deal with the class imbalanced problem

Contribution 2: AGLA performs effectively in a rehearsal-based CL setting by guiding its loss component to balance its stability on previously learned tasks and currently learned task. However, it can not handle catastrophic forgetting in a rehearsal-free setting. Meanwhile, leveraging the generalization ability from the pre-trained model, the prompt-based methods has demonstrated its success in rehearsal-free settings [37], [38], [39]. However, it still suffers from catastrophic forgetting due to inter-task vector similarity and unfitted new components of previously learned tasks. On the other hand, the language-guided approach falls short of its full potential due to minimum utilized knowledge and participation in the prompt tuning process [60]. To overcome these challenges, we design LEAPGen, a new prompt generation and its learning method to handle catastrophic forgetting in a rehearsal-free setting. The key principles of our approach are (1) LEAPGen utilizes fixed language embedding as input for prompt generation instead of a tasks-shared learnable vector that is continuously updated. The input is selected from the catalog of language descriptors based on their similarity to the input. (2) LEAPGen utilizes task-wise generators instead of growing generators to generate a prompt component; thus, the current task generators are frozen in the upcoming tasks' training. (3) LEAPGen utilizes a new soft task ID prediction to limit the search space of matching descriptors. The mechanism is different from the task ID prediction in the previous methods. (4) LEAPGen utilizes the generated language-based prompt as the auxiliary data to be appended to the input embedding.

Contribution 3: AGLA and LEAPGen works on rehearsal-based and rehearsal-free continual learning where each class carries an abundant of labeled samples. However, the methods can not handle catastrophic forgetting under data scarcity. To overcome the challenge, we propose a FLat-tO-WidE appRoach (FLOWER) for few-shot continual learning problems. FLOWER is configured as a prototypical network [61] driven by the prototypical loss. FLOWER puts forward the flat-to-wide learning idea [42], [62], to discover flat-wide minimum regions. It minimizes the losses of noise-perturbed parameters of the feature extractor over M runs. The flat-wide parameters are maintained in the subsequent few-shot learning tasks by constraining feature extractor parameters to prevent the catastrophic forgetting

problem. Our approach differs from [42] heavily focusing on the flat minimum regions, resulting in over-dependence on the base learning task. In addition, no memory is imposed by FLOWER, and the wide learning concept is integrated as an attempt to drive flat regions to be wide, thus scaling well for large-scale problems. The concept of ball generator is integrated in FLOWER to prevent the over-training problem due to scarce samples in few-shot learning tasks.

Contribution 4: FLOWER has proven its effectiveness in supervised few-shot continual learning. However, in real-world application such as remote sensing, where the data has varying camera parameters, spectral ranges, resolutions, etc, class labels are hardly available due to the high cost of annotation. To address the catastrophic forgetting under the unlabeled data scarcity we propose a novel unsupervised flat-wide learning approach (UNISA) for unsupervised few-shot continual learning for remote sensing image scene classifications, which do not depend on any labeled samples for its model updates. UNISA is developed from the idea of prototype scattering and positive sampling for learning representations, while the catastrophic forgetting problem is tackled with the flat-wide learning approach combined with a ball generator to address the data scarcity problem. The positive sampling leverages the pseudo-label generated by the unsupervised method, i.e., k-means clustering.

Contribution 5: AGLA and LEAPGen works excellently on a stand-alone system learning, where CL is conducted by a single agent that carries all the data. However, the methods can not perform adequately in a distributed system where CL is conducted by collaborative agents in a federated manner. Meanwhile, the current federated class incremental learning (FCIL) state-of-the-art (SOTA), i.e., GLFC[47] and LGA[59], train and share whole backbone parameters, resulting in a large number of parameters to train that leads to longer training time, and a high communication cost. Second, in the current SOTAs, the clients privately share perturbed images with the central server for the aggregation process. It may violate data privacy principles since perturbation does not guarantee information leakage. To address the challenges, we propose a new approach named prototype-injected prompt (PIP). In our approach, a client trains and shares only a small set of parameters called a prompt, while the backbone model remains frozen and

unshareable. As an advantage, we could utilize any backbone with any complexity, number of parameters, and size. Our approach utilizes deep features as a shareable prototype to handle the non-i.i.d. data between clients. As the second advantage, our approach offers far smaller prototypes and guarantees data privacy as deep features represent a totally different structure and value compared to raw or perturbed samples.

Contribution 6: The existing FCIL works, i.e., LGA [59], TARGET [63], and LANDER [64], and PIP assume that each class carries an abundant labeled sample that may not be applicable in real-world applications. We emphasize the data scarcity issue leading to the prototype-bias and over-fitting problems in FCIL and define a new problem, namely Federated Few-Shot Class Incremental Learning (FFSCIL). We propose a **novel** method for the FFSCIL problem termed Unified Optimized Prototype Prompt (UOPP) built upon a prompt learning framework coupled with static and dynamic prototypes optimized by Neural Ordinary Differential Equation (ODE). The proposed method utilizes an adaptive dual-head to enhance its predictive accuracy. To our knowledge, our proposed method is the **first** prompt-based method that integrates prompt tuning, prototype rectification by a trainable network, and adaptive dual classifiers in a single pipeline.

Contribution 7: LEAPGen works excellently on rehearsal-free continual learning, where the current learned task is stored. Thus, the training phase can be repeated multiple times. However, LEAPGen can not overcome the catastrophic forgetting in streaming data. To overcome the gap, we develop a novel learning method that achieves two compulsory aspects in OCL„ i.e., (1) achieving stability-plasticity and (2) being efficient both in training and inference phases. To achieve those 2 mandatory components, we develop a novel prompt online learning (PROL) method that accommodates the following components: (1) a single lightweight prompt generator for all tasks, (2) a set of scalers and shifters associated with the class-wise learnable keys, (3) PTM generalization preservation via cross-correlation matrices, and (4) hard-soft learning updates for a better parameter tuning. The generator is trained only on the first task to avoid forgetting and achieve stability, while scalers and shifters are trained on every task to achieve plasticity. To achieve high throughput, our lightweight generator has fewer parameters than a single

generator in [65] and fewer than a single task-specific prompt in [38].

Contribution 8: The previous methods are proposed to tackle catastrophic forgetting, which is applied in the classification domain and image datasets. The last contribution proposes an adaptation of the continual learning concept into a regression domain, i.e., dynamic long-term time-series forecasting. A reliable long-term time-series forecaster is highly demanded in practice but comes across many challenges, such as low computational and memory footprints as well as robustness against dynamic learning environments. We propose Meta-Transformer Networks (MANTRA) to deal with the dynamic long-term time-series forecasting tasks. MANTRA relies on the concept of fast and slow learners where a collection of fast learners learns different aspects of data distributions while adapting quickly to changes. A slow learner tailors suitable representations to fast learners. Fast adaptations to dynamic environments are achieved using the universal representation transformer layers, producing task-adapted representations with a small number of parameters.

1.6 Organization of The Thesis

This thesis is organized into 11 chapters, where each chapter discusses a particular issue. Chapter 1 presents the background, problem statements, motivation for this research, objectives, major contributions, and the structure of this thesis. Chapter 2 introduces the literature review on continual learning, few-shot continual learning, online continual learning, federated class incremental learning, and brief summary on the related area, i.e., meta learning, few-shot learning, federated learning, and data augmentation. Chapter 3 presents Assessor Guided Learning (AGLA), that dedicated to solve rehearsal-based Continual Learning (CL). The material of chapter 3 is published in the Information Sciences journal [Core A]. Chapter 4 proposes a novel Language as Prompt Generator (LEAPGen) to address catastrophic forgetting in rehearsal-free Continual Learning. The material of chapter 4 is published in The Thirteenth International Conference on Learning Representations (ICLR 2025) [Core A*]. Chapter 5 discusses Few-Shot Continual Learning (FSCL) and proposes Flat-to-Wide Approach (FLOWER), where the

materials is published in the IEEE Transactions on Neural Networks and Learning Systems journal [IF 10.2, CORE A*]. Chapter 6 pushes forward FSCL into a new sub-problem, i.e., Unsupervised Few-Shot Continual Learning (UFSCL), and proposes a novel Unsupervised Flat-Wide Approach (UNISA) to tackle UFSCL. The material of Chapter 6 is published in the IEEE Transactions on Geoscience and Remote Sensing [IF 7.5, #2 in RS]. Chapter 7 is dedicated to address Federated Class Incremental Learning (FCIL), where we propose new baselines and a novel Prototype Injected Prompt (PIP). The material of Chapter 7 is published in he 33rd ACM International Conference on Information and Management (CIKM 2024) [CORE A]. Chapter 8 defines a new problem named Federated Few-Shot Class Incremental Learning (FFSCIL) and proposes a novel Unified Optimized Prototype Prompt (UOPP) to tackle FFSCIL. The material of Chapter 8 is published in ICLR 2025 [CORE A*]. Chapter 9 is focused on rehearsal-free Online Continual Learning (OCL) and proposes a novel method named Prompt Online Learning (PROL), where the material is accepted at the International Conference on Computer Vision (ICCV 2025). Chapter 10 proposes Meta Transformer Networks (MANTRA) for dynamic long-term time series forecasting, where the material is published in the IEEE Transactions on Artificial Intelligence journal. The last chapter presents the concluding remarks of this thesis and the future directions of the study.

Chapter 2

Literature Review

This chapter presents a comprehensive literature review on continual learning, few-shot continual learning, online continual learning, and federated class incremental learning. This chapter also presents brief reviews on few-shot learning, federated learning, meta learning, and data augmentation.

Part of this chapter is published as: MA Ma'sum, M Pratama, I Skrjanc. Latest Advancements Towards Catastrophic Forgetting under Data Scarcity: A Comprehensive Survey on Few-Shot Class Incremental Learning. ArXiv:2502.08181. https://doi.org/10.48550/arXiv.2502.08181

2.1 Continual Learning

In continual learning, a machine learning model, e.g., a deep neural network, is trained in a sequence of tasks where the tasks are disjoint from each other [24] [54]. For example, in image classification, in the first task, a model is trained for 10 classes. In the next task (task 2), the model is trained for the other 20 classes (classes 10-29), where the 20 new classes are different from the previous 10 classes. The data from the previous 10 classes may not be available anymore, but the model needs to recognize all the learned classes (0-29). Then the model is trained for other tasks i.e,. task 3, task 4, and so on. In continual learning, a deep neural network is trained to learn gradually, like a human being, instead of being trained in a single task that contains all classes. Practically, continual learning accommodates

dynamic environment conditions where not all data is available at one time, but is continuously gathered. The main challenge in continual learning is that the model should be able to acquire knowledge on new tasks while maintaining its knowledge in previously learned tasks. A deep neural network model tends to experience catastrophic forgetting, where a model achieves high performance in a new task and degrades (forgets) its performance on previously learned tasks. Generally, the goal of continual learning is to develop a learning mechanism to make the model achieve high performance in new tasks (plasticity) while maintaining its high performance in previous tasks (stability). The difference of continual learning and conventional learning mechanisms is shown in Figure 2.1

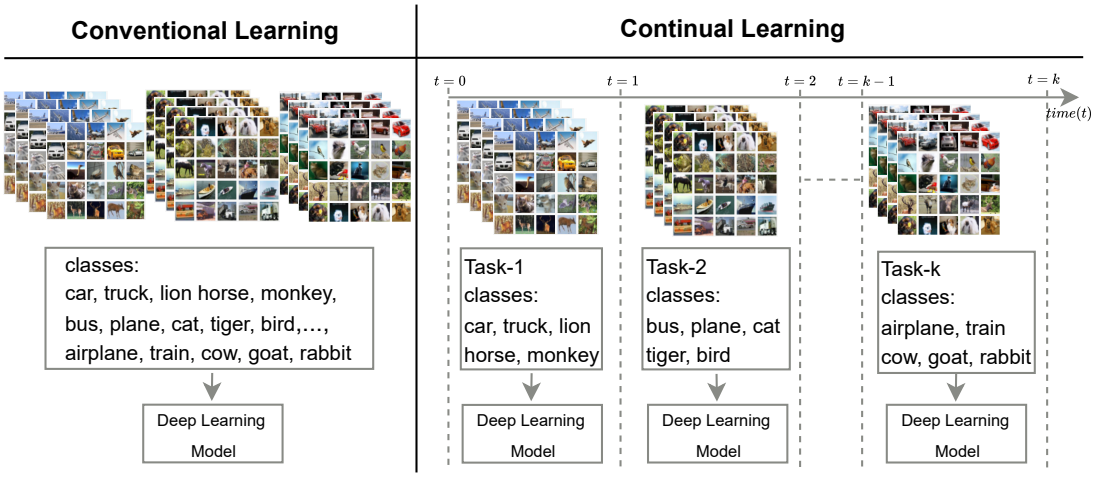


Figure 2.1: Difference between conventional learning and continual learning.

2.1.1 Type of Setting

a). Task Incremental Learning (TIL) is a problem setting where a model is trained to a sequence of tasks $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}_t, t \in \{1, ..., T\}$, each task $\mathcal{T}^t$ has its training set $T^t = \{x_n, y_n, id_n\}_{n=1}^{N^t}$, $x_n \in \mathcal{X}$ represents an input sample and $y_n \in \mathcal{Y}$ denotes its true class label, id_n stands for the task id. T, N^t respectively denote the number of tasks and the size of $t-th$ task.

b). Class Incremental Learning (CIL) is a similar setting with TIL, but the task id id_n is not visible to the model in the inference phase.

c). Domain Incremental Learning (DIL). Different from the previous two settings, in DIL, $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}_t, t \in \{1, ..., T\}$ carry the same class labels, but the domain of tasks are different, i.e., $\mathcal{D}^1 \neq \mathcal{D}^2 \neq ... \neq \mathcal{D}^t, t \in \{1, ..., T\}$.

The illustration of the CL settings is visualized in figure 2.2. Class incremental learning (CIL) is the most challenging problem among the 3 types of CL settings. Thus, some of the previous CL studies focused on CIL only.

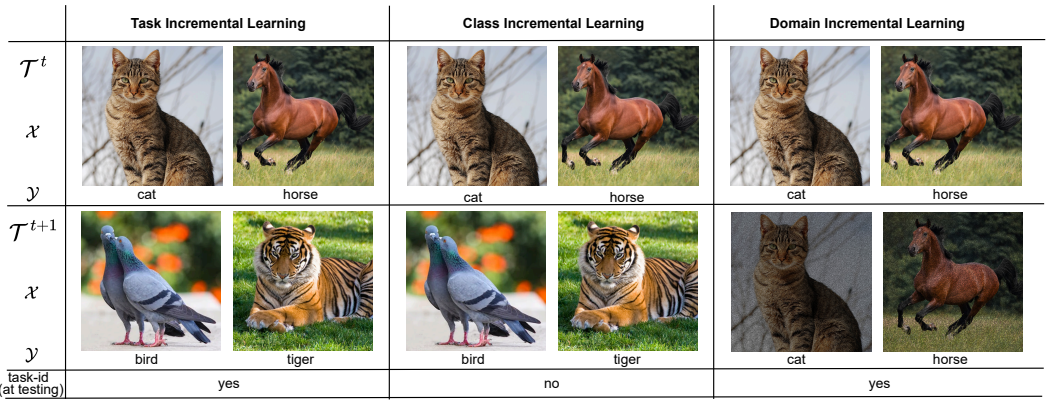


Figure 2.2: Types of CL settings, i.e., task-incremental learning, class incremental learning and domain incremental learning.

2.1.2 Continual Learning Approaches

Figure 2.3 visualize the existing CL approaches that generally can be classified into 2 main categories, i.e., without (w/o) pre-trained model and with (w/) pre-trained model. The first category trains the whole backbone model from scratch, which can be divided into 3 main sub-categories, i.e., regularization-based, structure-based, and memory-based approaches. The memory approach has a sub-category, namely the bias correction approach, which adds trainable bias layers to its backbone model. The second main category leverages the weight of the open-access foundation model as the initial state of its backbone. Having generalization ability, this approach requires a far smaller number of trainable parameters to learn the task sequence. This approach usually utilizes a pre-trained visual transformer (ViT) as the backbone, while the earlier approach utilizes ResNet as its backbone. The following sub-sections discuss the details of each approach.

2.1.3 Regularization-based Approach

Regularization-based method designs a regularization term that prevents important parameters of old tasks from deviating. Important parameters of old tasks are estimated, and an important parameter matrix is integrated in the regularization term. Elastic Weight Consolidation (EWC) [66] adopts the Fisher information matrix (FIM) to estimate the importance of network synapses. Synaptic intelligence (SI) [26] utilizes an accumulated gradient to quantify the significance of network parameters and incurs less expensive computation than FIM. An alternative is offered by memory aware synapses using an unsupervised and online approach [25]. Learning without forgetting (LWF) [67] applies the knowledge distillation approach making sure that current network outputs are close to previous network outputs. An online version of EWC is put forward in [27] where the parameter importance is approximated with the Laplace approximation. It is found that the regularization mechanism is better performed in the neuron level than at the synaptic level because of the hierarchical nature of neural networks [68]. That is, the regularization step is achieved by controlling the learning rates of the stochastic gradient descent (SGD) method. Similar approach is adopted in [69], but it considers the common information of each task enabling a node to be shared by different tasks. This strategy is capable of scaling up the regularization-based method for large-scale problems. Another approach to scale the regularization-based approach is done with the classifier's projection [62]. This approach induces wide local optimum regions, i.e., overlapping regions. The regularization-based approach depends on the task IDs and task boundaries, thus performing poorly for the class-incremental learning problems.

2.1.4 Structure-based approach

Structure-based approach is pioneered by progressive neural networks [70], where new network components are added to deal with new tasks while freezing old network parameters to handle the catastrophic forgetting problem. The complexity of this approach grows as the learning tasks increase. An error-based network growing method is put forward in [71]. It utilizes a selective-based retraining approach to handle the catastrophic forgetting problem. Learn-to-grow approach is developed

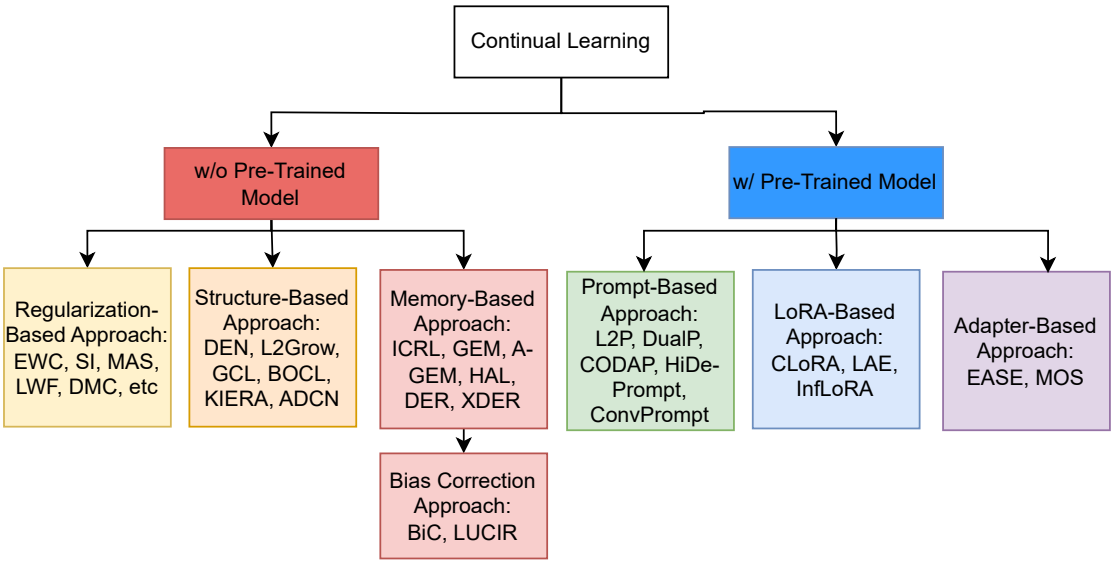


Figure 2.3: Types of CL approaches, i.e., regularization-based approach, structure-based approach, memory-based approach, bias correction approach, prompt-based approach, LoRA-based approach, and adapter approach.

in [72] where neural architecture search (NAS) is integrated to obtain the best network configuration of new tasks while isolating old network parameters. The same concept is implemented for graph continual learning in [73]. The Bayesian approach is incorporated in [74]. The idea is akin to [72], but the Bayesian approach is adopted instead of NAS to find the best network structure. These approaches are computationally prohibitive and calls for the presence of tasks IDs and task's boundaries. [75], [76] offer a data-driven structural learning approach for unsupervised continual learning where the bias-variance decomposition is put forward to grow or prune the network structure. The catastrophic forgetting problem is dealt with by the centroid-based experience replay mechanism [75] or the knowledge-distillation approach [76]. Although it is free of the task IDs and task boundaries for learning and predicting, the data-driven structural learning technique does not guarantee an optimal structure.

2.1.5 Memory-based Approach

Memory-based approach replays old samples stored in the memory when learning new tasks. iCaRL [31] utilizes the exemplar set of each class, where the classifica-

tion step is performed via the nearest-mean strategy. GEM [77] and A-GEM [78] store past examples to determine the forgetting case used to constrain the model update. HAL [79] puts forward the concept of anchor points optimized to maximize the forgetting case via a bilevel optimization approach. Prediction should not change to these anchor points when learning new tasks. DER [57] combines the concept of knowledge distillation and experience replay. [80] proposes the concept of knowledge amalgamation as a post-processing approach of class-incremental continual learning. These methods suffer from the class imbalanced problem since the size of the memory buffer is much less than the new samples.

2.1.6 Bias Correction Approach

The bias correction approach is the extension of the memory-based approach by adding trainable bias correction layers to its backbone. BiC implements the layer as task-wise trainable fully connected layers [35]. Thus, task-associated layers projects the samples' features into more separable forms. This mechanism enhance classes' separability and recognition of the rehearsal-based method. Another work, i.e., LUCIR, improves the bias correction layer with a better rebalancing between the currently trained bias correction layer and the layers that belong to previously learned tasks [34].

2.1.7 Prompt-based Approach

The prompt-based approach proposes a breakthrough solution for Class Incremental Learning (CIL) by training tiny task-aware parameters called **prompts** on top of a frozen pre-trained feature extractor, e.g., visual transformer (ViT)[81] that contains far bigger parameters. ViT is a sequence of L-layered model $f_\theta(.)$ where a layer-l ($l \in [1..L]$) contains H Multi-Head Self-Attention (MSA) combined with a Multi-Layer Perceptron (MLP) block, Norm, addition, and residual connections. Given an input patched embedding $x_{e_{l,h}}$, the hth head MSA of layer-l, parameterized by key-query-value matrices, i.e., $W^Q_{l,h}, W^K_{l,h}, W^V_{l,h}$ performs self-attention function as defined in equation below:

$$A(Q_{l,h}, K_{l,h}, V_{l,h}) = Softmax((Q_{l,h}K^T_{l,h})/\sqrt{D_h})V_{l,h} \tag{2.1}$$

where $Q_{l,h} = x_{e_{l,h}} W^Q_{l,h}, K_{l,h} = x_{e_{l,h}} W^K_{l,h}, V_{l,h} = x_{e_{l,h}} W^V_{l,h}$ are query, key, and value and D_h denotes their dimension. Prompt-based CL attaches a small set of trainable parameters called prompts into a frozen pre-trained ViT model (fixed ψ) [37], [38]. The attachment techniques are mainly divided into 2 types, i.e., prompt tuning [82] and prefix tuning [83]. Prompt tuning prepends a prompt vector $P \in \mathbb{R}^{L_P \times D}$ into query, key, and value of MSA. Suppose the prompt is divided into head-wise parts, i.e., $P = [P_1, P_2, ...P_H]$, then $P_h \in \mathbb{R}^{L_P \times D/H}$ is prepended into h^{th} head of lth layer, and the attention mechanism is computed as $A([P_h : Q_{l,h}], [P_h : K_{l,h}], [P_h : V_{l,h}])$. The symbols L_P, D, and ":" denote prompt length, embedding dimension and concatenation operator respectively. Prefix tuning as utilized in DualPprompt [38] divides P_h into pair of P^K_h and P^V_h for key and value respectively, then the attention mechanism becomes $A(Q_{l,h}, [P^K_h : K_{l,h}], [P^V_h : V_{l,h}])$.

Based on the prompt structure point of view, we can categorize the prompt-based approach into several categories, i.e., (1)Pool-based approach, e.g., L2P[37] that uses a pool as prompt container and select a prompt based on input-similarity, (2) Task-specific approach, e.g., DualPrompt[38], SPrompt[84], HiDE-Prompt[85] and CPrompt [86] that train specific prompt for a specific task and uses task identifier in the inference process, (3) Growing component approach, e.g., CODA-P[39], EvoPrompt[87], and ConvPrompt[65] that increase prompt components to adapt with a new task, instead of the task identifier. DualPrompt and CPrompt utilize learnable keys as a task predictor, while HiDe-Prompt utilizes an unstructured representation (clusters) as the task predictor. CODA-P adds new hand-crafted (fixed-size) learnable parameters as new prompt components, while EvoPrompt and ConvPrompt generate the new prompt components using newly added generators. Therefore, EvoPrompt and ConvPrompt can be considered as a growing generator approach.

2.1.8 LoRA-based Approach

Similar to the prompt-based approach, the Low Rank Adaptation (LoRA)-based approach trains a relatively small number of parameters to learn the task sequence. Suppose that W is the frozen weight of ViT, the LoRA-based method add W with a pair of trainable matrices A^t and B^t to learn each new task t. Given t is in the

range of $[1..T]$, then the accumulated weight for the ViT after learning task 1 to t is formulated as in the equation below.

$$W^t = W^{t-1} + A^t.B^t = W + \sum_{i=1}^{t} A^i.B^i \tag{2.2}$$

Several CLoRA models utilize cross-attention along with customized token selection to handle task similarity [88]. InfLoRA designs a matrix B reduction mechanism by leveraging its gradient magnitude [89]. In addition, it minimizes the task interference by reducing their similarity in a projected subspace. LAE combines the LoRA concept with prompt and adapter tuning [90].

2.1.9 Adapter-based Approach

Adapter-based approach train a sequence of adapter $A^1, A^2, ..., A^t$ to learn the task sequence $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^t$, where each A^i associated with task i. Different from the previous 2 approaches that infused the trainable parameters inside the ViT layers, the adapter approach works on the feature level where A^t takes a ViT layer's output as its input [91], [92]. However, besides its high-level simplified structure and intuitive flow, this approach has a remarkable drawback, i.e., it requires samples from previously learned classes for tuning the adapters. Thus, in the rehearsal-free setting, its performance is lower than in a rehearsal-based setting.

2.2 Few-Shot Continual Learning

Few-shot continual learning (FSCL), or more specifically known as few-shot class incremental learning (FSCIL), simulates a real-world problem where a machine learning model is deployed to learn dynamic environments with very few samples [40]. FSCIL problem is broadly applied in various areas, e.g., medical[93], remote sensing[94], audio processing[95], and data mining [96]. The presence of the data scarcity constraint in FSCIL hardens the catastrophic forgetting problem to be solved. As clear proof, rehearsal-based class incremental learning (CIL) methods [97], and event joint-training mechanism fail to overcome the FSCIL problem.

Thus, it calls for advanced solutions beyond CIL methods. On the other hand, popular rehearsal-based methods in the CIL problem cannot be applied in the FSCIL problem because they might result in all samples being stored in memory. Few-Shot Class-Incremental Learning (FSCIL) is defined as: Given a sequence of tasks $[0, 1, 2, ..., T]$ where each task t carries a labeled training set $\mathcal{T}^t = \{(x_i^t, y_i^t)\}_{i=1}^{|\mathcal{T}^t|}$, where $x_i^t \in \mathcal{X}^t$ denotes an input image and $y_i^t \in \mathcal{Y}^t$ denotes its label, and $|.|$ denotes the cardinality. Each task t is disjoint from another task t'., i.e., $\forall_{t,t'} \mathcal{T}^t \cap \mathcal{T}^{t'} = \emptyset$. The first task ($t = 0$) carries abundant training samples while the remaining tasks ($t > 0$) carries only far smaller samples than the first task i.e $|\mathcal{T}^0| >> |\mathcal{T}^{t>0}|$. For a convenient way, task-0 is called the base task, while the rest are called the few-shot task (FS task). Let a deep neural network $g_\phi(f_\theta(.))$ be parameterized by θ and ϕ where $f(.)$ and $g(.)$ are the feature extractor and classifier respectively. **The objective of FSCIL** is to achieve the optimum parameters $\theta*$ and $\phi*$ so that $g_{\phi*}(f_{\theta*}(.))$ recognizes all the learned classes from the first task until the current task t, i.e., $\{\mathcal{T}^0, ..., \mathcal{T}^t\}$. Formally, FSCIL's objective is defined as in equation 2.3.

$$\theta*, \phi* = \arg\min_{\theta,\phi} \mathbf{E}_{(x,y)\in\mathcal{T}^0\cup\mathcal{T}^1...\mathcal{T}^t} \mathbf{I}(y = g_\phi(f_\theta(x))) \tag{2.3}$$

where $\mathbf{E}$ denotes the expectation, $\mathbf{I}$ denotes the binary function that outputs 1 if the condition is met and 0 otherwise.

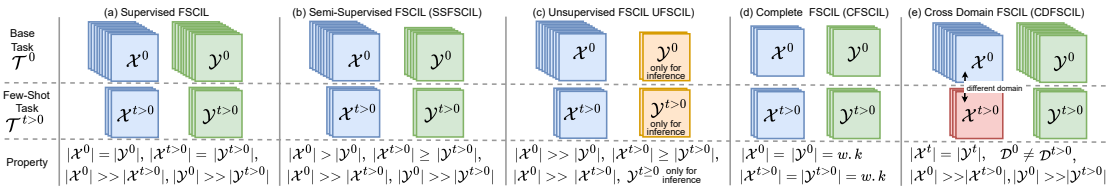


Figure 2.4: Types of FSCIL settings and their properties (a) Fully-Supervised, (b) Semi-Supervised (c) Unsupervised (d) Complete (e) Cross-Domain FSCIL.

2.2.1 Types of FSCIL Setings

There are 3 types of FSCIL setting based on the training mechanism with respect to the number of images and labels, i.e., $|\mathcal{X}|$ and $|\mathcal{Y}|$ as follows:

a). Supervised FSCIL or just FSCIL [40] utilizes the fully supervised training mechanism that is conducted in the case where the number of images and labels

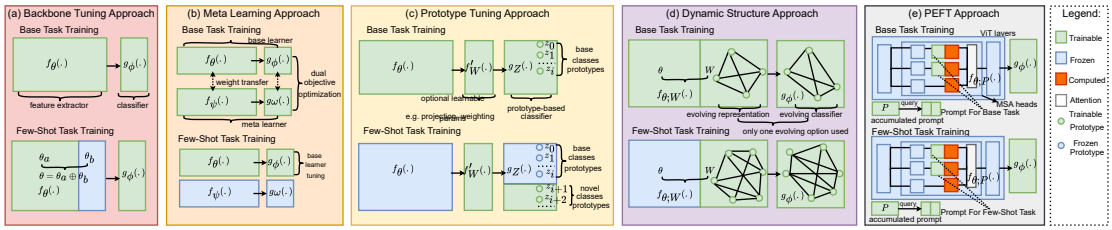


Figure 2.5: Visualization of FSCIL Approaches, i.e., (a) backbone tuning approach, (b) meta learning approach, (c) prototype tuning approach, (d) dynamic structure approach, and (e) parameter efficient fine tuning (PEFT) approach.

are the same for every task, i.e., $|\mathcal{X}^t| = |\mathcal{Y}^t|$ for all $t \in [1, 2..T]$. In the few-shot task ($t > 0$), the number of images and labels is the same and equal to the k-shot value.

b). Semi-Supervised FSCIL (SSFSCIL)[58] deploys supervised and unsupervised training mechanisms in the case where the number of images is greater than the number of labels, i.e., $|\mathcal{X}^t| > |\mathcal{Y}^t|$ for any $t \in [1, 2..T]$. The semi-supervised training is not necessarily conducted in all tasks; it can be conducted only in the base task where abundant samples are available, but only a few labels are provided.

c). Unsupervised FSCIL (UFSCIL)[98] deploys only unsupervised training mechanisms to deal with the unavailability of training label, i.e., $|\mathcal{Y}^t| = 0$ for all $t \in [1, 2..T]$. A small fraction of labels are available, i.e., $|\mathcal{Y}^t| << |\mathcal{X}^t|$, only for inference purpose since the model needs to give a predicted label for any input x.

d). Complete FSCIL (CFSCIL)[99] remove the assumption that the base task has abundant labeled samples. Thus, all task only hold a few labeled samples and organized in n-way k-shot setting, i.e., $|\mathcal{X}^t| = |\mathcal{Y}^t| = w.k$ for all $t \in [1, 2..T]$, where w is the number of classes in each task, and k is the number of samples (shot) for each class.

e). Cross Domain FSCIL (CDFSCIL)[99] is similar to supervised FSCIL, but assumes that the few-shot task and base task have different domains, i.e., $\mathcal{D}^0 \neq \mathcal{D}^{t>0}$. The illustration and property of all settings is visualized in fig.2.4.

2.2.2 Backbone Tuning Approach

The backbone tuning approach trains the whole backbone and classifier parameters in the base task, then trains the classifier and some of the backbone parameters in the few-shot tasks, as illustrated in Figure. 2.5 (a). Suppose that $\theta = \theta_a \oplus \theta_b$, where θ_a and θ_b are the front part (earlier layers) and the back part (latest layers) of the feature extractor parameters, and $\oplus$ denotes concatenation. In the base task, the method optimizes θ and ϕ, while in the few-shot tasks, the method freezes θ_a and optimizes θ_b and ϕ. In general, the objective of the backbone tuning approach is defined as in equation 2.4.

$$\begin{aligned} \theta*, \phi* &= \arg\min_{\theta,\phi} \mathbf{E}_{(x,y)\in\mathcal{T}^0} \mathbf{I}(y = g_\phi(f_\theta(x))) \text{ if } (t = 0) \\ \theta_b*, \phi* &= \arg\min_{\theta_b,\phi} \mathbf{E}_{(x,y)\in\mathcal{T}^0\cup\ldots\mathcal{T}^t} \mathbf{I}(y = g_\phi(f_\theta(x))) \text{ else} \end{aligned} \tag{2.4}$$

Several strategies are enforced in this approach, i.e., GPTree [100] utilizes a Gaussian Process to approximate the distribution of classes with few-shot labeled samples. Similarly, LDC[101] enforces a learnable model to approximate the distribution. F2M[42] and FLOWER[102] create a flat loss region by perturbating latest part of the backbone parameters, i.e., $\mathcal{L}((x, y); \theta_a \oplus (\theta_b + \epsilon); \phi))$ is minimal for any $-b \leq \epsilon \leq b$, where ϵ and b is perturbation noise and bound respectively. ERDR[103] and [104] utilize customized regularization strategies, i.e., entropy and subspace regularization, respectively. FeSSS[105] utilizes a self-supervised method and feature fusion to obtain more discriminative features. Mixed-up features are also utilized by M2SD[106] and combined with mixed-up distillations. SoftNet[107] adds a binary mask m and a gating function to the sub-network of θ. It enables different weights of θ for different samples. On the other hand, WaRP[108] transforms a sub-net of θ with multiple axis rotation to gain distinctive representations.

2.2.3 Meta-Learning Approach

Inspired by MAML[109] meta learning approach (fig. 2.5 (b)) trains both meta learner and base learner with the different sets of data, i.e., training and validation set. Validation set can be obtained from auxiliary (open) dataset or by transforming the training set with a sequence of operations. Suppose that $f_\psi(.)$ is the meta learner parametrized by ψ, and $\mathcal{T}_v^t =$ Transform$(\mathcal{T}^t)$ is validation set transformed

from $\mathcal{T}^t$. Then the dual objective of meta-learning-based FSCIL is formulated in Eq. 2.5.

$$\begin{aligned} \psi* &= \arg\min_{\psi} \mathbf{E}_{(x,y)\in\mathcal{T}_v^0\cup\ldots\mathcal{T}_v^t}\mathcal{L}(f_\psi(x)) \\ \text{s.t. } \theta*, \phi* &= \arg\min_{\theta,\phi} \mathbf{E}_{(x,y)\in\mathcal{T}^0\cup\ldots\mathcal{T}^t}\mathbf{I}(y = g_\phi(f_\theta(x))) \end{aligned} \tag{2.5}$$

MetaFSCIL[43] utilizes bidirectional guided modulation (BGM) that allows meta and base learners to map their weights one to another. C-FSCIL[99] utilizes meta learning to perform 3 modes of prototypes training, i.e., averaged prototypes, retraining on bipolarized-prototypes, and retraining on nudged-prototypes. Both MetaFSCIL and C-FSCIL train the meta learner only in the base task; it is frozen afterwards. XTarNet[110] deploys CNN as a meta learner, utilizes a pretrained backbone, and adds two additional modules, i.e., MergeNet and TConNet, that perform weights merger and conditioning classifier, respectively. A similar mechanism is performed by LIMIT[111] that utilizes meta learning for prototype tuning. LIMIT generates multiple prototypes from an input instance, then the prototypes are calibrated by a meta learner, and then aggregated with the inner product into a final prototype.

2.2.4 Prototype-Tuning Approach

The prototype tuning approach is the most common method in FSCIL due to its simplicity and scalability. Suppose that z_c is the prototype of class c, $Z^t = z_c^t$ is the prototype set of task t contains prototypes of any class c exists in $\mathcal{T}^t$, and $Z = Z^0 \cup Z^1, \ldots \cup Z^t$ is the prototype set of all learned classes from task 0 to task t. The approach utilizes prototype-based classifier $g_Z(.)$ instead of standard MLP classifier $g_\phi(.)$. Thus, it optimizes Z in every task, while the feature extractor parameter θ is optimized only in the base task. A prototype-based method may utilize an additional network $f'_W(.)$ parameterized by W between backbone $f_\theta(.)$ and prototype-based classifier $g_Z(.)$ for projection, weighting, or refinement. The objective of the prototype tuning approach is defined in eq. 2.4.

$$\begin{aligned} \theta*, W*, Z* &= \arg\min_{\theta,W,Z} \mathbf{E}_{(x,y)\in\mathcal{T}^0}\mathbf{I}(y = g_Z(f'_W(f_\theta(x)))) \text{ if}(t=0) \\ W*, Z* &= \arg\min_{Z} \mathbf{E}_{(x,y)\in\mathcal{T}^0\cup\ldots\mathcal{T}^t}\mathbf{I}(y = g_Z(f'_W(f_\theta(x)))) \text{ else} \end{aligned} \tag{2.6}$$

IDLVQ optimizes vector Z to directly classify backbone feature $f_\theta(x)$. There are several variations to implement $f'_W(.)$, i.e., dynamic relation projection as in SPRR[112], subspace projection as in SFMS[113], orthogonal projection as in ORCO[45], merged multiple embeddings by attention as in SAKD[114], kernel analytics module as in GKEAL[115], positive-negative weighting matrices as in CLOM[116], compositional weighting as in Comp-FSCIL[117], prototype rectification as in TEEN [118] and YOURSELF[119] or any customized refinement. Along with prototype tuning, this approach also applies conceptual learning methods, e.g., contrastive learning as in SAVC[120], virtual(augmented) prototypes as in FACT[121] and ALICE[122], neural collapse as in NC-FSCIL[44], and Slow-and-Fast learning as in MgSvf[123].

2.2.5 Dynamic Architecture Approach

A dynamic architecture approach tackles FSCIL challenges by incrementally evolving its architecture to adapt to the new tasks. Fig. 2.5 (d) shows that this approach can be implemented into two types of growing component, i.e., growing representation W or growing classifier ϕ. TOPIC[40] grows neural gas W, a graph-like structure to obtain a more discriminative class representation. BiDist[124] adds a learnable weight for each task W^t, then uses bilateral distillation of the current and previous tasks' representation. CEC[125] transforms an incrementally added linear classifier into a graph structure. S3C grows stochastic classifiers by continually adding 4-angles (0,90,180,270) representations. LEC-Net[126] expands the learnable compression network W to mitigate overfitting due to data scarcity. Similarly, DSN[127] expands the self-activation and compression network. The objective of the dynamic architecture approach is defined in Eq. 2.7.

$$\begin{aligned} \theta*, W*, \phi* &= \arg\min_{\theta,W,\phi} \mathbf{E}_{(x,y)\in\mathcal{T}^0} \mathbf{I}(y = g_\phi(f_{\theta;W}(x))) \text{ if } (t=0) \\ W*, \phi* &= \arg\min_{W,\phi} \mathbf{E}_{(x,y)\in\mathcal{T}^0\cup\ldots\mathcal{T}^t} \mathbf{I}(y = g_\phi(f_{\theta;W}(x))) \text{ else} \end{aligned} \tag{2.7}$$

2.2.6 Parameter Efficient Fine Tuning (PEFT) Approach

The PEFT approach proposes a unique new solution for FSCIL by training small-sized parameters (**prompts**) on top of a frozen pre-trained backbone, e.g., visual

transformer (ViT). As in CL, the prompt-based method attaches small learnable prompts into a frozen pre-trained ViT model (fixed ψ) [37]. The attachment techniques are mainly divided into 2 types, i.e., prompt tuning and prefix tuning [83]. Prompt tuning as in L2P [37], inserts a prompt vector P into query, key, and value of MSA, then the attention mechanism is computed as $A([P \oplus Q], [P \oplus K], [P \oplus V])$, where $\oplus$ denotes the concatenation operation. Prefix tuning, as in DualP [38] divides P into pair of P^K and P^V for key and value, respectively, then the attention mechanism becomes $A(Q, [P^K \oplus K], [P^V \oplus V])$. The structure of P can be defined by several options, i.e., pool-based structure as in L2P, task-wise structure as in DualP, growing component structure as in CODA-P[39], or any customized structure.

ASP[128] deploys task-aware task-invariant prompts to improve prompt selection during the testing phase. Besides, it utilizes a prompt encoder that produces an additional (second) prompt. Privilege [129] leverages vision and language modalities and prepends a concatenation of vision and language prompts. Similarly, FineFMPL[130] performs language-guided prompting, i.e., tuning the vision prototype by language prototypes. In addition, it utilizes two types of vision prototypes, i.e., global level and object level prototypes. ApproxFSCIL[131] concatenates ViT layers with norm layers and the tanh function instead of the prompt, then bounds the training process with transfer risk and consistency risk. Formally, the objectives of the PEFT approach are presented in Eq.2.8.

$$P*, \phi* = \arg\min_{P,\phi} \mathbf{E}_{(x,y)\in\mathcal{T}^0\cup\ldots\mathcal{T}^t} \mathbf{I}(y = g_\phi(f_{\theta;P}(x))) \tag{2.8}$$

2.3 Online Continual Learning

2.3.1 Non-PTM-based Approach

Recent Online Continual Learning (OCL)studies shed light on various approaches to handle catastrophic forgetting in streaming data. The backbone tuning approach tunes the whole model with unique mechanisms,, e.g., cross-task class discrimination as in GSA[132], dual-view as in DVC[133], proximal point method as in LPR[134], real-synthetic similarity maximization as in ESRM[135], and adaptive layer freezing as in ALF[136]. The other approaches tune the class prototype

along with the backbone as in ONPRO[51], augmenting and refining the prototype as in DSR[137], or generating multiple features in angular space as in EARL[138]. Another unique approach uses a student-teacher model and knowledge distillation between them, as in MKD[139], CCLDC[140], and MOSE[141].

2.3.2 PTM-based Approach

The recent development of the foundation model gives rise to the PTM-based OCL, e.g., NSCE[142] that transfers PTM weight and fine-tunes it, RandDumb[143] that trains an RBF kernel from PTM generated features, and RanPAC[144] that projects the feature with a randomized matrix. Tackling OCL can be achieved by a customized optimization technique, e.g., POCL[52] that utilizes Pareto optimization, and CAMA[53] that performs confidence-aware moving average. However, these aforementioned methods still save exemplar or features for rehearsal. It is not always applicable in real-world applications since the old data may not be available or accessible. Note that rehearsal-based CL methods [31], [33], [35], [57], [97] work poorly in OCL and even worse than the aforementioned methods.

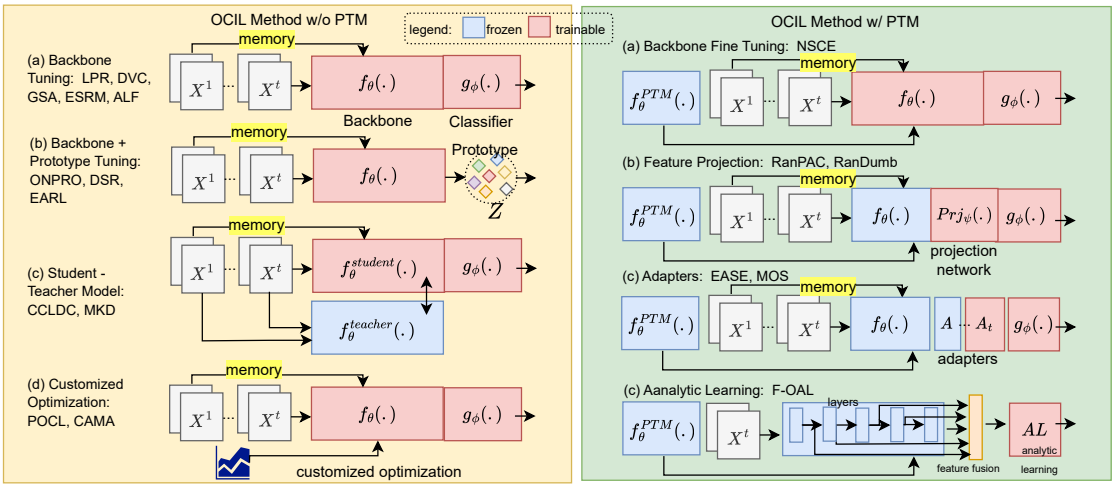


Figure 2.6: OCL approaches and methods.

2.4 Federated Class Incremental Learning

2.4.1 Whole Backbone Tuning Approach

Federated Class Incremental Learning (FCIL) studies address the catastrophic forgetting problem while preserving data privacy, e.g., FedWeIT [46]. The majority of existing FCIL SOTAs, i.e., GLFC [47], and LGA [59] optimize the global model by aggregating the whole locally optimized models. The current SOTAs prove their effectiveness rather than combining FedAvg [145] and class incremental learning (CIL) method, such as ICARL[31] and BiC [35]. However, the SOTAs tune and send the whole backbone, producing long training times and high communication costs as a consequence. Furthermore, the SOTAs assume that a client saves several samples in memory that may not be practically applicable. Another study, i.e., TARGET [146] leverages synthetic samples for rehearsal instead of real samples. It achieves higher performance than FedWeIT [46] but is still outperformed by LGA. A different approach, i.e., FedCIL [147] trains a generative model, i.e., ACGAN [148], on the local client's side to generate fake samples for aggregation on the central server's side. It achieves a higher performance than the combination of FedAvg or FedProx [149] with ACGAN, DGR[150] or LWF-2T [151], but it needs more expensive communication costs and training time due to the generative model.

2.4.2 Parameter-Efficient Fine-Tuning (PEFT) Approach

Similar to CL, FSCL, and OCL, several FCIL methods leverage the generalization property of a pre-trained model and train small-sized additional parameters. FedCPrompt customizes CODA-P for continual learning in distributed systems and adds a contrastive loss in the prompt tuning process [152]. FCILP tunes and shares task-aware and task-agnostic prompts to handle the catastrophic forgetting under non-i.i.d. distribution [153]. PILoRA applies the LoRA concept for the FCIL problem, enhanced by prototype-guided learning to handle the non-i.i.d. distribution [154]. Generally, the PEFT approach achieves higher performance than the whole backbone tuning approach, but it still suffers from catastrophic forgetting. In addition, both PEFT and the earlier approach do not anticipate the data scarcity in FCIL.

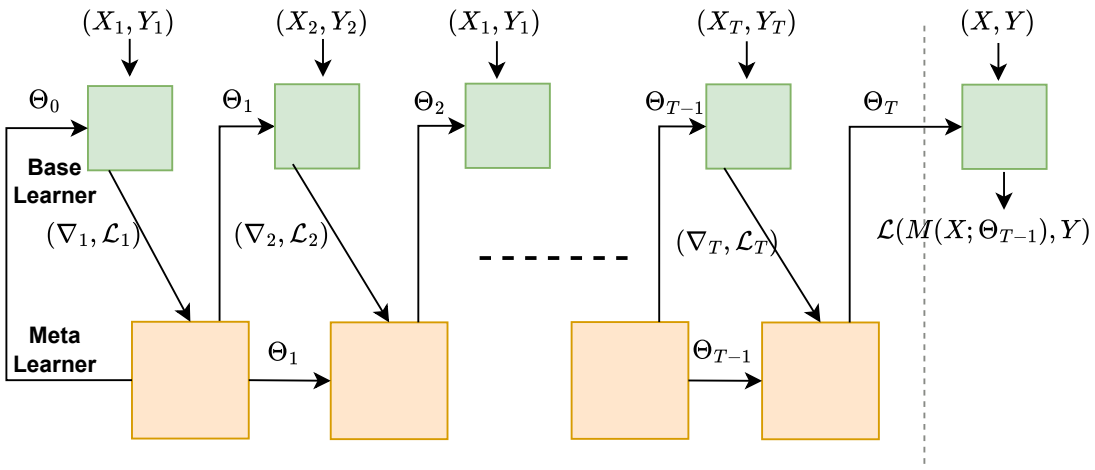


Figure 2.7: Meta-learning illustration.

2.5 Meta Learning

Meta-learning is a learning mechanism that brings the idea of "learning-to-learn" by a meta learner to achieve a better performance of the base model [155]. For instance, normally, we train deep neural models with hyperparameter tuning using grid search or other search methods to achieve the optimal performance of the models. In meta-learning, we utilize another learner model called a meta-learner to guide the deep-learning models (base learners) to achieve their best hyperparameters [155]. Hyperparameter tuning via grid search or try-and-error is time-consuming and does not adaptively lead to the best parameters during training iterations. While meta-learning approaches adaptively lead to the models' best parameters on each iteration and are more efficient than the try-and-error or grid search approaches.

The illustration of meta-learning is presented in Figure 2.7. In meta-learning, the base learner and meta learner are trained concurrently. The base learner is trained using a training set, while the meta learner is trained using the validation set. The validation set may be a processed version of the training set by using a sequence of transformations. Meta-learning is related to continual learning, as meta-learning approaches can be applied in continual learning so that the base model achieves a better performance.

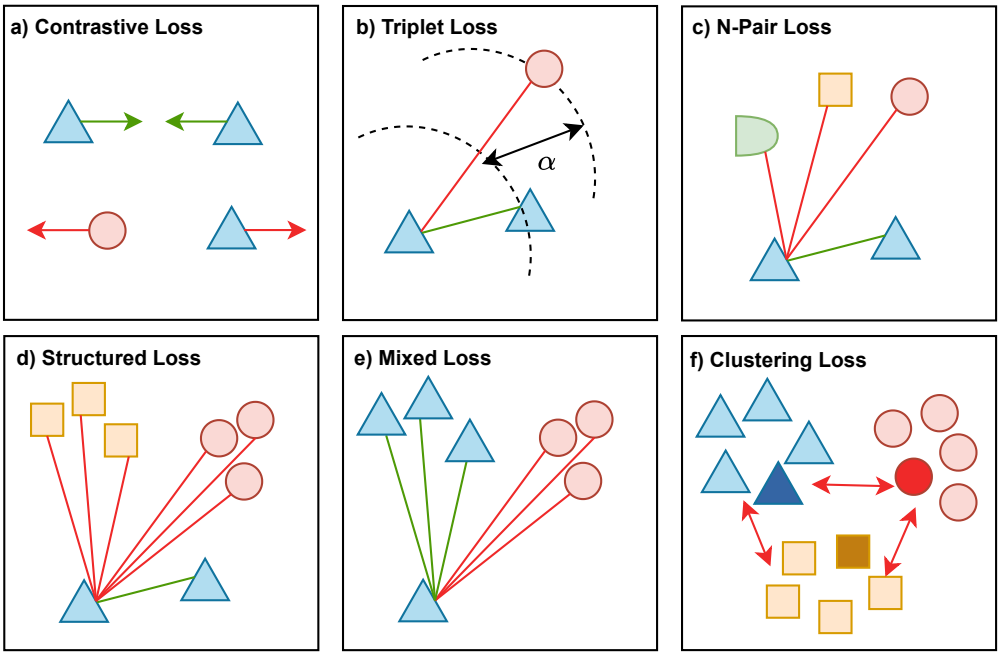


Figure 2.8: Metric Loss Illustration.

2.6 Few-Shot Learning

In general, Few-Shot Learning (FSL) is similar to conventional learning except that the available data is small (few), usually 1-5 samples. To make it a more realistic problem, the data scarcity is applied in the few-shot session, not in the base session. The base session assumes that each class has abundant labeled samples. The main challenge of FSL is the representation bias, where a class representation only represents the acquired samples, not the whole class distribution. In FSL, the loss function for conventional learning, such as cross-entropy loss and binary cross-entropy loss, can not be utilized; otherwise, it leads to over-fitting for the models. Recent development shows that metric losses such as contrastive loss, triplet loss, quadruple loss, and n-pair loss perform effectively in few-shot learning problems [156]. The illustration of metric losses is shown in Figure 2.8. The metric losses have a high possibility of performing effectively for a few-shot continual learning problem where the tasks only have a few samples of data.

2.7 Federated Learning

Federated learning is a learning mechanism that offers the idea of "sharing the model but not the data" due to data privacy issues. In conventional learning, several clients, i.e., institutions, build a machine-learning model for their local goals, with their limited local dataset. This may lead to the underperformance of the models. In the federated learning approach, the clients will be organized by a central server to build a general model that fits all clients' goals. However, due to privacy issues, the clients can not share their data; instead, they share their model parameters with the server. The server is responsible for coordinating the training process of the clients [157].

The mechanism of federated learning is illustrated in Figure 2.9. The mechanism can be modeled as a cycle of three training steps. In the first step, the server initiated the model for all clients. Then, the clients will conduct local training for their models. In the next step, the clients will send their models' weights to the server. After receiving all clients' weights, the server then aggregates the models' weights. Then the server sends weight updates to all clients. After receiving the weights updates, the clients conduct local training again. This cycle is repeated until the model converges or the number of predefined iterations is achieved.

The most difficult challenge in federated learning is that clients may have non-independent and identically distributed (non-i.i.d.) data. It will lead to poor performance of the global model compared to when the data is i.i.d., as the clients' models fit different distributions that produce different optimal parameters. It can be handled by deploying a personalization layer to customize the global model to local goals. However, it is still an open problem to handle the non-i.i.d. data in federated learning. Federated learning is related to continual learning, as there may be several clients that conduct continual learning and are coordinated by a server to exchange their models' parameters. The federated learning approaches may enhance the models' performance on the clients that conduct continual learning [46] [47].

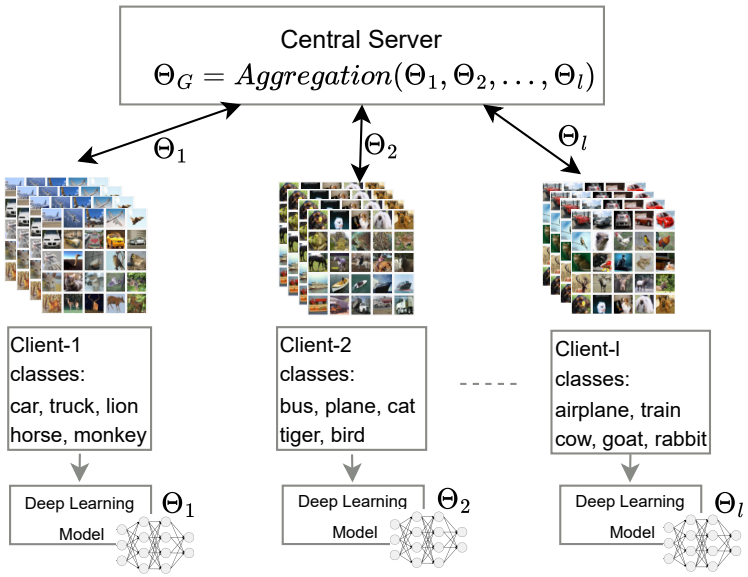


Figure 2.9: Federated Learning Illustration

2.8 Data Augmentation

Data augmentation is a method to generate artificial data from real data to support a limited training set. Data augmentation can be conducted in raw image space or feature space. Image augmentation is more commonly used than feature space augmentation. There are three main categories of image augmentation, i.e., basic image augmentation, and advanced image augmentation [158]. Basic image augmentation performs basic image operations such as geometric transformations, color space transformations, kernel filters, and random erasing. The advanced methods utilize trainable deep neural networks to generate fake samples, such as diffusion models, adversarial training, neural style transfer, and reinforcement learning. The meta-learning approach combines both basic image manipulation and deep learning approaches. The taxonomy of image augmentation is described in Figure 2.10. In addition, augmentation can be conducted in feature space (FS) aside from in data (image) space. Related to continual learning, data augmentation plays an important role in handling the class imbalance problem between current tasks and previous tasks, especially where low or no exemplar memory from previous tasks is available.

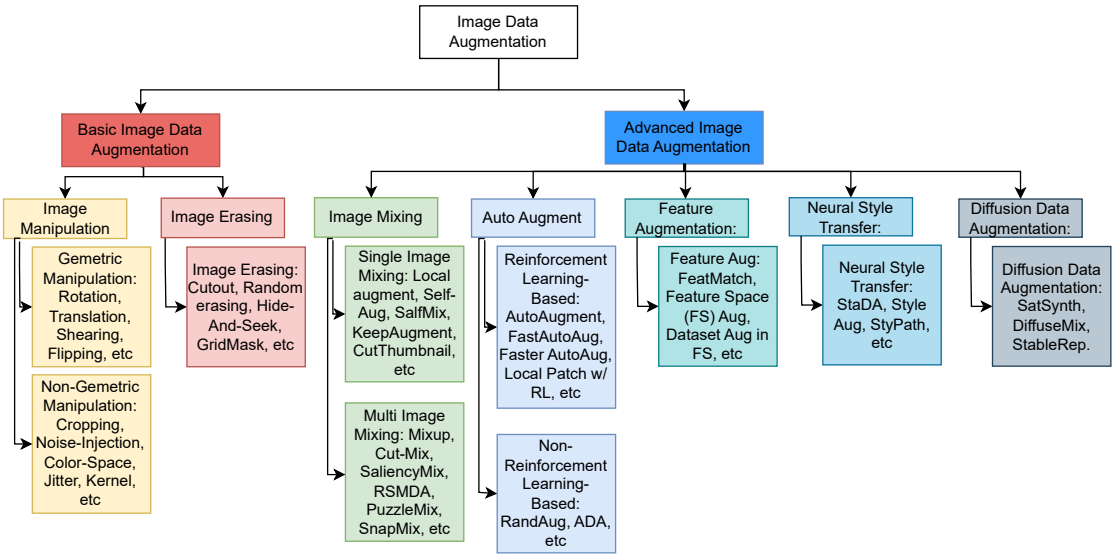


Figure 2.10: Image Augmentation Taxonomy.

2.9 Research Gap

Continual learning and its subproblems attract a huge amount of attention both in theoretical and applied research. However, there are bottlenecks in the existing methods, as summarized in the following sequence:

- Most of the rehearsal(memory)-based methods utilize the constant loss coefficients that lead to sup-optimality in achieving stability-plasticity balance [31], [35], [57]. In addition, the gap between memory size and the size of the currently learned task causes the methods to suffer from a class-imbalance problem. Without proper handling, the imbalance problem leads to a higher stability loss.

- The existing prompt-based methods indeed perform well on rehearsal-free continual learning [37], [38], [39], [85], [117]. However, the methods still suffer from the following drawbacks: (1) Task-wise prompt methods have similar task-wise keys that leads to inaccurate task prediction during inference time and (2) growing prompt component methods suffers a dilemma where prompt components/generators that are already optimal for previous tasks not optimal for currently learned task. As an implication, the model

faces a higher difficulty in achieving plasticity.

- The previous FSCL methods heavily rely on the base session training, where abundant samples are available, e.g, tuning the backbone only in the base task and freezing most of its parameters afterward [42], [56]. The few-shot tasks training is minimized to avoid overfitting on the novel classes, where each class carries only a few samples. This mechanism makes the model barely achieve plasticity and the balance between base and novel classes. In addition, the highest portion of measured performance comes from the base task that contains $\geq 50\%$ of the total classes. It could subtly hide the actual plasticity capability of the methods that is worth investigating.

- Aligned to the previous point, FSCL methods assume abundant labeled samples are available in the base task [40], [42], [45], [121]. It may not be applicable in real-world applications where the annotation is expensive. It is worth pushing forward FSCL into a more challenging yet realistic sub-problem where the base task carries far smaller labels. In addition, by the time this thesis was written, there was no previous study that addressed FSCL in unlabeled samples.

- The previous studies on FCIL train and share the whole backbone models, which implies a longer local training and higher communication cost [46], [47], [48], [63], [147]. In addition, sending perturbation images to the central server to assist models aggregation could breach the privacy preserving since it could reveal part of the local information [47], [48]. On the other hand, sending synthetic data [63] or a generative model [147] for the same purpose still incurs additional communication costs.

- The existing FCIL studies assume that each class in each client has abundant labeled samples [46], [47], [48], [63], [147]. This could be less realistic since data scarcity is a common occurrence in many real-world applications, as discussed in the FSCL problem. This bottleneck motivates us to push forward the FCIL problem into a more challenging yet more realistic problem named federated few-shot class incremental learning (FSCIL). To our knowledge, the problem is new at the time of this thesis writing. In addition, the

previous FCIL methods have not yet addressed the data scarcity in FCIL, given the assumption of abundant labeled samples for each class.

- The majority of both PTM-based and non-PTM-based OCL methods save exemplars from previously learned classes to be replayed along with currently streamed data [53], [132], [135], [142], [143], [144]. The exemplar saving and replay may breach data privacy policy, e.g., when the data is permissible only for one-time access. On the other hand, the prompt-based CL methods work on rehearsal-free CL, but need to repeat the training on the currently learned task for several epochs [37], [38], [39], [85], [117]. Similarly, the methods may breach the one-time access policy, as they need to temporarily store the current task data for the repetitions. Thus, it is worth highlighting the rehearsal-free constraint in OCL and developing an effective and efficient method for the OCL problem.

Chapter 3

AGLA: Assessor Guided Learning for Rehearsal-Based Continual Learning

Rehearsal-based CL method leverages stored exemplar memory to maintain its knowledge on previously learned tasks while gaining new knowledge on the currently learned task. The existing methods utilize a constant loss weighting mechanism for both currently and previously learned tasks, leading to less optimal stability-plasticity. This chapter proposes an assessor-guided learning (AGLA) strategy for rehearsal-based continual learning where an assessor guides the learning process of a base learner by controlling the direction and pace of the learning process, thus allowing an efficient learning of new environments while protecting against the catastrophic interference problem. The assessor is trained in a meta-learning manner with a meta-objective to boost the learning process of the base learner. It performs a soft-weighting mechanism on every sample, accepting positive samples while rejecting negative samples. The training objective of a base learner is to minimize a meta-weighted combination of the cross-entropy loss function, the dark experience replay (DER) loss function, and the knowledge distillation loss function, whose interactions are controlled in such a way as to attain an improved performance. A compensated over-sampling (COS) strategy is developed to overcome the class imbalanced problem of the episodic memory due to limited memory budgets. AGLA has been evaluated in the class-incremental

and task-incremental learning problems. AGLA achieves improved performance compared to its competitors, while the theoretical analysis of the COS strategy is offered.

Part of this chapter is published as: MA Ma'sum, M Pratama, E Lughofer, W Ding, W Jatmiko. Assessor-guided learning for continual environments. Information Sciences 640, 119088 https://doi.org/10.1016/j.ins.2023.119088 [**CORE A**]

3.1 Introduction

Continual learning problem aims to build a learning model throughout the life span of the model in use and gain its improved intelligence as the number of learning tasks increases. Unlike conventional learning algorithms limited to only a single task, a continual learner is exposed to streaming tasks where each task possesses varying characteristics, i.e., different data distributions, different target classes, or combinations of distributional changes and class changes. A continual learner has to adapt quickly to new environments without losing relevance to old tasks. This problem is not trivial for a deep neural network because of the catastrophic interference problem, where old parameters are overwritten when learning a new task, thereby losing its generalization power to the old tasks. Because of uncertain and possibly infinite problem sizes, a retraining process from scratch is undesired. In other words, the learning process occurs without complete access to old data samples.

The continual learning problem goes one step toward human-like intelligence, where the continual learner must be capable of accumulating knowledge from already seen experiences. As a result, this area has picked up substantial research attention. Existing works are categorized into three groups: regularization-based approach [27], structure-based approach [70], and memory-based approach [77]. The regularization-based approach introduces an extra regularization term, preventing important parameters of old tasks from deviating. This approach is simple to implement and computationally light. These approaches, however, do not scale well for large-scale problems because an overlapping region of all tasks is difficult to find with the regularization-based approach. The structure-based

approach increases network capacity to deal with new tasks while isolating old network parameters to avoid the catastrophic forgetting problem. This approach is, however, computationally expensive and usually involves complex learning procedures. The memory-based approach takes another route where a small subset of old data samples is stored in the memory and interleaved with current samples when learning a new task. *Conventional experience replay mechanism requires hundreds of samples to be stored in the memory, thus incurring expensive memory footprints. There also exists the class imbalanced problem because old samples are often lower in quantity than new samples. The assessor-guided learning approach is put forward here to address these drawbacks, where an assessor controls the learning process of the base learner via a soft-weighting mechanism of loss functions for every sample. Memory augmentation is applied in our method to address the class-imbalance problem between the current task and the previous tasks. The compensated over-sampling mechanism is integrated, where self-corrections while over-sampling are performed to avoid the out-of-distribution cases.*

Assessor-Guided Learning Approach (AGLA) is proposed here, where a sequence-aware assessor is integrated to navigate the learning process of a base model to attain an improved learning performance. The assessor is trained with a meta-objective to boost the generalization power of the base model via a soft-weighting mechanism of loss functions for every sample. A high-quality sample is assigned a high weight, whereas a low weight is assigned to poor samples. High-quality samples are those leading to positive forward and backward transfers, whereas poor samples are those imposing high losses associated with catastrophic forgetting. Two data subsets, the training subset and the validation subset, are created for each task to simulate the training-testing procedure [159], where both subsets represent current and old concepts. The concept of random transformation [160] is integrated to craft the validation subset. The training procedure follows the meta-learning principle, where the training subset is used to train the base model in the outer loop, while the assessor utilizes the validation subset for its updates in the inner loop. The assessor produces a set of weights, cross-entropy weight, dark experience replay (DER) weight and distillation weight controlling the interaction of loss functions and, in turn, the influence of every sample. This is made

possible by formulating the loss function as a meta-weighted combination of the cross entropy loss function, the DER loss function [57], and the knowledge distillation loss function [31]. The cross-entropy loss focuses on current samples and past samples of the memory, while the DER loss and the knowledge distillation loss target past samples of the memory to maintain previous knowledge. In other words, the assessor steers a base model to address the stability-plasticity dilemma. It determines how much a base model should learn from the current condition and the past condition with respect to every sample. Note that both DER and knowledge distillation target previous experiences because the number of previous tasks is larger than the current task, but underrepresented in the training process, i.e., memory samples are much smaller than current samples. This aspect confirms the importance of the meta-weighting mechanism regulating the influence of multi-objective functions seamlessly.

The class imbalanced problem due to disproportionate proportions of memory samples and current samples is tackled using the compensated over-sampling (COS) strategy, where compensations are performed while over-sampling via well-known data augmentation protocols to protect against out-of-distribution augmented samples undermining the model's generalization. A lemma analyzing such compensation with respect to the bias-variance decomposition is provided, as well as a theorem demonstrating reductions of MSEs as a result of the compensations.

This chapter conveys four major contributions:

1. The proposed method puts forward the concept of assessor-guided continual learning where the sequence-aware assessor is deployed to guide the learning process of the base learner, balancing the issue of stability and plasticity;
2. The compensated over-sampling (COS) strategy is proposed to deal with the class imbalanced problem underpinned by theoretical analyses;
3. A meta-training strategy via a bi-level optimization is put forward to train the assessor, where the concept of random transformation is implemented to craft the validation subset;
4. A meta-weighted combination of three loss functions is proposed to train

a base learner, where each loss function is associated with either current or past conditions. This design offers an intuition of learning strategies for every sample, i.e, whether to focus on the current or previous contexts.

The advantage of AGLA over existing approaches has been numerically validated under the class-incremental and task-incremental learning configurations. It is demonstrated that AGLA delivers improvements compared to recently published algorithms in the realm of average accuracy, while each learning component contributes positively to the performance. AGLA achieves comparable average forgetting indexes, where it mostly attains the second place and maintains decent performances with various memory sizes compared to baseline algorithms.

3.2 Problem Formulation

Supervised continual learning problems are considered here, where a model is trained on a sequence of fully labelled tasks $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K$. Each task $\mathcal{T}_k = \{x_n, y_n\}_{n=1}^{N_k}$, $k \in \{1, ..., K\}$ is sampled from the i.i.d distribution $\mathcal{D}_k$ where $x_n \in \mathcal{X}$ stands for an input sample and $y_n \in \mathcal{Y}$ denotes its true class label. K, N_k respectively denote the number of tasks, which might be infinite, and the size of the $k-th$ task. Each task does not possess the same characteristic, causing non-stationary environments to a continual learning model $g_\phi(f_\theta(.))$ with feature extractor's parameters θ and classifier's parameters ϕ. There exist three continual learning variants in the literature: domain-incremental, task-incremental, and class-incremental [161]. The domain-incremental problem refers to different data distributions of each task $P(X,Y)_k \neq P(X,Y)_{k+1}, k \in \{1, ..., K\}$ while having the same problem structure, i.e., input and target variables. The task-incremental and class-incremental problems feature different class labels for each task. Suppose that $L_k, L_{k'}, k, k' \in \{1, .., K\}$ stand for label sets of the $k-th$ task and the $k'-th$ task, $\forall k, k', L_k \cap L_{k'} = \emptyset$. The difference between the task-incremental and class-incremental problems lies in **the absence of the task IDs** for the class-incremental problem. A prediction relies on a single classifier $g_\phi(.)$ rather than one classifier per task $g_{\phi_k}(.)$. A task $\mathcal{T}_k$ is accessed at the $k-th$ session and discarded once completed. This issue leads to the catastrophic interference problem where

learning a new task $\mathcal{T}_k$ overwrites previously valid parameters.

3.3 AGLA: Assessor Guided Learning

AGLA is developed from the assessor-guided learning method where an assessor $\kappa_\psi(.)$ is deployed to guide the learning process of the base learner $g_\phi(f_\theta(.))$. The assessor produces a set of weights, the cross-entropy weight, the DER weight, and the distillation weight, for every sample. The soft-weighting mechanism not only controls the pace and direction of the learning process based on the quality of a data sample, but also governs the interaction of each loss function, making it possible for an adaptive selection of a suitable learning strategy for every sample, i.e., which losses to be favoured. The compensated over-sampling (COS) strategy is integrated to cope with the class-imbalanced issue on continual learning, where corrections are performed while over-sampling to prevent the adverse impacts of out-of-distribution samples.

3.3.1 Loss Function

The cost function is formulated as a meta-weighted combination of the cross entropy loss function, the DER loss function [57], and the knowledge distillation loss function [31]. The three loss functions have been rigorously validated, where the absence of one term results in significant performance drops, as shown in our ablation study.

$$\mathcal{L}_{bl} = \underbrace{\mathbb{E}_{(x,y)\backsim\mathcal{D}_k\cup\hat{\mathcal{M}}_{k-1}}[\alpha l(s_W(o),y)]}_{L_{CE}} + \underbrace{\mathbb{E}_{(x,y)\backsim\hat{\mathcal{M}}_{k-1}}\lambda[||o-h|| + l(s_W(o),y)]}_{L_{DER++}} + \underbrace{\mathbb{E}_{(x,y)\backsim\hat{\mathcal{M}}_{k-1}}[\pi l(o,h)]}_{L_{distill}} \tag{3.1}$$

$$\lambda = \begin{cases} 0, & k=1 \\ k*\beta, & k\geq 1 \end{cases}, \qquad \pi = \begin{cases} 0, & k=1 \\ k*\gamma, & k\geq 1 \end{cases}$$

where α, λ, π are respectively the cross entropy weight, the DER weight, and the distillation weight generated by the sequence-aware assessor $\{\alpha, \beta, \gamma\} = \kappa_\psi(x) \in [0, 1]$. $o = g_\phi(f_\theta(.))$ is a presoftmax response known as output logit. In addition to determining the sample's influence in the learning process, these three weights determine the learning strategies for a sample of interest, i.e., it is capable of selecting a suitable loss function for every sample. Note that the soft-weighting mechanism is applied and provides better flexibility than the binary hard-sampling mechanism because the learning process of the base learner is governed in a smooth manner. $h = g_\phi(f_\theta(x))_{k-1}$ and $||.||_2$ is the L-2 distance function. Note that the L-2 distance function functions similarly to the KL divergence. No softmax function is applied to avoid the effect of the squashing function, and the real labels are included here L_{DER++}, to prevent the distribution shift [57]. Both the DER loss L_{DER++} and the distillation loss $L_{distill}$ aim to maintain the stability of previously learned knowledge, while the cross entropy loss L_{CE} aims to enhance the plasticity to new knowledge. The three losses are meta-weighted by the assessor to achieve a proper tradeoff between plasticity and stability. Two loss functions, focusing on old tasks, the DER loss function and the distillation loss function, are integrated here because the number of old tasks is usually larger than the current task, but under-represented in the learning process due to a small memory size. In addition, L_{CE} involves both current and old samples to avoid biased responses toward new classes and has been a common design choice in the literature [35], [162]. The use of meta-weights also avoids a tweaking problem in continual learning algorithms using a large weight > 100 to the losses corresponding to the past states.

The base learner $g_\phi(f_\theta(.))$ is formulated as the MLP or CNN where the feature extractor $f_\theta(.)$ is created as stacked linear layers or convolutional layers while the classifier $g_\phi(.)$ is formed as a fully connected layer. One classifier per task $g_{\phi_k}(.)$ is used for the task-incremental learning problem, while a single classifier $g_\phi(.)$ is applied for the class-incremental learning problems without any task IDs. The assessor $\kappa_\psi(.)$ is formed as an LSTM followed by a fully connected layer with the sigmoid activation function at the last layer. The convolutional layers are integrated as the feature extractor in the assessor. The use of LSTM aims to provide short-term memory, hidden state, and long-term memory, cell state,

making it possible for a sequence of past weights to be preserved in the memory. $l(.)$ is a cross-entropy loss function and $s_W(.)$ is a softmax layer parameterized by W.

3.3.2 Meta-Training Strategy

The assessor is trained with a meta-objective to boost the learning process of the base learner, i.e., the influence of a current sample and an old sample, as well as the interaction of the three loss functions, is controlled to achieve a tradeoff between the issue of plasticity and stability. That is, the assessor $\{\alpha, \beta, \gamma\} = \kappa_\psi(x) \in [0, 1]$ controls the speed, direction, and learning strategies of the base learner. Suppose the current $k-th$ task, two data partitions, training set and validation set, are created where the training set is constructed from current samples and memory samples $\mathcal{T}^k_{train} = \mathcal{T}^k_{train} \cup \hat{\mathcal{M}}_{k-1}$ whereas the validation set is created by applying random transformation to the training set $\mathcal{T}^k_{val} = \{T(x_i), y_i\}_{i=1}^{N^k_{train}} \backsim \mathcal{T}^k_{train}$. $T(.)$ is the random transformation operator, such as color/geometric transformation or noise injection [160], and N^k_{train} is the size of the training set. We follow the same way as [160] where $T(.)$ is taken from one of the possible transformation sets Φ, i.e., three transformations, image invert, Gaussian noise perturbation, and RGB-rand perturbation are applied here to the original training samples. $\hat{\mathcal{M}}_{k-1}$ denotes an augmented memory set including those of the data augmentation procedure. The training set $\mathcal{T}^k_{train}$ is exploited to update the base learner, whereas the validation set $\mathcal{T}^k_{val}$ is used to train the assessor.

The training process of the base learner and the assessor is formulated as a bi-level optimization problem [163] where the base learner is updated using the training set while the assessor is trained using the validation set. That is, the bi-level optimization problem is formulated as follows:

$$\begin{aligned} \min_{\psi} \mathbb{E}_{(x,y)\backsim\mathcal{T}^k_{val}}[\mathcal{L}_{bl}(g_{\phi^*}(f_{\theta^*}(x)), y)] \\ s.t, \phi^*, \theta^* = \arg\min_{\phi,\theta} \mathbb{E}_{(x,y)\backsim\mathcal{T}^k_{train}}[\mathcal{L}_{bl}(g_{\phi}(f_{\theta}(x)), y)] \end{aligned} \tag{3.2}$$

where $\{\phi^*, \theta^*\}$ stand for optimal base parameters with respect to the current assessor ψ. The meta-learning strategy is implemented here because of the absence of

ground truth of the assessor, i.e., the ideal weights. This case implies the optimal parameters of the assessor ψ^* minimizing the validation loss of the base network. The bi-level optimization approach is solved by first updating the assessor. That is, the base learner is evaluated with the validation set $\mathcal{T}_{val}^k$, returning the validation loss. The validation loss is used to update the assessor. In other words, the learning process of the assessor aims to minimize a meta-objective as follows:

$$\psi^* = \arg\min_{\psi} \sum_{(x,y)\in\mathcal{T}_{val}^k} \mathcal{L}_{bl}(g_\phi(f_\theta(x)), y) \tag{3.3}$$

The validation loss is differentiable with respect to the assessor because the validation loss is a factor of the three weights α, β, γ produced by the assessor. Since the optimal base model depends on the assessor, the assessor has to be updated first:

$$\psi' = \psi - \eta \sum_{(x,y)\in\mathcal{T}_{val}^k} \nabla_\psi \mathcal{L}_{bl}(g_\phi(f_\theta(x)), y) \tag{3.4}$$

where η is a learning rate. Once the assessor updates, the base learner is adjusted using the meta-weights generated by the updated assessor:

$$\{\phi, \theta\} = \{\phi, \theta\} - \mu \sum_{(x,y)\in\mathcal{T}_{train}^k} \nabla_{\{\phi,\theta\}} [\mathcal{L}_{bl}(g_\phi(f_\theta(x)), y)] \tag{3.5}$$

where μ is a learning rate. This strategy implies a joint update of the base learner and the assessor, where the adjustment of the base learner involves the tuning step of the assessor. It guarantees an improved performance of the base learner because it is undertaken using the cross entropy weight α, the DER weight β, and the distillation weight γ produced by an updated assessor $\{\alpha, \beta, \gamma\} = \kappa_{\psi'}(x)$. This strategy also overcomes the tweaking issue of the memory-based approach in continual learning, where memory samples are assigned with large weights, e.g., > 100, confirming over-dependence on the experience replay strategy while ignoring new tasks.

3.3.3 Compensated Over-Sampling Approach

The main challenge of the memory-based approach is to resolve the class imbalanced problem because only a small fraction of old samples are stored. The class

imbalanced problem results in a biased problem where a model favours new classes because old samples of the memory are under-represented in the data batches. [35] shows that the last fully connected layer is heavily biased because it is not shared during the training process. We apply the over-sampling approach to correct the bias problem. That is, a memory set $\mathcal{M}_{k-1}$ is augmented using the random transformation method, thus generating an augmented memory set $\hat{\mathcal{M}}_{k-1}$. The augmented memory set rectifies disproportionate class proportions between new classes and old classes.

A naive oversampling approach might induce out-of-distribution samples, undermining the model's generalization power [164]. This chapter proposes a compensated over-sampling (COS) strategy for continual learning where the learning process of augmented samples is compensated to offset out-of-distribution samples. Ideally, the over-sampling approach via a family of the transformation functions $T(.)$ should sample from the same distribution p^* and semantics of the original samples x_i. Because such a distribution p^* is unknown in practice, we rely on a good assumption of heuristically chosen $T(.) \backsim \tilde{p}$:

$$\mathcal{L}^* = \sum_{i=1}^{|\mathcal{M}_{k-1}|} \sum_{k=1}^{M} \mathcal{L}_{bl}(T(x_i)_k, y_i; \theta, \phi) \tilde{p}_k \frac{p_k^*}{\tilde{p}_k} \tag{3.6}$$

The ratio $p_k^*/\tilde{p}_k$ is hard to access in practice. Alternatively, we define $w_k = \frac{p_k^*}{\tilde{p}_k}$ following the Radon Nikodym derivative [164] and analyze how it affects the MSE of the deep model. The following lemma describes the MSE of the deep estimator.

Lemma 1[164]: Define $\theta_0 = \arg\min_\theta \mathbb{E}_x \mathcal{L}_\theta(x), \theta_G = \arg\min_\theta \mathbb{E}_x \int \mathcal{L}_\theta(x, n) dp^*$, and $\hat{\theta}_G = \arg\min_\theta \mathcal{L}^*$. Let V_0 be the Hessian of $\theta \to \mathbb{E}_x \mathbb{L}_\theta(x)$ at θ_0, and V_G the Hessian of $\theta \to \mathbb{E}_x[\int \mathcal{L}_\theta(x, n) d\tilde{p}(n) w(n)]$ at θ_G. Let $M_0(x) = \nabla \mathcal{L}_{\theta_0}(x) \nabla \mathcal{L}_{\theta_0}(x)^T$ and $M_G(x) = \nabla \mathcal{L}_{\theta_G}(x) \nabla \mathcal{L}_{\theta_G}(x)^T$ where $\nabla \mathcal{L}_{\theta_0}(x)$ and $\nabla \mathcal{L}_{\theta_G}(x)$ correspond to the gradients of $\mathcal{L}_\theta(x)$ at $\theta = \theta_0$ and $\theta = \theta_G$ respectively. Suppose $M_G(x, n) = \nabla \mathcal{L}_{\theta_G}(x, n) \nabla \mathcal{L}_{\theta_G}(x, n)^T$. $tr(X)$ stands for the trace of matrix X. Hence, with C a constant invariant of w, under mild conditions, we obtain:

$$MSE(\hat{\theta}_G) \backsim C + ||\theta_G - \theta_0||_2^2 + \frac{1}{N}\mathbb{E}_x[\int tr(V_G^{-1}(M_G(x,n) - M_G(x))V_G^{-1})d\tilde{p}(n)w(n)] \tag{3.7}$$

$$+\frac{1}{N}\mathbb{E}_x[tr(V_G^{-1}(M_G(x) - M_0(x))V_G^{-1})]$$

$$+\frac{1}{N}tr((V_G^{-1} - V_0^{-1})Cov_x\nabla\mathcal{L}_{\theta_0}(x)(V_G^{-1} - V_0^{-1})) \tag{3.8}$$

$$-\frac{1}{N}tr(V_G^{-1}\mathbb{E}_x[Cov_w\nabla\mathcal{L}_{\theta_G}(x,n)]V_G^{-1}) \tag{3.9}$$

where $Cov_w(\nabla\mathcal{L}_{\theta_G}(x,n))$ is the covariance matrix of $\mathcal{L}_\theta(x,n)$ at θ_G under measure $p^*(n)$.

This lemma tells that large data variance via sampling from p^* is desired to decrease the model's variance. Another case also explains the increase of MSEs or biases if there exist significant differences between the gradient of augmented samples and those of original samples. The key observation is the presence of $w(n)$, which can be used to control the bias-variance trade-off. In other words, it is capable of compensating for possible out-of-distribution augmented samples. We apply the same principle as [164] where w_k is determined as the likelihood ratio in sampling $T(x_i)_k$ to reduce the MSE in Lemma 1. Suppose that $z_{i,k} = f_\theta(T(x_i)_k)$, $w_{i,k}$ is arranged

$$\frac{p^*}{\tilde{p}} \propto w_{i,k} = \exp -(z_{i,k} - \mu_i)(\tau\Sigma)^{-1}(z_{i,k} - \mu_i) \tag{3.10}$$

$$\mu_i = \frac{1}{M}\sum_{k=1}^{M} z_{i,k}; \Sigma = \frac{1}{NM}\sum_{i}^{N}\sum_{k}^{M}(z_{i,k} - \mu_i)(z_{i,k} - \mu_i)^T \tag{3.11}$$

where $k \in \{1, ..., M\}$, M denotes the number of random transformations applied an image x_i and τ is a temperature that regulates the influence of the distance measure. $w_{i,k}$ is further normalized across a data batch leading to $\overline{x}_i$:

$$\frac{p^*}{\tilde{p}} \propto \overline{w}_{i,k} = \frac{w_{i,k}}{\sum_{i=1}^{N}\sum_{k=1}^{M} w_{i,k}} \tag{3.12}$$

The normalized ratio is applied to govern the learning process of augmented samples:

$$\mathcal{L}_{bl} = \sum_{i=1}^{N}\sum_{k=1}^{M} \overline{w}_{i,k}\mathcal{L}_{bl} \tag{3.13}$$

At first, a forward propagation is performed to produce μ_i, Σ leading to $\overline{w}_{i,k}$. $\overline{w}_{i,k}$ is frozen afterward and inserted during the learning process of $\mathcal{L}_{bl}$. A theorem can be derived.

3.3.4 Algorithm

Algorithm 1 exhibits the pseudo-code of AGLA.

Algorithm 1 Learning Policy of AGLA

Input: continual dataset $\mathcal{D}$, learning rates μ, η, size of pseudo memory m, iteration number E
Output: parameters of base learner $\{\phi, \theta\}$, parameters of the assessor ψ
for $k = 1$ to K **do**
  $\hat{\mathcal{M}}_{k-1} = Augment(\mathcal{M}_{k-1})$ /* Augment the memory to address the class imbalanced problem/*
  $\mathcal{T}^k_{train} = \hat{\mathcal{M}}_{k-1} \cup \mathcal{T}_k$ /*Construct the training set/*
  $\mathcal{T}^k_{val} = \{T(x_i), y_i\}_{i=1}^{N^k_{train}} \backsim \mathcal{T}^k_{train}$ /*Apply random transformation to construct the validation set/*
  for $e = 1$ to E **do**
    Update assessor parameters ψ using (3.4)
    Update base learner parameters $\{\phi, \theta\}$ using (3.5)
  end for
  $B_k = Sample(\mathcal{T}_k)$ /* Perform reservoir sampling on the current task /*
  $\mathcal{M}_k = \mathcal{M}_{k-1} \cup B_k$ /*Update the current memory/*
end for

3.4 Theoretical Analysis

Proof of Lemma 1 [164] Because $\theta_G = \arg\min_\theta \mathbb{E}_x[\int \mathcal{L}_\theta(x, n)dp^*(n)]$ and

$$\hat{\theta}_G = \arg\min_\theta \mathcal{L}^* = \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{M} \mathcal{L}_\theta(x_i, n_k)w_{i,k} \tag{3.14}$$

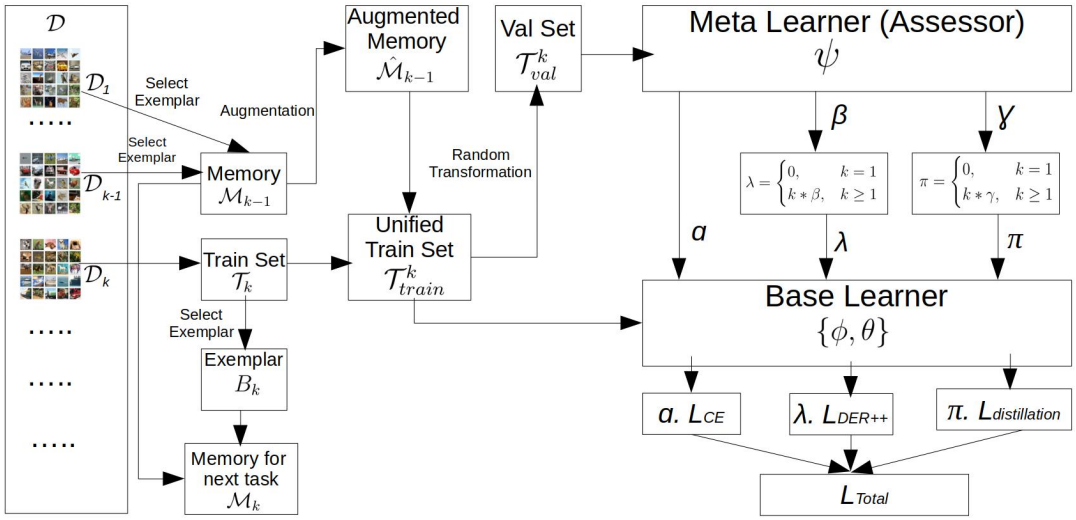


Figure 3.1: The flow-chart of the proposed method (AGLA).

Based on Theorem 5.23 of [165], we obtain

$$\sqrt{N}(\hat{\theta}_G - \theta_G) = -V_G^{-1} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{M} \nabla \mathcal{L}_{\theta_G}(x_i, n_k) w_{i,k} + o_p \tag{3.15}$$

where $o_p(1)$ can be safely ignored because of $o_p(1) \to 0$ as $N \to \infty$. This leads to the following.

$$MSE(\hat{\theta}_G) = ||\theta_G - \theta_0||^2 + \frac{1}{N} tr(V_G^{-1} Cov_x[\nabla \int \mathcal{L}_{\theta_G}(x, n) w(n) d\tilde{p}(n)] V_G^{-1}) \tag{3.16}$$

where ∇ denotes the gradient w.r.t θ and V_G denotes the Hessian matrix $Hess(\mathbb{E}_x[\int \mathcal{L}_\theta(x, n) w(n) d\tilde{p}(n)])$ at θ_G. The term $Cov_x[\nabla \int \mathcal{L}_{\theta_G}(x, n) w(n) d\tilde{p}(n)]$ can be expressed similarly as per Lemma 1 of [166]:

$$Cov_x[\nabla \int \mathcal{L}_{\theta_G}(x, n) w(n) d\tilde{p}(n)] - \mathbb{E}_x \mathbb{E}_{p^*} M_G(x, n) \tag{3.17}$$

$$= \mathbb{E}_x(\nabla \mathbb{E}_p^*(\mathcal{L}_{\theta_G}(x, n)) . \nabla \mathbb{E}_p^*(\mathcal{L}_{\theta_G}(x, n))^T) - \mathbb{E}_x(\nabla \mathbb{E}_p^*(\mathcal{L}_{\theta_G}(x, n))) . \mathbb{E}_x(\nabla \mathbb{E}_p^*(\mathcal{L}_{\theta_G})) - \mathbb{E}_x(\mathbb{E}_{p^*}(\nabla \mathcal{L}_{\theta_G}(x, n) . \nabla \mathcal{L}_{\theta_G}(x, n)^T) \tag{3.18}$$

$$= -\mathbb{E}_x(Cov_w \nabla \mathcal{L}_{\theta_G}(x, n)) \tag{3.19}$$

Note that $\mathbb{E}_x(\nabla\mathbb{E}_{p^*}(\mathcal{L}_{\theta_G}(x,n))) = 0$ [164]. The rests can be proved by decomposing $\mathbb{E}_x\mathbb{E}_{p^*}M_G(x,n)$ further following the proof of Lemma 1 in [167].

Theorem 1 [164]: given mild conditions, the learning process of augmented samples via (3.13) where $\overline{w}_{i,k}$ is determined via (3.12), (3.10) leads to reductions of MSEs as per Lemma 1.

Proof of Theorem 1: Based on Lemma 1, the mean squared error of $\hat{\theta}_G$ can be further simplified as

$$MSE(\hat{\theta}_G) \backsim C + ||\theta_G - \theta_0||^2 + \frac{1}{N}tr(V_G^{-1}.Cov_x[\nabla\mathbb{E}_{p^*}(\mathcal{L}_{\theta_G}(x,n))].V_{G^{-1}}) \quad (3.20)$$

where C is a constant invariant of the choice of $\{w_{i,j}\}$. Without loss of generality, the model θ is assumed to be orthogonalized, leading to further derivation as follows:

$$||\theta_G - \theta_0||^2 + \frac{1}{N}\frac{1}{V_G^2}\mathbb{E}_x[(E_{p^*}(\mathcal{L}^{'}{}_{\theta_G}(x,n)))^2] \quad (3.21)$$

where $\mathcal{L}^{'}_{\theta_G}(x,n)$ denotes the diagonal gradient matrix of $\mathcal{L}$ w.r.t θ and V_G^2 is the diagonal Hessian matrix at θ_G.

Assume the loss function $\mathcal{L}$ satisfies a symmetric condition: there exists an interval of (θ_1,θ_2) such that $\theta_0, \theta_G \in (\theta_1,\theta_2)$. Suppose for all $\theta \in (\theta_1.\theta_2), \mathcal{L}_\theta(x_i,n_j) = \mathcal{L}_\theta(x_i,n_k)$ whenever the Mahalanobis distance of the corresponding features $z_{i,j}$ and $z_{i,k}$ are equal: $(z_{i,j}-\mu_i)\Sigma^{-1}(z_{i,j}-\mu_i) = (z_{i,k}-\mu_i)\Sigma^{-1}(z_{i,k}-\mu_i)$. As a result, for $\theta_G \in (\theta_1,\theta_2)$ the second term of (3.20) can be written as

$$\frac{1}{N}\frac{1}{V_G^2}\mathbb{E}_x[\sum_{j=1}^{M}\overline{\mathcal{L}}^{'}_{\theta_G}(d^2(z_{i,j},\mu_i)w_{i,j})^2] \quad (3.22)$$

where $\overline{\mathcal{L}}$ is the function as a result of rewriting $\mathcal{L}$ in terms of the Mahalanobis distance denoted by $d^2(z_{i,j},\mu_i)$. Note $\arg\min_\theta \mathbb{E}_x\sum_{j=1}^{M}\overline{\mathcal{L}}_\theta(d^2(z_{i,j},\mu_i))w_{i,j} = \arg\min_\theta \mathbb{E}_x[\int \mathcal{L}_\theta(x,n)dp^*(n)] = \theta_G$. Suppose the distribution of the Mahalanobis distance of the generated $\{z_{i,j}\}$ is approximately uniform, and $\mathbb{E}_x[\sum_{j=1}^{M}\overline{\mathcal{L}}^{'}_{\theta_G}(d^2(z_{i,j},\mu_i)w_{i,j})^2]$ can be well approximated by $(\int\overline{\mathcal{L}}\overline{w}(x))^2$ where $\overline{w}$ is the representation of w as a function of the Mahalanobis distance. Then for all

Table 3.1: Details of Continual Learning Problems, nTasks: number of tasks, nClasses / Task: number of classes per task, CL Setting: Continual Learning Setting.

Problem	nTasks	nClasses / Task	CL Setting
SMNIST	5	2	Class and Task - IL
SCIFAR10	5	2	Class and Task - IL
SCIFAR100	10	10	Class and Task - IL
SMINIIMAGENET	10	10	Class and Task - IL

loss function $\mathcal{L}$ such that $[\mathbb{E}_1(\overline{\mathcal{L}}_{\theta_G}(x))] < [\mathbb{E}_2(\overline{\mathcal{L}}_{\theta_G}(x))]$ where E_1 and E_2 are expectations under standard exponential and uniform distribution defined on an interval of admissible values of the Mahalanobis distance. The proposed $w_{i,j}$ contributes to the reduction of (3.22) compared to equal weights. That is, $z_{i,j}$ with a larger Mahalanobis distance is down-weighted to reduce its leverage in the estimating region [164].

The squared bias $||\theta_G - \theta_0||^2$ is reduced by selecting measure $p(n)$ to approach the invariance $\mathbb{E}_x \int \mathcal{L}_\theta(x,n)dp(n) = \mathbb{E}_x[\mathbb{L}_\theta(x,Id)]$ on interval (θ_1,θ_2) as $||\theta_G - \theta_0|| \backsim |\mathbb{E}_x \int \mathcal{L}_{\theta_G}(x,n)dp(n) - \mathbb{E}_x[\mathcal{L}_{\theta_0}(x,Id)]|/l$ where l is square-integrable function not depending on θ. Given our assumption that $w(n)p(n) = p^*(n)$ improves invariance comparing to $\tilde{p}(n)$ with $|\mathbb{E}_x \int \mathbb{L}_\theta(x,n)dp^*(n) - \mathbb{E}_x[\mathcal{L}_\theta(x,Id)]| < |\mathbb{E}_x \int \mathcal{L}_\theta(x,n)d\tilde{p}(n) - \mathbb{E}_x[\mathcal{L}_\theta(x,Id)]|$ for $\theta \in (\theta_1,\theta_2)$ using triangular inequality we conclude that imposing the weight adjustments with w can effectively reduce the MSE [164].

3.5 Experiment Results and Analysis

The advantage of AGLA is numerically validated in both class-incremental and task-incremental learning problems. In addition, an ablation study and memory analysis are offered to analyze the contribution of each component and the memory size to the final performance. Our experiments are conducted by five independent runs using different random seeds, where the numerical results are averaged and reported in Table 3.3 and Table 3.4. Two performance metrics, average accuracy and forgetting measure [170], are applied to measure the learning performances. The source codes and other supporting data of AGLA, including raw numerical results,

Table 3.2: Hyperparameters of Consolidated Algorithms.

Methods	**Params**	**Value**
All: AGLA, DER, BIC, ICARL, EEIL, HAL EWC, LWF, MAS, SI, DMC, Finetuning, Joint	batch_size	100
	clipping	10000
	eval_on_train	False
	fix_bn	False
	gridsearch_tasks	-1
	keep_existing_head	False
	last_layer_analysis	False
	log	['disk']
	lr	0.05
	lr_factor	1
	lr_min	0.0001
	lr_patience	5
	momentum	0.9
	multi_softmax	False
	nepochs	100
	network	resnet18
	no_cudnn_deterministic	False
	num_workers	4
	pin_memory	False
	pretrained	False
	use_valid_only	False
	warmup_lr_factor	1
	warmup_nepochs	0
	weight_decay	0.0002
Memory Based: AGLA, DER BIC, ICARL, EEIL, HAL	num_exemplars_per_class	50
	exemplar_selection	random
BIC	T	2
	lamb	-1
	num_bias_epochs	200
	val_exemplar_percentage	0.1
ICARL	lamb	1
EEIL	T	2
	lamb	1
	lr_finetuning_factor	0.01
	nepochs_finetuning	40
	noise_grad	FALSE
HAL	anchor_optimization_steps	100
	lamb	0.1
	beta	0.5
	gamma	0.1
EWC	alpha	0.5
	fi_num_samples	-1
	fi_sampling_type	max_pred
	lamb	5000
LWF	T	2
	lamb	1
MAS	alpha	0.5
	fi_num_samples	-1
	lamb	1
SI	damping	0.1
	lamb	0.1
DMC	aux_dataset	imagenet_32

Table 3.3: Classification result (accuracy %) for continual learning benchmarks, averaged across 5 runs. $*$ 4 tasks only due to crash.

	S-CIFAR-100		S-CIFAR-10		S-MINI-IMAGENET		S-MNIST	
Method	Task IL	Class IL	Task IL	Class IL	Task IL	Class IL	Task IL	Class IL
Finetuning	49.01±1.84	24.21±0.76	74.18±5.18	42.88±1.57	44.06±1.78	20.99±0.89	78.33±4.28	40.92±3.24
Joint	75.45±1.71	55.88±2.03	95.86±1.53	86.94±0.93	66.45±1.95	44.43±1.55	99.88±0.06*	99.62±0.09*
EWC[27]	53.46±2.02	26.18±1.21	77.16±6.22	40.51±4.54	48.41±2.36	23.26±0.94	83.07±2.83	43.19±3.57
LWF[67]	51.28±14.82	26.61±4.73	91.14±3.61	56.32±1.89	52.04±2.5	24.34±1.33	79.84±12.60	51.38±8.69
MAS[25]	52.3±1.95	26.01±1.15	76.72±5.21	38.94±4.77	46.57±2.36	22.16±1.08	86.50±4.83	44.97±2.31
SI[26]	52.19±1.85	25.56±1	76.44±5.67	39.2±2.6	46.72±2.22	22.15±1.06	82.10±3.13	41.48±1.20
DMC[168]	62.31±1.91	35.72±2.07	81.90±11.40	52.70±10.10	53.42±1.61	27.91±1.73	77.90±15.50	48.50±15.30
EEIL[162]	67.24±1.56	40.75±1.49	93.99±1.67	71.2±2.93	57.99±2.51	30.84±1.69	86.51±6.58	72.74±11.74
ICARL[31]	67.13±1.51	44.15±2.02	93.91±2.35	78.54±1.4	58.53±2.63	34.93±2.14	99.79±0.07	98.67±0.15
BIC[35]	67.82±2.89	44.63±3.02	89.41±10.59	71.1±15.41	59.16±3.05	35.19±2.11	76.48±4.52	51.02±2.25
DER++[57]	65.89±1.84	41±1.72	94.7±1.6	77.42±1.98	55.91±2.12	30.78±1.3	99.77±0.08	98.54±0.28
HAL[169]	53.77±10.76	30.08±8.52	74.60±3.64	46.90±1.57	45.94±8.11	22.69±5.06	71.30±4.55	45.80±0.19
AGLA(Ours)	**69.51±1.68**	**46.95±2.19**	**95.64±1.53**	**82.64±1.14**	**60.54±2.09**	**36.23±1.66**	**99.86±0.04**	**98.86±0.18**

Table 3.4: Average forgetting for continual learning benchmarks, averaged across 5 runs. $*$ 4 tasks only due to crash.

	S-CIFAR-100		S-CIFAR-10		S-MINI-IMAGENET		S-MNIST	
Method	Task IL	Class IL	Task IL	Class IL	Task IL	Class IL	Task IL	Class IL
Finetuning	31.28±1.72	59.54±2.14	37.71±9.5	77.9±9.12	24.58±1.37	50.11±2.41	37.67±7.78	69.24±8.44
Joint	-1.21±0.2	5.79±0.59	-0.73±0.17	6.26±1.63	-1.14±0.17	6.23±1.18	-0.02±0.02*	0.15±0.06*
EWC[27]	19.23±2.37	39.62±2.9	29.47±11.44	50.86±13.93	14.6±1.56	32.51±2.8	29.66±4.69	64.96±9.68
LWF[67]	13.71±7.94	39.19±22.02	6.22±3.78	48.94±5.61	12.57±0.69	42.3±1.55	0.07±0.10	18.45±26.53
MAS[25]	21.91±1.94	43.52±2.37	30.56±8.57	51.11±14.79	17.36±2.33	35.75±2.8	23.36±8.88	58.75±8.34
SI[26]	21.56±2.25	46.15±2.32	31.46±9.71	56.27±7.46	17.53±0.99	38.31±1.8	31.16±5.44	68.53±13.28
DMC[168]	4.08±0.95	19.20±2.80	10.80±12.70	33.79±10.13	3.98±2.43	19.22±3.53	18.40±13.0	48.46±19.14
EEIL[162]	5.65±0.68	39.62±1.51	1.16±0.8	42.72±8.26	5.25±0.56	40.51±1.52	15.33±8.74	29.53±20.87
ICARL[31]	**0.61±0.4**	11.79±1.24	0.8±0.66	18.96±4.28	**0.75±0.33**	**11.97±0.63**	0.06±0.09	0.67±0.26
BIC[35]	1.21±0.94	**11.15±2.33**	**0.33±0.27**	18.43±7.71	1.78±0.44	12.44±1.26	**0±0**	5.62±3.16
DER++[57]	9.71±1.03	40.85±0.87	1.06±0.83	30.22±5.38	10.47±0.67	41.93±1.77	0.11±0.18	2.18±0.30
HAL[169]	9.63±4.10	38.15±3.13	1.57±2.21	11.65±16.8	8.29±2.87	36.53±1.32	45.80±0.19	**0.02±0.04**
AGLA(Ours)	5.35±1.16	16.29±5.35	0.59±0.57	**17.13±1.65**	3.78±0.26	15.63±6.77	0.06±0.04	0.89±0.37

are made public in https://github.com/anwarmaxsum/AGLA for further study. The models are evaluated in two continual learning settings, i.e., task incremental learning (Task IL) and class incremental learning (Class IL). **Task IL** is a problem setting where a model is trained to a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K, k \in \{1, ..., K\}$, each task $\mathcal{T}_k$ has its training set $T_k = \{x_n, y_n, id_n\}_{n=1}^{N_k}$, $x_n \in \mathcal{X}$ represents an input sample and $y_n \in \mathcal{Y}$ denotes its true class label, id_n stands for the task id. K, N_k respectively denote the number of tasks and the size of $k-th$ task. **Class IL** is a similar setting with Task IL but the task id id_n is not visible to the model.

3.5.1 Datasets

Four datasets examine the advantage of AGLA: split MNIST (SMNIST) [26], [171], split CIFAR10 (SCIFAR10) [172], [173], split CIFAR100 (SCIFAR100) [172], [174],

Table 3.5: Classification result (accuracy) of AGLA based on various configurations.

Code	Configuration						S-CIFAR-100		S-CIFAR-10	
	Assr	Aug	R.Tr	$(\overline{w}_{i,k})$	$\mathcal{L}_{DER++}$	$\mathcal{L}_{dist}$	Task IL	Class IL	Task IL	Class IL
A		✓	✓	✓	✓	✓	69.16	46.26	**96.42**	**82.96**
B	✓		✓	✓	✓	✓	**70.44**	45.83	96.00	80.50
C	✓	✓		✓	✓	✓	68.52	46.16	96.00	**82.94**
D	✓	✓	✓		✓	✓	69.50	**46.82**	96.18	82.74
E	✓	✓	✓	✓		✓	**70.01**	**47.61**	**96.28**	82.68
F	✓	✓	✓	✓	✓		67.76	43.98	95.56	78.78
AGLA	✓	✓	✓	✓	✓	✓	**69.76**	**47.70**	**96.22**	**83.20**

Table 3.6: Forgetting rate of AGLA based on various configurations.

Code	Configuration						S-CIFAR-100		S-CIFAR-10	
	Assr	Aug	R.Tr	$(\overline{w}_{i,k})$	$\mathcal{L}_{DER++}$	$\mathcal{L}_{dist}$	Task IL	Class IL	Task IL	Class IL
A		✓	✓	✓	✓	✓	**4.04**	19.78	**0.35**	21.03
B	✓		✓	✓	✓	✓	5.30	31.71	0.68	27.20
C	✓	✓		✓	✓	✓	**4.16**	19.00	0.78	**17.35**
D	✓	✓	✓		✓	✓	4.74	**10.94**	**0.43**	20.85
E	✓	✓	✓	✓		✓	**4.36**	**11.56**	0.80	**19.40**
F	✓	✓	✓	✓	✓		10.07	36.44	1.20	29.83
AGLA	✓	✓	✓	✓	✓	✓	5.11	**13.41**	**0.40**	**19.35**

and Split Mini-ImageNet (SMImageNet) [163], [175]. SMNIST and SCIFAR10 feature the task-incremental and class-incremental learning problems with 5 tasks, where each task contains two disjoint classes, whereas SCIFAR100 and SMImageNet also present the two problems with 10 tasks, where each task features 10 disjoint classes. Note that no task IDs and the single-head structure are applied in the class-incremental learning problem, while the task-incremental learning problem benefits from the presence of task IDs and the multi-head network structure. Table 3.1 tabulates our experimental setting. The hyperparameter settings for the consolidated algorithms are presented in Table 3.

3.5.2 Baseline Algorithms

AGLA is compared with **ten algorithms**: EWC [27], SI [26], LWF [67], MAS [25], iCaRL [31], DER++ [57], EEIL [162], BIC [35], DMC [168] and HAL [169] . In addition, the lower (fine-tuning) and upper bounds (joint) are provided, where the naive SGD depicts the lower bound case and the joint training approach presents the upper bound case. Comparison is done in the same computational

Table 3.7: Classification result (accuracy) of the benchmarks based on memory size.

Memory	Method	S-CIFAR100 Accuracy		S-CIFAR100 Forgetting	
		Task IL	Class IL	Task IL	Class IL
2000	EEIL	63.88	33.43	8.14	49.90
	ICARL	63.94	40.07	**2.47**	15.71
	BIC	67.41	42.33	2.96	**15.54**
	DER++	62.53	34.33	14.99	52.81
	HAL	55.92	28.02	10.72	47.46
	AGLA (Ours)	**67.60**	**43.50**	8.37	31.37
3000	EEIL	65.70	36.94	7.78	46.44
	ICARL	65.08	42.01	**2.52**	15.32
	BIC	68.21	44.10	2.58	**13.99**
	DER++	64.26	36.95	11.80	48.23
	HAL	57.23	30.75	9.02	41.80
	AGLA (Ours)	**69.07**	**45.64**	5.21	19.58
4000	EEIL	66.12	38.54	5.84	42.58
	ICARL	65.34	42.13	**0.69**	**12.59**
	BIC	67.26	44.28	2.23	13.69
	DER++	65.06	39.43	10.18	42.94
	HAL	57.77	32.19	9.02	40.32
	AGLA (Ours)	**69.34**	**46.19**	5.66	24.88
5000	EEIL	66.74	40.18	5.71	40.14
	ICARL	66.09	43.03	**0.31**	**10.73**
	BIC	67.84	44.80	1.87	13.32
	DER++	66.57	41.33	8.33	39.87
	HAL	57.76	32.32	8.18	37.20
	AGLA (Ours)	**69.34**	**47.70**	5.11	14.54

environments, eight NVIDIA DGX A100 GPUs with 40 GB each, under the FACIL framework [176] to ensure fair comparison.

3.5.3 Experiment Setup

For all problems, ResNet18 is deployed without the pretraining process. A grid search approach is applied to all consolidated algorithms, where detailed hyper-parameters are provided in Table 3. On the other hand, the assessor is formed as the LSTM network: ResNet18 feature extractor, two LSTM layers, and a single fully connected layer where the number of nodes in each layer is set as 64. The memory size of the memory-based approaches, AGLA, BIC, DER++, EEIL, and ICaRL, is set equally to 5000 samples for all problems, but the memory analysis is offered here as well. The image size of the SMImageNet is resized to 32 by 32 from

the original size of 84 by 84. The hyperparameters of consolidated algorithms are listed in Table VII. Note that AGLA only carries a single hyperparameter, memory size, common to other memory-based approaches. Other hyperparameters are shared by all benchmarked algorithms.

3.5.4 Numerical Results

From Table 3.3, it is shown that AGLA outperforms other algorithms in all four problems with noticeable gaps. AGLA produces better accuracy than BIC having the second highest accuracy by about $2-3\%$ in the SCIFAR100 problem for both the task-incremental and class-incremental learning problems. This finding confirms the advantage of the meta-trained assessor in guiding the learning process of the base model. The assessor enables a soft sample selection mechanism, assigning high weights only to positive samples and a dynamic weighting mechanism of loss functions controlling interactions of the three loss functions. Note that BIC features the bias-correction layer to handle the class-imbalanced problem. The class-imbalanced problem is addressed in AGLA with the compensated over-sampling technique, which does not call for an additional training stage as per BIC. This is done by simply augmenting the old samples in the memory with corrections to prevent out-of-distribution samples. The same pattern is observed in the SCIFAR10 problem, where AGLA delivers the highest accuracy with about 1.2% gap to DER++ in the task-incremental learning problem and 4% gap to ICARL in the class-incremental learning problem.

AGLA is also superior to other algorithms in the SMImagenet problem. It beats the BIC by 1.5% margin in both the task-incremental and class-incremental learning problems. Once again, this aspect confirms that the assessor-guided learning process of AGLA achieves a better tradeoff between plasticity and stability than other algorithms. The use of the meta-trained assessor also performs a selection of suitable learning strategies for data samples. It controls how new concepts/semantics and past concepts are to be accepted. On the other hand, AGLA also outperforms other consolidated algorithms with a tiny margin in the task-incremental learning and class-incremental learning of the SMNIST problem. Note that the SMNIST problem is considered an easy problem for continual learn-

ing algorithms, i.e., most algorithms produce decent results in this problem. The performance of AGLA is also stable across the four problems. That is, it does not suffer from any sudden performance losses, i.e., this occurs for BIC (SCIFAR10) and DER++ (SMINIIMAGENET). Another important finding is seen in the fact that the memory-based approaches are better than the regularization-based approaches. This fact becomes obvious in the class-incremental learning problem. Nonetheless, the regularization-based approach is simple to implement and faster than the memory-based approach. Table 3.4 presents the average forgetting of all consolidated algorithms. It is perceived that AGLA produces comparable average forgetting to other memory-based approaches, where it mostly occupies the first or second place, and is still far better than the regularization-based approaches. This should not hinder the advantage of AGLA because the average forgetting is referred to only if the accuracy of compared algorithms is on par, i.e., AGLA beats other algorithms in all four cases. Low average forgetting score might occur in the case of poor average accuracy.

3.5.5 Ablation Study

This section demonstrates the contribution of each learning component of AGLA. Five aspects are studied here: 1) the learning performance of AGLA without the presence of an assessor is presented - configuration (A); 2) the learning performance of AGLA without the data augmentation strategy is demonstrated - configuration (B). That is, no oversampling strategy is performed; 3) the learning performance of AGLA without the random transformation strategy to construct the validation set of the meta-learning phase is depicted - configuration (C). This implies the use of the training set in updating the assessor; 4) the learning performance of AGLA without the out-of-distribution compensation is depicted - configuration (D). This implies that the augmented data is not guaranteed to be within the original samples' distribution; 5) the learning performance of AGLA without the DER loss is shown - configuration (E). This implies the application of the two loss functions only, the cross entropy loss function and the distillation loss function; 6) the learning performance of AGLA without the distillation loss function is presented - configuration (F). This implies the use of the cross-entropy loss function

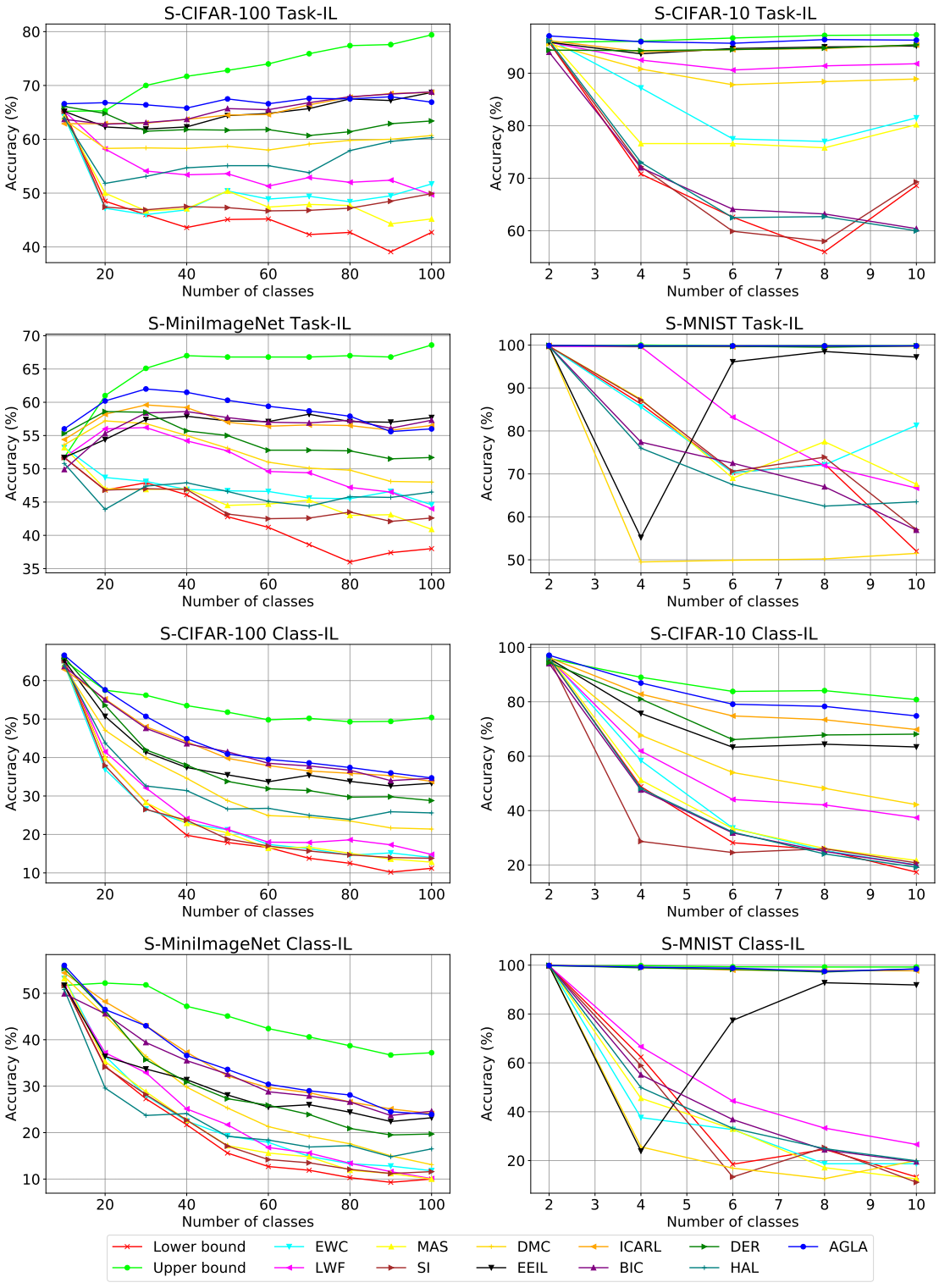


Figure 3.2: Classification Accuracy of Consolidated Algorithms across All Datasets.

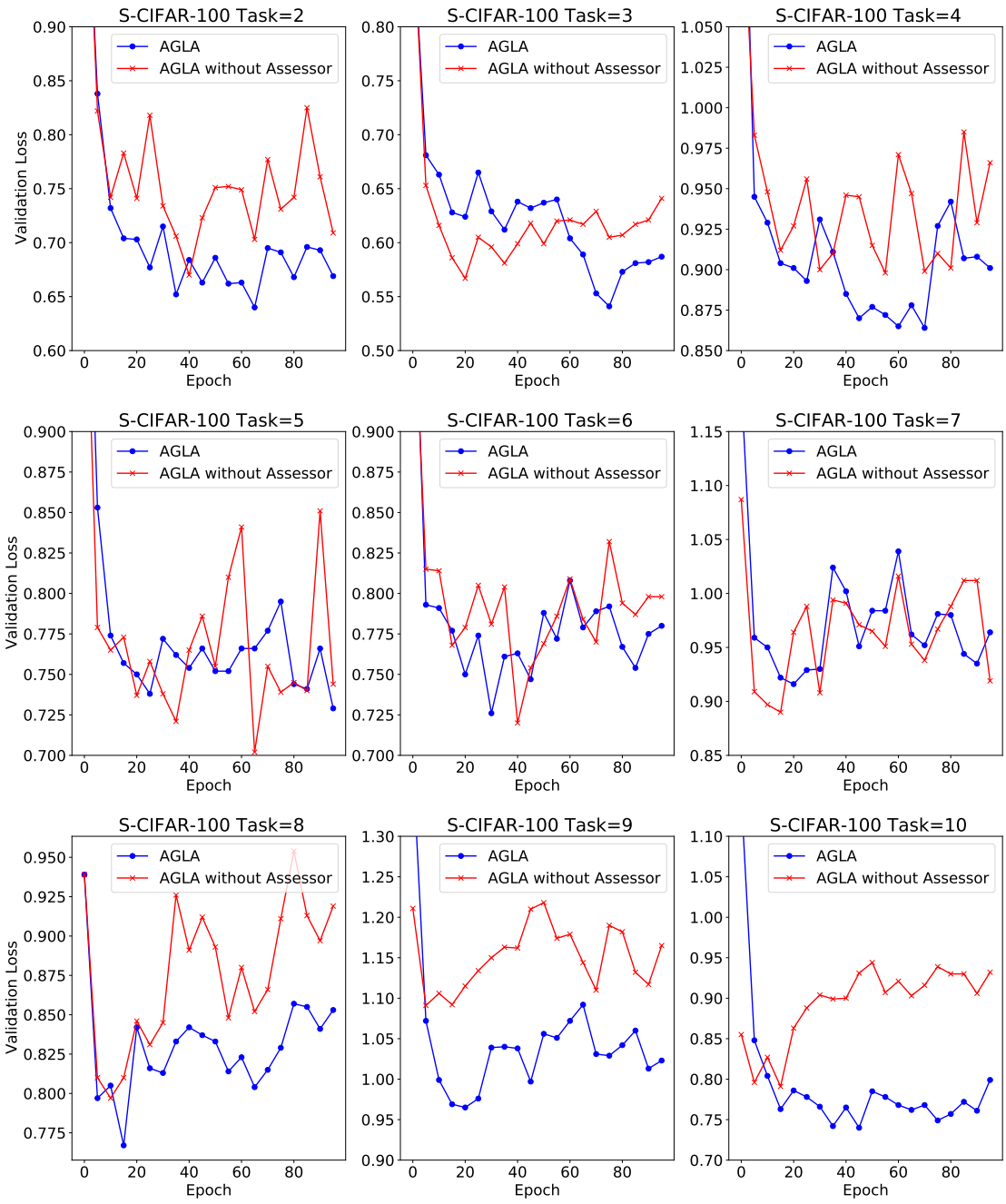


Figure 3.3: The trace of losses for AGLA with or w/o Assessor in the S-CIFAR-100 problem.

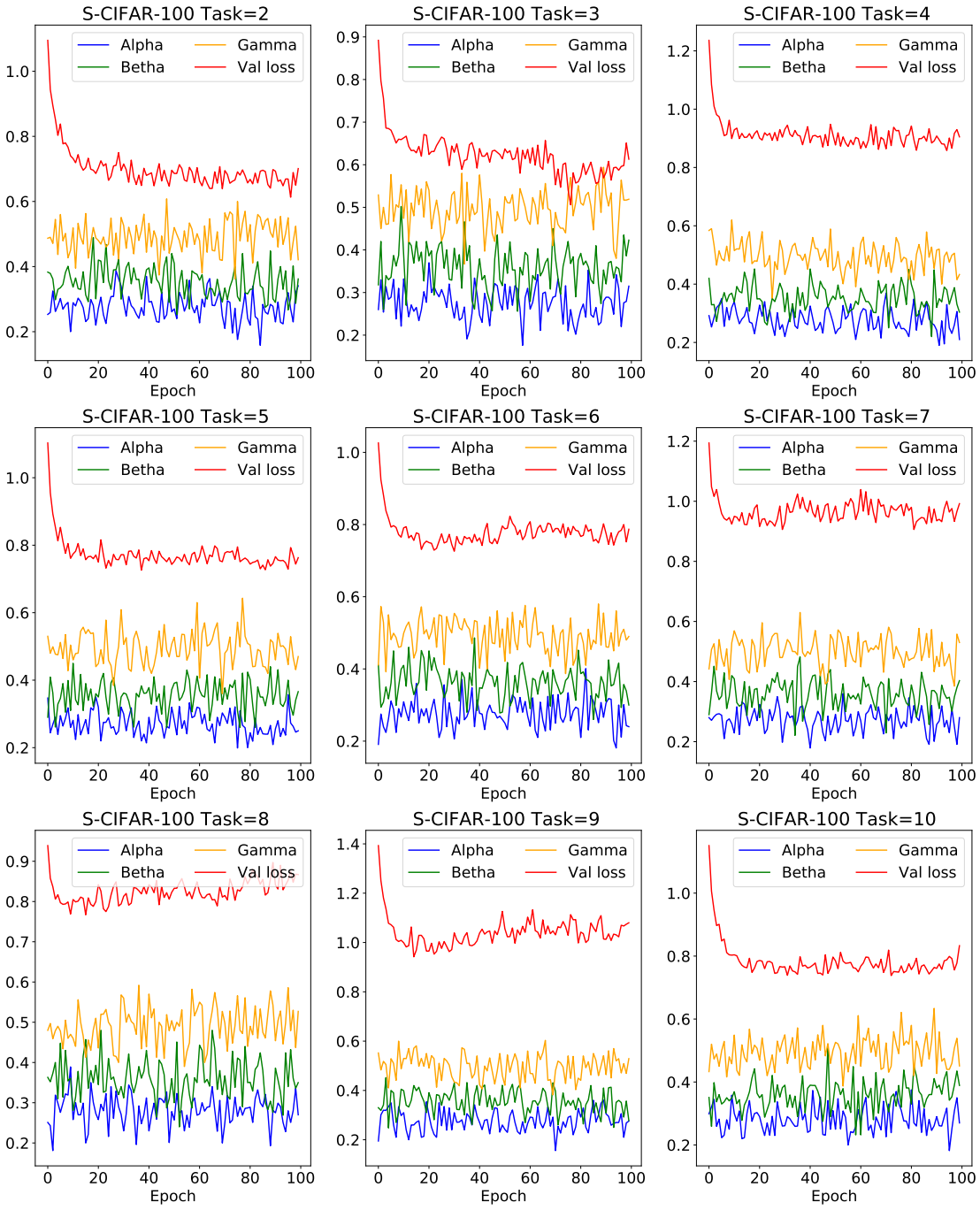


Figure 3.4: The trace of weighting coefficients and loss for AGLA in the S-CIFAR-100 problem.

and the DER loss function. All numerical results are tabulated in Table 3.5 and in Table 3.6, where they are produced using the SCIFAR10/100 problems in both the class-incremental learning problem and the task-incremental learning problem. Note that we are only able to report numerical results of a single run for the ablation study due to limited access to computational resources.

The absence of the assessor as depicted in configuration (A) leads to significant performance deterioration in all cases with up to 2% drop in accuracy and 7% drop in average forgetting. This finding confirms the importance of the assessor in balancing between plasticity and stability. The three loss functions of AGLA contribute positively to the final performance. The absence of the distillation loss function in configuration (F) or the DER loss function in configuration (E) causes a major performance drop, i.e., up to 15% for configuration (F) and up to 1% for configuration (E). The DER loss function and the distillation loss function protect against catastrophic forgetting. This finding is further borne out by the major increase of average forgetting. Table 3.6 shows that the absence of distillation loss and DER loss increases the forgetting rate up to 23% for configuration (F) and 2% for configuration (E). The over-sampling procedure plays a vital role, where configuration (B) shows performance degradation in accuracy (up to 3%) and average forgetting (up to 18%). This mechanism is designed to cope with the class imbalanced problem of the memory-based approach. The random transformation technique to construct the validation set for the assessor training process improves the model's generalization, i.e., configuration (C) results in noticeable loss in performance (up to 2%) and increasing forgetting rate (up to 6%). This aspect demonstrates that the meta-training strategy should be carried out using different training and validation sets. Last but not least, the compensation strategy also plays an important role in the proposed method, i.e., configuration (D) leads to drops in accuracy by up to 1% and an increase in forgetting rate by up to 2%. This finding shows that the augmented samples should be protected to be within the original samples' distribution.

3.5.6 Memory Analysis

This section depicts the performance of AGLA and other consolidated memory-based approaches with different memory sizes in the SCIFAR100 problem under both class-incremental and task-incremental configurations. Specifically, the memory size is set to 2000, 3000, 4000, respectively, where consolidated numerical results are provided in Table 3.7. As with the ablation study, the memory analysis is performed under a single run due to limited access to computational resources. However, this should not bias our findings in the ablation study and the memory analysis since AGLA is superior to all baseline algorithms in all cases in our main results reported in Table 3.3 and 3.4, where five runs are committed under different random seeds.

It is obvious that AGLA still maintains superior performance compared to other memory-based approaches under smaller memory sizes (2000, 3000, 4000) than that reported in Table 3.3 and 3.4 (5000) with clear margins in the range of $1-3\%$. The performances of AGLA are relatively stable under different memory budgets, where performance drops due to reductions of memory sizes are less than 4%. Although ICARL and BIC produced a smaller average forgetting than AGLA, their accuracy is far worse than AGLA. This finding confirms the advantage of the compensated over-sampling approach to cope with the severe class imbalanced problem without any extra parameters. It is also observed that the performance drops due to the memory constraint are higher in the class-incremental learning problem than in the task-incremental learning problem because of the absence of task IDs and the application of single-head classifiers.

3.5.7 Performance Evaluations per Task

The evolution of classification accuracy of all consolidated algorithms in all datasets in both the class-incremental and task-incremental learning problems in one of the runs is pictorially illustrated in Fig. 3.2, where the advantage of AGLA is clearly demonstrated. For the class-IL problems, AGLA produces higher classification accuracy in all tasks across all datasets than its counterparts.

For the task-IL problem, AGLA outperforms other algorithms across all tasks

in all datasets. That is, it returns the highest classification accuracy per task among other algorithms. The positive contribution of the assessor-guided continual learning is portrayed here, where it beats the DER++ algorithm across all tasks in all datasets, regardless of the task-IL or class-IL problem. Note that AGLA shares similar loss functions as DER++ except for the application of the distillation loss function, the assessor, and the meta-weighting strategy.

3.5.8 Assessor vs w/o Assessor

This subsection confirms the positive roles of the assessor in guiding the continual learning process of AGLA where Fig. 3.3 visualizes the trace of AGLA's losses in the SCIFAR100 problem with and w/o the assessor under the class-IL setting in one of the experiments. Clearly, the use of an assessor to perform the soft-weighting mechanism of data samples and the selection of appropriate learning strategies result in faster convergence and lower losses than in the absence of an assessor in most tasks. Importantly, AGLA remains stable across all tasks with the assessor-guided learning process, i.e., the losses signify decreasing trends across all tasks. This fact implies the success of our meta-learning-based bi-level optimization approach to fine-tune both the base network and the assessor.

3.5.9 Evolution of Meta-weights

Figure 3.4 shows the plots visualizing the evolutions of meta-weights (α, β, γ) in CIFAR-100 dataset. The figure shows that the value of γ and β that correspond to distillation loss and dark experience replay loss, respectively, are higher than the value of α that corresponds to cross-entropy loss. It shows that the proposed method needs higher weights for the memory samples than the current samples. This is in line with the fact that the number of memory samples is less than the current task samples, so it needs **high** weights in current task training to handle class-imbalance problems. The figure also shows that the dynamic evolution of the three weights leads to decreasing trends of validation losses.

3.5.10 Complexity Analysis

This sub-section discusses the complexity analysis of the proposed method. Suppose that N is the total samples of a dataset, $k \in \{1, ..., K\}$ is the index of task, N_k is the number of samples on task k **which satisfies** $\sum_{k=1}^{K} N_k = N$, e is the number of epoch for networks training, M_k is the size of memory on task k **that satisfies** $\sum_{k=1}^{K} M_k < N$. Following the pseudo-code in Algorithm 1, there are **several processes**, i.e., augmentation (Aug), union of train data and memory (Union), random transformation (R.Trans), assessor and base networks update (N.Update), memory sampling (M.Sampling), and memory update (M.Update). Please note that augmentation and random transformation are done in a few operations ($c < 10$). Let C denote the complexity of a process. Following Algorithm 1, the complexity of the proposed method can be written as follows:

$$\begin{aligned} \mathcal{C}(AGLA) = \sum_{k=1}^{K} (\mathcal{C}(Aug) + \mathcal{C}(Union) + \mathcal{C}(R.Trans) + \\ \mathcal{C}(N.update) + \mathcal{C}(M.Samping) + \mathcal{C}(M.Update)) \end{aligned} \tag{3.23}$$

$$\begin{aligned} \mathcal{C}(AGLA) = \sum_{k=1}^{K} ((c.N_k) + max(N_k, M_k) + (c.(N_k + M_k)) + \\ (2.e.(N_k + M_k)) + (N_k + M_k) + max(N_k + M_k, M_{k+1})) \end{aligned} \tag{3.24}$$

$$\begin{aligned} \mathcal{C}(AGLA) = ((c.\sum_{k=1}^{K} N_k) + max(\sum_{k=1}^{K} N_k, \sum_{k=1}^{K} M_k) + \\ (c.(\sum_{k=1}^{K} N_k + \sum_{k=1}^{K} M_k)) + \\ (2.e.(\sum_{k=1}^{K} N_k + \sum_{k=1}^{K} M_k)) + \\ (\sum_{k=1}^{K} N_k + \sum_{k=1}^{K} M_k) + \\ max(\sum_{k=1}^{K} N_k + \sum_{k=1}^{K} M_k, \sum_{k=1}^{K} M_{k+1})) \end{aligned} \tag{3.25}$$

Since $\sum_{k=1}^{K} N_k = N$, $\sum_{k=1}^{K} M_k < N$, and $\sum_{k=1}^{K} M_{k+1} < N$. Then the complexity of AGLA can be derived as:

$$\begin{aligned}\mathcal{C}(AGLA) \leq ((c.N) + N + (c.(N+N)) + \\ (2.e.(N+N)) + (N+N) + (N+N))\end{aligned} \tag{3.26}$$

$$\mathcal{C}(AGLA) \leq (cN + N + 2cN + 4eN + 2N + N) \tag{3.27}$$

$$\mathcal{C}(AGLA) \leq (4N + 3cN + 4eN) \tag{3.28}$$

$$\mathcal{C}(AGLA) = (O(N) + O(cN) + O(eN)) \tag{3.29}$$

Considering that c is a small number (< 10), then the complexity of AGLA can be derived to:

$$\begin{aligned}\mathcal{C}(AGLA) &= (O(N) + O(N) + O(eN)) \\ &= (O(N) + O(eN)) \\ &= O(eN)\end{aligned} \tag{3.30}$$

The equation above concludes that the complexity of the proposed method is $O(eN)$, where N is the total number of instances across all tasks, and e is the number of epochs for training. In case the epoch is set as constant, e.g., 100, then the complexity of the proposed method will be $O(N)$.

3.6 Summary

This chapter proposes an assessor-guided learning approach (AGLA) for rehearsal-based continual learnin,g where it puts forward a sequence-aware assessor to guide the learning process of the base learner. The assessor performs a soft-weighting mechanism controlling the influence of a data sample and the interaction of loss functions. It is made possible by forming the loss function of the base learner as the meta-weighted combination of the cross-entropy loss function, the DER loss

function, and the distillation loss function. The underlying objective is to arrive at a proper tradeoff between the plasticity and the stability because not only positive samples are selected, but also suitable learning strategies are chosen for every sample, i.e., the assessor balances the stability and the plasticity for every sample. Another major contribution is perceived in the proposal of compensated over-sampling (COS) to address the class-imbalanced problem on continual learning, where corrections are carried out while augmenting data samples to avoid the effect of out-of-distribution augmented samples. Rigorous numerical results have been carried out using four popular continual learning problems in both the task-incremental and class-incremental learning problems. AGLA has been compared with ten recently published algorithms under the same computational environments, where AGLA demonstrates improved and stable accuracy across all four problems in both the task-incremental learning problems and the class-incremental learning problems, with noticeable margins. In the realm of average forgetting, AGLA delivers comparable performances to those of memory-based approaches, where it mostly occupies the second-lowest average forgetting index, and better than those of regularization-based approaches. Our ablation study demonstrates the advantage of each learning component of AGLA, where the absence of one component results in major performance drops. AGLA also maintains decent performance with tiny memory sizes, as indicated by our memory analysis. Existing continual learning approaches, including the AGLA function with a large number of samples, overfit quickly in the case of a few samples in each task.

The upcoming chapter addresses rehearsal-free continual learning problems where there is no exemplar memory from previously learned tasks. In real-world applications, the samples from the previous task might be unavailable due to data privacy constraints. The constraint complicates the CL method in maintaining stability on previously learned tasks.

Chapter 4

LEAPGen: Vision and Language Synergy for Rehearsal Free Continual Learning

In the previous chapter, AGLA is proposed to handle catastrophic forgetting in a rehearsal-based manner. The demerit of AGLA is that it requires memory samples from previously learned classes that may be unavailable due to privacy constraints. This chapter proposes a novel Language as Prompt Generator (LEAPGen) as the successor of AGLA to address catastrophic forgetting in a rehearsal-free manner. Leveraging a pre-trained model, the prompt-based approach has demonstrated its success for rehearsal-free continual learning problems. However, it still suffers from catastrophic forgetting due to inter-task vector similarity and unfitted new components of previously learned tasks. On the other hand, prompt-based methods with a language-guided approach fall short of their full potential due to the minimal utilization of knowledge and participation in the prompt tuning process. LEAPGen offers a novel prompt-based structure and algorithm that incorporates 4 key concepts: (1) language as input for prompt generation, (2) task-wise generators, (3) limiting matching descriptors search space via soft task ID prediction, and (4) generated prompt as auxiliary data. The experimental analysis shows the superiority of LEAPGen to existing SOTAs in CIFAR100, ImageNet-R, and CUB datasets with significant margins in terms of final average accuracy, cumulative average accuracy, final forgetting measure, and cumulative forgetting measure.

The historical analysis confirms LEAPGen successfully maintains the stability-plasticity trade-off in every task. LEAPGen proves its consistency in achieving high performances in various prompt lengths, layer depths, and the number of generators per task compared to the SOTAs. In addition, this chapter provides a comprehensive theoretical analysis and extensive numerical results.

Part of this chapter is published as: MA Ma'sum, M Pratama, S. Ramasamy, L Liu, H Habibullah, R Kowalczyk. Vision and Language Synergy for Rehearsal Free Continual Learning. The Thirteenth International Conference on Learning Representations (ICLR 2025). https://openreview.net/forum?id=9aZ2ixiYGd **[CORE A*]**

4.1 Introduction

Continual learning (CL) attracts tremendous interest in the world of artificial intelligence (AI) for dynamic environments. CL methods address the catastrophic forgetting when handling a sequence of tasks [54], [55]. CL methods are proven promising in various applications, e.g., computer vision, NLP, graph, and Automation [177], [178], [179], [180]. However, most of the CL methods require a small portion of previous task exemplars for a rehearsal process to maintain the knowledge from previously learned tasks [31], [35], [97]. This approach is not practical since old data can be unavailable anymore due to privacy constraints or data openness policy. Besides, the rehearsal approach incurs additional memory and computational expenses. To this end, a prompt-based approach comes as a breakthrough solution. Leveraging a frozen pre-trained backbone, the prompt-based approach demands only small added learnable parameters (prompt) to adapt to a new task. This approach leads to efficient yet accurate solutions for catastrophic forgetting.

Despite its promising performance and efficiency, the prompt-based approach suffers from the following dilemmas: First, the task-specific prompt approach, e.g., DualPrompt[38], SPrompt[84], HiDE-Prompt[85] and CPrompt [86], relies on a task identifier, where a misidentified task ID leads to misclassification, e.g., in the case where the trained keys of two or more tasks have a similar vector. Second,

the growing component approach, e.g., CODA-P[39], EvoPrompt[87], and ConvPrompt[65], on the other hand, increase prompt components to adapt to a new task, instead of the task identifier. However, the (new) extra components are optimal for the current task, but not for previous tasks. In addition, the old task components/generators that are trained side-by-side with the current task components/generators lead to forgetting. Third, the shared learnable parameter approach, such as G-Prompt in DualPrompt or semantic embedding in ConvPrompt has the risk of forgetting since it is adjusted in all tasks. This condition is similar to the pool-based approach, i.e., L2P[37] where any prompt in the pool may be trained in every task, as it is chosen by the input-to-prompt similarity mechanism. On the other hand, a language-guided approach, e.g., LGCL [60] and GMM [181] do not fully reach the optimum potential of language assistance in prompt tuning since they only use the name of the class as additional information.

In retrospect to the aforementioned drawbacks, this chapter proposes a novel approach for prompt-based continual learning. We design a new prompt generation and its learning mechanism to handle catastrophic forgetting. The key principles of our approach are (1) language as input for prompt generation instead of a task-shared learnable vector. The input is selected from the catalog of language descriptors based on their similarity to the input. (2) task-wise generators instead of growing generators to generate a prompt component, thus the current task generators are frozen in the upcoming tasks' training. (3) A new soft task ID prediction to limit the search space of matching descriptors. Our mechanism is different from the task ID prediction in the previous methods. (4) generated language-based prompts as the auxiliary data to be appended to the input embedding. To our knowledge, these four principles are new in the prompt-based CL method and have not yet been explored in the previous State-of-the-art(s) (SOTAs).

This chapter proposes four main contributions as follows:

1. A novel rehearsal-free prompt-based method for the CL problem named Language as Prompt Generator (LEAPGen), consisting of four main components as aforementioned.

2. A new task ID predictor and joint loss function for LEAPGen.

3. Rigorous experiments and analysis prove the superiority of LEAPGen compared to the existing SOTAs in general and historical performance in terms of accuracy and forgetting index, and robustness of the proposed method in various settings, i.e., the number of layers, prompt length, and number of generators.

4. Extended analysis and discussion of LEAPGen, both theoretically and numerically.

4.2 Preliminary

4.2.1 Problem Formulation

In this study, we focus on Class incremental learning (CIL) since it is the most challenging sub-problem in CL. Class incremental learning (CIL) problem is defined as the problem of learning a sequence of fully supervised tasks $\{\mathcal{T}^t\}_{t=1}^{T}$, where after finishing learning a task $\mathcal{T}^t$, a model must recognize all learned tasks, i.e., $\mathcal{T}^1$,$\mathcal{T}^2$,...$\mathcal{T}^{t-1}$,$\mathcal{T}^t$. Symbol T represents the number of consecutive tasks. Each task carries pairs of training samples, i.e., $\mathcal{T}^t = \{(x_i^t, y_i^t)\}_{i=1}^{|\mathcal{T}^t|}$ where $x_i \in \mathcal{X}^t$ and $y_i \in \mathcal{Y}^t$ denotes input image and corresponding label, while $|.|$ denotes cardinality. $\mathcal{C}^t$ denotes the unique class labels in $\mathcal{Y}^t$, i.e., $\mathcal{C}^t = unique(\mathcal{Y}^t)$, and $|\mathcal{C}^t|$ denotes the number of classes in $\mathcal{T}^t$. Each task t is disjoint from another task t', i.e., $\forall t, t' \neq t, (\mathcal{T}^t \cap \mathcal{T}^{t'} = \emptyset)$.

4.2.2 ViT and Prompt-based CL

Prompt-based CL approach is built upon Visual Transformer (ViT) architecture [81], i.e., a sequence of L-layered model $f_\theta(.)$ where a layer-l ($l \in [1..L]$) contains H Multi Head Self Attention (MSA) combined with Multi-Layer Perceptron (MLP) block, Norm, addition and residual connections. Given an input patched embedding $x_{e_{l,h}}$, the hth head MSA of layer-l, parameterized by key-query-value matrices, i.e., $W_{l,h}^Q, W_{l,h}^K, W_{l,h}^V$ performs self-attention function as defined in equation below:

$$A(Q_{l,h}, K_{l,h}, V_{l,h}) = Softmax((Q_{l,h}K_{l,h}^T)/\sqrt{D_h})V_{l,h} \tag{4.1}$$

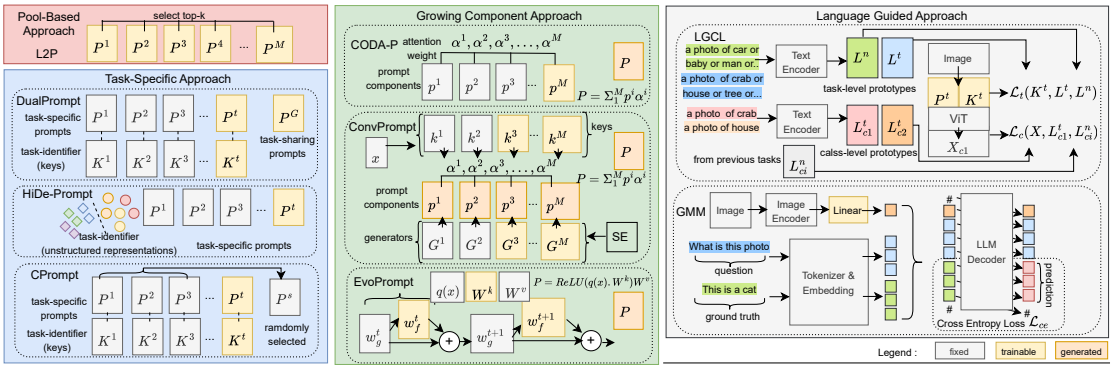


Figure 4.1: Visualization of Pool-Based Prompting (red area), Task-Specific Prompting (blue area), Growing Component Prompting (green area), and Language Guided Approach for Continual Learning (gray area).

where $Q_{l,h} = x_{e_{l,h}} W^Q_{l,h}, K_{l,h} = x_{e_{l,h}} W^K_{l,h}, V_{l,h} = x_{e_{l,h}} W^V_{l,h}$ are query, key, and value and D_h denotes their dimension. Prompt-based CL attaches a small set of trainable parameters called prompts into a frozen pre-trained ViT model (fixed θ) [37], [38]. The attachment techniques are mainly divided into 2 types, i.e., prompt tuning [82] and prefix tuning [83]. Prompt tuning prepends a prompt vector $P \in \mathbb{R}^{L_P \times D}$ into the query, key, and value of MSA. Suppose the prompt is divided into head-wise parts, i.e., $P = [P_1, P_2, ...P_H]$, then $P_h \in \mathbb{R}^{L_P \times D/H}$ is prepended into h^{th} head of lth layer, and the attention mechanism is computed as $A([P_h : Q_{l,h}], [P_h : K_{l,h}], [P_h : V_{l,h}])$. The symbols L_P, D, and ":" denote prompt length, embedding dimension, and concatenation operator, respectively. Prefix tuning as utilized in DualPprompt [38] divides P_h into pair of P^K_h and P^V_h for key and value respectively, then the attention mechanism becomes $A(Q_{l,h}, [P^K_h : K_{l,h}], [P^V_h : V_{l,h}])$.

4.2.3 Types of Prompt-based Approaches

Figure 4.1 visualizes the topology of prompt-based approaches. **The pool-based approach**, e.g., L2P[37] selects top-k prompts from the prompt pool $[P^1, P^2, ...P^M]$ where M is the pool size. Any prompt P^m in the pool is trainable at any task $t \in [1..T]$. **The task-specific prompt approach**, e.g., DualPrompt[38] trains only the corresponding prompt P^t during the training process on task t while the prompts associated with the previous tasks, i.e., $[P^1, P^2, ...P^{(t-1)}]$ are frozen. Each task t is accommodated with a learnable key K^t to identify the task ID and select

the prompt P^t during the inference process. The prompt identification is based on the highest cosine similarity to the input. HiDe-Prompt[85] utilizes unstructured representation (centroids) produced by unsupervised learning instead of learnable keys. CPrompt[86] randomly selects a frozen prompt from $[P^1, P^2, ..., P^{t-1}]$ to support the tuning of P^t. **The growing component approach** dynamically increases the number of prompt components, e.g., $p^i, p^{i+1}, ..p^M$ to adapt to a new task. The final prompt is generated by a weighted sum of the components with similarity coefficients, e.g., $P = \Sigma_{i=1}^{M} p^i \alpha^i$. In CODA-P[39], p^i is a pre-defined learnable parameter, while $\alpha^i = \gamma(q(x), k^i)$ is attention similarity between input query $q(x)$ and learnable key k^i associated to p^i. In ConvPrompt[65], p^i is generated by the respective generator network G^i from the shared embedding (SE). Similarly, α^i is the cosine similarity between input x and k^i. EvoPrompt[87] grows prompt memory W that forms the fusion of referred (previously learned) memory w_g^t and current working memory w_f^t. W is implemented as a linear model with 2 parts, i.e., W^k and W^v, and the prompt P is generated by the formula $P = ReLU(q(x).W^k)W^v$.

4.2.4 Language Guided Approaches

The gray colored area of figure 4.1 shows how the role of language modality for the prompt-based method (LGCL) and generative method (GMM). LGCL[60] generates task level prototypes, i.e., $L^1, L^2, ..L^t$ and class level prototypes, e.g., $L_1^t, L_2^t, ..L_i^t$, then utilizes the prototypes for loss computation, i.e., $\mathcal{L}_t(K^t, L^t, L^n)$ and $\mathcal{L}_c(X, L_c^T, L_{ci}^n)$, where $\mathcal{L}_t$ and $\mathcal{L}_c$ are task-wise and class-wise losses respectively, X and L_c^t are the ViT output and language prototype for current learned class respectively, K^t is the learnable key associated with prompt P^t, and L^n and L_c^n are task-level and class-level prototypes taken from the previously learned tasks. GMM[181] transforms question-and-answer (ground truth) sentences into an embedding and then passes the embedding into the LLM decoder. The decoder generates a predicted embedding associated with the ground truth embedding. The cross-entropy loss of predicted and ground truth tunes a trainable linear model.

4.2.5 Dilemma and Drawbacks

The pool-based approach has a high risk of forgetting since all the prompts $[P^1, P^2, ...P^M]$ are possible to be trained in all tasks $t \in [1..T]$. A prompt P^i is optimal for the tth task but is tuned again in the $t+1^{th}$ task. Therefore, it is no longer optimal for t^{th} task and leads to forgetting. In a task-specific approach, two different tasks, e.g., $t \neq t'$ could produce similar key vectors i.e. $0.73 \leq cos(K^t, K^{t'}) \leq 0.96$ (Please see section 4.2.7) that leads to inaccurate task identification, which leads to misclassification and forgetting. Note that K^t is trained based on all samples of all classes in $\mathcal{T}^t$. $\mathcal{T}^t$ and $\mathcal{T}^{t'}$ are indeed disjoint so that $\mathcal{X}^t \cap \mathcal{X}^{t'} = \emptyset$, but they could have similar representation, i.e., $f_\theta(\mathcal{X}^t) \approx f_\theta(\mathcal{X}^{t'})$ such as in CUB dataset where all the images are the photos of bird. Similarly, HiDe-Prompt suffers from the same drawback where two representations from different tasks have similar values $f_i \in \mathcal{T}^t \approx f_j \in \mathcal{T}^{t'}$. In the growing component approach, newly added components, i.e., $p^i, p^{i+1}, ..p^M$ disrupt the previous components, i.e., $p^1, p^2, ..p^{i+1}$ that are already optimal for previous tasks. In addition, utilizing a continuously learned parameter, e.g., shared embedding (SE)[65] or task-sharing prompt P^G[38] increases the chance of forgetting. The existing language guidance is not fully explored. LGCL produces class prototype L_c^n by encoding string "the photo of class name". However, the prototypes could be misleading due to high similarity between different classes, e.g., the prototype of class "Great White Shark" has 0.9 cosine similarity to the prototypes of class "Tree Frog" and "Iguana", please see section 4.2.8. With such different classes, we can not directly utilize language representations as references or guidance for model training.

4.2.6 Preliminary Analysis

Table 4.1 presents the average accuracy and average forgetting of all learned classes in the first, second, and last tasks on ImageNet-R dataset with the 10-task setting. The table shows that the existing prompt-based methods suffer from 9-15% average accuracy drop between the first and the final tasks. In the second task, the methods already suffer from up to 5% average forgetting, and the amount tends to increase to up to 8% in the last tasks. Instead of reducing the average forgetting, almost all methods experience higher forgetting in the latest tasks. HiDe-prompt manages

Table 4.1: Preliminary results on Imagenet-R with 10 tasks setting.

Method	**Avg. Accuracy**			**Avg. Forgetting**		
	T1	T10	Drop	T2	T10	Inc.
L2P	76.89	62.50	14.39	5.52	5.01	-0.51
DualPrompt	78.97	68.59	10.39	3.83	4.61	0.79
CODA-P	89.24	73.77	15.47	5.04	7.94	2.90
LGCL	78.59	68.65	9.93	2.91	4.75	1.85
HiDe-Prompt	85.22	75.75	9.47	3.29	2.29	-1.00
PGP	78.97	68.62	10.35	3.83	4.53	0.70
EvoPrompt	89.27	76.00	13.27	5.17	4.22	-0.95
CPrompt	90.91	76.32	14.59	4.55	6.10	1.54
ConvPrompt	89.53	77.08	12.45	2.57	4.17	1.61

to reduce the average forgetting with the highest amount, i.e., 1%, but in the last task, it achieves $1-2\%$ lower accuracy than the best achiever, i.e., ConvPrompt. Despite its highest performance in the final tasks, ConvPrompt suffers from a fairly high accuracy drop, i.e., 12%, and an increase of 1.5% forgetting. This preliminary numerical result confirms our aforementioned analysis.

4.2.7 High Similarity of Task Key Vectors

We investigate the further details of conventional task ID prediction computed by cosine similarity between input x and task-key K as implemented by DualPrompt. After completing the 10-task learning on the ImageNet-R dataset, we evaluate the similarity between task keys. Table 4.2 shows the cosine similarity between K^t and $K^{t'}$ for all $t, t' \in [1..T]$. The tables show that a pair of different task keys has a high similarity. For example, except for task-3, task-1 has 0.8 to 0.96 cosine similarity to other task keys. During the inference step, this high similarity between different task keys causes a false task prediction that leads to the wrong selected prompt, unexpected feature value, and finally, misclassification.

This empirical analysis is one of our motivations to develop a better task ID prediction, i.e., by utilizing both task-wise key K^t and class-wise key L_c^t. Combining the similarity score of input x to both K^t and key L_c^t produces a more discriminative task prediction than using K^t only. Another way to solve this issue is by creating an uninstructed representation where a class may have several prototypes,

as used in HiDe-Prompt. However, this mechanism takes a longer time since the uninstructed representation optimization needs prototype augmentation and many iterations updates.

Table 4.2: Cosine similarity of task-key vectors after completing 10-task training on ImageNet-R by DualPrompt method.

	Task-1	**Task-2**	**Task-3**	**Task-4**	**Task-5**	**Task-6**	**Task-7**	**Task-8**	**Task-9**	**Task-10**
Task-1	1.000	0.961	0.728	0.857	0.932	0.901	0.866	0.827	0.809	0.877
Task-2	0.961	1.000	0.749	0.866	0.932	0.909	0.879	0.844	0.825	0.899
Task-3	0.728	0.749	1.000	0.873	0.738	0.785	0.725	0.672	0.660	0.687
Task-4	0.857	0.866	0.873	1.000	0.872	0.893	0.828	0.771	0.755	0.806
Task-5	0.932	0.932	0.738	0.872	1.000	0.921	0.893	0.850	0.843	0.908
Task-6	0.901	0.909	0.785	0.893	0.921	1.000	0.933	0.893	0.873	0.893
Task-7	0.866	0.879	0.725	0.828	0.893	0.933	1.000	0.937	0.920	0.907
Task-8	0.827	0.844	0.672	0.771	0.850	0.893	0.937	1.000	0.905	0.857
Task-9	0.809	0.825	0.660	0.755	0.843	0.873	0.920	0.905	1.000	0.872
Task-10	0.877	0.899	0.687	0.806	0.908	0.893	0.907	0.857	0.872	1.000

4.2.8 High Similarity of Language Embedding

We continue our deeper investigation on the encoded language embedding as implemented in previous SOTAs, i.e., LGCL. LGCL generates the language embedding by forming the string "The photo of class name", then patches it into the CLIP encoder. Surprisingly, prototypes from some classes have high similarity to the other classes' prototypes. For example, in class 'Great White Shark' has 0.9 similarities to the prototypes of class 'Tree Frog' and 'Iguana', despite those classes being totally different and not the sub-classes of the same superclass. On the contrary, the class 'Great White Shark' has lower similarity to the class 'Goldfish', despite both of them being species from the same super-category, i.e., fish.

In that case, we can not utilize the language embedding as a reference or anchor in the training process, as it may pull the adjusted (ViT) prototype closer to its value, while its value is highly similar to other classes' prototypes. Thus, in dealing with this challenge, we choose to use the language embedding as an input for prompt generation. In such a way, the trainable generators can produce a more discriminative prompt than the undiscriminating language embedding.

Table 4.3: Encoded Language Embedding similarity of the first 20 classes of ImageNet-R, encoded by CLIP encoder as implemented in LGCL.

	'Gold-fish'	'Great White Shark'	'Ham-mer-head'	'Sting-ray'	'Hen'	'Os-trich'	'Gold-finch'	'Junco'	'Bald Eagle'	'Vul-ture'	'Newt'	'Axo-lotl'	'Tree Frog'	'Igu-ana'	'Afri-can Chame-leon'	'Co-bra'	'Scor-pion'	'Taran-tula'	'Centi-pede'	'Pea-cock'
'Goldfish'	1.000	0.711	0.732	0.724	0.659	0.766	0.746	0.750	0.697	0.424	0.589	0.734	0.714	0.759	0.724	0.682	0.595	0.734	0.565	0.736
'Great White Shark'	0.711	1.000	0.916	0.608	0.760	0.719	0.848	0.887	0.744	0.538	0.719	0.901	0.904	0.896	0.654	0.668	0.751	0.824	0.577	0.882
'Hammer-head'	0.732	0.916	1.000	0.631	0.654	0.714	0.811	0.878	0.657	0.445	0.573	0.850	0.871	0.846	0.584	0.608	0.648	0.806	0.575	0.815
'Stingray'	0.724	0.608	0.631	1.000	0.616	0.676	0.652	0.693	0.680	0.425	0.579	0.669	0.655	0.651	0.644	0.635	0.606	0.674	0.617	0.680
'Hen'	0.659	0.760	0.654	0.616	1.000	0.693	0.798	0.807	0.858	0.780	0.927	0.794	0.805	0.875	0.613	0.682	0.858	0.713	0.589	0.886
'Ostrich'	0.766	0.719	0.714	0.676	0.693	1.000	0.626	0.736	0.742	0.425	0.614	0.752	0.665	0.734	0.646	0.689	0.671	0.742	0.621	0.696
'Goldfinch'	0.746	0.848	0.811	0.652	0.798	0.626	1.000	0.813	0.778	0.685	0.772	0.778	0.916	0.917	0.677	0.598	0.767	0.718	0.564	0.919
'Junco'	0.750	0.887	0.878	0.693	0.807	0.736	0.813	1.000	0.774	0.542	0.719	0.929	0.873	0.913	0.590	0.716	0.726	0.859	0.562	0.893
'Bald-Eagle'	0.697	0.744	0.657	0.680	0.858	0.742	0.778	0.774	1.000	0.669	0.832	0.765	0.794	0.821	0.639	0.690	0.805	0.742	0.665	0.856
'Vulture'	0.424	0.538	0.445	0.425	0.780	0.425	0.685	0.542	0.669	1.000	0.815	0.462	0.686	0.690	0.360	0.389	0.723	0.502	0.371	0.718
'Newt'	0.589	0.719	0.573	0.579	0.927	0.614	0.772	0.719	0.832	0.815	1.000	0.721	0.762	0.830	0.609	0.623	0.835	0.655	0.557	0.844
'Axolotl'	0.734	0.901	0.850	0.669	0.794	0.752	0.778	0.929	0.765	0.462	0.721	1.000	0.822	0.879	0.665	0.745	0.713	0.866	0.597	0.864
'Tree Frog'	0.714	0.904	0.871	0.655	0.805	0.665	0.916	0.873	0.794	0.686	0.762	0.822	1.000	0.922	0.584	0.623	0.786	0.753	0.554	0.926
'Iguana'	0.759	0.896	0.846	0.651	0.875	0.734	0.917	0.913	0.821	0.690	0.830	0.879	0.922	1.000	0.657	0.671	0.804	0.815	0.572	0.941
'African Chameleon'	0.724	0.654	0.584	0.644	0.613	0.646	0.677	0.590	0.639	0.360	0.609	0.665	0.584	0.657	1.000	0.629	0.594	0.585	0.648	0.645
'Cobra'	0.682	0.668	0.608	0.635	0.682	0.689	0.598	0.716	0.690	0.389	0.623	0.745	0.623	0.671	0.629	1.001	0.651	0.709	0.674	0.692
'Scorpion'	0.595	0.751	0.648	0.606	0.858	0.671	0.767	0.726	0.805	0.723	0.835	0.713	0.786	0.804	0.594	0.651	1.000	0.664	0.665	0.826
'Tarantula'	0.734	0.824	0.806	0.674	0.713	0.742	0.718	0.859	0.742	0.502	0.655	0.866	0.753	0.815	0.585	0.709	0.664	1.000	0.581	0.794
'Centipede'	0.565	0.577	0.575	0.617	0.589	0.621	0.564	0.562	0.665	0.371	0.557	0.597	0.554	0.572	0.648	0.674	0.665	0.581	1.000	0.619
'Peacock'	0.736	0.882	0.815	0.680	0.886	0.696	0.919	0.893	0.856	0.718	0.844	0.864	0.926	0.941	0.645	0.692	0.826	0.794	0.619	1.000

4.3 LEAPGen: Language as Prompt Generator

4.3.1 Overview

In this study, we propose a novel LanguagE As Prompt Generator (LEAPGen) accommodating our main principles that are emphasized in the introduction section. The structure and flow of LEAPGen are visualized in figure 4.2. LEAPGen generators produce prompts from the top-k selected embedding as input. LEAPGen also produces auxiliary (aux) data from the top-k embedding. The prompts are prepended into ViT MSAs while the aux is appended into input patches, thus producing feature and final prediction by ViT layers and MLP head, respectively. The top-k embedding is selected based on the cosine similarities between an input and the class-wise keys. LEAPGen limits the search space into task 1 to the predicted task t, by performing soft task ID prediction. In each task of the training phase, LEAPGen updates task-associated learnable parameters, i.e., generator, task-wise key, and class-wise keys. In the inference phase, LEAPGen selects the generators based on the predicted task t. LEAPGen utilizes cross-entropy loss and cosine similarity loss to optimize its parameters. Task-wise generators, language embedding for prompt generation, and soft task ID prediction are unique to recent SOTAs of evolving generator methods, e.g., [65] and [87], task-wise fixed prompt methods such as [85] and [86], and pool-based method [37] in terms of prompt

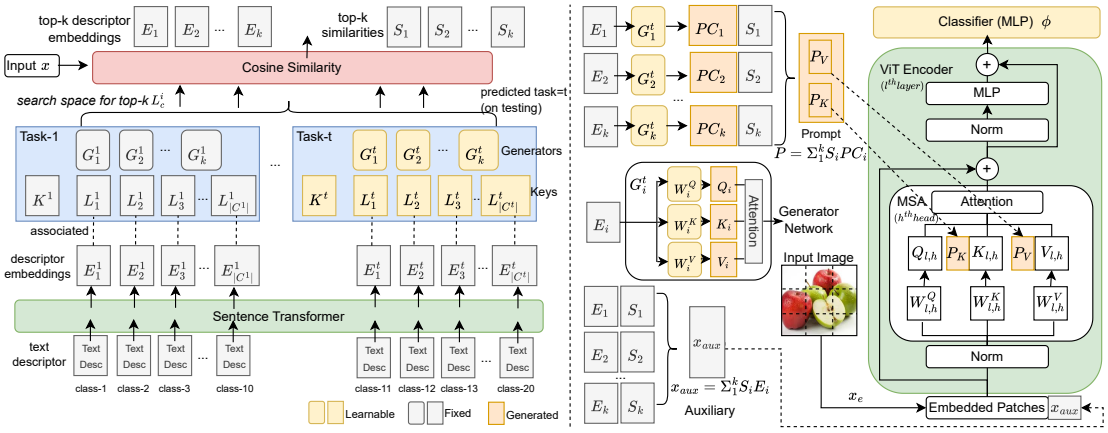


Figure 4.2: The LEAPGen architecture. Each task contains a set of generators, a task-wise learnable, and class-wise learnable parameters associated with the descriptor embedding. Given an input image, LEAPGen predicts the task ID and finds top-k descriptors matched to the input via cosine similarity afterward. The descriptors are dispatched to the generators producing the prompt components. LEAPGen produces the final prompt and an auxiliary embedding by the weighted sum of the prompt components and the descriptors, respectively. The prompt is prepended into the ViT MSA layer, while the auxiliary is appended into the input patches.

generation/selection, task-prediction mechanism, and modality for prompt generation. The detailed architecture, flow, and learning mechanism are presented in sub-section 4.3.2 and 4.3.3.

4.3.2 Architecture and Prompt Generation

4.3.2.1 Input and Learnable Parameters

While the existing works use the language to compute loss as in LGCL[60] or calculate the number of generators per task as in ConvPrompt [65], we take the benefit of language modality as the input for prompt generation. LEAPGen accommodates both language text, i.e., generated descriptors by GPT as in ConvPrompt and class names (as utilized in LGCL) as descriptors. Descriptors list carries text describing the visual attributes of the classes, e.g., the descriptors for class "apple" are ["round shape", "smooth, glossy skin", "red, green, or yellow color", "stem and leaves", "five-pointed star shape when cut in half", "white flesh inside"]. The descriptors are more distinctive and representative than the words "the photo

of apple" utilized in the existing works. The list of descriptors is then unified into a single long string and encoded into a numerical vector, i.e., $E_c^t \in \mathbb{R}^D$ called a descriptor embedding. We store the descriptor embedding from the learned classes as the catalog for prompt generation. Each task t contain k generators $\{G_i^t\}_{i=1}^k$, a task-wise learnable key $K^t \in \mathbb{R}^D$, and a set of class-wise learnable parameters $\{L_c^t\}_{c=1}^{|C^t|}, L_c^t \in \mathbb{R}^D$, where k is a predefined number, and $|C^t|$ is the number of classes on task t, each L_c^t is associated to E_c^t. These learnable parameters are tuned only on task-t.

4.3.2.2 Prompt Generation

Before generating the prompt for an input image x, LEAPGen predicts the task ID via a soft task prediction (ref. point c). In the training process, the task ID is easily given in a supervised way instead of being predicted. Let the task ID be t, then LEAPGen selects k descriptor embeddings associated with the top-k matched L by performing cosine similarity. Note that the search space is from the first task (task-1) until task-t, i.e., $\{E_c^1\}_{c=1}^{|C^1|}$, $\{E_c^2\}_{c=1}^{|C^2|}$,.., $\{E_c^t\}_{c=1}^{|C^t|}$. Let the k selected embedding be $\{E_i\}_{i=1}^k$ and associated similarity values $\{S_i\}_{i=1}^k$, then the embedding is inputted into the respective generators, i.e., $\{G_i^t\}_{i=1}^k$ producing prompt components $\{PC_i\}_{i=1}^k$. Note that each generator is a trainable network, so it satisfies $PC_i = G_i^t(E_i)$. The final prompt is generated by a weighted sum of the prompt components and similarity, i.e., $P = \Sigma_{i=1}^k S_i PC_i$. In a simplified expression, the prompt generation is formulated by equation 4.2.

$$P = \Sigma_{i=1}^k S_i G_i^t(E_i) \tag{4.2}$$

4.3.2.3 Generator Networks and Detailed Generation

We design a single-head self-attention (SSA) network G_i^t parameterized by W_i^Q, W_i^K, and W_i^V without bias as visualized in figure 4.2. Following [65] and prefix tuning [83], we implement G_i^t as a key-value pair generators, i.e., (G_i^{tK}, G_i^{tV}) and distribute them into corresponding layers and heads. Therefore, a hth head of MSA in l^{th} layer corresponds to a pair of generators $(G_{i,l,h}^{tK}, G_{i,l,h}^{tV})$ parameterized by $(WK_{i,l,h}^Q, WKV_{i,l,h}^K, WK_{i,l,h}^V)$ and $(WV_{i,l,h}^Q, WKV_{i,l,h}^K, WV_{i,l,h}^V)$, where $WK_{i,l,h}^Q$ is

$(D/H) \times (D/H)$ matrix and so do the other parameters. The language embedding E_i is then divided into H parts, i.e., $E_i = [E_{i,l,1}, E_{i,l,2}, ..E_{i,l,H}]$, where a part $E_{i,l,h}, h \in [1..H]$ plays as the input for $(G^{tK}_{i,l,h}, G^{tV}_{i,l,h})$. Thus, the detailed formula of prompt generation for respective head h on l^{th} layer is expressed as in equation 4.3.

$$P^K_{l,h} = \Sigma^k_{i=1} S_i G^{tK}_{i,l,h}(E_{i,l,h}), \quad P^V_{l,h} = \Sigma^k_{i=1} S_i G^{tV}_{i,l,h}(E_{i,l,h}) \tag{4.3}$$

4.3.2.4 Soft task ID Prediction

We design a soft task ID prediction to limit the search space to find top-k E_i. We use the term "soft" as the search for top-k E_i is conducted not only within predicted task t but also the previously learned tasks, i.e., $[1, 2, ..t-1]$. The reason is that most similar language embedding comes from the respective class from within respective tasks, the second-ranked and the rest may come from classes from different (previous) tasks. Therefore, they will contribute to a better distinctive prompt than the embedding from the other classes within the same task. For instance, class "pear" has higher similar descriptors to class "apple" than to class "palm tree". But class "apple" is located in a different (previous) task, while class "palm tree" is within the same task as class "pear", thus selecting "apple" descriptors from the previous task is better than choosing "palm tree" descriptors from the same task. Different from the task ID prediction in previous works [38], [85], we utilize both task-wise learnable parameter K^t and class-wise learnable parameter L^t_c. We predict the task ID t for an input x by finding the maximum product of cosine similarity between x and K^t and similarity between x and L^t_c as formulated in equation 4.4.

$$t = \underset{t \in [1..T], c \in C^t}{\arg\max} \{(x.K^t)(x.L^t_c)/((max(||x||_2.||K^t||_2, \epsilon))(max(||x||_2.||K^t||_2, \epsilon))\} \tag{4.4}$$

4.3.2.5 Learning with Auxiliary Data

LEAPGen generates auxiliary embedding based on k selected language embeddings used for prompt generation. The auxiliary embedding $x_{aux} \in \mathbb{R}^D$ helps LEAPGen to recognize the input image x as it is generated based on the similarity between the input and the catalog of language embedding. The previous study

[182] presents insightful knowledge of leveraging the auxiliary modality for deep learning training. The auxiliary embedding x_{aux} is generated by the weighted sum of language embedding, i.e., $x_{aux} = \Sigma_{i=1}^{k} s_i E_i$. The auxiliary embedding is concatenated into the input patches list; therefore, the patches list that enters the ViT layers now becomes $[x_{e_1}, x_{e_2}, ..., x_{e_{N_p}}, x_{aux}]$, where N_p is the number of patches.

4.3.3 Formulation of Learning Mechanism

4.3.3.1 Theoretical Foundation

The goal of CIL is to learn a sequence of tasks, i.e., $\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^T$, where each task $\mathcal{T}^t, t \in [1..T]$ carries a pair of image and label set $(\mathcal{X}^t, \mathcal{Y}^t)$. Suppose that C^t is the set of available classes in $\mathcal{Y}^t$, and $c \in C^t$ represents a class label c available in task t. Suppose that ":" denotes the concatenation and $\mathcal{T}^{1:t} = \cup_{i=1}^{t} \mathcal{T}^i$ denotes the concatenation of task 1 to task t. Similarly, $\mathcal{X}^{1:t}$ and $\mathcal{Y}^{1:t}$ denote the concatenation of input and label space, respectively, from task 1 to task t. Given a sequence of experienced events $\mathcal{T} = \{\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^T\}$, and a deep learning model $f_\theta(.)$. The objective of CIL is to recognize any input $x \in X^{1:T}$ by maximizing $P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta)$. Following [85], $P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta)$ can be decomposed into task prediction and class prediction within task as in equation 4.5.

$$P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta) = P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta) P(x \in \mathcal{X}^t | \mathcal{T}, \theta) \tag{4.5}$$

Analogically, we can decompose the $P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta)$ in a softer task ID prediction, i.e., by decomposing the term $P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta)$ w.r.t. $P(x \in \mathcal{X}^{1:t} | \mathcal{T}, \theta)$. Then, the decomposition can be derived into equation 4.6.

$$\begin{aligned} P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta) &= P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta) P(x \in \mathcal{X}^t | \mathcal{T}, \theta) \\ &= P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^{1:t}, \mathcal{T}, \theta) P(x \in \mathcal{X}^{1:t} | \mathcal{T}, \theta) \end{aligned} \tag{4.6}$$

The first line of equation 4.6 expresses a hard decomposition where a classification is conducted into a strict two-step classification, i.e., predict the task ID first, then predict the class label within the task. On the other hand, the second line of the equation 4.6 expresses a soft way of decomposed classification, i.e., predict the possible task (1:t) of the sample, then suffert predict the class within the task range. In the first approach, a false task prediction directly leads to misclassification, while

in the second approach, it still holds a possibility for correct classification as long as the predicted task ID is higher than the ground truth task. Since event $x \in \mathcal{X}^t$ is subset of event $x \in \mathcal{X}^{1:t}$, Then $P(x \in \mathcal{X}^t|\mathcal{T},\theta) \leq P(x \in \mathcal{X}^{1:t}|\mathcal{T},\theta)$, that implies $P(x \in \mathcal{X}_x^t|x \in \mathcal{X}^t) \geq P(x \in \mathcal{X}_x^t|x \in \mathcal{X}^{1:t})$. The second approach has an easier way to predict the possible task but a harder step afterward, while the first approach has a more difficult task prediction but an easier way afterward. However, the equation above gives us insight that improving $P(x \in \mathcal{X}_c^t|\mathcal{T},\theta)$ can be conducted by improving $P(x \in \mathcal{X}_c^t|x \in \mathcal{X}^t,\mathcal{T},\theta)$,$P(x \in \mathcal{X}^t|\mathcal{T},\theta)$, $P(x \in \mathcal{X}_c^t|x \in \mathcal{X}^{1:t},\mathcal{T},\theta)$, $P(x \in \mathcal{X}^{1:t}|\mathcal{T},\theta)$, pair to the first line, pair of the second line of equation 4.6, or all of them.

4.3.3.2 Learning Objective

Based on the insights above, we develop a strategy to improve $P(x \in \mathcal{X}_c^t|\mathcal{T},\theta)$ during the learning process. Note that ψ is the frozen ViT parameter, $\theta = \{G_i^t, K^t, L_c^t\}_{t=1,c\in C^t}^{t=T}$ are our customized trainable parameters including generators, task-wise keys, and language class-wise keys, and ϕ is the parameter of MLP head (classifier) and the class label of an input x is predicted by computing $h_\phi(f_{\psi,\theta}(x))$. First, utilizing cross-entropy function, we improve $P(x \in \mathcal{X}_c^t|x \in \mathcal{X}^t,\mathcal{T},\theta)$ by intra-task loss $\mathcal{L}_{intra}$ as formulated in equation 4.7.

$$\mathcal{L}_{intra}(x,(\psi,\theta,\phi)) = -\sum_{c\in C^t} log\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{c'\in C^t} exp(h_\phi(f_{\psi,\theta}(x))[c'])} \tag{4.7}$$

where "[]" denotes the index filter for the respective class. Note that $h_\phi(f_{\psi,\theta}(x))$ produces a list of softmax values for all classes. Second, similar to the intra-task loss, utilizing cross-entropy function, we improve $P(x \in \mathcal{X}_c^t|x \in \mathcal{X}^{1:t},\mathcal{T},\theta)$ by inter-task loss $\mathcal{L}_{inter}$ as defined in equation 4.8.

$$\mathcal{L}_{inter}(x,(\psi,\theta,\phi)) = -\sum_{t=1}^{t=T}\sum_{c\in C^t} log\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{t=1}^{t=T}\sum_{c'\in C^t} exp(h_\phi(f_{\psi,\theta}(x))[c'])} \tag{4.8}$$

Then, we improve task prediction $P(x \in \mathcal{X}^t|\mathcal{T},\theta)$ that leads to improve $P(x \in \mathcal{X}^{1:t}|\mathcal{T},\theta)$ by computing negative cosine similarity between the input x and task-wise and class wise learnable parameters K^t and L_c^t respectively as formulated in

equations 4.9 and 4.10. Note that our designed task ID predictor utilizes both K^t and L_c^t as explained in sub-section 4.1.c.

$$\mathcal{L}_t(x, K^t) = -(x.K^t)/(max(||x||_2.||K^t||_2, \epsilon)) \tag{4.9}$$

$$\mathcal{L}_c(x, L_c^t) = -(x.L_c^t)(max(||x||_2.||L_c^t||_2, \epsilon)) \tag{4.10}$$

Finally, we define a joint loss function as a total loss accommodating the four loss components above to train the learnable parameters, generators, and classifier head formulated in the equation 4.11. We add loss coefficients, i.e., $\lambda_1, \lambda_2, \lambda_3$, to enhance the flexibility of our proposed method, e.g., the higher number of learned classes needs higher capability on inter-task discrimination. We do not utilize any regularization, since all learned parameters on task-t will be frozen after finishing the task.

$$\mathcal{L}_{total} = \mathcal{L}_{intra}(x, (\psi, \theta, \phi)) + \lambda_1 \mathcal{L}_{inter}(x, (\psi, \theta, \phi)) + \lambda_2 \mathcal{L}_t(x, K^t) + \lambda_3 \mathcal{L}_c(x, L_c^t) \tag{4.11}$$

4.3.3.3 Algorithm

In this sub-section, we present the procedures of LEAPGen training and inference that are shown in algorithm 2 and 3, respectively.

4.4 Theoretical Analysis

We state Theorems 1, 2, and 3, which prove the importance of the generator network, loss minimization, and better task prediction, respectively.

Theorem 1: *Let f_ψ and h_ψ are the ViT encoder and classifier head respectively, ψ is frozen, $\theta = \{G_i^t, K^t, L_c^t\}_{t=1,c\in C^t}^{t=T}$ is the set of trainable parameters and generators, $\theta' = \{K^t, L_c^t\}_{t=1,c\in C^t}^{t=T}$ is the set of trainable parameters only, and $\mathcal{L}$ is loss function. Assume that G, f, h, and $\mathcal{L}$ are continues functions in $\mathbb{R}$, if G_i^t can produce identity output, then there is $\epsilon \geq 0$ that satisfies $\mathcal{L}(h_\phi(f_{\psi,\theta'}(x))) - \mathcal{L}(h_\phi(f_{\psi,\theta}(x))) = \epsilon$..*

Theorem 2: *Assuming a pre-trained ViT as the backbone, if the loss error $\mathcal{L} \leq \delta$ then exist $\mathcal{L}_{intra} \leq \delta$ and $\mathcal{L}_{inter} \leq \delta$.*

Theorem 3: *Let F and F' denote the task-predictor functions parameterized by $(\{K^t\}, \{L_c^t\})$ and $\{K^t\}$ respectively, and $cos(,)$ denote cosine similarity function.*

Algorithm 2 LEAPGen Training

Input: A sequence of tasks $\mathcal{T}^1$, $\mathcal{T}^2$,...,$\mathcal{T}^T$, a frozen L-layered (with H MSA heads/layer) pre-trained ViT $f_{\psi(.)}$, training epochs E, and batch size B.
for $t = 1 : T$ **do**
    Initiate trainable parameters for task t, $\theta^t = \{\{G_i^t\}_{i=1}^k, K^t, \{L_c^t\}_{c=1}^{|C^t|}\}$
    $\mathcal{B} \leftarrow$ Split $\mathcal{T}^t$ into B sized batches
    for $e = 1 : E$ **do**
        for $b = 1 : \mathcal{B}$ **do**
            Find top-k L_c^j where $j \in [1..t]$, $c \in [1..|C^j|]$
            Generate prompt $P = \{(P_{l,h}^K, P_{l,h}^V)\}_{l\in[1..L],h\in[1..H]}$ as in eq. 4.3
            Compute logits by forwarding the input, i.e., $h_\phi(f_{\psi,\theta}(x))$
            Compute $\mathcal{L}_{intra}$ as in eq. 4.7
            Compute $\mathcal{L}_{inter}$ as in eq. 4.8
            Compute $\mathcal{L}_t$ as in eq. 4.9
            Compute $\mathcal{L}_c$ as in eq. 4.10
            Compute $\mathcal{L}_{total}$ as in eq. 4.11
            Update parameters $\theta^t = \{\{G_i^t\}_{i=1}^k, K^t, \{L_c^t\}_{c=1}^{|C^t|}\}$ based on $\mathcal{L}_{total}$
        end for
    end for
end for
Output: Optimum parameters $\theta = \{theta^t\}_{t=1}^T = \{\{G_i^t\}_{i=1}^k, K^t, \{L_c^t\}_{c=1}^{|C^t|}\}_{t=1}^T$

Algorithm 3 LEAPGen Inference

Input: An input x, a frozen L-layered (with H MSA heads/layer) pre-trained ViT $f_{\psi(.)}$, and optimized parameters $\theta = \{theta^t\}_{t=1}^T = \{\{G_i^t\}_{i=1}^k, K^t, \{L_c^t\}_{c=1}^{|C^t|}\}_{t=1}^T$
Predict task ID t using soft task ID prediction as in eq. 4.4
Find top-k L_c^j where $j \in [1..t]$, $c \in [1..|C^j|]$
Generate prompt $P = \{(P_{l,h}^K, P_{l,h}^V)\}_{l\in[1..L],h\in[1..H]}$ as in eq. 4.3
Compute logits by forwarding the input, i.e., $logits = h_\phi(f_{\psi,\theta}(x))$
Compute Pedicted label $\hat{y} = argmax(logits)$
Output: Predicted label $\hat{y}$

If K^t *and* L_x^t *are trained well enough so that* $cos(x, K^t) \approx cos(x, L_c^t) \geq S$, *and* $cos(x, K^{t'}) \approx cos(x, L_{c'}^t) \approx cos(x, L_c^{t'}) \leq 2/3S$ *for any* $x \in \mathcal{X}_c^t$ *then it guarantees* $P(x \in \mathcal{X}^t | \mathcal{T}, \theta_F) \geq P(x \in \mathcal{X}^t | \mathcal{T}, \theta_{F'})$.

Theorem 1 theoretically shows that there is always a chance to achieve a smaller loss with generators than with descriptor embedding only. Theorem 2 shows that minimizing the loss error implies minimizing both inter-task and intra-task losses. Theorem 3 shows that if the learnable parameters are well-trained, then it guarantees our task predictor, as presented in equation 4.4, is more accurate than the existing task ID predictor (based on $\{K^t\}$ only).

Proof of Theorem 1: Recall that f_ψ is pre-trained ViT encoder with a frozen parameter ψ, h_ϕ is classifier head parameterized by ϕ (learnable) , $\theta = \{G_i^t, K^t, L_c^t\}_{t=1,c\in C^t}^{t=T}$ is the set of learnable parameters and generators,
$\theta' = \{K^t, L_c^t\}_{t=1,c\in C^t}^{t=T}$ is the set of learnable parameters only. Note that G_i^t produces a prompt component $PC_i = G_i^t(E_i)$. $\theta' = \{K^t, L_c^t\}_{t=1,c\in C^t}^{t=T}$ is the same as using identity function I as generator. $\mathcal{L}$ is the loss function that, in this case is cross-entropy. First we adopt the definition of continuity of $f(x)$ at a, i.e., for all $\eta_1 > 0$ there is some δ_1 such that:

$$|x - a| < \delta_1 \Rightarrow |f(x) - f(a)| < \eta_1 \tag{4.12}$$

Now, following the assumption that $f(.)$ is a continuous function in $\mathbb{R}$, now we substitute $f(.)$ with $f_{\psi,\theta}(.)$, substitute x with $G_i^t(E_i)$ and a with $I(E_i)$. Not that G_i^t is a trainable generator that can produce identity, function I as well as nearby vectors, i.e., $G_i^t(E_i)$. Also note that $f_{\psi,\theta,(G=I)}(.) = f_{\psi,\theta'}(.)$ Therefore, the equation 4.12 can be derived into:

$$|G_i^t(E_i) - I(E_i)| < \delta_1 \Rightarrow |f_{\psi,\theta}(x) - f_{\psi,\theta'}(x)| < \eta_1 \ . \tag{4.13}$$

Then, incorporating the continuity of $h(.)$ in $\mathbb{R}$ and the definition of contituinous function then we have

$$|f_{\psi,\theta}(x) - f_{\psi,\theta'}(x)| < \eta_1 \Rightarrow |h_\phi(f_{\psi,\theta}(x)) - h_\theta(f_{\psi,\theta'}(x))| < \eta_2 \ . \tag{4.14}$$

Analogically, applying the continuity of $\mathcal{L}(.)$ in $\mathbb{R}$, then we have

$$|h_\phi(f_{\psi,\theta}(x)) - h_\theta(f_{\psi,\theta'}(x))| < \eta_2 \Rightarrow |\mathcal{L}(h_\phi(f_{\psi,\theta}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta'}(x)))| < \eta_3 \tag{4.15}$$

Combining the equations 4.13, 4.14, 4.15, then we have

$$|G_i^t(E_i) - I(E_i)| < \delta_1 \Rightarrow |\mathcal{L}(h_\phi(f_{\psi,\theta}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta'}(x)))| < \eta_3 \ . \tag{4.16}$$

In the equation above, we can pick two constants $0 < \eta_3' < \eta_3$ and $0 < \delta_1' < \delta_1$ that satisfy:

$$|G_i^t(E_i) - I(E_i)| = \delta_1' \Rightarrow |\mathcal{L}(h_\phi(f_{\psi,\theta}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta'}(x)))| = \eta_3' \ . \tag{4.17}$$

The term $|\mathcal{L}(h_\phi(f_{\psi,\theta}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta'}(x)))| = \eta_3$ can be satisfied by two conditions: (1) $\mathcal{L}(h_\phi(f_{\psi,\theta'}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta}(x))) = \eta_3$ if $\mathcal{L}(h_\phi(f_{\psi,\theta'}(x))) > \mathcal{L}(h_\theta(f_{\psi,\theta}(x)))$. In this condition, then we can state the following statement:

$$\mathcal{L}(h_\phi(f_{\psi,\theta'}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta}(x))) > 0 \tag{4.18}$$

The other condition (2) $\mathcal{L}(h_\phi(f_{\psi,\theta}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta'}(x))) = \eta_3$ if $\mathcal{L}(h_\phi(f_{\psi,\theta}(x))) > \mathcal{L}(h_\theta(f_{\psi,\theta'}(x)))$. However, we didn't use this condition as the optimized G_i^t, instead, we select $G_i^t = I$ so that $\mathcal{L}(h_\phi(f_{\psi,\theta}(x))) = \mathcal{L}(h_\theta(f_{\psi,\theta'}(x)))$. The second condition shows the case where the best possible value of G_i^t is the identity function itself. In the second condition, then we can state

$$\mathcal{L}(h_\phi(f_{\psi,\theta'}(x))) - \mathcal{L}(h_\theta(f_{\psi,\theta}(x))) = 0 \tag{4.19}$$

Combining both inequality 4.18 and 4.19, then we prove **Theorem 1** that states : Assume that G, f, h, and $\mathcal{L}$ are continues functions in $\mathbb{R}$, if G_i^t can produce identity output, then there is $\epsilon \geq 0$ that satisfies $\mathcal{L}(h_\phi(f_{\psi,\theta'}(x))) - \mathcal{L}(h_\phi(f_{\psi,\theta}(x))) = \epsilon$.

Theorem 1 shows that a possibility to achieve better or at least similar learning performance with learnable generator networks than without them.

Proof of Theorem 2: Given a model with loss error $\mathcal{L} \leq \delta$, then

$$-\sum_{t=1}^{t=T}\sum_{c\in C^t} log\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{t=1}^{t=T}\sum_{c'\in C^t} exp(h_\phi(f_{\psi,\theta}(x))[c'])} \leq \delta \tag{4.20}$$

$$\mathcal{L}_{inter}(x,(\psi,\theta,\phi)) = -\sum_{t=1}^{t=T}\sum_{c\in C^t} log\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{t=1}^{t=T}\sum_{c'\in C^t} exp(h_\phi(f_{\psi,\theta}(x))[c'])} \leq \delta \tag{4.21}$$

Or simply

$$\mathcal{L}_{inter}(x,(\psi,\theta,\phi)) \leq \delta \tag{4.22}$$

From equation 4.5 We get intra-task probability as

$$P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta) = P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta) / P(x \in \mathcal{X}^t | \mathcal{T}, \theta) \tag{4.23}$$

Since $P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta)$, $P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta)$, and $P(x \in \mathcal{X}^t | \mathcal{T}, \theta)$ are in the range of $(0, 1)$ then we have:

$$P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta) \geq P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta) \tag{4.24}$$

Applying log to both sides, then we have

$$log(P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta)) \geq log(P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta)) \tag{4.25}$$

Multiplying both sides with -1, then we have

$$-log(P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta)) \leq -log(P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta)) \tag{4.26}$$

Let inter-task probability and intra-task probability be represented as:

$$\begin{aligned} P(x \in \mathcal{X}_c^t | \mathcal{T}, \theta) &= \sum_{t=1}^{t=T}\sum_{c\in C^t} \frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{t=1}^{t=T}\sum_{c'\in C^t} exp(h_\phi(f_{\psi,\theta}(x))[c'])} \\ P(x \in \mathcal{X}_c^t | x \in \mathcal{X}^t, \mathcal{T}, \theta) &= \sum_{c\in C^t} \frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{c'\in C^t} exp(h_\phi(f_{\psi,\theta}(x))[c'])} \end{aligned} \tag{4.27}$$

Substituting equations 4.27 into 4.26 and combining equation 4.22, then we have:

$$-log(\sum_{c\in C^t}\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{c'\in C^t}exp(h_\phi(f_{\psi,\theta}(x))[c'])}) \leq -log(\sum_{t=1}^{t=T}\sum_{c\in C^t}\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{t=1}^{t=T}\sum_{c'\in C^t}exp(h_\phi(f_{\psi,\theta}(x))[c'])}) \leq \delta \quad (4.28)$$

$$-\sum_{c\in C^t}log\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{c'\in C^t}exp(h_\phi(f_{\psi,\theta}(x))[c'])} \leq \sum_{t=1}^{t=T}\sum_{c\in C^t}log\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{t=1}^{t=T}\sum_{c'\in C^t}exp(h_\phi(f_{\psi,\theta}(x))[c'])} \leq \delta \quad (4.29)$$

Therefore, we have:

$$\mathcal{L}_{intra}(x,(\psi,\theta,\phi)) = -\sum_{c\in C^t}log\frac{exp(h_\phi(f_{\psi,\theta}(x))[c])}{\sum_{c'\in C^t}exp(h_\phi(f_{\psi,\theta}(x))[c'])} \leq \delta \quad (4.30)$$

or in the simplified form

$$\mathcal{L}_{intra}(x,(\psi,\theta,\phi)) \leq \delta \quad (4.31)$$

The explanation above proves Theorem 2 that assumes a pre-trained ViT as the backbone, if the loss error $\mathcal{L} \leq \delta$. then there are exist $\mathcal{L}_{intra} \leq \delta$ and $\mathcal{L}_{inter} \leq \delta$.

Proof of Theorem 3: Let F and F' denote the task-predictor functions parameterized by $(\{K^t\}, \{L_c^t\})$ and $\{K^t\}$ respectively therefore, F is presented in equation 4.4 and simplified into equation 4.32 while F' is presented in equation 4.33.

$$F(x) = t = \underset{t\in[1..T],c\in C^t}{\arg\max}\{cos(x,K^t)cos(x,L_c^t)\} \quad (4.32)$$

$$F'(x) = t = \underset{t\in[1..T],}{\arg\max}\{(x.K^t)/(max(||x||_2.||K^t||_2,\epsilon)\} = \underset{t\in[1..T],}{\arg\max}\{cos(x,K^t)\} \quad (4.33)$$

Let $cos(.)$ denote the cosine similarity function, $\mathcal{L}$ denotes similarity loss, i.e., the difference between the right and wrong representation. Therefore, for any $x \in \mathcal{X}_c^t$ then we have

$$\begin{aligned}\mathcal{L}(F(x)) = cos(x,K^t)cos(x,L_c^t) &- \frac{1}{(T-1)|C^{t'}|}\sum_{t'\neq t}\sum_{c\in C^{t'}}cos(x,K^{t'})cos(x,L_c^{t'}) \\ &- \frac{1}{(|C^t|-1)}\sum_{c'\neq c\in C^t}cos(x,K^t)cos(x,L_{c'}^t)\end{aligned} \quad (4.34)$$

$$\mathcal{L}(F'(x)) = cos(x, K^t) - \frac{1}{T-1}\sum_{t' \neq t} cos(x, K^{t'}) \tag{4.35}$$

Thus minimizing $\mathcal{L}(F(x))$ and $\mathcal{L}F'(x)$ imply maximizing F and F' respectively. Suppose that K^t and L_x^t are trained well enough so that for any $x \in \mathcal{X}_c^t$, it satisfy $cos(x, K^t) \approx S_1$, and $cos(x, K^{t'}) \approx S_a = S_1 - \epsilon_1$, $cos(x, L_c^t) \approx S_2$ and $cos(x, L_{c' \neq c}^t) \approx cos(x, L_c^{t'}) \approx S_b = S_2 - \epsilon_2$, where S_1, S_2, S_a, S_b, ϵ_1, and ϵ_2 have positive values. Then equations 4.34 can be derived into

$$\mathcal{L}(F(x)) = S_1 S_2 - \frac{1}{(T-1)|C^{t'}|}\sum_{t' \neq t}\sum_{c \in C^{t'}} S_a S_b - \frac{1}{(|C^t|-1)}\sum_{c' \neq c \in C^t} S_1 S_b \tag{4.36}$$

$$\mathcal{L}(F(x)) = S_1 S_2 - S_a S_b - S_1 S_b \tag{4.37}$$

and 4.35 can be derived into

$$\mathcal{L}(F'(x)) = S_1 - \frac{1}{T-1}\sum_{t' \neq t} S_a \tag{4.38}$$

$$\mathcal{L}(F'(x)) = S_1 - S_a \tag{4.39}$$

Now, assuming that $S_1 \approx S_2 \approx S$ and $\epsilon_1 \approx \epsilon_2 \approx \epsilon$, then we have

$$\mathcal{L}(F(x)) = S^2 - (S-\epsilon)(S-\epsilon) - S(S-\epsilon) \tag{4.40}$$

$$\mathcal{L}(F(x)) = S^2 - (S^2 - 2S\epsilon + \epsilon^2) - (S^2 - S\epsilon) \tag{4.41}$$

$$\mathcal{L}(F(x)) = S^2 - S^2 + 2S\epsilon - \epsilon^2 - S^2 + S\epsilon \tag{4.42}$$

$$\mathcal{L}(F(x)) = 3S\epsilon - S^2 - \epsilon^2 \tag{4.43}$$

and we have,

$$\mathcal{L}(F'(x)) = S_1 - S_a = S - (S-\epsilon) = \epsilon \tag{4.44}$$

Suppose that we want to find a condition where $\mathcal{L}(F(x))\mathcal{L} \leq (F'(x))$, then we have

$$3S\epsilon - (S^2 + \epsilon^2) \leq \epsilon \tag{4.45}$$

$$3S\epsilon - S^2 - \epsilon^2 \leq \epsilon \tag{4.46}$$

Note that $\epsilon < S$. Then we can choose $\epsilon = 1/3S$, so the equation above becomes

$$3S(1/3)S - S^2 - \epsilon^2 \leq \epsilon \tag{4.47}$$

$$S^2 - S^2 - \epsilon^2 \leq \epsilon \tag{4.48}$$

$$-\epsilon^2 \leq \epsilon \tag{4.49}$$

that always true for any $\epsilon > 0$. Note that $\epsilon = (1/3)S$ means $S_a = S - \epsilon = S - (1/3)S = (2/3)S$. Choosing $\epsilon < (1/3)S$ also satisfies the inequality above. The derivation above proves Theorem 3:i.e. let F and F' denote the task-predictor functions parameterized by $(\{K^t\}, \{L_c^t\})$ and $\{K^t\}$ respectively, and $cos(,)$ denote cosine similarity function. If K^t and L_x^t are trained well enough so that $cos(x, K^t) \approx cos(x, L_c^t) \geq S$, and $cos(x, K^{t'}) \approx cos(x, L_{c'}^t) \approx cos(x, L_c^{t'}) \leq 2/3S$ for any $x \in \mathcal{X}_c^t$ then it guarantees $\mathcal{L}(F(x)) \leq \mathcal{L}(F'(x))$ that implies $P(x \in \mathcal{X}^t | \mathcal{T}, \theta_F) \geq P(x \in \mathcal{X}^t | \mathcal{T}, \theta_{F'})$.

4.5 Experiment Results and Analysis

4.5.1 Datasets

We evaluate LEAPGen in three CIL benchmarks, i.e., CIFAR100[172], ImageNet-R[183], and CUB[184]. The CIFAR100 has 100 classes of small-scale images, while ImageNet-R contains large-scale images that cover 200 classes. ImageNet-R contains art, cartoon, paint, and deviant images. The CUB dataset includes fine-grained images of 200 classes of birds. We follow 5, 10, and 20 task splits as in the SOTAs [65], [86], [87].

4.5.2 Baseline Methods

We compare our proposed method with the existing SOTAs of prompt-based approaches, i.e., L2P[37], DualPrompt[38], CODA-P[39], LGCL[60], PGP[185], HiDE-Prompt[85], EvoPrompt[87], CPrompt[86], and ConvPrompt[65]. We also compare LEAPGen to the Low-Rank Adaptation (LoRA) approach, i.e., CLoRA[88], LAE[90], InfLoRA[89], the slow learner approach, i.e., SLCA[186], and the generative approach, i.e., GMM[181].

4.5.3 Experiment Setup:

Performance Metrics: Adapted from HidePrompt, we measure both accuracy and forgetting of the methods. Suppose that $A_{i,t}$ denotes the accuracy on the t-th task after learning the t-th task. The average accuracy of all learned tasks is defined as $AA_t = (1/t)\Sigma_{i=1}^{t} A_{i,t}$. Suppose that T is the number of all tasks, we measure final average accuracy (FAA), cumulative average accuracy(CAA), final forgetting measure (FFM), and cumulative forgetting measure (CFM) as defined in the equations below.

$$FAA = AA_T \tag{4.50}$$

$$CAA = \frac{1}{T}\Sigma_{t=1}^{T} AA_t \tag{4.51}$$

$$FFM = \frac{1}{T-1}\Sigma_{i=1}^{T-1} max_{t\in 1,\ldots,T-1}(AA_{i,t} - AA_{i,T}) \tag{4.52}$$

$$CFM = \frac{1}{T-1}\frac{1}{T-1}\Sigma_{j=2}^{T}\Sigma_{i=1}^{j-1} max_{t\in 1,\ldots,j-1}(AA_{i,t} - AA_{i,j}) \tag{4.53}$$

Implementation Details: We implemented LEAPGen and its lite version, i.e., LEAPGen-lite, in Pytorch utilizing the pre-trained ViT-B/16 backbone following the common practice. LEAPGen-lite utilizes lightweight Conv1d generators (note: different from ConvPrompt) without descriptors; it uses class names instead. All the methods are run under the same environment, i.e., a single NVIDIA A100 GPU with 40 GB RAM.

Existing Methods: The existing SOTAs are run by executing the official implementation (code) of the respective methods. The hyperparameter setting is chosen based on the official setting. HiDe-Prompt utilizes S-Prompt[84], i.e., similar to DualPrompt but without the global (task-shared) prompt. LGCL and PGP do not propose a specific prompt structure but rather utilize L2P and DualPrompt. The reported results for PGP and LGCL are obtained with DualPrompt structure, which is the best of their result. The other methods, i.e., L2P, DualPrompt, CODA-P, EvoPrompt, CPrompt, and ConvPrompt utilize their proposed structure. All the evaluated methods utilize ViT B/16 pre-trained on ImageNet 21K as the backbone model. LGCL utilizes a pre-trained CLIP text encoder, while ConvPrompt utilizes SentenceTransformer pre-trained on BERT as its text encoder.

LEAPGen and LEAPGen-lite: Our proposed methods are implemented on top of the ViT backbone pre-trained on ImageNet-21K. The prompt structure is as defined in section 4. The prompt length is set to 30, and the prefix tuning layer is set to 7, i.e., [0,1,2,3,4,5,6] for all main experiments. We utilize Adam optimizer with a cosine learning rate scheduler. For CIFAR100 dataset, we set 0.01 initial learning rate and 3, 10, and 10 epochs for 5-task, 10-task, and 20-task settings, respectively. For ImageNet-R dataset, we choose 5, 10, and 20 epochs for 5-task, 10-task, and 20-task settings, respectively. The initial learning rate is chosen from the best of 0.04,0.05,0.06. For CUB dataset, we choose 20 epochs and 0.005 initial learning rate. LEAPGen accommodates a flat learning rate (not decayed) for K^t in a different optimizer; the soft task ID predictor is computed in a batch-wise manner for convenience. Similar to ConvPrompt, we utilize SentenceTransformer as the text encoder. The pre-trained models, i.e., ViT, SentenceTransformer, and CLIP(LGCL) parameters are frozen (not fine-tuned) following ConvPrompt [65] implementation. The λ_2 and λ_3 are set to 1.0, while λ_1 is set to 1.0 for CIFAR100, and 0.1 for CUB and ImageNet-R. The number of generators per task that is the same as top-k descriptors is set to 3. All the consolidated methods are run under the same machine and computing environment, i.e., a single NVIDIA A100 GPU with 40 GB memory, Python 3.8, and Pytorch 2.2.0.

4.5.4 Overall Performance

Table 4.4 shows the summarized numerical results of consolidated algorithms in the split CIFAR100 and ImageNet-R dataset, while Table 4.5 shows the summarized results in the split CUB dataset. In the split CIFAR100 dataset, our proposed method (LEAPGen) achieves the highest accuracy with a 5-16% margin of FAA and a 5-11% margin of CAA. The table shows that a higher number of tasks results in a higher gap in performance. For the forgetting measure, LEAPGen consistently achieves the lowest FFM and CFM in all settings of CIFAR100. In addition, the magnitude of its FFM and CFM is lower than 0.7%, while the best-of-competitor method, i.e., HiDE-Prompt, suffers more than 1.5 %, and more than 2% for the rest of the SOTAs. Note that HiDe-Prompt failed to finish the experiment in the 20 tasks setting, and the result is obtained from its first 10 tasks. However, LEAPGen

Table 4.4: Summarized numerical result on split CIFAR100 and ImageNet-R in 5, 10, and 20 tasks settings, * and ∘ denote the results obtained from 1 seeded run and the first 10 tasks, respectively due to system crash.

Method	Split CIFAR100				Split ImageNet-R			
	FAA	CAA	FFM	CFM	FAA	CAA	FFM	CFM
	5 Task @20 classes/task				**5 Task @40 classes/task**			
L2P	84.77 ± 0.48	88.67 ± 0.30	6.18 ± 0.57	5.99 ± 0.29	64.62 ± 0.32	68.01 ± 0.42	3.94 ± 0.16	3.55 ± 0.20
DualPrompt	86.41 ± 0.21	89.95 ± 0.10	5.37 ± 0.21	4.77 ± 0.46	69.71 ± 0.11	72.78 ± 0.14	3.32 ± 0.16	2.78 ± 0.25
CODA-P	88.22 ± 1.06	92.25 ± 1.28	7.05 ± 2.18	6.06 ± 2.66	74.89 ± 0.36	79.71 ± 1.27	8.89 ± 0.65	7.65 ± 0.98
LGCL	86.90 ± 0.40	90.45 ± 0.18	5.01 ± 0.35	4.36 ± 0.13	69.93 ± 0.21	72.91 ± 0.19	3.04 ± 0.36	2.50 ± 0.38
HiDe-Prompt	91.99 ± 0.03	93.95 ± 0.09	2.52 ± 0.18	2.33 ± 0.15	75.40 ± 0.27	78.88 ± 0.04	3.15 ± 0.46	2.64 ± 0.16
PGP	87.69 ± 0.06	91.26 ± 0.13	5.32 ± 0.18	4.60 ± 0.15	69.71 ± 0.15	72.77 ± 0.07	3.36 ± 0.23	2.85 ± 0.25
EvoPrompt	89.07 ± 0.38	92.32 ± 0.26	5.25 ± 0.65	5.39 ± 0.24	77.27 ± 0.40	81.67 ± 0.18	1.79 ± 0.31	1.41 ± 0.32
CPrompt	89.22 ± 0.05	93.09 ± 0.06	5.02 ± 0.17	4.31 ± 0.35	78.65 ± 0.00*	82.44 ± 0.00	6.00 ± 0.00	5.49 ± 0.00
ConvPrompt	90.26 ± 0.44	93.49 ± 0.19	3.64 ± 0.28	3.25 ± 0.16	79.36 ± 0.08	82.93 ± 0.24	3.42 ± 0.05	2.36 ± 0.16
LEAPGen-lite	**97.07 ± 0.08**	**97.28 ± 0.10**	**0.05 ± 0.01**	**0.02 ± 0.01**	**82.44 ± 0.63**	**84.37 ± 0.90**	**0.43 ± 0.08**	**0.17 ± 0.06**
LEAPGen	**96.84 ± 0.12**	**96.85 ± 0.26**	**0.08 ± 0.07**	**0.06 ± 0.04**	**82.79 ± 0.32**	**85.06 ± 0.29**	**0.51 ± 0.04**	**0.18 ± 0.07**
	10 Task @10 classes/task				**10 Task @20 classes/task**			
L2P	83.84 ± 0.32	88.67 ± 0.16	6.55 ± 0.34	5.16 ± 0.14	62.50 ± 0.51	67.05 ± 0.47	5.01 ± 0.40	4.41 ± 0.43
DualPrompt	85.36 ± 0.20	89.77 ± 0.20	5.41 ± 0.33	4.33 ± 0.15	68.59 ± 0.24	72.18 ± 0.20	4.61 ± 0.07	3.70 ± 0.18
CODA-P	86.44 ± 0.16	91.27 ± 0.56	6.38 ± 1.46	5.09 ± 1.19	73.77 ± 0.50	79.38 ± 1.48	7.94 ± 0.08	6.72 ± 0.79
LGCL	85.68 ± 0.43	90.16 ± 0.29	5.46 ± 0.22	4.25 ± 0.32	68.65 ± 0.25	72.57 ± 0.19	4.75 ± 0.33	3.38 ± 0.58
HiDe-Prompt	92.89 ± 0.11	95.01 ± 0.08	1.98 ± 0.05	1.56 ± 0.15	75.75 ± 0.40	79.27 ± 0.17	2.29 ± 0.27	2.33 ± 0.17
PGP	86.36 ± 0.19	90.83 ± 0.17	5.49 ± 0.35	4.28 ± 0.27	68.62 ± 0.14	72.19 ± 0.20	4.53 ± 0.40	3.63 ± 0.35
EvoPrompt	88.17 ± 0.51	92.18 ± 0.49	5.39 ± 0.45	3.97 ± 0.73	76.00 ± 0.26	80.97 ± 0.30	4.22 ± 0.42	3.59 ± 0.52
CPrompt	86.92 ± 1.04	91.73 ± 0.66	5.43 ± 0.74	4.01 ± 0.81	76.32 ± 0.53	81.50 ± 0.30	6.10 ± 0.75	5.60 ± 1.35
ConvPrompt	88.77 ± 0.24	92.71 ± 0.04	4.12 ± 0.44	2.67 ± 0.11	77.08 ± 0.26	81.47 ± 0.10	4.17 ± 0.04	3.11 ± 0.17
LEAPGen-lite	**98.58 ± 0.03**	**98.69 ± 0.10**	**0.11 ± 0.03**	**0.06 ± 0.03**	**82.38 ± 1.04**	**85.14 ± 0.52**	**3.01 ± 1.19**	**2.13 ± 0.60**
LEAPGen	**98.38 ± 0.15**	**98.15 ± 0.39**	**0.10 ± 0.03**	**0.05 ± 0.00**	**84.09 ± 0.93**	**85.54 ± 0.65**	**1.46 ± 1.25**	**2.11 ± 1.21**
	20 Task @5 classes/task				**20 Task @10 classes/task**			
L2P	81.89 ± 0.38	87.16 ± 0.33	8.81 ± 0.10	6.79 ± 0.33	57.40 ± 0.31	63.33 ± 0.21	10.76 ± 0.45	7.88 ± 0.17
DualPrompt	82.32 ± 0.22	87.47 ± 0.24	6.88 ± 0.35	5.63 ± 0.23	65.19 ± 0.17	70.31 ± 0.29	7.30 ± 0.18	5.16 ± 0.34
CODA-P	81.29 ± 0.16	87.72 ± 0.44	6.82 ± 1.60	4.98 ± 0.95	70.55 ± 0.71	77.08 ± 1.02	8.23 ± 0.86	6.95 ± 0.70
LGCL	83.18 ± 0.40	88.63 ± 0.18	7.22 ± 0.47	4.91 ± 0.56	64.96 ± 0.67	70.18 ± 0.37	7.35 ± 0.65	5.05 ± 0.32
HiDe-Prompt∘	-	97.62 ± 0.14	-	0.74 ± 0.03	-	81.60 ± 0.48	-	2.23 ± 0.38
PGP	83.41 ± 0.35	89.23 ± 0.13	7.95 ± 0.23	5.66 ± 0.22	65.24 ± 0.25	70.36 ± 0.26	7.17 ± 0.21	5.09 ± 0.25
EvoPrompt	84.63 ± 0.21	89.47 ± 0.21	9.19 ± 0.41	7.39 ± 0.65	74.93 ± 0.64	79.92 ± 0.13	6.72 ± 0.90	5.67 ± 0.26
CPrompt	83.60 ± 0.00*	90.10 ± 0.00	6.47 ± 0.00	4.78 ± 0.00	74.23 ± 0.17	79.82 ± 0.51	5.98 ± 0.24	5.54 ± 0.48
ConvPrompt	87.21 ± 0.20	91.60 ± 0.36	5.47 ± 0.33	3.92 ± 0.33	73.93 ± 0.36	78.92 ± 0.37	4.87 ± 0.57	3.57 ± 0.25
LEAPGen-lite	**95.28 ± 3.37**	**98.38 ± 1.13**	**1.08 ± 1.54**	**0.70 ± 0.99**	**83.67 ± 0.39**	**85.65 ± 0.33**	**1.06 ± 0.24**	**0.47 ± 0.14**
LEAPGen	**96.51 ± 2.16**	**98.73 ± 0.26**	**0.66 ± 0.75**	**0.52 ± 0.39**	**87.03 ± 0.12**	**87.81 ± 0.48**	**2.17 ± 0.17**	**2.54 ± 0.77**

still achieves higher performance in the setting by approximately 1% margin. In Imagenet-R dataset, the proposed method outperforms the existing methods with a higher margin, i.e., 3-30% for FAA and 2-24% for CAA. LEAPGen also attains the lowest forgetting in both FFM and CFM with $\geq 1\%$ margin to the competitors. Similar to CIFAR100 dataset, the difference in accuracy and forgetting is higher for a higher number of tasks. In CUB dataset, LEAPGen keeps its superiority over the previous methods. For the final performance, i.e., FAA and FFM, LEAPGen outperforms the SOTAs with 1-21% and 0.6-10% respectively, while in terms of cumulative performance, i.e., CAA and CFM, LEAPGen gains 3-16% and 1.4-8.5% respectively.

Table 4.5: Summarized result on split CUB (10-tasks).

Method	Split CUB 10 Tasks @20 classes/task			
	FAA	**CAA**	**FFM**	**CFM**
L2P	66.95 ± 0.13	74.03 ± 0.32	5.18 ± 0.11	7.39 ± 0.23
DualPrompt	73.95 ± 0.73	80.29 ± 0.15	7.87 ± 0.76	8.60 ± 0.37
CODA-P	72.99 ± 0.30	82.64 ± 0.64	11.71 ± 1.49	9.91 ± 1.04
LGCL	79.93 ± 0.30	83.07 ± 0.23	5.45 ± 0.33	5.58 ± 0.39
HiDe-Prompt	87.21 ± 0.18	87.66 ± 0.01	1.90 ± 0.45	2.66 ± 0.13
PGP	78.35 ± 0.68	82.21 ± 0.56	5.76 ± 0.10	6.10 ± 0.26
EvoPrompt	76.23 ± 0.51	81.00 ± 0.40	3.96 ± 0.43	3.57 ± 0.83
CPrompt	77.14 ± 1.16	85.67 ± 0.56	11.65 ± 0.47	8.69 ± 0.31
ConvPrompt	80.12 ± 1.37	84.70 ± 0.64	6.04 ± 0.97	4.61 ± 0.45
LEAPGen-lite	**86.00 ± 0.32**	**88.03 ± 0.47**	**2.46 ± 1.61**	**2.29 ± 1.15**
LEAPGen	**88.45 ± 0.58**	**90.90 ± 0.81**	**1.32 ± 0.52**	**1.29 ± 0.82**

Table 4.6: Comparison of the proposed method to the LoRA approach, SLCA, and GMM. All the results of the competitor methods are taken from the respective references.

Method	**S-ImageNet-R 5 Tasks**		**S-ImageNet-R 10 Tasks**		**S-ImageNet-R 20 Tasks**		**S-CIFAR100 10 Tasks**	
	FAA	**CAA**	**FAA**	**CAA**	**FAA**	**CAA**	**FAA**	**CAA**
C-LoRA	75.85 ± 0.31	78.85 ± 0.34	71.89 ± 0.45	75.33 ± 0.28	65.71 ± 0.60	70.63 ± 0.85	82.97 ± 0.47	87.06 ± 0.25
LAE	73.84 ± 0.14	77.29 ± 0.45	71.70 ± 0.39	76.71 ± 0.10	66.98 ± 0.35	73.72 ± 0.05	88.81 ± 0.34	91.59 ± 0.13
InfLoRA	77.52 ± 0.37	82.01 ± 0.12	75.65 ± 0.14	80.82 ± 0.24	71.01 ± 0.45	77.28 ± 0.45	89.84 ± 0.03	91.70 ± 0.32
SLCA	-	-	77.00 ± 0.33	81.17 ± 0.64	-	-	91.53 ± 0.28	94.09 ± 0.87
GMM	-	-	80.72	-	-	-	87.59	-
LEAPGen-lite	**82.44 ± 0.63**	**84.37 ± 0.90**	**82.38 ± 1.04**	**85.14 ± 0.52**	**83.67 ± 0.39**	**85.65 ± 0.33**	**98.58 ± 0.03**	**98.69 ± 0.10**
LEAPGen	**82.79 ± 0.32**	**85.06 ± 0.29**	**84.09 ± 0.93**	**85.54 ± 0.65**	**87.03 ± 0.12**	**87.81 ± 0.48**	**98.38 ± 0.15**	**98.15 ± 0.39**

4.5.5 Comparison to Non-Prompting Approaches

Table 4.6 compares the performance of LEAPGen to SOTAs of different approaches, i.e., LoRA, slow learner (SLCA), and generative (GMM) approaches. Compared to LoRA approach, LEAPGen achieves a better performance with a significant margin, i.e., 5-22% and 3-17% for FAA and CAA, respectively. These margins are even higher than the margins of the prompt-based approach. SLCA achieves a promising performance as its performance is on par with ConvPrompt and Hide-Prompt in ImageNet-R 10 tasks and CIFAR100 10 tasks settings, respectively. However, LEAPGen is still higher performance than SLCA with 4-7% margin. LEAPGen also achieves higher performance than the GMM method with 3-10% margin.

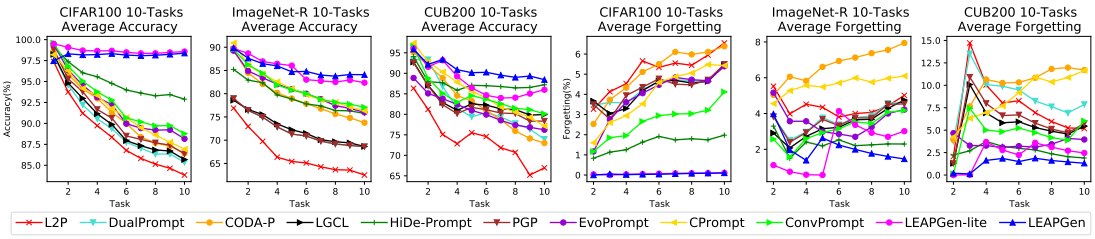


Figure 4.3: Historical performance plot of the consolidated methods in 3 CIL benchmark datasets.

4.5.6 Historical Performance

4.5.6.1 On 10-Task Setting

Figure 4.3 shows the historical average accuracy, i.e., the final average accuracy of all learned classes after finishing learning on each task t. LEAPGen's gentle slope shows that our proposed method experiences the least performance degradation compared to the existing methods. In the CIFAR100 dataset, LEAPGen even manages to increase its performance in the later tasks. The existing methods suffer from a significant performance drop, i.e., up to 13%, 15%, 22% in CIFAR100, ImageNet-R, and CUB datasets, respectively. For the average forgetting measure, LEAPGen once again demonstrates an excellent trend. Except on the first 5 tasks of ImageNet-R datasets, LEAPGen suffers from the lowest average forgetting in all tasks and all datasets. In CIFAR100 datasets, LEAPGen experiences nearly zero forgetting from the first until the last task. In Imagenet-R and CUB datasets, LEAPGen manages to decrease its average forgetting in the 5 latest tasks, and consistently achieves the lowest magnitude, i.e., $1.4 - 2.2\%$ and $1.2 - 1.8$ in CUB dataset. On the contrary, the competitor methods suffer from the increasing average forgetting during incremental learning, as shown by the rising curves. The methods suffer from a high average forgetting, i.e., for up to 7%, 8%, and 15% in CIFAR100, ImageNet-R, and CUB datasets, respectively. Their increase of average forgetting is significant, i.e., 4%, 3%, and 8%, respectively. Overall, the historical analysis confirms how LEAPGen handles catastrophic forgetting better than the existing methods.

4.5.6.2 On 5-Task and 20-Task Settings

We continue our analysis of the historical performance of consolidated methods in CIFAR100 and ImageNet-R with 5 and 20-task settings. Figures 4.4 and 4.5 plot the historical performance of the consolidated methods. The figures show that, both in the 5-task and 20-task settings, both in CIFAR100 and ImageNet-R datasets, LEAPGen consistently outperforms the existing methods. Figure 4.4 shows that in the 5-task setting, LEAPGen manages to maintain its performance in CIFAR100 in all tasks, and suffers from less than 4% accuracy degradation in Imagent-R. The figure also shows that LEAPGen experiences less than 1% forgetting in all tasks in both datasets.

Figure 4.5 shows that in the 20-task setting, LEAPGen suffers from less than 2% accuracy drop CIFAR100, while the existing methods suffer from up to 17% accuracy drop. In the Imagenet-R dataset, LEAPGens experience less than 7% accuracy drop CIFAR100, while the existing methods suffer from up to 25% accuracy degradation. This is a remarkable achievement of our proposed method since the larger task number implies a more challenging setting. In such a difficult setting, LEAPGen can maintain its performance so that it experiences the least degradation among the evaluated methods. The figure also shows that LEAPGen experiences nearly zero forgetting in the first ten tasks of CIFAR100, and experiences less than 2% forgetting afterward. LEAPGen also manages to decrease its forgetting measure in the latest tasks on ImageNet-R datasets. On the other hand, the existing methods suffer from increasing forgetting both in CIFAR100 and Imagenet-R, so its magnitude gets higher, i.e., up to 9% and 11% in CIFAF100 and ImageNet-R, respectively.

This historical analysis emphasizes the robustness of LEAPGen in a variety of task settings and confirms its superiority to the existing SOTAs.

4.5.7 LEAPGen-lite's Performance

As shown in Table 4.4-4.6 and Figure 4.3 Despite utilizing far smaller (2.67% params) generators and without class descriptors generated by LLM, LEAPGen-lite still outperforms the existing methods significantly, i.e., 4.7-25% FAA and 3-

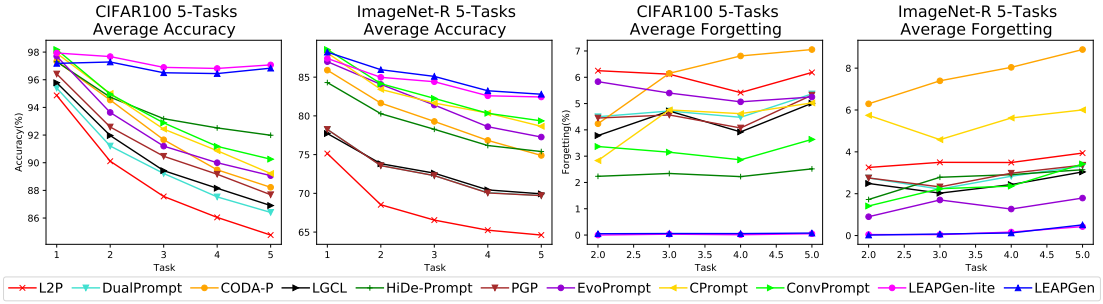


Figure 4.4: Visualization of the historical performance of consolidated algorithms in CIFAR100 and ImageNet-R with 5 tasks setting.

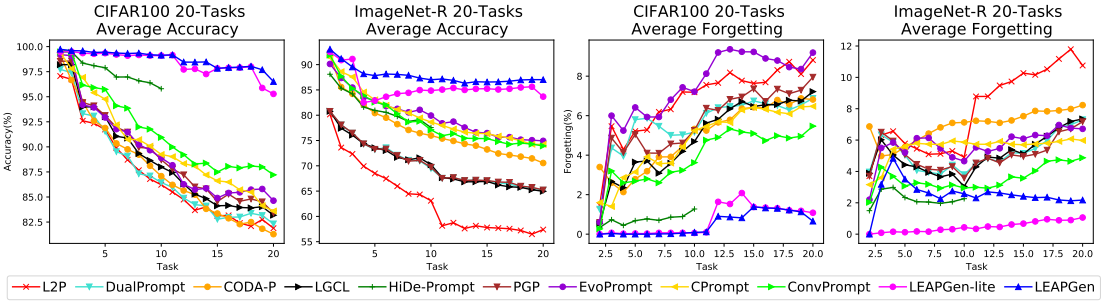


Figure 4.5: Visualization of the historical performance of consolidated algorithms in CIFAR100 and ImageNet-R with 20 tasks settings.

Table 4.7: Ablation study on the impact of LEAPGen components to its performance, run on 10 tasks Imagenet-R.

Component	Loss	FAA	CAA	FFM	CFM
FT	$\mathcal{L}_{intra}$	61.5	65.3	5.9	5.6
FT+E	$\mathcal{L}_{intra}$	72.9	78.5	5.8	5.3
FT+E+K^t	$\mathcal{L}_{intra}$, $\mathcal{L}_t$	74.9	79.5	2.8	3.2
FT+$G^t(E)$+K^t	$\mathcal{L}_{intra}$, $\mathcal{L}_t$	76.6	80.5	1.8	2.4
FT+$G^t(E)$+K^t+L_c^t	$\mathcal{L}_{intra}$, $\mathcal{L}_t$, $\mathcal{L}_c$	77.2	81.0	0.5	1.3
FT+$G^t(E)$+K^t+L_c^t+Aux	$\mathcal{L}_{intra}$, $\mathcal{L}_t$, $\mathcal{L}_c$	78.4	82.0	0.5	1.2
FT+$G^t(E)$+K^t+L_c^t+Aux	$\mathcal{L}_{intra}$, $\mathcal{L}_{inter}$, $\mathcal{L}_t$, $\mathcal{L}_c$	83.7	86.0	2.5	1.9
FT+$G^t(E)$+K^t+L_c^t+Aux+ Soft task ID Predictor	$\mathcal{L}_{intra}$, $\mathcal{L}_{inter}$, $\mathcal{L}_t$, $\mathcal{L}_c$	84.7	86.1	0.9	1.4

11% CAA in CIFAR100, and 3-30% FAA and 2-26% CAA in ImageNet-R dataset. LEAPGen-lite also achieves a low forgetting rate in these 2 datasets for all task settings. In the CUB dataset, LEAPGen-lite archives a comparable performance to HiDe-Prompt and outperforms other SOTA with a significant margin, i.e., 6-20% FAA and 3.3-14% CAA. This evidence proves our ideas, i.e., language embedding as input for prompt generation, task-wise generators, soft task ID predictor, and learning with auxiliary data are not bounded by the generated descriptors and the size of generators.

4.5.8 Ablation Study

Table 4.7 shows the impacts of the proposed method's components on its performance. **(1) Descriptor Embedding** E as one of LEAPGen's main components, elevates fine tuning (FT) baseline significantly, i.e., $>$ 11% FAA and $>$ 13% CAA. It shows the promising impact of our idea . **(2) Task Key K^t and Loss $\mathcal{L}_t$** transform the method into task-wise prompting based on input to K^t and $\mathcal{L}_t$ similarity. They produce a better model with 1-2% FAA and CAA improvement and reduce FFM and CFM by $>$ 2%. It shows the effectiveness of task-wise decomposition as applied in the task-wise prompt approach. **(3) Task-wise generator** $G^t()$ enhances the model recognition capability by 2% and 1% for FAA and CAA respectively. It also decreases its forgetting rate by 1%. It means the trainable generator produces a more discriminative prompt than the descriptor embedding only. **(4) L_c^t Class-wise Key L_c^t and Loss $\mathcal{L}_c$** for top-k embedding selection, i.e., E_i for $i \in [1...k]$ improves the model performance with 1% margin. It also reduces

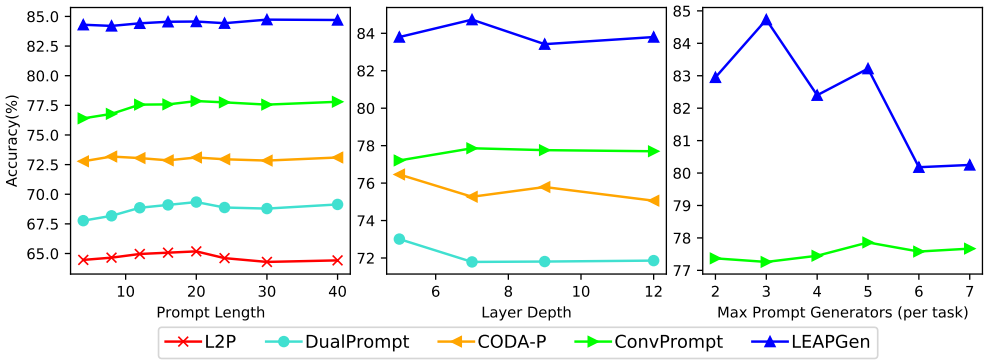


Figure 4.6: FAA of the consolidated methods under different prompt lengths (left), layer depth (middle), and maximum generators per task (right).

the model forgetting rate with up to 1% margin. Thus, measuring the input similarity to L_c^t offers a better embedding selection than toE_c^t directly.**(5) Auxiliary** embedding improves the LEAPGen's performance with a more than 1% margin. It shows the secondary contribution of descriptor embedding as an auxiliary modality along with its primary role in prompt generation. **(6) Inter-tasks loss** $\mathcal{L}_{inter}$ improves LEAPGen accuracy significantly, i.e., 5.4% and 4.0% for FAA and CAA, respectively, even though it increases FFM and CFM by $0.5 - 2\%$. It contributes to balancing the knowledge of previously learned classes (stability) and currently learned classes (plasticity), and emphasizes the risk of forgetting previously learned classes. **(7) Soft task ID predictor** substitution to conventional task ID [38] improves the accuracy with up to 1% and reduces FFM and CFM by 1.5% and 0.5% respectively. It implies that our designed task ID predictor outperforms the existing task ID predictor both in accuracy and forgetting.

4.5.9 Sensitivity Analysis

Figure 4.6 shows the sensitivity analysis of our proposed method in various settings, i.e., different **prompt lengths (left), layer depths (center), and number of generators per task(right)**. LEAPGen maintains its stable FAA in any prompt length values, i.e., within the range of $84.2 - 84.7\%$, while the existing methods, i.e., ConvPrompt, CODA-P, DualPrompt, and L2P, achieve significantly lower FAA, i.e., $\leq 78\%$. LEAPGen with the shortest prompt (4) has a higher FAA than

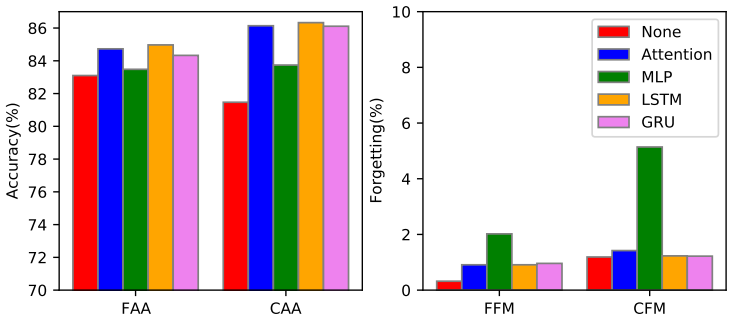


Figure 4.7: FAA, CAA, FFM, and CFM of the proposed method w.r.t types of generator.

the existing methods with the longest prompt (40), despite the 10 times smaller size. LEAPGen holds its stable performance at various depths of prompted VIT layers. LEAPGen consistently manages to achieve $83.5 - 84.7\%$ FAA, while other methods achieve far lower FAA, i.e., $\leq 78\%$. The best layer depth for LEAPGen and ConvPrompt is 7, while the best depth for the others is 5. The number of generators turns out to be the most sensitive parameter to LEAPGen, as it is saturated after 4 generators per task. Considering the number of generators in LEAPGen is the same number as the selected descriptor embedding, it is quite logical that LEAPGen becomes counterproductive after some threshold k.

4.5.10 Performance on Various Generators

LEAPGen's default generator $(G^{tK}_{i,l,h}, G^{tV}_{i,l,h})$ is a single-head self-attention (SSA) network parameterized by 3 matrices as explained in section sub-4.d. However, LEAPGen accommodates the different types of generators, e.g., MLP (3-layered), LSTM, and GRU. Figure 4.7 shows that LAPGen with any type generator achieves higher performance than without generators (None). The proposed SSA generator achieves fairly higher performance than MLP, slightly higher than GRU, and comparable to LSTM. However, LSTM and GRU has approximately 2.5 and 2 times larger numbers of parameters than SSA, while MLP has the same number of parameters as SSA.

4.5.11 Performance on Various Descriptors

We explore 3 types of descriptors, i.e., short, long, and narrative, generated by popular LLMSs, i.e., GPT, Gemini, and Llama. The examples of descriptors for classes "Strawberry" and "Lemon" for respective types and LLMs are shown in Table 4.8. The table shows that GPT generates a longer phrase than Gemini and Llama for the same descriptor type. We evaluated the performance of LEAPGen and compared it to ConvPrompt, which is the second-best achiever in ImageNet-R dataset and also uses descriptors. Note that ConvPrompt utilizes descriptors to define the number of new generators, while LEAPGen utilizes them as input to generate prompt components.

Table 4.9 shows the summarized result of LEAPGen and ConvPrompt performance on various descriptors. Overall, in any descriptor type generated by any LLMs, LEAPGen achieves at least 83.6% FAA and 85.0% CAA, where ConvPrompt achieves a maximum 77.6% FAA and 81.5% CAA. This implies that regardless of any type of descriptors generated by any LLM, LEAPGen consistently outperforms ConvPrompt with at least 6% and 3.5% margin for FAA and CAA, respectively. Looking at the average forgetting metrics, LEAPGEn also consistently outperforms ConvPrompt with at least 3% and 1% margin for FFM and CFM, respectively. Figure 4.8 shows the historical performance of LEAPGEn and ConvPrompt in various types of detectors and LLMs. The figure shows that despite LEAPGen achieving slightly lower average accuracy in the first task, it consistently achieves higher accuracy in the second until the last task. In addition, the gap is getting higher with the increasing number of learned tasks. The figure also shows that despite LEAPGen achieving a slightly higher forgetting in the second task, it manages to decrease the forgetting measure in the following task, until the last task. On the other hand, ConvPrompt suffers from the increasing average forgetting, so its average forgetting becomes far higher than LEAPGen's average forgetting in the later tasks.

Table 4.9 shows that the best result is achieved by LEAPGen with short descriptors generated by LLama. However, LEAPGen with short descriptors generated by Gemini achieves the least performance. Surprisingly, short descriptors generated

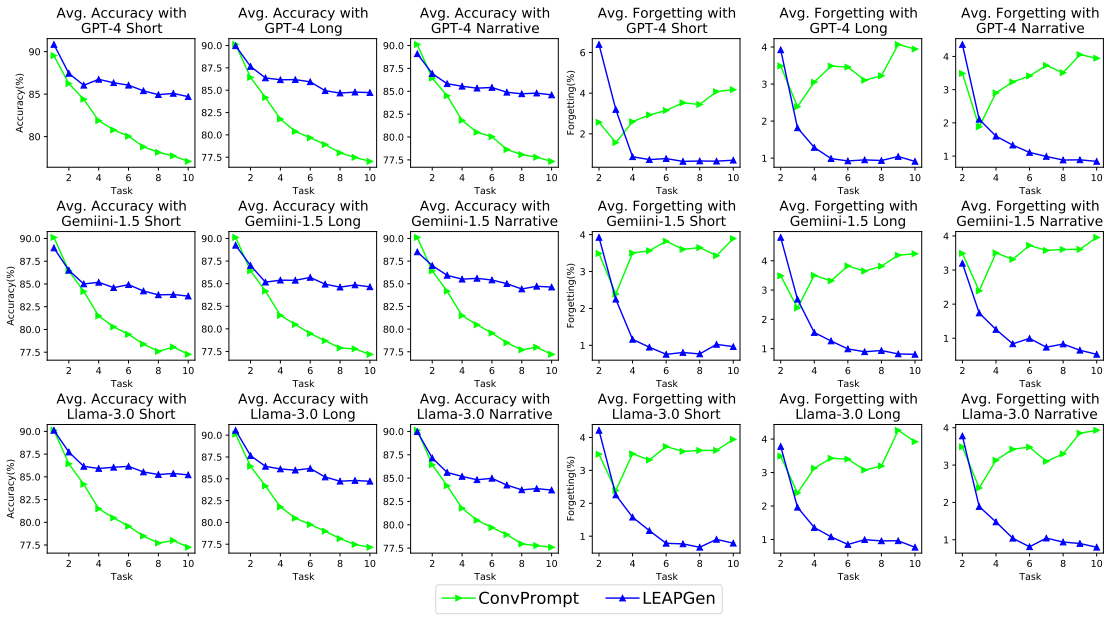


Figure 4.8: Visualization of the historical performance of consolidated algorithms in ImageNet-R with 10 tasks setting, in various descriptors and LLMs.

by Llama are the shortest descriptors among all the evaluated types. This result gives us insight that the length of descriptors is not the main factor in achieving maximum performance, rather the discriminative text has more impact. According to table 4.9, we also get insight that GPT produces more stable descriptors than the other LLSm, as the performance difference between types of descriptors is insignificant ($< 0.15\%$). Thus, regardless type of preferred descriptors, GPT is preferable to the other LLMs.

4.5.12 Trade-offs Between Cost and Performance

We extend our analysis on the trade-offs between Cost and Performance. Table 4.10 shows the cost performance on LEAPGen in comparison to ConvPrompt as the SOTA of the growing component approach and HiDe-Prompt as the SOTA of the task-wise prompt approach. We also compare our proposed method to the bigger version of ConvPrompt and HiDe-Prompt, i.e., ConvPrompt-Large and HiDe-Prompt, that utilize a larger number of executed parameters than LEAPGen. Please note that the number of executed parameters in the table means the total

Table 4.8: Types and examples of descriptors generated by popular LLMs.

LLM	Type of Descriptors		
	Short	**Long**	**Narrative**
GPT-4	"Strawberry": ["red, heart-shaped fruit", "small, green leaves", "white seeds on the surface", "yellow or white center", "pointed tip", "juicy texture"] "Lemon": ["yellow or greenish-yellow in color", "oval or round shape", smooth, glossy skin", "dimpled surface", "acidic smell", "sour taste", "thick, white pith", "seeds inside"]	"Strawberry": ["Red, heart-shaped fruit", "Small, green leafy crown", "Juicy and sweet", "Tiny seeds on surface", "Bright red color", "Soft, tender texture", "Used in desserts and salads", "Often enjoyed fresh or in preserves", "Distinctive aroma", "Popular in summer"] "Lemon": ["Bright yellow, oval fruit", "Tangy, acidic flavor", "Thick, dimpled skin", "Juicy, citrusy interior", "Used for flavoring and zest", "Common in cooking and beverages", "Refreshing and tart", "Variety of sizes", "Often used in garnishes", "Distinctive, vibrant color"]	"Strawberry": ["A strawberry rests temptingly, its red surface dotted with tiny seeds hinting at juiciness.", "The green leaves crown the top, contrasting beautifully with its vibrant hue.", "With a sweet spirit, it embodies the essence of summer."] "Lemon": ["A lemon sits cheerfully, its bright yellow skin and dimpled texture promising a burst of citrus flavor.", "The zesty aroma and vibrant color invite culinary exploration.", "With a tangy spirit, it embodies the essence of refreshment."]
Gemini-1.5	"Strawberry": ["red fruit with seeds on the outside", "red color", "small size and Sweet taste"] "Lemon": ["yellow citrus fruit", "yellow skin", "sour taste"]	"Strawberry": ["red fruit with seeds on The outside", "red color", "small size and sweet taste", "fruit", "red color", "small size and sweet taste", "used in desserts and jams"] "Lemon": ["yellow citrus fruit", "yellow skin", "sour taste", "citrus fruit", "yellow skin", "sour taste", "used in drinks and cooking"]	Strawberry": ["A small, red fruit with seeds on the outside. It is a popular fruit and is often used in jams, jellies, and desserts."] "Lemon": ["A yellow citrus fruit with a sour taste. It is often used in drinks, cooking, and baking."]
Llama-3.0	"Strawberry": ["fruit", "red color", "sweet"] "Lemon": ["fruit", "yellow skin", "sour"]	"Strawberry": ["fruit", "sweet", "juicy", "red color", "summer", "shortcake", "vitamin-rich", "antioxidant", "delicious", "romantic"] "Lemon": ["fruit", "citrus" "sour taste", "yellow skin", "juicing", "cooking", "flavorful", "refreshing", "zesty", "invigorating"]	"Strawberry": ["Sweet fruit delights, luscious landscape indulging, happy heart enjoying like a Joyful, little treat."] "Lemon": ["Zesty slice refreshes, citrusy landscape revitalizing, Rejuvenated heart refreshing Like a bright, little sunrise."]

Table 4.9: Performance of LEAPGEn vs ConvPrompt in various descriptors.

LLM	Desc. Type	LEAPGen				ConvPrompt			
		FAA	CAA	FFM	CFM	FAA	CAA	FFM	CFM
GPT-4	Short	84.70	86.36	0.70	1.74	77.08	81.47	4.17	2.98
GPT-4	Long	84.73	86.14	0.91	1.48	77.03	81.40	3.95	3.28
GPT-4	Narrative	84.60	85.72	0.84	1.66	77.33	81.53	3.95	3.28
Gemini-1.5	Short	83.67	85.07	0.96	1.45	77.27	81.33	3.89	3.43
Gemini-1.5	Long	84.65	85.69	0.81	1.74	77.18	81.37	4.24	3.52
Gemini-1.5	Narrative	84.63	85.68	0.53	1.28	77.22	81.37	3.96	3.40
Llama-3.0	Short	85.22	86.35	0.79	1.54	77.25	81.37	3.94	3.40
Llama-3.0	Long	84.72	86.24	0.77	1.49	77.15	81.45	3.92	3.29
Llama-3.0	Narrative	83.72	85.33	0.80	1.49	77.59	81.49	3.93	3.27

executed parameters in the forward operation, excluding the ViT backbone and classifier head.

4.5.12.1 Performance vs Parameter and Running Time

LEAPGen outperforms the default version of ConvPrompt and HiDe-Prompt with up to 9% average accuracy margin and up to 4% average forgetting measure, with the drawback of the higher number of executed parameters. However, with the higher number of executed parameters, LEAPGen requires a significantly lower running time than both of the methods, i.e., 1.5 h vs 2.11h and 5.4h for ConvPrompt and Hide-Prompt, respectively. Even though utilizing far smaller parameters, Hide-Prompt requires extremely high running time since it executes too many operations, i.e., augmenting the prototypes and then forming an uninstructed representation that requires high epochs to converge. Thus, it spends far more running time in total.

The large version of ConvPrompt and HiDe-Prompt executes a higher number of parameters than LEAPGen. In line to the higher parameters, they require far higher running time than their original version. Unfortunately, with the extremely high number of parameters, they didn't achieve a higher performance, but rather a lower performance than the performance of their standard version. Compared to the large versions of ConvPrompt and HiDe-Prompt, LEAPGen has better performance and lower cost. This evidence confirms our sensitivity analysis that a model has the best setting that achieves optimal performance. The higher number

of parameters or training epochs does not imply a higher performance. Thus, we evaluate the competitor methods following their advised setting described in the respective papers or official code.

LEAPGen-lite outperforms ConvPrompt and Hide-Prompt with a significant margin, i.e., 6-8% even though it utilizes a smaller number of parameters on average and without class descriptors as in ConvPrompt and LEAPGen. However, it achieves a smaller final (task-10) average accuracy than LEAPGen with more than 1.5% margin.

4.5.12.2 Total Running Time

Table 4.11 shows the simulation time for LEAPGen, ConvPrompt as SOTA in 3 benchmark datasets, i.e., CIFAR100, ImageNet-R, and CUB. The table shows a similar trend to the analysis on the ImageNet-R dataset. Descriptor generation consumes a fair running time, i.e., 0.53-1.09 h, since it is conducted via online query through the internet connection. In such a manner, we do not need extra storage to save the LLM model. Please note that descriptor generation is conducted only once before the training phase. Thus, the LLm is not utilized during the training and testing phases of LEAPGen. Despite requiring additional time to generate classes' descriptors, LEAPGen still has significantly lower total grinning time than HiDe-Prompt, i.e., 3 times, 3 times, and 2.8 times smaller in CIFAR100, ImageNet-R, and CUB datasets, respectively. Compared to LEAPGen, ConvPrompt requires a higher training time in those 3 datasets with a significant gap, i.e., up to 7.7 hours. Please note that ConvPrompt utilizes evolving generators, and the number of its generators grows with the increase of tasks. Thus, in the later tasks, its forward operation takes more time than in the earlier tasks. Having no additional time for descriptor generation, LEAPGen-lite spends the smallest total running time and training+inference time.

4.5.12.3 Inference time only

Table 4.11 shows that LEAPGens, i.e., LEAPGen and LEAPGen-lite have a moderate inference time, i.e., lower than ConvPrompt and higher than HiDe-Prompt. This empirical evidence is in line with our previous analysis, where the growing

components/generators approach executes more generators in its forward function with the increasing number of tasks. Thus, on average, it takes more inference time than our task-wise generator approach. Hide-Prompt requires less inference time since it utilizes task-wise prompting, where the prompts are not generated, but rather directly prepended into the ViT MSA heads. Thu,s it takes fewer operations to execute, which implies it has a lower inference time. In spite of its lower inference time, HiDe-Prompt requires far higher training time, which makes it a higher total running time than LEAPGen. In summary, this evidence proves the running efficiency of our proposed method both in terms of inference time and the total running time in a simulation, despite the fact that LEAPGen executes a higher number of parameters.

4.5.12.4 Performance vs Storage

Table 4.12 shows the performance vs storage of the consolidated methods in the ImageNet-R dataset with 10 task settings. Model saving is conducted by executing the method "torch.save(model.state_dict().filename)". Please note that this storage does not include the LLM model, since it is used via online (remote) query, not saved in our storage. The table shows that LEAPGen requires a bigger storage size than ConvPrompt, as LEAPGen has more attributes and parameters. Please note that the model saving is represented in a dictionary object, thus, more attributes require more dictionary keys and items that imply more storage consumption. Different from ConvPrompt and HiDe-Prompt, LEAPGen has a more complex generator structure and task ID predictors that utilize task-wise key and class-wise keys. In addition, LEAPGen also saves the descriptor embedding for the testing phase, different from ConvPrompt, which does not save the embeddings as it uses the descriptor embedding to calculate the number of generators during the training phase only. Therefore, it requires a bigger storage to save its model. The storage consumption can be minimized by using a lightweight generator. In comparison to large versions of ConvPrompt and HiDe-Prompt that have a higher number of parameters, LEAPGen achieves a lower storage consumption while achieving a higher performance, i.e., FAA and CAA.

Table 4.10: Trade-offs between cost and performance on ImageNet-R dataset with 10 task settings. ConvPrompt-Large is run for 9 tasks only since it crashed on the final task.

Method	Metrics	Performance After Finishing Task										AVG	Running Time (h)
		1	2	3	4	5	6	7	8	9	10		
HiDe-Prompt	Avg.Accuracy	85.22	82.93	82.35	79.74	78.85	77.81	77.37	76.38	76.31	75.75	79.27	-
HiDe-Prompt-Large	Avg.Accuracy	85.76	83.12	82.03	79.49	78.55	76.55	76.00	74.82	74.98	74.30	78.56	-
ConvPrompt	Avg.Accuracy	89.53	86.24	84.40	81.88	80.79	80.04	78.80	78.15	77.74	77.08	81.47	-
ConvPrompt-Large	Avg.Accuracy	90.12	86.22	83.96	81.45	80.52	79.73	76.40	74.19	74.56	-	80.79	-
LEAPGen-lite	Avg.Accuracy	89.73	88.66	86.94	86.43	86.04	83.03	82.75	82.52	82.94	82.38	85.14	-
LEAPGen	Avg.Accuracy	89.83	87.63	86.46	85.97	84.71	84.79	84.01	83.77	84.12	84.09	85.54	-
HiDe-Prompt	Avg.Forgetting	3.29	1.50	2.41	2.19	2.53	2.20	2.25	2.30	2.29	2.33	2.33	-
HiDe-Prompt-Large	Avg.Forgetting	-	3.05	1.89	2.14	2.15	3.25	2.91	2.93	2.66	2.70	2.63	-
ConvPrompt	Avg.Forgetting	-	2.57	1.57	2.60	2.93	3.15	3.53	3.44	4.07	4.17	3.11	-
ConvPrompt-Large	Avg.Forgetting	-	4.65	3.39	4.62	4.69	5.56	8.91	10.55	10.50	-	6.61	-
LEAPGen-lite	Avg.Forgetting	1.11	0.75	0.58	0.56	4.13	3.37	2.92	2.70	3.01	2.13	2.13	-
LEAPGen	Avg.Forgetting	-	3.97	1.96	1.38	2.63	2.24	1.99	1.77	1.60	1.46	2.11	-
HiDe-Prompt	#Parameters (M)	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	5.40
HiDe-Prompt-Large	#Parameters (M)	12.29	12.29	12.29	12.29	12.29	12.29	12.29	12.29	12.29	12.29	12.29	15.15
ConvPrompt	#Parameters (M)	0.55	0.60	0.70	0.80	0.85	0.89	0.99	1.09	1.19	1.28	0.89	2.11
ConvPrompt-Large	#Parameters (M)	5.18	5.82	6.65	7.28	7.91	8.60	9.81	10.94	12.01	13.23	8.74	15.28
LEAPGen-lite	#Parameters (M)	0.02	0.03	0.05	0.07	0.08	0.10	0.11	0.13	0.15	0.16	0.09	0.53
LEAPGen	#Parameters (M)	6.21	6.23	6.24	6.26	6.27	6.29	6.31	6.32	6.34	6.35	6.28	1.79

4.5.13 Complexity Analysis

Following the pseudo-code in Algorithm 2, LEAPGen has several atomic operations, e.g., selecting top-k L_c^i (line 7), prompt generation (line 8), computing logits (line 9), computing loss (line 10-14), and updating parameters (line 15). Let $N^t = |\mathcal{T}|^t$, and accumulating the operations on all batches, then selecting top-k L_c^i costs $O(N^t.T.|C^t|)$, the prompt generation costs $O(N^t.k)$, computing losses costs $O(N^t)$, and parameters update cost costs $O(N^t)$, then the the complexity of LEAPGen will be:

$$O(LEAPGen) = T.E.(O(N^t.T.|C^t|) + O(N^t.k) + O(N^t) + O(N^t)) \tag{4.54}$$

$$O(LEAPGen) = T.E.(O(N^t.T.|C^t|) + O(N^t.k) + O(N^t)) \tag{4.55}$$

$$O(LEAPGen) = E.(O(T.N^t.T.|C^t|) + O(T.N^t.k) + O(T.N^t)) \tag{4.56}$$

Note that $T.N^t = N$ that represents the number of all samples from the first task until the last task, and $T.|C^t| = |C|$ that represents the number of all learned

Table 4.11: Total running time including descriptor generation, training, and testing.

Method	Approach	Running time (h)			
		Descriptor Generation	Inference Only	Training + Inference	Total
		CIFAR100			
HiDe-Prompt	Task-wise Fixed prompt	-	0.037	4.63	4.63
ConvPrompt	Evolving Prompt generator	0.53	0.023	2.01	2.54
LEAPGen-lite	Task-wise Prompt generator	-	0.016	1.20	1.20
LEAPGen	Task-wise Prompt generator	0.53	0.017	0.98	1.51
		ImageNet-R			
HiDe-Prompt	Task-wise Fixed prompt	-	0.019	5.40	5.40
ConvPrompt	Evolving Prompt generator	1.07	0.033	1.04	2.11
LEAPGen-lite	Task-wise Prompt generator	-	0.028	0.53	0.53
LEAPGen	Task-wise Prompt generator	1.07	0.025	0.43	1.50
		CUB			
HiDe-Prompt	Task-wise Fixed prompt	-	0.017	3.94	3.94
ConvPrompt	Evolving Prompt generator	1.09	0.035	8.08	9.17
LEAPGen-lite	Task-wise Prompt generator	-	0.028	0.38	0.38
LEAPGen	Task-wise Prompt generator	1.09	0.025	0.32	1.41

Table 4.12: Performance vs Storage Trade-offs in ImageNet-R dataset, * denotes measurement at task-9 since the method crashed after task-9.

Method	Performance		Storage (MB)
	FAA	CFA	
HiDe-Prompt	75.75	79.27	334
HiDe-Prompt-Large	74.30	78.56	797
ConvPrompt	77.08	81.47	346
ConvPrompt-Large	74.56*	80.79	632
LAEPGen-lite	82.38	85.14	332
LEAPGen	84.09	85.54	567

classes where $|C| << N$, and k is a small constant < 10. Then LEAPGen complexity becomes

$$O(LEAPGen) = E.(O(N|C|) + O(Nk) + O(N)) \tag{4.57}$$

$$O(LEAPGen) = E.(O(N|C|)) \tag{4.58}$$

$$O(LEAPGen) = O(EN|C|) \tag{4.59}$$

Assuming that learning epoch E and number of classes $|C|$ are constant, then the

equation above is derived into:

$$O(LEAPGen) = O(EN|C|) = O(N) \tag{4.60}$$

4.6 Summary

This chapter identifies the prompt-based approach dilemma and proposes a novel prompt-based structure and algorithm that incorporates 4 key concepts: (1) language as input for prompt generation, (2) task-wise generators, (3) limiting matching descriptors search space via soft task ID prediction, and (4) generated prompt as auxiliary data. The experimental analysis shows the superiority of LEAPGen to existing SOTAs in CIFAR100, ImageNet-R, and CUB datasets with a significant margin, i.e., up to 30% final average accuracy, 24% cumulative average accuracy, 8% final forgetting measure, and 7% cumulative forgetting measure. The historical analysis confirms LEAPGen successfully maintains stability-plasticity in every task. The robustness analysis shows LEAPGen consistently achieves higher performance in various prompt lengths, layer depths, and the number of generators per task compared to the SOTAs. The extended analysis shows LEAPGen complies with various generator networks and various descriptor types. The cost-to-performance analysis shows that even though it executes a higher number of parameters, LEAPGen requires a smaller total running time and relatively small inference time.

The upcoming chapter is directed to answer the few-shot continual learning problems, handling catastrophic forgetting under data scarcity. The few-shot continual learning is a challenge since it is difficult to achieve plasticity while maintaining stability with few-shot samples. Furthermore, the replay methods that save previous task samples in memory can not be applied.

Chapter 5

FLOWER: Few-Shot Continual Learning via Flat-to-Wide Approaches

In the previous chapters, AGLA and LEAPGen propose new methods for standard CL where each class carries abundant labeled samples. In many real-world problems, a class may carry only a very limited sample due to many reasons, such as an unavailable observed object or the high cost of data acquisition. This chapter addresses the few-shot continual learning problem, i.e., continual learning under data scarcity constraints. This chapter proposes a few-shot continual learning approach, termed FLat-tO-WidE AppRoach (FLOWER), where a flat-to-wide learning process, finding the flat-wide minima, is proposed to address the catastrophic forgetting problem. The issue of data scarcity is overcome with a data augmentation approach, making use of a ball generator concept to restrict the sampling space into the smallest enclosing ball. Our numerical studies demonstrate the advantage of FLOWER, achieving significantly improved performances over prior arts, notably in the small base tasks.

Part of this chapter is published as: MA Ma'sum, M Pratama, E Lughofer, L Liu, H Habibullah, R Kowalczyk. Few-shot continual learning via flat-to-wide approaches. IEEE Transactions on Neural Networks and Learning Systems. 36(5), 8966-8978. https://doi.org/10.1109/TNNLS.2024.3415048 [**IF: 10.2, CORE A***]

5.1 Introduction

Existing studies on continual learning (CL) mostly focus on solving the catastrophic forgetting (CF) problem, where old parameters of a deep neural network are overwritten when learning new tasks without relying on any information from old samples, thus losing their generalization power to old tasks. Three approaches, namely the regularization-based approach, the structure-based approach, and the memory-based approach, have been well-established in the literature [187], where some of the above-mentioned approaches have successfully tackled the class-incremental learning problem [161], i.e., a single head scenario without any task IDs, more challenging than the task-incremental learning problem. However, these approaches suffer from the over-fitting problem in the case of limited samples. Hence, they are unfit for many real-world problems, such as a slow-speed manufacturing process, where the problem of data scarcity is present. This drawback also does not coincide with the natural learning where human beings can learn from very few samples [109]. Our interest here is to attack the problem of few-shot continual learning [40] where each task carries very few samples in the N-way K-shot setting, i.e., each task comprises N classes where each task presents K samples per class. Such a case requires a model to handle not only a sequence of tasks without the CF problem, but also to avoid the over-fitting problem due to scarce samples.

The few-shot continual learning approach is pioneered by the work in [40] where the concept of neural gas (NG) is adopted. Old nodes of the NG structure are stabilized to avoid the CF problem while expanding the NG structure to adapt to new concepts. [56] presents a few-shot continual learning algorithm using the concept of session trainable parameters, deemed unimportant for previous tasks, to alleviate the CF problem. The training process is governed by a triplet loss functio,n and classification is performed using the nearest prototype-based classification approach. The vector quantization concept in the deep embedding space is successfully implemented in [188] to deal with both regression and classification problems in few-shot continual learning environments. A continually evolved classifier (CEC) is proposed in [125] using a graph attention network (GAT) to adapt

the classifier's weights. [42] introduces the concept of flat learning, finding flat regions in the base learning task to be maintained in the few-shot learning task. However, it relies on a memory storing exemplar sets of previous tasks. The use of memory might lead to most samples of few-shot learning tasks being kept in the memory. It heavily focuses on the stability of base tasks but ignores the plasticity of new tasks.

This chapter presents a FLat-tO-WidE appRoach (FLOWER) for few-shot continual learning problems. FLOWER is configured as a prototypical network [61] driven by the prototypical loss. FLOWER puts forward the flat-to-wide learning idea [42], [62], to discover flat-wide minimum regions. It minimizes the losses of noise-perturbed parameters of the feature extractor over M runs. The flat-wide parameters are maintained in the subsequent few-shot learning tasks by constraining feature extractor parameters to prevent the catastrophic forgetting problem. Our approach differs from [42], heavily focusing on the flat minimum regions, resulting in over-dependence on the base learning task. In addition, no memory is imposed by FLOWER, and the wide learning concept is integrated as an attempt to drive flat regions to be wide, thus scaling well for large-scale problems. The concept of ball generator is integrated in FLOWER to prevent the over-training problem due to scarce samples in few-shot learning tasks.

A feature-space data augmentation approach is applied in a sequence of few-shot learning tasks. The data augmentation strategy follows the ball generator concept [189] where the sampling space is limited to the smallest enclosing ball of few-shot samples. A generator loss is incorporated such that generated samples are adjacent to the center of their own class and far from the center of other classes. The projection-based memory-aware synapses method is implemented to avoid the classifier's parameters being overwritten when embracing new few-shot tasks. That is, the memory-aware synapses regularization approach is combined with the classifier's projection concept of [62] and the clamping procedure, inducing flat-wide local optimum regions scaling well for large-scale continual learning tasks. The main network and the ball generator are trained to minimize a joint loss function of the prototypical loss and the generator loss, added with the regularizer. A constraint is implemented to feature extractor parameters, where they are clamped

if going beyond the flat-wide regions [42]. This approach also prevents the overfitting problem because only few classifier parameters are adapted while leaving feature extractor parameters in a fixed range.

Five major contributions are proposed:

1. This chapter proposes flat-to-wide approach (FLOWER) for few-shot continual learning, synergizing the flat learning concept and the wide learning concept. The flat-wide minima are enforced to address the CF problem. This enables improved plasticity when learning new tasks as well as maintains decent performances with small base tasks. Note that the good average performance of [42] often comes at the cost of poor performances on new tasks and small base tasks;

2. The concept of ball generator is proposed to perform the feature space data augmentation as a solution to data scarcity. This approach is modified from [189] designed for texts and non-continual learning problems;

3. New loss functions unifying the flat learning loss, wide learning loss, and ball-generator loss are derived for both base learning tasks and few-shot learning tasks, leading to an end-to-end training process; A rigorous theoretical study is performed to confirm the convergence of FLOWER, where its losses converge toward zero.

4. The efficacy of FLOWER has been validated and compared with state-of-the-art algorithms, where FLOWER delivers improved performances and works well with the small base tasks.

5.2 Problem Formulation

A few-shot continual learning problem is defined as a learning problem of a sequence of fully-labelled tasks $\mathcal{T}_1, \mathcal{T}_k, ..., \mathcal{T}_K$ where $k \in \{1, ..., K\}$ denotes the number of tasks unknown before a process runs. Each task carries N_k pairs of training samples $\mathcal{T}_k = \{x_i^k, y_i^k\}_{i=1}^{N_k}$ where $x_i \in \mathcal{X}$ stands for an input image and $y_i \in \mathcal{Y}$ denotes its corresponding class label. All data samples are drawn from the same domain $\mathcal{X} \times \mathcal{Y} \in \mathcal{D}$. Each task features the same image size but possesses disjoint

target classes. Suppose that Y_k labels a class set of a $k-th$ task and Y_{k-1} denotes a class set of a $k-1$ task, $\forall k, k-1, Y_k \cap Y_{k-1} = \emptyset$. *The few-shot continual learning problem is formulated as the class-incremental learning problem rather than the task-incremental learning problem [161] where a class prediction is inferred by a single-head network structure and a pair of data samples of every task arrive without the task IDs* t_i^k.

$\mathcal{T}_1$ is a base learning task possessing a reasonably large number of samples, while the remainder of the tasks are a few-shot learning task lacking samples, i.e., each few-shot learning task is set in the N way K shot configuration where it has N target classes with K samples per class. In addition, a task $\mathcal{T}_{k>1}$ is only accessible at the $k-th$ session and completely discarded once seen. This constraint causes the catastrophic interference problem, where previously valid parameters are overwritten when learning a new task. The memory-based approach is unfeasible for the few-shot continual learning problems because it may lead to the storage of all past samples due to the data scarcity constraint, a retraining case. Hence, a few-shot continual learner is supposed to not only accumulate knowledge from already seen experiences but also present efficient sample utilization to avoid the over-fitting problem.

Our few-shot continual learner is defined as $g_{W_C}(f_{W_F}(.))$ where $f_{W_F}(.)$ stands for a feature extractor with parameters W_F while $g_{W_C}(.)$ denotes a fully-connected network with parameters W_C. Specifically, it is configured as a prototypical network [61] where the final prediction is expressed as follows:

$$P(\hat{y} = c|x; W_C, W_F) = \frac{\exp\left(-d(g_{W_C}(f_{W_F}(x)), p_c)\right)}{\sum_{c \in M} \exp\left(-d(g_{W_C}(f_{W_F}(x)), p_c)\right)} \tag{5.1}$$

$c^* = \arg\max_{c \in M} P(\hat{y} = c|x; W_C, W_F)$ stands for a predicted class label and M denotes the number of classes seen thus far. $d(.)$ is a distance metric implemented as the L_2 distance metric in this chapter. $p_c = \frac{1}{|Y_1|} \sum_{(y) \in Y_1} f_{W_F}(x_i)$ is the prototype of the $c-th$ class. The learner is tasked to perform well across all tasks and minimizes $\frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_k$ and $\mathcal{L}_k \triangleq \mathbb{E}_{(x,y) \backsim \mathcal{T}_k}[\mathcal{L}(g_{W_C}(f_{W_F}(x)), y)]$ where $\mathcal{L}(.)$ is a loss function of interest where a cross-entropy loss of the prototypical network is adopted here.

$$\mathcal{L}_{ce} = -\sum_{i}\sum_{c} \mathbf{1}(y=c)\frac{\exp\left(-d(g_{W_C}(f_{W_F}(x_i)), p_c)\right)}{\sum_{c\in M}\exp\left(-d(g_{W_C}(f_{W_F}(x_i)), p_c)\right)} \tag{5.2}$$

where $\mathbf{1}(y=c)$ is an indicator function returning 1 if $y=c$. The inference adopts the nearest-mean classification strategy, making use of a class label of the nearest prototype.

5.3 The Flat-to-Wide Approach (FLOWER)

The few-shot continual learning problem consists of two learning stages: a base learning phase and a sequence of few-shot learning phases.

5.3.1 Base Learning Phase ($k=1$)

We apply the concept of b-flat-wide local minima where, for the parameters of interest in the flat-wide region, i.e., the feature extractor parameters W_F, the losses are still minimized, leading to well separations of the classes of interest and robustness against the catastrophic forgetting problem [42]. Once found in the base learning phase, these parameters are only tuned within this region during the few-shot learning phases. In other words, it is sufficient to adjust the network parameters within the flat-wide interval to avoid the over-training problem. A definition of the b-flat-wide local minima is formalised as:

Definition 1 [42]: given a real-valued loss function $\mathcal{L}(.)$, for any $b>0$ and network parameters $\Theta = \{W_C, W_F\}$, the b-flat-wide local minima Θ^* follow the following condition:

- Condition 1: for $\mathcal{L}(x;\Theta^*+\epsilon)$, where $-b \leq \epsilon \leq +b$.
- Condition 2: there exist $c_1 < \Theta^* - b$ and $c_2 > \Theta^* + b$ s.t. $\mathcal{L}(x;\Theta) > \mathcal{L}(x;\Theta^*)$ where $c_1 < \Theta < \Theta^* - b$ and $\Theta^* + b < \Theta < c_2$

These conditions are extremely hard to hold in reality, and an approximation is carried out using a noise perturbation. That is, we run M trials with different noise perturbations $\epsilon_j, -b \leq \epsilon_j \leq +b \quad \& \quad j \in M$, against network parameters Θ^*. Under different noise perturbations, the network parameters within the estimated

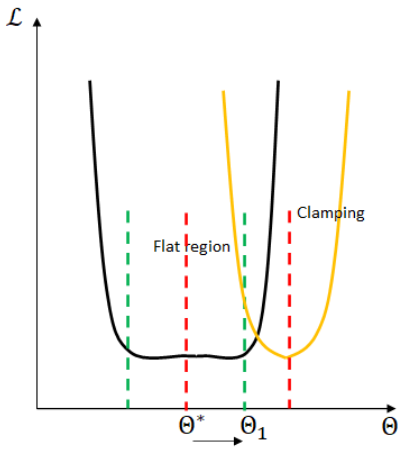


Figure 5.1: Flat Continual Learning: during the base learning phase $k = 1$, FLOWER finds the network parameters Θ^* located in the flat local optimum region. These parameters might shift to non-flat local minima regions when learning the few-shot learning tasks $k > 1$ via $\Theta^* \rightarrow \Theta_1$. Clamping is performed for Θ_1 to stay around a flat local optimum region.

b flat-wide regions are supposed to return similar losses and to have small function values [42]. Figure 5.1 visualizes the flat learning paradigm. In a nutshell, the base learning phase is designed to minimize the following loss function.

$$\begin{aligned} \mathcal{L}_{k=1} &= \frac{1}{M}\sum_{j=1}^{M}\mathcal{L}_{base}^{j}(g_{W_C}(f_{W_F+\epsilon_j}(x)), y) \\ \mathcal{L}_{base}^{j} &= \mathbb{E}_{(x,y)\backsim\mathcal{T}_1}[\mathcal{L}_{ce}(g_{W_C}(f_{W_F+\epsilon_j}(x)), y)+ \\ \mathcal{L}_{KL}(g_{W_C}(f_{W_F+\epsilon_j}(x)), P_U)] &+ \lambda_1\mathbb{E}_{y\backsim Y_1}||p_c - p_c^*||_2 \end{aligned} \tag{5.3}$$

where p_c, p_c^* stand for the prototype of the $c-th$ class before and after noise perturbations respectively. The first term of $\mathcal{L}_{base}$ is designed to unveil the b-flat-wide region, while the second term prevents the prototype drift problem. P_U is a uniform distribution and $\mathcal{L}_{KL}(.)$ promotes wide local minima via output's projections to uniform distributions. Note that (5.3) extends [42] where only the flatness is solicited and ignores the wide learning issue.

The goal of (5.3) is to unveil a flat-wide region working well on all tasks, thus overcoming the catastrophic forgetting problem. It is done by sampling ϵ_j M times to vary network parameter leading to $\mathcal{L}_{k=1}$ considered as a non-convex loss

function [42]. It is an approximation of the expected loss $R(\Theta) = \mathbb{E}_{z,\epsilon}[\mathcal{L}_{t=1}]$, which is impossible to minimize in practise. z_k is the $k-th$ data batch.

Assumption 1 (L-smooth risk function [42]): $R : \Re^D \to \Re$ is continuously differentiable and L-smooth $L > 0$:

$$||\nabla R(\Theta) - \nabla R(\Theta^{'})||_2 \leq L||\Theta - \Theta^{'}|| \tag{5.4}$$

This assumption constrains the pace of the gradients with respect to the parameter vectors.

Assumption 2: the expected loss function $R(\Theta)$ holds the following properties [42]:

- R is bounded below a scalar R^* given $\{\Theta_k\}$ where k indicates the $k-th$ data batch. It implies R to be bounded by a minimum value R^*;
- For all $k \in \mathbb{N}$ and $j \in [1, M]$:

$$\mathbb{E}_{z_k,\epsilon_j} \nabla \mathcal{L}_{base}^{j,k} = \nabla R(\Theta_k) \tag{5.5}$$

  where $\nabla \mathcal{L}_{base}^{j,k}$ is an unbiased estimation of $\nabla R(\Theta_k)$ to simplify the proof;
- given $m_1 \geq 0, m_2 \geq 0$, for all $k \in \mathbb{N}$ and $j \in [1, M]$:

$$\mathbb{V}_{z_k,\epsilon_j}[\nabla \mathcal{L}_{base}^{j,k}] \leq (m_1 + m_2)||\nabla R(\Theta_k)||_2^2 \tag{5.6}$$

  meaning that the variance of the gradient cannot be arbitrarily large.

$\mathbb{E}_{z_k,\epsilon_j}, \mathbb{V}[.]$ respectively denote the expectation w.r.t the joint distribution of random variables z_k, ϵ_j and the variance. These three conditions are reasonable in practice and applicable for convergence analysis.

Assumption 3: the learning rates λ_k satisfy [42]

$$\sum_{k=1}^{\infty} \lambda_k = \infty, \sum_{k=1}^{\infty} \lambda_k^2 < \infty \tag{5.7}$$

which can be easily met due to $\lambda_k < 1$ and a decreasing function w.r.t k. These assumptions establish a theorem.

Theorem 1: Given the assumptions 1-3 and R is twice differentiable, we have:

$$\lim_{k\to\infty} \mathbb{E}[||\nabla R(\Theta_k)||_2^2] = 0 \tag{5.8}$$

the convergence of the flat-wide minima strategy.

5.3.2 Few-Shot Learning Phase ($k > 1$)

The few-shot learning phase of FLOWER is driven by a feature-space data augmentation approach to address the data scarcity issue while applying a combination of the flat and wide learning principles to address the CF problem without any memory. Network parameters, namely those of feature extractors, are adjusted in such a way to maintain the b-flat-wide local minima conditions to mitigate the over-training problems, and the projection concept is implemented in the memory-aware synapses (MAS) method [26] to induce a wide local optimum.

5.3.2.1 Feature Space Data Augmentation

The concept of ball generator [189] is adopted and operates in the feature space $z = f_{W_F}(x) \in \Re^D$ rather than the high-dimensional data space $x \in \Re^{w\times H\times c}$. The ball generator produces synthetic samples of the smallest enclosing ball of the $c-th$ class $\omega(Z_c) = \{C_c, \sigma_c\}$ where C_c, σ_c denote the centroid and radius of the $c-th$ ball/class. Unlike [189], the ball generator is directly executed in the few-shot learning tasks where there exist the data scarcity problem rather than to that of the support set. A synthetic sample $\hat{z} \in \Re^D$ is generated:

$$\hat{z} = C_c + u^{\frac{1}{D}}\sigma_c \frac{\phi}{||\phi||_2} \tag{5.9}$$

where u denotes a shifting constant sampled from a uniform distribution $[0, 1]$ and $\phi \in \Re^D$ labels a random vector generated from a normal distribution $\phi \backsim \mathcal{N}(0, 1)$.

We feed a synthetic sample $\hat{z}$ to a transformation module $\hat{\hat{z}} = \kappa_{W_T}(\hat{z}) \in \Re^D$ arranged as three fully connected layer network as per [189]. The goal of the transformation module is to avoid the bias of synthetic samples. That is, a ball

generator loss is introduced as follows:

$$\mathcal{L}_{ball} = \sum_{\hat{\hat{z}} \in c_i, \hat{\hat{z}} \notin c_j} \lambda_2 \max\{0, d(\hat{\hat{z}}, C_i) + r - d(\hat{\hat{z}}, C_j)\} \tag{5.10}$$

where r is a predefined margin and $\hat{\hat{z}} = \kappa_{W_T}(\hat{z})$ stands for a synthetically generated sample of the transformation module. C_i, C_j respectively denote the centroid of the $i-th$ and $j-th$ ball or class. (5.10) is inspired by the triplet loss of the metric learning, aiming to pull the synthetic sample $\hat{\hat{z}}$ close to its centroid and to push it away from other centroids. λ_2 is a predefined constant steering the influence of the ball generator loss function.

5.3.2.2 Projection-based Memory Aware Synapses

FLOWER relies on a regularizer $\mathcal{R}_{PMAS}$ integrating the projecting concept [62] into the memory-aware synapses approach to address the catastrophic forgetting problem. The application of the projection concept induces wide local optimum regions, improving the scalability of a continual learner since it increases the chance of an overlapping region being found. The PMAS method is formalized:

$$\begin{aligned} \mathcal{R}_{PMAS} = \frac{\lambda_3}{N_k} \sum_{i=1}^{N_k} \mathcal{L}_{KL}(g_{W_C}(f_{W_F}(x_i^k)), P_U) \\ + \lambda_4 \sum_i \Xi_i^{k-1} (\Theta_i^k - \Theta_i^{k-1})^2 \end{aligned} \tag{5.11}$$

where Ξ is a parameter importance matrix which can be estimated using different approaches: EWC [27], SI [26], MAS [25], etc. We apply the MAS method based on the online and unsupervised method, since it is faster to compute than EWC and SI. $\Theta = \{W_F, W_C\}$ where no regularization mechanism is applied to the transformation module $\kappa_{W_T}(.)$ because it only functions to enrich the current concept. The first term of (5.11) promotes a wide local minimum where $\mathcal{L}_{KL}(.)$ denotes the KL divergence loss function, minimizing the discrepancies of the two components, while the second term of (5.11) is the MAS-based regularization approach inspired by the L_2 regularization technique except that of $(\Theta_i^k - \Theta_i^{k-1})$. $\Theta_i^{k-1}, \Theta_i^k$ respectively denote the network parameters before and after seeing the current $k-th$ task. λ_3, λ_4 stand for predefined constants controlling the influence of each term.

(5.11) introduces the KL divergence term to induce wide local optimum regions where it is perceivable as the optimization $\min_{Q \in \mathcal{Q}} \mathcal{L}_{KL}(Q|P)$ where P is a given distribution and $\mathcal{Q}$ is a convex set of distributions in the probability simplex Δ_m [62] where $g_{W_C}(.) : \Re^D \rightarrow \Delta_M$ and M is the label space dimension. After minimizing this term, P^* is obtained and represents a distribution in $\mathcal{Q}$ that is the closest to P. Such a case presents $\mathcal{L}_{KL}(Q|P)$ as the Euclidean distance where (Q, P^*, P) constructs a right triangle.

Lemma 1 [62]. Suppose $P^* \in Q$ such that $\mathcal{L}_{KL}(P^*|P) = \min_{Q \in \mathcal{Q}} \mathcal{L}_{KL}(Q|P)$, then

$$\mathcal{L}_{KL}(Q|P) \geq \mathcal{L}_{KL}(Q|P^*) + \mathcal{L}_{KL}(P^*|P), \forall Q \in \mathcal{Q} \tag{5.12}$$

The classifier's output $g_{W_C}(.)$ is projected to a uniform distribution in a set $\mathcal{C}$ and interpreted as conditional distributions $Q_{Y|X}$ between target Y and input X. After learning the $k-th$ task with (5.11), $P^{k*}_{Y|X} \in \mathcal{C}$ is obtained. Suppose the presence of two classifiers in the $k+1-th$ task $P^{k+1}_{Y|X} \notin \mathcal{C}$ w/o the projection and $P^{(k+1)*}_{Y|X} \in \mathcal{C}$ with the projection, a right triangle is established across $(P^{k*}_{Y|X}, P^{(k+1)}_{Y|X}, P^{(k+1)*}_{Y|X})$ as per Lemma 1. $P^{k*}_{Y|X}$ is closer to $P^{(k+1)*}_{Y|X}$ than that without the projection $P^{(k+1)}_{Y|X}$ when being evaluated on the $k-th$ task. A uniform distribution P_U is selected because it represents the centroid of Δ_M that results in an upper bound of divergence $\log M$. Note that an ideal classifier performing well on all tasks does not exist in the continual learning context, and thus we turn our attention to the set of possible classifiers $\mathcal{C}$ to be a KL divergence ball centered at the uniform distribution P_U [62]. The wide learning strategy is illustrated in Figure 5.2.

5.3.2.3 Loss Function of Few-Shot Learning Phase

The few-shot learning phase of FLOWER is controlled by three loss functions: cross entropy loss function $\mathcal{L}_{ce}$, ball generator loss function $\mathcal{L}_{ball}$, and regularizer $\mathcal{R}_{PMAS}$. The cross entropy loss function (5.2) learns the current concept supported by the feature space data augmentation method, while the ball generator loss function (5.10) addresses the synthetic sample bias problem, where a synthetic sample is pulled closely to its centroid and pushed away from other centroids. The

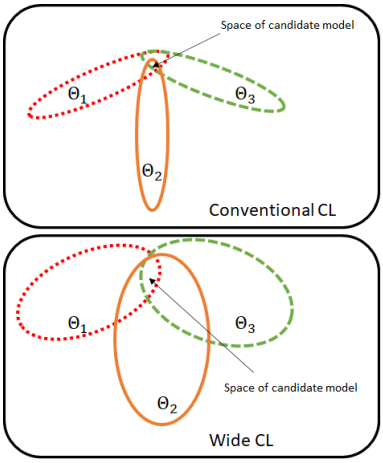


Figure 5.2: Wide Continual Learning: conventional continual learning approaches generate narrow ellipsoids where the local minima region is sharp, leaving very small space for model candidate parameters, i.e., intersections of all ellipsoids. On the contrary, the wide continual learning generates wide ellipsoids having wide local minima regions, allowing enough space for model candidate parameters.

regularizer (5.11) functions to protect against the catastrophic forgetting problem:

$$\mathcal{L}_{k>1} = \mathcal{L}_{ce} + \mathcal{L}_{ball} + \mathcal{R}_{PMAS} \tag{5.13}$$

The network parameters are maintained in such a way as to satisfy the b-flat-wide local minima conditions. That is, parameters of the feature extractor are clamped if they go beyond the flat-wide region. Besides, the wide learning paradigm is actualized in the projection mechanism to scale FLOWER up to long sequences of tasks. (5.13) takes the advantage of the flat learning concept [42] and the wide learning concept [62] to address the CF problem without any memory and thus realizes the flat-wide learning strategy. Pseudo-codes of FLOWER are provided in Algorithm 1, while an overview of flat-wide learning strategies is pictorially illustrated in Figure 5.3.

5.3.3 Algorithm

Algorithm 1 provides an overview of FLOWER learning algorithm. In the base learning phase, FLOWER is trained to discover a flat-wide local region minimizing (5.3). This is achieved by injecting small perturbations ϵ_j to the network paramet-

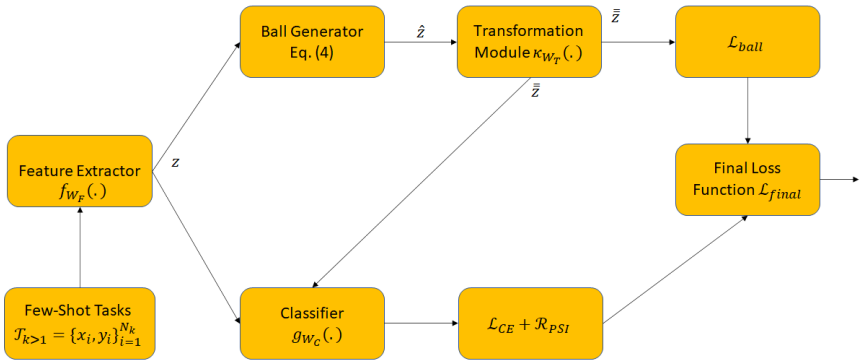


Figure 5.3: Few Shot Learning Phase of FLOWER is driven by the three loss functions, the cross entropy loss function, the ball generator loss function, and the projection-based synaptic intelligence regularizer. It combines wide and flat learning paradigms supported by the feature space data augmentation module, with a transformation module.

ers of FLOWERS and projections of output logits to the uniform distributions. Note that the projections to the uniform distributions are absent in [42]. This loop is repeated M times to unveil the flat-wide local regions. In the few-shot learning tasks, the learning objective is to minimize (5.13) comprising the cross-entropy loss function, the ball generator loss function, and the wide regularization strategy. The learning process is guided by the ball generator, augmenting a few data samples of the current tasks. It is worth noting that the ball generator prevents the synthetic sample bias problem because of (5.10), making the synthetic samples revolve around a compact region. Besides, the MAS-based regularization (5.11) promotes the wide learning term, driving the flat local region to be wide. The flat-wide regions are maintained during the few-shot learning phase because the network parameters are clamped whenever going beyond the flat-wide region to overcome the catastrophic forgetting problem $W_{F^*} - b \leq W_F \leq W_{F^*} + b$.

Algorithm 4 Learning Policy of FLOWER

Input: The flat-wide region bound b, randomly initialized parameters $\Theta = \{W_F, W_C\}, W_T$, hyperparameters $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, number of epochs E, number of synthetic samples S.
Output: Updated Network Parameters $\Theta = \{W_F, W_C\}$
//Base Learning Phase $k = 1$
for $e = 1 : E$ **do**
  for $j = 1 : M$ **do**
    Sample a noise vector $\epsilon_j \backsim \mathcal{E}$ where $-b \leq \epsilon_j \leq b$
    Perturb the feature extractor parameters $\Theta = \{W_F + \epsilon_j, W_C\}$
    Compute the base loss $\mathcal{L}_{base}$
    Reset the network parameters $\Theta = \{W_F, W_C\}$
  end for
  Update network parameters with $\mathcal{L}_{k=1}$ as per Eq. (5.3)
end for
//Few Shot Learning Phase $k > 1$
for $k = 2 : K$ **do**
  for $e = 1 : E$ **do**
    $\mathcal{T}'_k = \emptyset$
    Calculate the ball/class parameters $\omega(Z_c) = \{C_c, \sigma_c\}$
    for $i' = 1 : S$ **do**
      Generate a synthetic sample $\hat{z}_{i'}$ as per Eq. (5.9)
      Apply the transformation module $\hat{\hat{z}} = \kappa_{W_T}(\hat{z}_{i'})$ to induce a unbiased synthetic sample $\hat{\hat{z}}_{i'}$
      $\mathcal{T}'_k = \mathcal{T}'_k \cup \{\hat{\hat{z}}_{i'}, y_i\}$
    end for
    $\hat{\mathcal{T}}^e_k = \mathcal{T}'_k \cup \mathcal{T}_k$
    Calculate the ball generator loss with $\hat{\mathcal{T}}^e_k$ as per Eq. (5.10)
    Calculate the cross entropy loss with $\hat{\mathcal{T}}^e_k$ as per Eq. (5.2)
    Calculate the final loss as per Eq. (5.13)
    Update network parameters $\Theta = \{W_F, W_C\}, W_T$ with $\mathcal{L}_k$
    Clamp feature extractor parameters $W_{F^*} - b \leq W_F \leq W_{F^*} + b$ to meet the b-flat-wide local minima conditions
  end for
end for

5.4 Theoretical Analysis

Lemma 2 [42]**:** Under assumptions 1 and 2, the following can be derived:

$$\mathbb{E}_{z_k,\epsilon_j}[R(\Theta_{k+1})] - R(\Theta_k) \leq -\lambda_k \frac{2M - \lambda_k L(m_2 + M)}{2M} ||\nabla R(\Theta_k)||_2^2 + \frac{\lambda_k^2 L m_1}{2M} \tag{5.14}$$

Proof. Assumption 1 leads to the following for all $\{\Theta, \Theta'\} \in \Re^d$.

$$R(\Theta) \leq R(\Theta') + \nabla R(\Theta')^T(\Theta - \Theta') + \frac{1}{2}L||\Theta - \Theta||_2^2 \tag{5.15}$$

Given the parameter update $\Theta_{k+1} = \Theta_k - \frac{\lambda_k}{M}\sum_{j=1}^{M} \nabla \mathcal{L}_{base}^{j,k}$ and $\hat{s} = \frac{1}{M}\sum_{j=1}^{M} s_{j,k}$ where $s_{j,k} = \mathcal{L}_{base}^{j,k}$, we can go further:

$$\begin{aligned} R(\Theta_{k+1}) - R(\Theta_k) &\leq \nabla R(\Theta_k)^T(\Theta_{k+1} - \Theta_k) + \\ \frac{1}{2}L||\Theta_{k+1} - \Theta_k||_2^2 &\leq -\lambda_k \nabla R(\Theta_k)^T \hat{s} + \frac{\lambda_k^2 L}{2}||\hat{s}||_2^2 \end{aligned} \tag{5.16}$$

Recall that $\mathbb{E}_{z_k,\epsilon_j}[.]$ denotes the expectation w.r.t the joint distribution of random variables z_k, ϵ_j which can be applied on both sides of (5.16).

$$\mathbb{E}_{z_k,\epsilon_j}[R(\Theta_{k+1})] - R(\Theta_k) \leq -\lambda_k \nabla R(\Theta_k)^T \mathbb{E}_{z_k,\epsilon_j}[\hat{s}] + \frac{\lambda_k^2 L}{2}\mathbb{E}_{z_k,\epsilon_j}[||\hat{s}||_2^2] \tag{5.17}$$

Because of the second condition of assumption 2 and the fact that Θ_{k+1} is a factor of z_k, ϵ_j, $\mathbb{E}_{z_k,\epsilon_j}[\hat{s}]$ meets the following condition.

$$\mathbb{E}_{z_k,\epsilon_j}[\hat{s}] = \frac{1}{M}\sum_{j=1}^{M} \mathbb{E}_{z_k,\epsilon_j}[s_{j,k}] = \nabla R(\Theta_k) \tag{5.18}$$

Suppose ϵ_j is sampled without replacement from $P(\epsilon)$, following [190] and condition 3 of assumption 2, we can obtain:

$$\mathbb{V}_{z_k,\epsilon_j}[\hat{s}] \leq \frac{\mathbb{V}_{z_k,\epsilon_j} s_{j,k}}{M} \leq \frac{m_1}{M} + \frac{m_2}{M}||\nabla R(\Theta_k)||_2^2 \tag{5.19}$$

Combining (5.18) and (5.19), we find:

$$\mathbb{E}_{z_k,\epsilon_j}[||\hat{s}||_2^2] = \mathbb{V}_{z_k,\epsilon_j}[\hat{b}] + ||\mathbb{E}_{z_k,\epsilon_j}[\hat{s}]||_2^2 \leq \frac{m_1}{M} + \frac{m_2 + M}{M}||\nabla R(\Theta_k)||_2^2 \tag{5.20}$$

The proof is completed by combining (5.17),(5.18),(5.20).

Lemma 3 [42]: Because of assumptions 1-3, the following lemma is derived.

$$\liminf_{k\to\infty} \mathbb{E}[||\nabla R(\Theta_k)||_2^2] = 0 \tag{5.21}$$

Proof. Note that $\lim_{k\to\infty} \lambda_k = 0$ as per assumption 3 leading to $k \in \mathbb{N}, \lambda_k L(m_2 + M) \leq M$ without loss of generalization. Suppose that $\mathbb{E}[.]$ stands for the total expectations across all random variables z_k, ϵ_j, we can express $\mathbb{E}[R(\Theta_k)]$:

$$\mathbb{E}[R(\Theta_k)] = \mathbb{E}_{z_0,\epsilon_0}\mathbb{E}_{z_1,\epsilon_1}...\mathbb{E}_{z_{k-1},\epsilon_{k-1}}[R(\Theta_k)] \tag{5.22}$$

Applying the total expectation to (5.14), we observe:

$$\mathbb{E}[R(\Theta_{k+1})] - \mathbb{E}[R(\Theta_k)] \leq -\frac{\lambda_k}{2}\mathbb{E}[||\nabla R(\Theta_k)||_2^2] + \frac{\lambda_k^2 L m_1}{2M} \tag{5.23}$$

For $k = [0, K]$, summation is applied to both sides of (5.23):

$$R^* - \mathbb{E}[R(\Theta_1)] \leq \mathbb{E}[R(\Theta_{K+1})] - \mathbb{E}[R(\Theta_0)] \leq \frac{-1}{2}\sum_{k=0}^{K} \lambda_l \mathbb{E}[||\nabla R(\Theta_k)||_2^2] + \frac{L m_1}{2M}\sum_{k=0}^{K} \lambda_k^2 \tag{5.24}$$

where R^* is the lower bound of condition 1 of assumption 2. we further derive (5.24):

$$\sum_{k=0}^{K} \lambda_k \mathbb{E}[||\nabla R(\Theta)||_2^2] \leq 2(\mathbb{E}[R(\Theta_1)] - R^*) + \frac{L m_1}{M}\sum_{k=0}^{K} \lambda_k^2 \tag{5.25}$$

Using the second condition of assumption 3, we obtain:

$$\lim_{K\to\infty} \mathbb{E}[\sum_{k=0}^{K} \lambda_k ||\nabla R(\Theta_k)||_2^2] \leq 2(\mathbb{E}[R(\Theta_0)] - R^*) + \lim_{k\to\infty} \frac{L m_1}{M}\sum_{k=0}^{K} \lambda_k^2 < \infty \tag{5.26}$$

We divide both sides (5.25) by $\sum_{k=1}^{K} \lambda_k$ and use assumption 3:

$$\lim_{K\to\infty} \mathbb{E}[\frac{\sum_{k=1}^{K} \lambda_k ||\nabla R(\Theta_k)||_2^2}{\sum_{k=1}^{K} \lambda_2}] = 0 \tag{5.27}$$

(5.27) is a weighted average operation with λ_k as a weight. As a result, $||\nabla R(\Theta_k)||_2^2$

must asymptotically approach zero, thus proving (5.21).

Proof of Theorem 1 [42]: Recall that because of assumptions 1-3 and that the risk function R is twice differentiable and as a result of assumption 1 $||\nabla R(\Theta)||_2^2$ is L_2-smooth with $L_2 > 0$, the following holds.

$$\lim_{k\to\infty} \mathbb{E}[||\nabla R(\Theta_k)||_2^2] = 0 \tag{5.28}$$

Proof. Let $Q(\Theta) = ||R(\Theta)||_2^2$, we observe:

$$\begin{aligned}
\mathbb{E}_{z_k,\epsilon_j}[Q(\Theta_{k+1})] - Q(\Theta_k) &\leq \nabla Q(\Theta_k)^T \mathbb{E}_{z_k,\epsilon_j}[(\Theta_{k+1} - \Theta_k)] \\
&\quad + \frac{1}{2} L_2 \mathbb{E}_{z_k,\epsilon_j}[||\Theta_{k+1} - \Theta_k||_2^2] \\
&\leq -\lambda_k \nabla Q(\Theta_k)^T \mathbb{E}_{z_k,\epsilon_j}[\hat{s}] + \frac{\lambda_k^2 L_2}{2} \mathbb{E}_{z_k,\epsilon_k}[||\hat{s}||_2^2] \\
&\leq -2\lambda_k \nabla R(\Theta_k)^T \nabla^2 R(\Theta_k) \mathbb{E}_{z_k,\epsilon_j}[\hat{g}] + \frac{\lambda_k^2 L_2}{2} \mathbb{E}_{z_k,\epsilon_j}[||\hat{s}||_2^2] \\
&\leq 2\lambda_k L ||\nabla R(\Theta_k)||_2^2 ||\nabla^2 R(\Theta_k)||_2 ||\mathbb{E}_{z_k,\epsilon_j}||_2 + \frac{\lambda_k^2 L_2}{2} \mathbb{E}_{z_k,\epsilon_j}[||\hat{g}||_2^2] \\
&\leq 2\lambda_k L ||\nabla R(\Theta_k)||_2^2 + \frac{\lambda_k^2 L_2}{2} (\frac{m1}{M} + \frac{m_2 + M}{M} ||\nabla R(\Theta_k)||_2^2)
\end{aligned} \tag{5.29}$$

Using the total expectations to both sides of (5.29) leads to

$$\mathbb{E}[Q(\Theta_{k+1})] - \mathbb{E}[F(\Theta_k)] \leq 2\lambda_2 L \mathbb{E}[||\nabla R(\Theta_k)||_2^2] + \frac{\lambda_k^2 L_2}{2} (\frac{m_1}{M} + \frac{m_2 + M}{M} \mathbb{E}[||\nabla R(\Theta_k)||_2^2]) \tag{5.30}$$

$2\lambda_k L \mathbb{E}[||\nabla R(\Theta_k)||_2^]$ is the term of a convergent sum as per (5.26) as well as that $\frac{\lambda_k^2 L_2}{2} (\frac{m_1}{M} + \frac{m_2+M}{M} \mathbb{E}[||\nabla R(\Theta_k)||_2^2])$ because of the convergence of $\sum_{k=1}^{\infty} \lambda_k^2$. Therefore, the upper bound of (5.30) implies a convergent sum. We define:

$$A_K^+ = \sum_{k=0}^{K-1} \max\left(0, \mathbb{E}[Q(\Theta_{k+1})] - \mathbb{E}[Q(\Theta_k)]\right) \tag{5.31}$$

$$A_K^- = \sum_{k=0}^{K-1} \max\left(0, \mathbb{E}[Q(\Theta_k)] - \mathbb{E}[Q(\Theta_{k+1})]\right) \tag{5.32}$$

As per (5.30), the bound of $\mathbb{E}[Q(\Theta_{k+1})] - \mathbb{E}[Q(\Theta_k)]$ are positive and the term of convergent sum. A_K^+ and A_K^- converge as the sum of the bound of $\mathbb{E}[Q(\Theta_{K+1})] -$

$\mathbb{E}[Q(\Theta_k)]$. $Q(\Theta_K)$ also converges due to $Q(\Theta_K) = Q(\Theta_0) + A_K^+ - A_K^-$ for $K \in \mathbb{N}$. We can then prove the theorem 1 also justified by Lemma 3.

5.5 Experiments Result and Analysis

This section presents numerical validations of FLOWER: comparisons with prior art approaches with different sizes of base tasks, an ablation study, and sensitivity analyses of the bound b and the number of trials M.

5.5.1 Datasets

Our numerical study is carried out using CIFAR100, miniImageNet, and CUB-200-2011. The CIFAR100 and miniImageNet problems are configured such that 60 classes are reserved for the base task, while the remaining 40 classes are the few-shot tasks under the 5-way 5-shots configuration, i.e., 5 classes are sampled where each class comprises 5 samples, thus leading to 8 few-shot tasks. For the CUB-200-2011 problem with 200 classes, the base task is built upon 100 classes, while the few-shot tasks utilize the remaining 100 classes with the 10-way 5-shot configuration, thus forming 10 few-shot tasks. In addition, we also evaluate consolidated algorithms in the case of small base tasks where only 20 classes are included in the miniImagenet and CIFAR100 problems, while 50 classes are presented for the CUB problem. Numerical results under a 1-shot configuration are also reported here.

5.5.2 Baselines

FLOWER is compared with 10 state-of-the-art algorithms: iCaRL [31], Rebalance [34], FSLL [56], cRT [191], TOPIC [40], EEIL [162], F2M [42], S3C [192], FACT [121], and SSFE-Net [193] (MiniImageNet and CUB only). The baseline algorithm follows [42], where the training process is only carried out in the base task, while only exemplar constructions are done for few-shot tasks for a simple nearest class mean classification algorithm. The FlatWide algorithm is added, where it presents the case of flat and wide learning, only without any data augmentation procedure. A joint training is also included, where the training process encompasses

base classes and few-shot classes. Although the joint-training method commonly serves as **the upper bound** for the continual learning problems, it is not strong enough here because the extreme imbalance problem between the base task and the few-shot tasks exists in the few-shot continual learning problems. Here, the cRT is added as **the additional upper bound** because it performs long-tailed classification trained with all encountered data [42]. Numerical results of ICARL, Rebalance, EEIL, TOPIC, cRT, SSFE-Net, and joint-training are taken from [40], [42], and [193] while we run ICARL, Rebalance, Baseline, FlatWide, F2M, S3C, FACT and FSLL in the same computational environments, applying the implementation of [42] with their best hyperparameters found with the grid search.

5.5.3 Experiment Setup

Implementation Details: our numerical study is executed under single NVIDIA A100 GPU with 40 GB memory across five runs. ResNet18 is adopted as the backbone network for all methods in the miniimagenet and CUB-200-2011 problems, while ResNet20 is implemented as a backbone network for all methods in the CIFAR100 problem. As with [42], M is set in between 2 to 4 and noise is injected to the last 4 or 8 convolutional layers while the flat region bound b is assigned as 0.01. Epoch per session is selected as 15 and the learning rate is assigned as 0.2 initially and truncated by 1e-6 after 5 epochs. hyperparameters are detailed in Table 5.1.

Evaluation Metrics: We apply three evaluation metrics, average accuracy (ACC), and Harmonic Mean Accuracy (HMACC), to assess the advantage of consolidated algorithms following [192]. HMACC is the balance between stability performance represented by base classes accuracy (BACC) and plasticity performance represented by novel classes accuracy (NACC) Suppose that $R_{i,j}$ stands for the test classification accuracy of the model on task t_j for the samples that belong to task

Table 5.1: Hyperparameters setting for our experiments.

hyperparameter	**value**
image_size	32 (CIFAR), 84 (MiniImageNet), 224(CUB)
networks	ResNet18, ResNet20(CIFAR)
optimizer	SGD
bacth_size	256
epoch	
- base task	- 1000, 600 (CUB)
- continual tasks	- 15, generate prototype only (CUB)
initial learning rate (lr)	
- base task	- 0.1, 0.001 (CUB)
- continual tasks	- 0.2
gamma	
- base task	- 0.1, 1e-6 (CUB)
- continual tasks	- 1e-6
weight_decay	5e-4
momentum	0.9
random_noise	
- distribution	- DiscreteBeta
- type	- suffix_conv_weight
- num_layers	- 4 or 8
- low	- 0.1
- high	- 5.0, 3.0 (CUB)
- reduction_factor	- 4
- bound_value	- 0.01, 0.0002 (SUB)
- random_times	- 2-4
lambda	
- lambda1	- 1.0
- lambda2	- 1.0
- lambda3	- 10
- lambda4	- 100

t_i, the evaluation metrics are expressed as follows:

$$\begin{aligned} ACC_{\uparrow} &= \frac{1}{T}\sum_{i=1}^{T} R_{i,T} \\ HMACC_{\uparrow} &= \frac{BACC + NACC}{2} \\ BACC_{\uparrow} = R_{1,T}\textbf{, and } NACC_{\uparrow} &= \frac{1}{T-1}\sum_{i=2}^{T} R_{i,T} \end{aligned} \tag{5.33}$$

where the larger these values, the better the model.

Table 5.2: Classification accuracy on split MiniImageNet dataset with 60 and 20 base classes averaged across 5 times run, #B indicates the number of base classes, * indicates the results are copied from F2M, + indicates the results copied from TOPIC.

Method	Session									Avg	Gap
	1	2	3	4	5	6	7	8	9		
Split MiniImageNet Base classes = 60											
cRT*	67.30	64.15	60.59	57.32	54.22	51.43	48.92	46.78	44.85	55.06	-4.5
Joint-Training*	67.30	62.34	57.79	54.08	50.93	47.65	44.65	42.61	40.29	51.96	-7.6
Baseline	66.78	61.29	57.25	54.17	51.54	49.00	46.56	44.99	43.78	52.82	-6.7
F2M	66.07	61.05	56.82	53.51	50.76	48.26	45.79	44.07	42.62	52.11	-7.4
FSLL	66.98	57.23	52.47	49.66	47.45	45.20	43.29	42.06	41.18	49.50	-10.0
ICARL	66.05	56.47	53.26	50.14	47.55	45.08	42.47	41.04	39.60	49.07	-10.5
Rebalance	66.45	60.66	55.61	51.38	47.93	44.64	41.40	38.75	37.11	49.33	-10.2
ICARL+	61.31	46.32	42.94	37.63	30.49	24.00	20.89	18.80	17.21	33.29	-26.3
EEIL+	62.31	46.58	44.00	37.29	33.14	27.12	24.10	21.57	19.58	35.08	-24.5
Rebalance+	63.31	47.80	39.31	31.91	25.68	21.35	18.67	17.24	14.17	31.05	-28.5
TOPIC+	64.31	50.09	45.17	41.16	37.48	35.52	32.19	29.46	24.42	39.98	-19.6
SSFE-Net	72.06	66.17	62.25	59.74	56.36	53.85	51.96	49.55	47.73	57.74	-1.8
FACT	67.90	62.75	58.52	54.84	51.81	48.96	46.36	44.23	42.40	53.09	-6.5
S3C	71.70	66.61	62.28	58.81	55.69	52.80	50.13	48.17	46.47	56.96	-2.6
FlatWide (Ours)	74.60	69.45	64.55	60.50	56.66	52.87	48.74	45.08	43.62	57.34	-2.2
FLOWER (Ours)	**74.60**	**69.43**	**65.14**	**61.49**	**58.16**	**55.18**	**52.36**	**50.52**	**49.06**	**59.55**	**0.0**
Split MiniImageNet Base classes = 20											
Baseline	67.90	46.32	37.24	30.13	25.56	22.11	20.03	18.14	16.80	31.58	-4.7
F2M	65.45	45.70	36.15	29.19	24.73	21.29	19.18	17.18	15.91	30.53	-5.8
FSLL	68.05	47.05	37.74	30.70	26.00	22.43	20.26	18.22	17.11	31.95	-4.4
ICARL	60.70	38.91	30.65	25.30	21.46	17.97	16.38	14.58	13.77	26.64	-9.7
Rebalance	67.55	44.89	33.67	29.04	25.32	22.19	20.73	18.88	17.70	31.11	-5.2
FACT	74.90	50.88	40.02	32.48	27.43	23.53	20.82	18.51	16.93	33.94	-2.4
S3C	76.80	53.31	42.97	**35.61**	**30.55**	**26.39**	**24.14**	**21.81**	**20.39**	**36.89**	**0.6**
FlatWide (Ours)	79.05	53.93	42.62	34.48	28.78	24.52	21.55	18.63	17.04	35.62	-0.7
FLOWER (Ours)	**79.05**	**54.13**	**43.03**	34.69	29.20	25.28	22.49	20.22	18.79	**36.32**	**0.0**
Split CIFAR100 Base classes = 60											
cRT*	65.18	63.89	60.20	57.23	53.71	50.39	48.77	47.29	45.28	54.66	-4.8
Joint-Training*	65.18	61.45	57.36	53.68	50.84	47.33	44.79	42.62	40.08	51.48	-7.9
Baseline	72.20	67.88	64.06	60.43	57.28	54.54	52.83	50.93	48.80	58.77	-0.6
F2M	71.40	66.65	63.20	59.54	56.61	54.08	52.20	50.49	48.40	58.06	-1.4
FSLL	72.52	64.20	59.60	55.09	52.85	51.57	51.21	49.66	47.86	56.06	-3.4
ICARL	71.72	64.86	60.46	56.61	53.76	51.10	49.10	46.95	44.64	55.47	-4.0
Rebalance	74.57	66.65	60.96	55.59	50.87	46.55	43.82	40.29	37.23	52.95	-6.5
ICARL+	57.78	46.31	33.79	28.59	24.98	21.33	19.07	17.05	16.25	29.46	-30.0
EEIL+	57.78	41.32	35.19	29.95	25.65	23.20	22.19	20.61	18.53	30.49	-28.9
Rebalance+	57.78	48.91	41.91	38.05	30.61	26.68	24.79	22.15	19.50	34.49	-24.9
TOPIC+	57.78	49.49	44.12	39.82	35.07	31.42	27.82	25.47	24.17	37.24	-22.2
FACT	68.18	63.48	59.24	55.49	52.12	49.37	47.00	44.81	42.67	53.60	-5.8
S3C	70.20	66.34	62.53	58.82	55.99	53.26	51.55	49.93	47.57	57.35	-2.1
FlatWide (Ours)	73.40	68.83	64.81	60.48	57.06	54.27	52.39	49.81	46.80	58.65	-0.8
FLOWER (Ours)	**73.40**	**68.98**	**64.98**	**61.20**	**57.88**	**55.14**	**53.28**	**51.16**	**48.78**	**59.42**	**0.0**
Split CIFAR100 Base classes = 20											
Baseline	74.75	54.39	41.83	35.24	30.72	27.67	24.91	22.82	20.94	37.03	-3.3
F2M	73.40	53.25	42.10	35.42	31.03	28.12	24.95	23.04	21.38	36.97	-3.4
FSLL	75.00	52.76	39.87	33.29	29.74	26.73	24.12	22.22	20.28	36.00	-4.3
ICARL	76.85	53.58	41.51	34.65	30.22	27.15	23.73	21.65	19.62	36.55	-3.8
Rebalance	73.90	49.27	36.95	31.73	28.62	26.68	24.44	22.62	20.86	35.01	-5.3
FACT	78.40	54.61	42.41	34.75	29.68	25.60	22.25	20.08	18.06	36.20	-4.1
S3C	76.15	55.02	43.30	**36.70**	**32.33**	**29.59**	**26.60**	**24.12**	**22.28**	**38.45**	**-1.9**
FlatWide (Ours)	83.20	58.89	46.02	38.28	33.09	28.99	25.77	23.29	19.75	39.70	-0.6
FLOWER (Ours)	**83.20**	**58.93**	**46.01**	**38.56**	**33.48**	**29.54**	**26.49**	**24.40**	**22.27**	**40.32**	**0.0**

Table 5.3: Classification accuracy on split CUB 2011 dataset with 100 and 50 base classes averaged across 5 runs, #B indicates the number of base classes, * indicates the results are copied from F2M, + indicates the results copied from TOPIC, ∘ indicates the method is run 1 time due to system crash.

Method	Session											Avg	Gap
	1	2	3	4	5	6	7	8	9	10	11		
Base classes = 100													
cRT*	80.83	78.51	76.12	73.93	71.46	68.96	67.73	66.75	64.22	62.53	61.08	70.19	4.52
Joint-training*	80.83	77.57	74.11	70.75	68.52	65.97	64.58	62.22	60.18	58.49	56.78	67.27	1.60
Baseline	76.92	72.83	68.96	64.74	62.38	59.63	58.44	57.25	54.97	54.33	53.49	62.18	-3.49
F2M	77.41	73.50	69.52	65.27	63.07	60.41	59.20	58.02	55.89	55.49	54.51	62.94	-2.73
FSLL	76.92	70.58	64.73	57.77	55.96	54.34	53.62	53.04	50.45	50.36	49.48	57.93	-7.74
ICARL	75.73	69.97	65.95	61.27	57.62	55.50	54.16	53.37	50.75	50.35	48.77	58.49	-7.18
Rebalance	74.83	55.51	48.36	42.38	36.98	32.82	30.15	27.82	25.96	24.44	22.83	38.37	-27.30
ICARL+	68.68	52.65	48.61	44.16	36.62	29.52	27.83	26.26	24.01	23.89	21.16	36.67	-29.00
EEIL+	68.68	53.63	47.91	44.20	36.30	27.46	25.93	24.70	23.95	24.13	22.11	36.27	-29.40
Rebalance+	68.68	62.55	50.33	45.07	38.25	32.58	28.71	26.28	23.80	19.91	17.82	37.63	-28.04
TOPIC+	68.68	62.49	54.81	49.99	42.25	41.40	38.35	35.36	32.22	28.31	26.28	43.65	-22.02
SSFE-Net	76.38	72.11	68.82	64.77	63.59	60.56	59.84	58.93	57.33	56.23	54.28	62.99	-2.68
FACT	77.93	74.30	70.93	67.16	65.10	62.44	61.27	60.49	58.51	58.22	57.07	64.86	-0.81
S3C	79.43	75.20	71.06	67.13	64.77	61.71	60.34	59.22	57.01	56.26	54.81	64.27	-1.40
FlatWide (Ours)	78.98	75.72	72.03	67.77	65.63	62.99	61.46	60.25	57.16	55.55	51.1	64.42	-1.25
FLOWER (Ours)	**79.02**	**75.77**	**72.01**	**67.96**	**65.99**	**63.38**	**62.14**	**61.12**	**58.96**	**58.52**	**57.49**	**65.67**	0.00
Base classes = 50													
Baseline	77.83	66.94	61.18	59.13	54.85	49.96	46.73	43.93	42.30	39.98	39.74	52.96	-4.86
F2M	78.11	68.52	63.50	61.61	57.58	52.64	49.26	46.43	44.81	42.71	42.41	55.23	-2.59
FSLL	77.61	65.96	60.37	58.59	55.96	51.75	48.35	45.84	44.92	42.80	42.62	54.07	-3.75
ICARL	75.88	62.54	57.70	54.73	49.16	43.96	40.84	38.09	36.63	34.38	34.79	48.06	-9.76
Rebalance	74.73	60.91	56.29	53.93	48.31	42.49	37.40	33.42	31.32	29.29	27.93	45.09	-12.73
FACT	78.62	70.54	65.98	63.17	59.54	54.97	**52.27**	**50.15**	**48.85**	**46.76**	**46.39**	**57.93**	**0.11**
S3C	80.99	70.77	63.68	59.93	55.09	50.45	47.16	44.46	42.94	40.62	40.01	54.19	-3.63
FlatWide (Ours)	79.63	71.48	66.39	64.38	60.19	55.41	51.61	48.91	47.43	44.86	44.18	57.68	-0.14
FLOWER (Ours)	**79.63**	**71.14**	**66.18**	**64.22**	**60.03**	**55.17**	51.74	49.13	47.79	45.68	45.29	**57.82**	**0.00**

Table 5.4: Summarized performance of consolidated methods with 1-shot setting on MiniImageNet, CIFAR-100, and CUB-200 dataset averaged across 5 time runs,* denotes only run 1 time due to system crash.

Method	MiniImageNet 1-shot				CIFAR100 1-shot				CUB2011 1-shot			
	Session			Gap	Session			Gap	Session			Gap
	1	9	Avg		1	9	Avg		1	11	Avg	
Baseline	66.78	40.51	51.59	-6.16	72.20	45.36	57.01	-0.42	76.92	43.13	56.43	-1.47
F2M	66.07	39.86	50.90	-6.85	71.40	44.57	56.07	-1.36	77.41	42.55	56.35	-1.55
FSLL	66.98	37.58	46.49	-11.26	72.52	43.61	52.89	-4.55	76.92	38.79	51.58	-6.32
ICARL	66.05	1.00	8.44	-49.32	71.72	1.00	9.07	-48.36	75.73	0.52	7.52	-50.38
Rebalance	66.45	39.86	51.19	-6.56	**74.57**	44.72	**57.44**	0.01	74.83	20.43	38.78	-19.12
FACT	67.90	41.02	52.40	-5.35	68.18	41.20	52.75	-4.68	77.93	44.28	57.52	-0.38
S3C	71.70	43.72	55.51	-2.25	70.20	44.14	55.50	-1.93	78.39	**44.45**	**58.13**	**0.23**
FlatWide (Ours)	74.60	41.19	56.06	-1.69	73.40	44.02	56.85	-0.58	78.98	43.60	57.84	-0.06
FLOWER (Ours)	**74.60**	**45.66**	**57.75**	**0.00**	**73.40**	**45.27**	**57.43**	**0.00**	**79.02**	**44.26**	**57.90**	**0.00**

Table 5.5: Harmonic Mean Accuracy (HMACC) on split MiniImageNet dataset with 60 and 20 base classes averaged across 5 runs.

Method	Session								Avg	Gap
	2	3	4	5	6	7	8	9		
Base classes = 60										
Baseline	44.31	41.74	41.32	41.28	40.48	39.52	39.55	39.81	41.00	-3.94
F2M	40.92	39.25	39.07	39.04	38.73	37.99	38.04	38.15	38.90	-6.05
FSLL	34.85	34.43	35.22	35.78	35.78	35.56	36.18	36.76	35.57	-9.37
FACT	38.08	37.94	37.78	37.63	37.22	36.81	36.72	36.72	37.36	-7.58
S3C	42.44	42.21	42.61	42.00	41.54	40.92	41.16	41.12	41.75	-3.19
FLOWER	47.29	46.48	45.96	44.93	44.13	43.31	43.53	43.91	44.94	**0.00**
Base classes = 20										
Baseline	39.38	37.24	34.41	32.90	31.78	31.26	30.69	30.27	33.49	-6.03
F2M	37.99	36.15	33.83	32.59	31.59	31.14	30.49	30.05	32.98	-6.54
FSLL	38.59	37.75	35.72	34.46	33.29	32.73	32.00	31.71	34.53	-4.99
FACT	40.63	40.02	38.58	37.71	36.94	36.48	35.96	35.58	37.74	-1.78
S3C	43.93	42.97	41.24	40.07	38.91	38.64	37.97	37.35	40.13	**0.62**
FLOWER	44.78	43.02	40.49	39.08	38.10	37.50	36.84	36.35	39.52	**0.00**

Table 5.6: Summarized Harmonic Mean Accuracy (HMACC) on split CIFAR100 dataset with 60 and 20 base classes, and split CUB 2011 with 100 and 50 base classes averaged across 5 runs.

Method	Split CIFAR-100				Split CUB 2011			
	Session		Avg	Gap	Session		Avg	Gap
	2	9			2	11		
	Base classes = 60				Base classes = 100			
Baseline	52.14	44.67	47.34	0.59	63.22	53.67	55.42	-3.16
F2M	51.98	44.58	47.24	0.49	63.01	54.70	55.95	-2.63
FSLL	45.82	43.49	43.61	-3.14	60.24	49.59	51.71	-6.87
FACT	41.72	37.17	38.56	-8.18	60.93	57.28	57.35	-1.23
S3C	57.70	44.03	47.96	**1.21**	65.27	55.03	56.92	-1.66
FLOWER	51.19	44.21	46.75	**0.00**	64.94	57.69	58.58	**0.00**
	Base classes = 20				Base classes = 50			
Baseline	47.93	33.88	38.58	-3.78	60.82	47.28	53.04	-5.57
F2M	46.20	33.08	38.14	-4.22	62.45	49.90	55.51	-3.11
FSLL	45.34	30.53	36.10	-6.26	59.54	47.39	53.34	-5.27
FACT	45.83	34.36	38.62	-3.74	63.67	54.58	58.65	**0.04**
S3C	47.74	34.99	39.83	-2.53	64.15	49.96	54.99	-3.63
FLOWER	50.39	37.72	42.36	**0.00**	64.50	53.84	58.61	**0.00**

Table 5.7: Ablation study on split MiniImagenet dataset with 60 base classes averaged across 5 runs.

Configuration			Session			Gap
FM	WM	Ball	1	9	Avg	
	✓	✓	73.42	46.50	57.77	-1.78
✓		✓	73.93	46.53	57.95	-1.60
✓	✓		74.60	43.62	57.34	-2.21
✓	✓	✓	**74.60**	**49.06**	**59.55**	0.00

Table 5.8: Sensitivity analysis of the impact of bound value to classification accuracy on the MiniImageNet dataset with 60 base classes averaged across 5 time runs.

Bounds (b)	Session		
	1	9	Avg (1-9)
0.0001	73.65	47.78	58.12
0.001	**74.60**	**49.06**	**59.55**
0.0025	**73.98**	**49.13**	**59.29**
0.005	70.40	45.60	55.62
0.01	68.80	43.95	54.30
0.025	67.30	43.60	53.07
0.05	67.52	43.38	53.10
0.075	67.58	43.54	53.42
0.1	67.73	43.67	53.57

Table 5.9: Sensitivity analysis of the impact of random times to classification accuracy on MiniImageNet dataset with 60 base classes averaged across 5 time runs, * denotes only run in base task due to poor performance/

Random Times (M)	Session		
	1	9	Avg (1-9)
1*	1.67	-	-
2	**74.60**	**49.06**	**59.55**
3	73.83	**48.52**	**59.05**
4	**74.08**	**48.39**	**59.00**
5	**74.67**	**48.85**	**59.48**
6	73.50	48.12	58.49
7	74.13	48.47	59.12
8	74.10	48.40	59.01
9	74.08	48.06	58.69
10	73.88	48.18	58.87

Table 5.10: Analysis of the impact of augmentation to classification accuracy on MiniImageNet dataset with 60 base classes averaged across 5-time runs.

Aug. Conf.			Session			Gap
R.Crop+R.Flip	Ball	$\mathcal{L}_{ball}$	1	9	Avg	
	✓	✓	74.60	49.06	59.54	-0.01
✓			74.60	43.62	57.34	-2.21
✓	✓		74.60	46.51	58.27	-1.28
✓	✓	✓	**74.60**	**49.06**	**59.55**	**0.00**

5.5.4 Numerical Results

Numerical results of consolidated algorithms across the CIFAR100, miniimagenet, and CUB problems under moderate base tasks, e.g., 50 for CIFAR100 and miniimagenet and 100 for CUB, are reported in Tables 5.2, and 5.3. FLOWER delivers the most encouraging performance in the CIFAR100 problem and even demonstrates better performances than the upper bounds, cRT, and joint-training. It respectively beats baseline with 0.6% gap, F2M with 1.4% gap, FACT with 5.8% gap, and S3C with 0.8% gap. The same finding is observed in the realm of per-session accuracy, where FLOWER outperforms other consolidated algorithms. FLOWER consistently exceeds other algorithms in the miniimagenet problem, even with higher differences than that of the CIFAR100 problem. It respectively beats the baseline with 6.7% gap, F2M with 7.4% gap, SSFE-Net with 1.8% gap, FACT with 6.5% gap, and S3C with 2.6% gap. FLOWER also delivers higher per-session accuracy than those other algorithms. As with the other two problems, FLOWER is superior to others in the CUB problem where it outperforms F2M (2.7% margin), Baseline (3.49% margin), SSFE-Net (2.7% margin), FACT (0.8% margin), and S3C(1.4% margin) while consistently attaining higher per-session accuracy than other algorithms. FlatWide (FLOWER without Ball augmentation) also shows a promising performance where in CIFAR100 dataset it achieves the second-best performance with a small margin to FLOWER, in MiniImagenet and CUB dataset it achieves higher performance than the other competitors except SSFE-Net.

FLOWER's performances are more promising in the small base task than in the moderate base task, where numerical results of consolidated algorithms are reported in Tables 5.2, and 5.3. It outperforms Baseline with 3.3% gap, F2M with

3.4%, FACT with 4.1% gap, and S3C with 1.9% gap in the CIFAR100 problem. It achieves higher accuracy than F2M in the MiniImageNet problem with 5.8% margin, and in the CUB problem with 2.59% margin. FLOWER achieves higher performance than FACT with 2.4% margin in MiniImageNet and comparable performance in the CUB problem. FLOWER also achieves higher performance than S3C in the CUB problem and comparable performance in MiniImageNet. FlatWide achieves competitive performance with a similar pattern as in the moderate base classes scenario. In CIFAR100, it achieves the second-best performance, and in MiniImageNet and CUB problems, it achieves higher performance than the competitors except S3C (in the MiniImageNet problem) and FACT (in the CUB problem). These support the efficacy of FLOWER for few-shot continual learning problems, notably for small base tasks. We also confirm that the Baseline, where only prototype constructions are performed in the few-shot learning sessions while leaving other parameters fixed, is capable of producing strong performances as observed in [42]. That is, a model is supposed to retain decent performances of the base task here for the few-shot continual learning problems due to a large number of classes.

Summarized numerical results in the 1-shot configuration are tabulated in Table 5.4. FLOWER remains superior to other algorithms in the three benchmark problems under the 1-shot setting, i.e., each class of few-shot learning phases comprises 1 sample only. It produces better accuracy than baseline with visible gaps, i.e., 2% for MiniImageNet, while its accuracy is on par with Rebalance in the CIFAR100 problem and S3C in the CUB problem. Note that the moderate base tasks are implemented here, and Rebalance is poor on the CUB. In the 1-shot setting, FlatWide also shows a competitive performance as shown in Table 5.4, i.e., it achieves the second-best performance in the MiniImageNet problem, comparable to FLOWER and Rebalance in CIFAR100, and comparable to FLOWER and FACT in CUB. Performance of consolidated algorithms for moderate base tasks under the 5-shot setting is pictorially illustrated in Figure 5.4, where ours consistently delivers the most improved trends. Visualizations for small base tasks under the 5-shot setting and that of the 1-shot setting are provided in Figure 5.5 and Figure 5.6 where both exhibit the same and consistent trends.

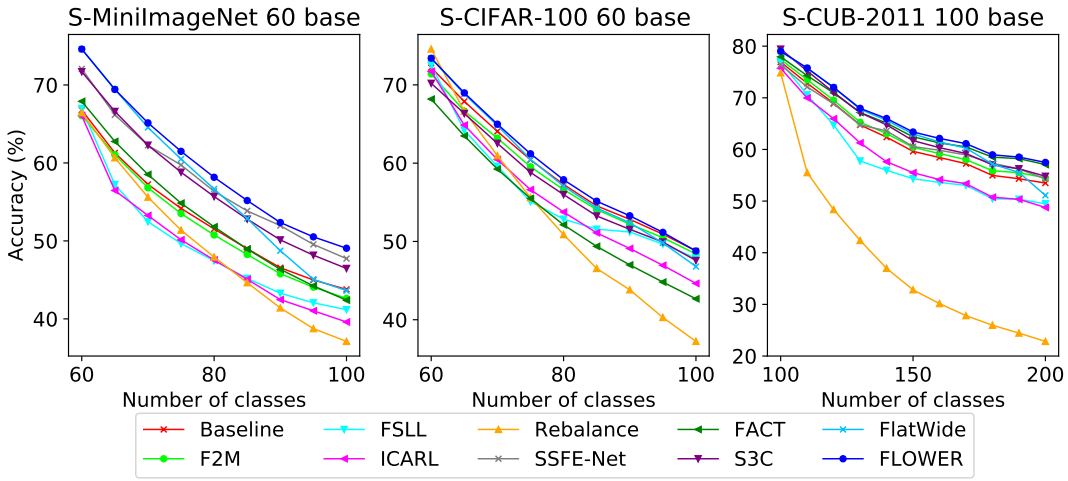


Figure 5.4: Visualization of Performances of Consolidated Algorithms across three problems under the moderate base tasks

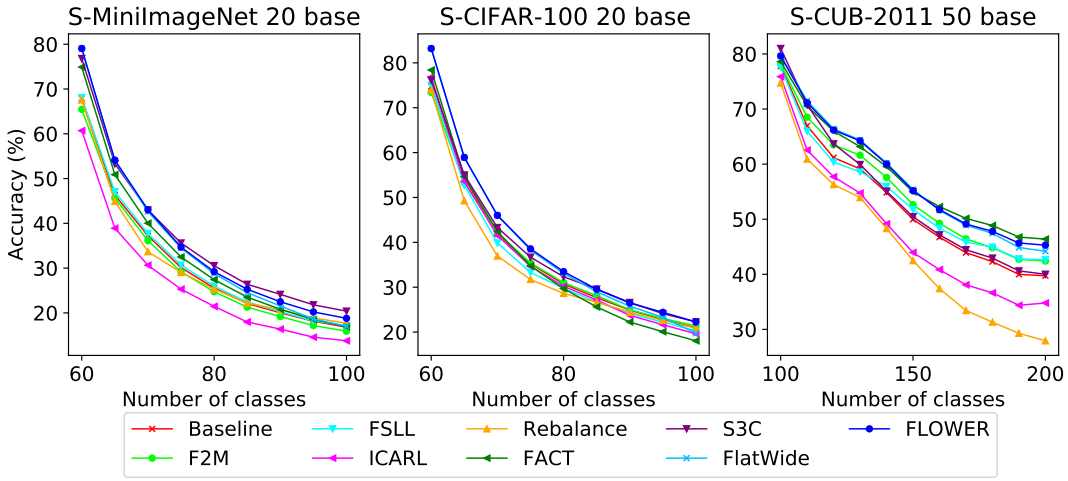


Figure 5.5: Visualization of Consolidated algorithms in MiniImagenet, CIFAR-100, and CUB-2011 dataset with 20, 20, and 50 base classes, respectively.

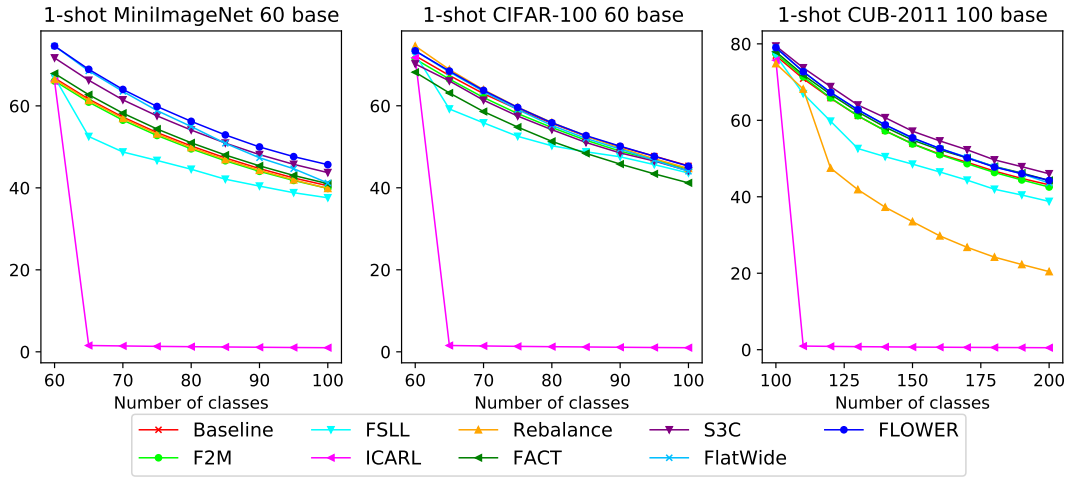


Figure 5.6: Visualization of Consolidated algorithms in MiniImagenet, CIFAR-100, and CUB-2011 dataset with 60, 60, and 100 base classes respectively, with 1-shot setting.

5.5.5 Stability-Plasticity Analysis

We analyze the stability-plasticity performance, showing how the consolidated methods handle the catastrophic forgetting problem. Tables 5.5 and 5.6 show harmonic mean accuracy that shows the stability-plasticity performance of the consolidated methods. In the MiniImageNet problem with a moderate base classes setting, FLOWER achieves the highest harmonic mean accuracy with $> 3.2\%$ margin compared to the competitors. Looking at the harmonic mean accuracy on each task, FLOWER consistently achieves the highest performance from task 2 until the last task (task 9). In the small base classes setting, FLOWER achieves a comparable performance to S3C and higher performance than the rest of the competitors with $> 1.8\%$ gap. In the CIFAR100 problem, FLOWER achieves a lower harmonic mean accuracy than S3C, F2M, and Baseline in a moderate base classes setting, and the highest harmonic mean accuracy in a small base classes setting with $> 2.5\%$ margin. A similar trend is shown in the CUB problem, where FLOWER achieves a comparable harmonic mean accuracy FACT in a moderate base classes setting, and the highest harmonic mean accuracy in a small base classes setting with $> 3.1\%$ margin. Looking at all the cases, FLOWER is considered to achieve better stability-plasticity performance than the competitor methods.

Figure 5.7 shows more detailed stability-plasticity performance by showing the base classes' accuracy and novel classes' accuracy across all tasks. In the MiniImageNet problem, FLOWER achieves the highest base classes' accuracy and second-best (lower than Baseline) novel classes' accuracy in all tasks. Please note that the base classes' accuracy of the Baseline method is far lower than FLOWER's. Thus, FLOWER achieves the highest harmonic mean accuracy in all tasks as shown in Table 5.5. In the CIFAR100 problem, FLOWER achieves the highest base classes' accuracy but moderate novel class accuracy in all tasks. Therefore, FLOWER achieves a fair harmonic mean accuracy. In the CUB problem, FLOWER achieves the highest base classes' accuracy and novel classes' accuracy in all tasks. Thus, it achieves the highest harmonic mean accuracy with a clear margin. This analysis confirms that FLOWER handles catastrophic forgetting better than the competitor methods.

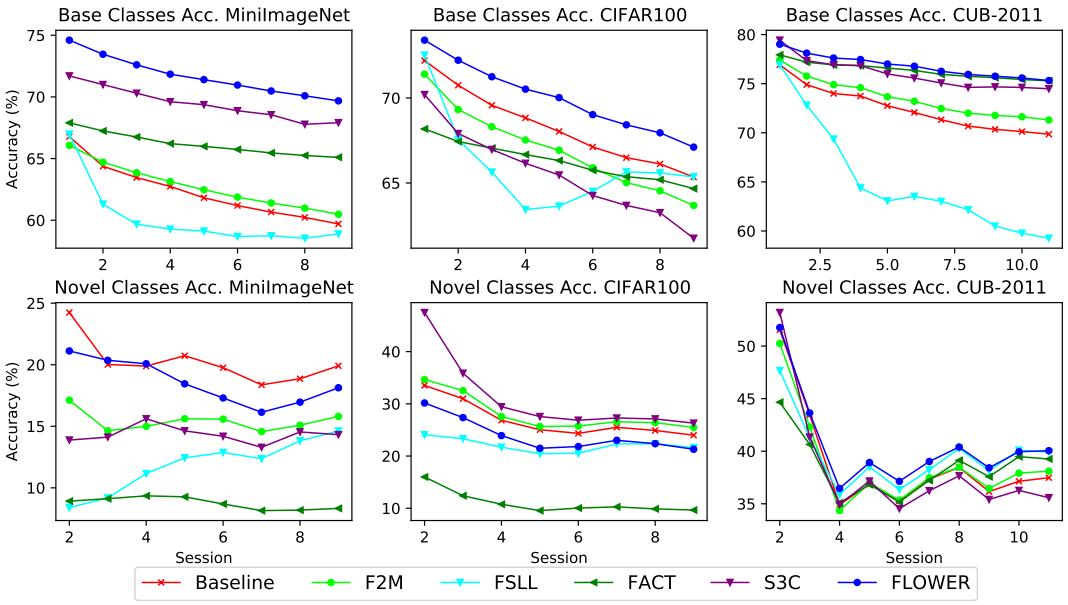


Figure 5.7: Visualization Base Classes and Novel Class Accuracy across all tasks in MiniImagenet, CIFAR-100, and CUB-2011 dataset with 60, 60, and 100 base classes respectively.

5.5.6 Ablation Study

Ablation study is executed to study the contribution of each learning module of FLOWER to the overall learning performance, where numerical results under the moderate base task of the miniimagenet problem are tabulated in Table 5.7. In a nutshell, each learning module of FLOWER contributes positively, where the absence of one component results in noticeable performance degradation. FLOWER is underpinned by the ball generator for feature space data augmentation to alleviate the issue of data scarcity, where its absence significantly brings down FLOWER's accuracy by 2.21%. The absence of flat minima (FM) learning and wide minima (WM) learning incurs performance losses of 1.78% and 1.6%, respectively.

5.5.7 Sensitivity Analysis

Sensitivity analyses are performed to study the effect of the flat-wide region bound b and the number of trials M. Numerical results across different bounds b are reported in Table 5.8. It is perceived that a too small bound leads to performance deterioration because of non-flat-wide regions, i.e., the confirmation of the presence of flat-wide local minima. On the other hand, a too large bound causes inaccurate approximations of flat-wide regions, thus ending up with the CF problem. $b = 0.001$ is applied in our experiments.

Numerical results across different random times M are presented in Table 5.9. $M = 1$ significantly compromises FLOWER's performance since no approximation of flat-wide regions is carried out in this case. The increase in the number of trials $M > 1$ results in trivial differences in performance. However, high M slows down execution times since FLOWER has to undergo many trials in the base learning tasks to find wide-flat regions. In this chapter, we set the number of trials $M \in [2, 4]$ for all experiments.

5.5.8 Analysis of Data Augmentation

This section is directed to analyze the advantage of the ball data generator of FLOWER.

Naive vs Ball Comparison: This section compares the ball data generator of FLOWER with naive data augmentation protocols,i.e., random crops and horizontal flips. Table 5.10 displays our numerical results. The table shows that the absence of Ball augmentation decreases the average performance with a significant margin, i.e., 2.21%. The absence of naive augmentation does not impact the average performance significantly. Please note that the absence of Ball implies the absence of $\mathcal{L}_{ball}$.

$\mathcal{L}_{ball}$ **Function:** The ball data generator is equipped with Eq. (5.10) that aims to minimize the synthetic sample bias problem due to the random nature of the enclosing ball. The absence of $\mathcal{L}_{ball}$ (5.10) decreases the performance with 1.28%. Please note that in this case, we still have ball augmented samples, i.e., $\hat{z}$ and $\hat{\hat{z}}$. This finding shows the presence of both $\mathcal{L}_{ball}$ and augmented samples ($\hat{z}$ and $\hat{\hat{z}}$) contribute to the improvement of FLOWER's performance.

5.5.9 Complexity and Running Time Analysis

Following the pseudo-code in Algorithm 4, FLOWER has several operations e.g. noise sampling (line 5) that costs O(1), parameter perturbation (line 6) that costs O(1), computing $\mathcal{L}_{base}$ (line 7) that costs O(N_1), reset parameter (line 8) that costs O(1), update network parameters (line 10 and 25) that costs O(1), ball augmentation (line 15-20) that costs O(S), computing $\mathcal{L}_{k>1}$ (line 22-24) that costs O(N_k), and clamping the parameter(line 26) that costs O(1) where N_k denotes number of samples on task k. The FLOWER complexity can be computed as:

$$O(FLOWER) = O(BaseTask) + O(FewStotTask) \tag{5.34}$$

$$\begin{aligned} O(FLOWER) =& (M.O(1) + M.O(1) + M.O(N_1) \\ &+ M.O(1) + O(1)) \\ &+ (K-1)(O(S) + O(N_k) + O(1)) \end{aligned} \tag{5.35}$$

$$\begin{aligned} O(FLOWER) =& O(M.N_1) + O((K-1.)S) \\ &+ O((K-1)N_k) \end{aligned} \tag{5.36}$$

Since M and S are small constants, then FLOWER complexity can be derived to the following equations.

$$O(FLOWER) = O(N_1) + O(K-1) + O((K-1)N_k) \tag{5.37}$$

Table 5.11: Training time of the consolidated methods in CIFAR100 dataset with 60 base classes.

Method	Running time per epoch (s)	
	Base Task	Few Shot Task
Baseline	4.72	0.02
F2M	0.31	0.56
FSLL	4.74	1.13
ICARL	4.8	0.69
Rebalance	7.51	0.54
FACT	10.93	18.21
S3C	15.5	0.44
FlatWide	8.82	0.69
FLOWER	8.82	2.83

$$O(FLOWER) = O(N_1) + O((K-1)N_k) \tag{5.38}$$

Please note that $N_1 + (K-1)N_k = N$, where N is the total samples across all tasks. then the complexity of FLOWER can be derived to the following equations.

$$O(FLOWER) = O(N_1 + (K-1)N_k) \tag{5.39}$$

$$O(FLOWER) = O(N) \tag{5.40}$$

Table 5.11 shows that FLOWER has moderate training time both in base task and few-shot tasks, proving its complexity O(N) which is the same as competitor methods. Please note that in Table 5.11, F2M applies the task sampler in base task training, implying that only a small portion of each batch will be selected for training. The baseline method only generates prototypes in a few few-shot tasks without doing parameter updates, which results in its short training time. FLOWER generates augmented samples (features), i.e., $\hat{z}$, and transforms them into $\hat{\hat{z}}$, which results in its longer training time in few-shot tasks.

5.6 Summary

This chapter presents FLOWER as a solution to few-shot continual learning problems. Our in-depth numerical study confirms the advantage of FLOWER, where it outperforms prior arts with $1-4\%$ margin in the three benchmark problems

for both average and per-session accuracy. This fact is even more obvious with small base tasks, where larger margins to their counterparts are observed than with moderate base tasks, whereas prior arts are compromised in such a setting because they heavily focus on the stability of base tasks. Ablation study and sensitivity analysis further demonstrate the advantage of each learning module and the robustness of FLOWER. Current few-shot continual learning relies on a strong assumption that each task is drawn from the same domain.

The upcoming chapter is directed to unsupervised few-shot continual learning problems to address unavailable labels in the dynamic environment. In real-world applications, a dataset is feasible to collect, but the labels (annotations) are hardly available. It needs expensive **human efforts** to annotate the dataset.

Chapter 6

UNISA: Unsupervised Few-Shot Continual Learning for Remote Sensing Image Scene Classification

In the previous chapter, FLOWER has proven its effectiveness in supervised few-shot continual learning. However, in real-world applications such as remote sensing, where the data has varying camera parameters, spectral ranges, resolutions, etc, class labels are hardly available due to the high cost of annotation. This chapter addresses this problem with a proposal of an unsupervised flat-wide learning approach (UNISA) for unsupervised few-shot continual learning approaches of remote sensing image scene classifications, which do not depend on any labeled samples for their model updates. UNISA is developed from the idea of prototype scattering and positive sampling for learning representations while the catastrophic forgetting problem is tackled with the flat-wide learning approach combined with a ball generator to address the data scarcity problem. The numerical study with remote sensing image scene datasets and a hyperspectral dataset confirms the advantages of our solution.

Part of this chapter is published as: MA Ma'sum, M Pratama, R Savitha, L Liu, H Habibullah, R Kowalczyk. Unsupervised Few-Shot Continual Learning for Remote Sensing Image Scene Classification. IEEE Transactions on Geoscience

and Remote Sensing 62, 4707214. https://doi.org/10.1109/TGRS.2024.3445887 **[IF: 7.5, #2 in RS]**

6.1 Introduction

The use of deep learning (DL) has been widespread in the domain of remote sensing (RS), such as earth observation, disaster monitoring, urban planning, etc. This progress is mainly driven by the operations of advanced satellite technologies, facilitating remote sensing imaging technologies for convenient data collection. Nonetheless, RS domains are known to be rapidly changing because of temporal, spatial, and spectral resolution limitations, thus calling for continual learning (CL) solutions, e.g., while earth observation sensors obtain images continuously over time, the land cover/ land use classes are made available sequentially [194]. RS applications characterize four unique problems [195]: *data openness*, *temporal openness*, *spatial openness*, and *spectral openness*. Naive retraining approaches impose considerable complexities that do not scale well for continuous environments.

Continual learning (CL) is a research area of growing interest in the RS domain, where several works have been recently proposed in the literature [194], [195], [196], [197], [198]. The concept of model agnostic meta learning (MAML) [109] is implemented in [196] for CL of RS images. The lightweight incremental learning (LIL) approach is proposed in [198] for continual RS image classification. [197] presents the notion of two trainable sub-networks and [194] is based on the curriculum learning. The concept of preserving network and transient network is developed in [195] for CL of RS image scene classifications. Nonetheless, these works require a lot of labeled samples, which do not fit the RS applications. Labeling cost is often expensive in the RS domains because ground truth is based on the field-based survey. Fig. 6.1 visualizes the case of CL for RS images. The number of classes cannot be fixed at once due to temporal, spatial, and spectral resolution limitations, leading to the case of CL, which should learn a sequence of new target classes without the catastrophic forgetting problem.

Some attempts have been made to address the problem of labeling costs in the

RS area. [199] proposes the concept of CNN and GCN for cross-scene hyperspectral image classifications. [200] offers a single-source domain generalization framework in the realm of cross-scene hyperspectral image classifications based on the supervised contrastive learning framework. Another domain generalization framework is proposed in [201], integrating linguistic modal information. [202] combines the hyperspectral data and the LiDAR data for collaborative land-cover classifications. Albeit these recent advances in handling the low-label situation, these works do not yet address the CL problem where a model has to cope with a sequence of different learning tasks. In addition, the problem of limited data remains an uncharted territory.

How to handle continual learning problems with limited samples is known as few-shot continual learning (FSCL) [40] and has attracted significant research interests in the realm of CL. [40] proposes the idea of neural gas, while [56] puts forward the concept of session trainable parameters. [125] integrates the graph attention network to boost the performance of the base task, while [188] addresses the FSCL problem using the concept of incremental vector quantization. [42] offers the idea of flat learning concept and [192] proposes the idea of stochastic classifier. All of these works have not been validated with RS data. In addition, they still assume the presence of fully labeled training data, albeit with the low sample constraint. Note that [203] does not address the CL problem, although it deals with cross-domain few-shot learning problems. The motivations of this chapter are outlined as follows:

- Existing works in the CL for RS data do not yet address the problem of data scarcity. That is, they suffer from the over-fitting problem in the case of few-shot continual learning.
- Although the problem of FSCL has been studied in some recent works, these approaches are not yet targeted for the RS data possessing unique characteristics.
- Existing FSCL works assume the presence of fully labeled samples, which may not hold in many practical environments. In practice, labels are scarce and expensive to obtain.

This chapter proposes an unsupervised flat-wide learning approach (UNISA) for unsupervised few-shot continual learning (UFSCL) of RS image scene classification. UFSCL can be perceived as an extension of few-shot continual learning (FSCL) [40], where no labeled samples are used for model updates. Very few samples are only made available for class inferences. UNISA is developed from the concept of a deep clustering network where the representation training process is driven by the concept of prototype scattering and positive sampling [204], avoiding the class collision issue while attaining uniform representation. This strategy combines the concept of contrastive learning and non-contrastive learning, where the prototype scattering loss assures the uniformity of representation, while the positive sampling loss overcomes the class collision issue. The catastrophic forgetting problem is resolved with the union of flat learning concept [42] and wide learning concept [62], seeking and maintaining the flat-wide local minima regions. That is, the base learning task explores the flat-wide minima regions via noise perturbation while clamping network parameters to stay around the flat-wide local optimum regions during the few-shot learning tasks. The idea of the ball generator [189] is implemented during the few-shot learning tasks as a feature-level data augmentation approach to address the problem of data scarcity.

This chapter contains four major contributions:

1. This chapter proposes a challenging but practical problem, unsupervised few-shot continual learning (UFSCL) problems. This problem emphasizes the low label problem in the few-shot continual learning (FSCL) setting, where no labeled samples are offered for model updates. This chapter proposes an unsupervised flat-wide learning (UNISA) approach as a solution to the UFSCL problem. To the best of our knowledge, existing FSCL approaches exclude the low-label problem and thus are not directly applicable to the UFSCL setting;

2. This chapter proposes a novel unsupervised flat-wide learning approach based on the concept of prototype scattering and positive sampling. This approach goes one step ahead of the current flat-wide learning approach [205], which is only designed in the fully supervised learning contexts.

3. A new joint loss function for UFSCL problems is put forward. The representation learning component aims to extract meaningful representation without any labels, and the ball data generator component is meant to enrich latent space representation, while the wide learning component and the regularization term protect against the catastrophic forgetting problem.

4. The rigorous experiment shows our proposed method is superior to existing state-of-the-art algorithms in six popular remote sensing datasets and robust in various scenarios.

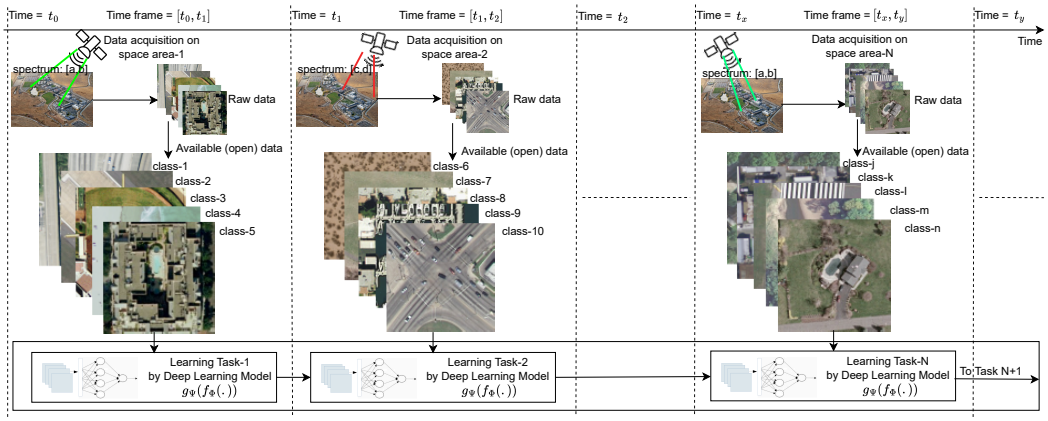


Figure 6.1: Visualization of Continual Learning for Remote Sensing Scene Classification.

6.2 Preliminaries

6.2.1 Continual Learning in RS

Continual learning (CL) has triggered active research in the RS domain. The concept of MAML [109] is integrated in [196] for lifelong learning of RS images. The concept of lightweight incremental learning (LIL) is proposed in [198] for CL of RS images. [197] adopts the idea of two trainable sub-networks while [194] proposes the notion of curriculum learning. The concept of preserving and transient networks is proposed in [195]. In [206], distribution reliability assessment based on incremental learning (DRAIL) is proposed for automatic target recognition. [207], [208] emphasize the importance of the knowledge distillation concept in RS clas-

sification. A novel exemplar selection is proposed in [209] for a class-incremental learning algorithm and applied to the RS data. Synthetic Aperture Radar Automatic Target Recognition (SARATR) is proposed in [210] and is based on the few-shot class-incremental learning algorithm. [211] proposed a label-consistent classifier and gradual regression to boost model performance on few-shot incremental object detection in RS. [212] utilized a prompting approach and dual frequency approaches for FSCL in RS. [213] proposed multi-scale feature representations and multi-scale fusion to enhance the model's discriminative capability for FSCL in RS. All these works are not compatible with the UFSCL problem proposed here because they are still designed for ordinary CL problems, excluding low-label and low-data situations in the CL problems.

6.2.2 Problem Definition

Unsupervised few-shot continual learning (UFSCL) is defined as a sequential learning problem over a sequence of tasks, $\mathcal{T}_1, ..., \mathcal{T}_K, k \in \{1, ..., K\}$ under the presence of both label and data scarcities. K stands for the number of tasks unknown in practice. Unlike conventional FSCL problem [40], a model is fed only by unlabeled samples $\mathcal{T}_k = \{x_i^k\}_{i=1}^{|T_k|}$ where $|.|$ stands for the cardinality symbol and $x_i^k \in X$ denotes the $i-th$ image of the $k-th$ task. A very small subset of labels can be accessed only for class inferences $Y_k = \{y_i\}_{i=1}^{|Y_k|}, |Y_k| << |\mathcal{T}_k|$ where $y_i \in \mathcal{Y}$ denotes a class label of the $i-th$ image and Y_k stands for the label set of the $k-th$ task. All images are drawn from the same domain $\mathcal{X} \times \mathcal{Y} \in \mathcal{D}$. We only focus on the challenging class-incremental learning setting [161] here, where a model is configured under a single head structure with the absence of any task IDs. That is, each task carries different and non-overlapping label sets $\forall_{k,(k-1)} Y_k \cap Y_{k-1} \in \emptyset$.

As with the FSCL problem, the UFSCL problem comprises a base learning task $\mathcal{T}_1$ and streaming few-shot learning tasks $\mathcal{T}_k$. There exist large samples of the base learning task $\mathcal{T}_1$, whereas the data scarcity issue is seen in the few-shot learning tasks $\mathcal{T}_k$. That is, the few-shot task $\mathcal{T}_k$ is configured in the N-way K-shot setting, i.e., it carries N classes where each class contains only K samples. We explore the case of $K \in [1, 5]$.

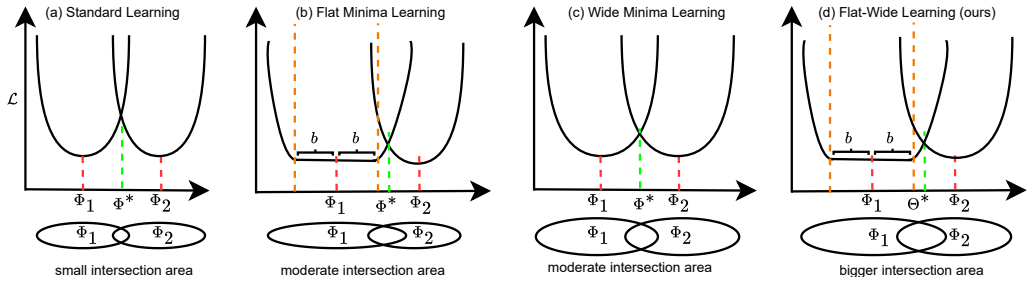


Figure 6.2: Visualization of Model Parameters in Flat Wide Learning (d) compared to Standard Learning (a), Flat Minima Learning (b), and Wide Minima Learning (d).

6.2.3 Our Solution: Unsupervised Flat-Wide Learning Approach (UNISA)

In this section, we introduce the main concept of our proposed method, i.e., unsupervised flat-wide learning illustrated in figure 6.2. Given a deep learning model $g_\Psi(f_\Phi(.))$ where f and g is feature extractor and classifier that parameterized by Φ and Ψ respectively. Standard FSCL learning optimizes Φ to recognize the learned classes from the base task (T_1) until the current task (T_2). However, since the tasks have different feature distributions, the optimal intersection of optimal parameters for T_1 and T_2 is small as shown in figure 6.2 (a). Flat minima learning shown in figure 6.2 (b) tries to search the flat region on the base task by perturbing Φ_1 with noise in the range [-b,b], then forces the clamp of the optimal parameter Φ^* into the flat region. Wide minima learning strives to enlarge the insertion between Φ_1 and Φ_2 by scratching both loss curves as shown in figure 6.2 (c). Our proposed method, i.e., Flat-Wide Learning, leverages both properties. In the base task, it finds a flat region while simultaneously scratching its loss curve. In the few-shot task, the model keeps applying the wide minima principle, so the intersection between Φ_1 and Φ_2 is bigger than the previous three approaches. The higher intersection area implies the model has a higher chance of recognizing classes in both tasks.

The second uniqueness is that our proposed method is trained in an unsupervised way, while the previous three approaches are trained in a supervised way. Instead of the actual label, our approach utilizes a pseudo-label produced by an unsupervised learning mechanism, i.e., clustering, to optimize Φ. Third, our proposed method

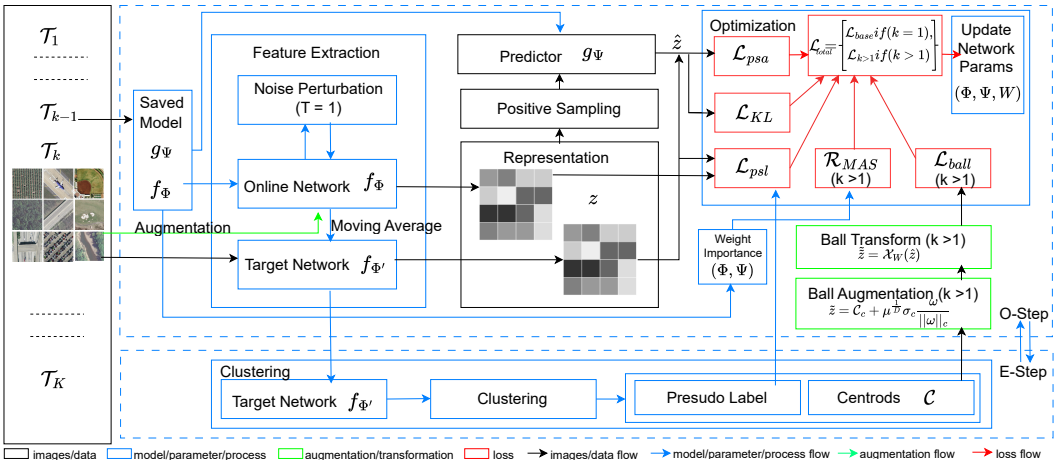


Figure 6.3: Workflow of the Proposed Method (UNISA): UNISA is trained minimizing a joint loss function of the prototype scattering loss, the positive sampling alignment loss, the wide local minimum loss, the ball generator loss, and the regularization term. The data augmentation procedure is implemented to address the data scarcity problem.

creates multiple representations for each input. In each iteration, given an input x, UNISA creates representations $f_\Phi(x)$ and $f_{\Phi'}(x)$ where Φ' is an alternate version of Φ by applying a moving average operation. This enrichment of representation plays an important role in optimizing Φ. These last two principles try to answer the data scarcity challenge, where only a small amount of labeled data is available. The details of our proposed method are presented in the following section.

In a formal expression, UNISA is structured as a deep clustering network composed of a feature extractor $z = f_\Phi(x) \in \Re^D : x \to z$ mapping an input image $x \in \Re^{w \times h \times c}$ to a feature space, fully connected layers $\hat{z} = g_\Psi(z) \in \Re^{\hat{D}} : z \to \hat{z}$, and a clustering module $\hat{y} = \kappa_C(\hat{z}) : \Re^{\hat{D}} \to \Delta_M$ delivering final predictions in which M denotes the output dimension.

6.3 UNISA Base Learning Session $(k = 1)$

This section outlines the procedure of the base learning session of UNISA, whose goal is to solicit flat-wide local minima to be retained throughout its learning process via representation learning of the prototype scattering and positive sampling

losses.

6.3.1 Representation Learning

The clustering-friendly latent space of UNISA is attained via a representation learning method minimizing the prototype scattering and positive sampling losses [204]. This concept combines the concepts of contrastive and non-contrastive learning under one roof, taking advantage of their strengths while eliminating their bottlenecks. Notwithstanding that the contrastive learning strategy [214] using the InfoNCE loss allows uniform representations, it suffers from the class collision issue where negative samples of the same clusters are wrongly pushed away. On the other hand, such a drawback is absent in the non-contrastive learning technique, but it often results in trivial solutions leading to too many empty clusters. Let x be an instance where x^+ is its augmented version via random augmentation and $\{x_1^-, ..., x_m^-\}$ be a set of m negative instances, the InfoNCE loss can be written as follows:

$$\begin{aligned}\mathcal{L}_{INCE} \approx & \underbrace{-\frac{g_\Psi(f_\Phi(x)^T) g_\Psi(f_\Phi(x^+))}{\tau}}_{alignment} \\ & \underbrace{+\log \sum_{i=1}^{m} \exp\left(\frac{g_\Psi(f_\Phi(x)^T) g_\Psi(f_\Phi(x^-))}{\tau}\right)}_{uniformity}\end{aligned} \tag{6.1}$$

pulling together the positive pair (x, x^+), the alignment term, while pushing away x from a set of m negative instances, the uniformity term. The absence of cluster representation causes the class collision issue due to the uniformity term. The uniformity term is discarded in the non-constraining learning technique, thus avoiding the class collision issue, but it suffers from non-uniform representations.

The prototype scattering loss works directly on the cluster level using the contrastive strategy. Suppose that there exists N_C clusters, one cluster is assigned as a positive cluster, while the remainder $N_C - 1$ clusters are selected as negative prototypes. N_C clusters are constructed at the bottleneck layer of the fully connected module $\{C_1, C_2, ..., C_{N_C}\}$ and at the same time their augmented versions are also created using the random augmentation approach $\{C_1', C_2', ..., C_{N_C}'\}$. The

prototype scattering loss function is formalized as follows:

$$\mathcal{L}_{psl} = \frac{1}{N_C}\sum_{c=1}^{N_C} -\frac{C_c^T C_c'}{\tau} + \frac{1}{N_C}\sum_{s=1}^{N_C}\log\sum_{\substack{c=1\\c\neq s}}^{N_C}\exp\left(\frac{C_s^T C_c}{\tau}\right) \tag{6.2}$$

(6.2) guarantees uniform representations and well-separated clusters without any risk of the class collision issue because the negative instance is of different clusters.

The positive sampling alignment approach is developed to improve within-cluster compactness while further overcoming the class collision problem. This is achieved by aligning the neighbouring examples with another view of a sample. Note that neighbouring samples are truly positive and belong to the same class [204]. Neighbouring samples themselves are constructed with the help of the Gaussian distribution, controlling the influence zone of a sample to be treated as a positive sample. The positive alignment loss is written as follows:

$$\mathcal{L}_{psa} = ||g_\Psi(f_\Phi(x+\epsilon\sigma)) - f^{'}(x^+)||_2^2 \tag{6.3}$$

where $\epsilon \backsim \mathcal{N}(0,1)$ and σ is a predefined constant. $f^{'}(.)$ is a target network, momentum updated. (6.3) should not cause any class collision issue because neighboring samples of an instance should be positive samples.

This strategy is implemented in the EM framework, where the k-means clustering strategy is applied in the E step, followed by the fine-tuning mechanism during the M step, minimizing the proposed loss function. The target network is updated using moving average formula as [204], [215] $\phi^{'} = m\phi^{'} + (1-m)\phi$ and used in the inference phase.

6.3.2 Flat-Wide Base Learning Phase ($k = 1$)

The base learning phase is devised to capture flat-wide local optimum regions in which the losses are still minimized while coping with the CF problems. The flat learning concept itself [42] does not suffice because it relies on too narrow local optimum regions, which do not scale well for large-scale problems. As a result, the flat learning concept usually works well with large base tasks and needs to be supplemented with an episodic memory to address the CF problem. The

concept of wide learning [62] is integrated here to drive the flat regions to be wide, thereby increasing the chance of overlapping regions across all tasks to be discovered. The flat local optimum regions $\Phi - b \leq \Phi \leq \Phi + b$ can be found by introducing small random noises $-b \leq \epsilon_t \leq +b$ to the model's parameters multiple times, such that similar but different losses are returned. Such losses can be minimized simultaneously to discover the flat minimum regions. Formally speaking, our interest is to minimize the expected loss function w.r.t. the joint distribution of data z and noise $\epsilon, -b \leq \epsilon \leq +b$:

$$\begin{aligned} R(\Phi) = \int_{\Re^{d_\epsilon}} \int_{\Re^{d_z}} \mathcal{L}(z; \Phi + \epsilon, \Psi) dP(z) dP(\epsilon) \\ = \mathbb{E}[\mathcal{L}(z; \Phi + \epsilon, \Psi)] \end{aligned} \tag{6.4}$$

where $P(z), P(\epsilon)$ respectively stand for the data distribution and the noise distribution. In practice, expected losses are impossible to be minimized, so their empirical version is formulated as follows:

$$\begin{aligned} \mathcal{L}_{k=1} = \frac{1}{T} \sum_{t=1}^{T} \mathcal{L}_{base}(z; \Phi + \epsilon_t, \Psi) \\ \mathcal{L}_{base} = \mathbb{E}_{(x) \backsim \mathcal{T}_1}[\mathcal{L}_{psl}(z; \Phi + \epsilon_t, \Psi, C) + \lambda_1 \mathcal{L}_{psa}(z; \Phi + \epsilon_t, \Psi) \\ + \lambda_2 \mathcal{L}_{KL}(g_\Psi(f_\Phi(x)), P_U)] + \lambda_3 ||C_c - C_c^*||_2 \end{aligned} \tag{6.5}$$

where $\lambda_1, \lambda_2, \lambda_3$ denote predefined constants. $\mathcal{L}_{KL}(.)$ is appended to promote wide local optima where latent features are projected to P_U, the uniform distribution. That is, the uniform distribution is selected because it offers the upper bound of divergence $\log M$. C_c, C_c^* denote the latent clusters before and after noise injections, where the last term is included to avoid the cluster's drifts. Note that we follow [215] where the target network $f_{\Phi'}(.)$ is momentum updated to allow stable representations. That is, the online network $f_\Phi(.)$ is updated first, minimizing (6.5) and applied to the moving average formula to adjust the target network.

6.4 UNISA Few-Shot Learning Phase $(k > 1)$

The few-shot learning phase features data scarcity issues, where it is configured under the N-way K-shot configuration. UNISA applies the feature-space data aug-

mentation strategy to overcome this issue while putting forward the regularization-based approach to handle the CF problem.

6.4.1 Ball Data Generator

The concept of ball data generator [189] is modified to fit images rather than texts, and also in an unsupervised manner. Synthetic samples of the ball data generator are induced within the smallest enclosing balls of the feature space to prevent out-of-distribution. Since there exist very few labels, the ball $\zeta(C_c, \sigma_c)$ is constructed via clusters in an unsupervised manner, and the episodic training phase, as practiced in the few-shot learning domain, is absent in the FSCL problem. (C_c, σ_c) denote the centroid and standard deviation of the $c-th$ cluster estimated via the k-means clustering procedure. A synthetic sample $\tilde{z} \in \Re^{\hat{D}}$ is generated as follows:

$$\tilde{z} = C_c + u^{\frac{1}{D}} \sigma_c \frac{\omega}{||\omega||_2} \tag{6.6}$$

where $u \backsim \mathcal{U}(0,1)$ denotes a shifting constant and $\omega \in \Re^{\hat{D}} \backsim \mathcal{N}(0,1)$ labels a random vector. $\tilde{z}$ is not directly injected to the training process rather fed to a projection module $\tilde{\tilde{z}} = \chi_W(\tilde{z}) \in \Re^{\hat{D}} : \hat{z} \rightarrow \tilde{\tilde{z}}$ to generate stable features. The projection module is trained in such a way as to avoid synthetic sample biases via the triplet loss.

$$\mathcal{L}_{ball} = \sum_{\substack{\tilde{\tilde{z}} \notin C_j \\ \tilde{\tilde{z}} \in C_i}} \lambda_4 \max\{0, d(\tilde{\tilde{z}}, C_i) + r - d(\tilde{\tilde{z}}, C_j)\} \tag{6.7}$$

where r, λ_4 are predefined constants respectively. The goal is such that synthetic samples stay close to their centroid while pushing them away from other centroids, where the distance margin is controlled by r. C_i, C_j are the centroid of the $i-th$ and $j-th$ clusters respectively.

6.4.2 Regularization-based Approach

UNISA relies on the regularization-based approach to combat the CF problem due to the sequences of the few-shot learning task. We apply the memory-aware synapses (MAS) [25], which can be computed in an online and unsupervised manner.

Suppose that $\Theta = \{\Phi, \Psi\}$, the MAS technique is derived as follows:

$$\mathcal{R}_{MAS} = \lambda_5 \sum_i \Gamma_i^{k-1} (\Theta_i^k - \Theta^{k-1})^2 \tag{6.8}$$

where Γ is a parameter importance matrix and λ_5 is a predefined constant. MAS is preferred over EWC [66] and SI [26], because it is faster to compute them, and applicable for unsupervised scenarios. We do not impose any regularization on the transformation module $\chi_W(.)$ because it only focuses on the current concept. (6.8) can be perceived as the L_2 regularization except for the presence of the difference across two consecutive weight matrices Θ^{k-1}, Θ^k.

6.4.3 Flat-Wide Few-Shot Learning Phases $k > 1$

The loss function of the few-shot learning phases is written as follows:

$$\begin{aligned} \mathcal{L}_{k>1} = \mathbb{E}_{(x) \backsim \mathcal{T}_k} [\mathcal{L}_{psl}(z; \Phi, \Psi, C) + \lambda_1 \mathcal{L}_{psa}(z; \Phi, \Psi) + \\ \lambda_2 \mathcal{L}_{KL}(g_\Psi(f_\Phi(x)), P_U) + \lambda_4 \mathcal{L}_{ball}] + \mathcal{R}_{MAS} \end{aligned} \tag{6.9}$$

That is, the prototype scattering loss $\mathcal{L}_{psl}$ and the positive sampling alignment loss $\mathcal{L}_{psa}$ learn the current concept with the support of synthetic samples induced by the ball generator. The ball generator loss L_{ball} is designed to produce quality samples, while L_{KL} exists to induce wide local optimum regions and drives flat regions to be wide. Last but not least, the regularization $\mathcal{R}_{MAS}$ addresses the CF problem. Note that the projection module is trained in an end-to-end manner alongside the online network. The flat region is retained by applying the clamping mechanism. That is, the network parameters are clamped when going beyond the flat regions. In addition, the target network is adjusted by the momentum strategy of the online network. The workflow of UNISA is pictorially depicted in Fig. 6.3. Pseudo-code of UNISA is provided in section 6.4.4.

6.4.4 Algorithm

The learning policy of UNISA, both for the base task and the few-shot task, is presented in algorithm 5.

Algorithm 5 Learning Policy of UNISA

Input: The flat-wide region bound b, randomly initialized parameters $\Theta = \{\Phi, \Phi', \Psi\}$, projection module parameter w_X, hyperparameetrs $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, number of epochs E, number of batches on each task O, number of perturbation M. number of synthetic samples S.
Output: Updated Network Parameters $\Theta = \{\Phi, \Phi', \Psi\}$, Updated projection module w_X
//Base Learning Phase $k = 1$
for $e = 1 : E$ **do**
  $PseudoLabel = KMeans_Clustering()$
  for $batch = 1 : B$ **do** //number of batches
    $\mathcal{B}$ = minibatch sample of $\mathcal{T}_k$
    for $m = 1 : M$ **do**
      Sample a noise vector $\epsilon_j \backsim \mathcal{E}$ where $-b \leq \epsilon_j \leq b$
      Perturb the feature extractor parameters $\Theta = \{\Phi + \epsilon_j, \Psi\}$
      for x in $\mathcal{B}$ **do**
        Randomly augment x and $x+$
        Compute cluster centers $\mathcal{C}$
        Compute loss $\mathcal{L}_{psl}$ and $\mathcal{L}_{psa}$ as per Eq.(6.2) and (6.3)
      end for
      Compute the base loss $\mathcal{L}_{base}$ as per Eq.(6.5)
      Update the network parameters $\{\Phi, \Psi\}$
      Update the network parameters Φ' using moving average
      Reset the network parameters Φ
    end for
  end for
end for
//Few Shot Learning Phase $k > 1$
for $k = 2 : K$ **do**
  for $e = 1 : E$ **do**
    $PseudoLabel = KMeans_Clustering()$
    for $batch = 1 : B$ **do** // number of batches
      $\mathcal{B}$ = minibatch sample of $\mathcal{T}_k$
      for x in $\mathcal{B}$ **do**
        Randomly augment x and $x+$
        Compute cluster centers $\mathcal{C}$
        Compute loss $\mathcal{L}_{psl}$ and $\mathcal{L}_{psa}$ as per Eq.(6.2) and (6.3)
      end for
      $\mathcal{B}' = \emptyset$
      for $i' = 1 : S$ **do**
        Generate a synthetic sample $\hat{z}_{i'}$ as per Eq.(6.6)
        Apply the transformation module $\hat{\hat{z}} = \kappa_{W_T}(\hat{z}_{i'})$ to induce a unbiased synthetic sample $\hat{\hat{z}}_{i'}$
        $\mathcal{B}' = \mathcal{B}' \cup \{\hat{\hat{z}}_{i'}, c_i\}$
      end for
      Calculate $\mathcal{L}_{ball}$ as per Eq.(6.7) with $\mathcal{B}'$
      Calculate $\mathcal{L}_{KL}$ with $\mathcal{B}$
      Calculate total loss $\mathcal{L}_{k>1}$ as per Eq.(6.9)
    end for
    Update network parameters $\{\Phi, \Psi\}, w_X$ with $\mathcal{L}_{k>1}$
    Clamp feature extractor parameters $\Phi^* - b \leq \Phi \leq \Phi^* + b$ to meet the b-flat-wide local minima conditions
    Update the network parameters Φ' using moving average
  end for
end for

Table 6.1: Classification accuracy on split UC Merced dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot.

Method	#S	Performance on Split UC Merced					
		Session				Avg	Gap to UNISA
		1	2	3	4		
Baseline	5	18.33±0.00	17.20±0.90	14.61±1.43	11.90±0.81	15.51±0.94	-39.79±1.47
F2M		25.00±0.00	16.80±0.57	13.83±0.85	12.14±1.06	16.94±0.74	-38.36±1.35
FSLL		22.50±0.00	10.80±1.40	9.67±1.14	7.76±0.45	12.68±0.93	-42.62±1.46
ICARL		23.50±1.47	18.80±1.49	16.72±0.90	14.48±1.32	18.38±1.32	-36.92±1.74
Rebalance		12.50±0.23	10.07±0.22	8.22±0.12	7.24±0.21	9.51±0.20	-45.79±1.15
S3C		50.00±0.00	42.07±2.10	34.83±1.34	30.24±1.29	39.29±1.40	-16.01±1.80
SCN		23.75±0.00	10.47±2.01	8.83±1.56	7.76±1.02	12.70±1.37	-42.6±1.78
ORCO		42.50±0.00	13.07±2.95	15.89±2.05	15.67±2.75	21.78±2.26	-33.52±2.53
UNISA(Ours)		**67.50±0.00**	**58.27±1.51**	**51.67±0.77**	**43.76±1.50**	**55.3±1.13**	**0.00±0.00**
Baseline	4	18.33±0.00	17.33±1.12	10.44±2.83	8.86±2.22	13.74±1.88	-38.37±2.62
F2M		25.00±0.00	16.80±0.88	13.83±1.32	11.86±0.76	16.87±0.88	-35.24±2.03
FSLL		22.50±0.00	10.47±0.92	8.61±0.90	8.05±0.81	12.41±0.76	-39.7±1.98
ICARL		23.92±1.28	6.67±0.00	5.56±0.00	4.76±0.00	10.23±0.64	-41.88±1.94
Rebalance		12.75±0.44	9.73±0.39	8.39±0.18	7.10±0.20	9.49±0.32	-42.62±1.86
S3C		50.00±0.00	41.47±0.68	34.44±2.05	29.00±1.26	38.73±1.25	-13.38±2.22
SCN		23.75±0.00	10.73±1.02	9.44±1.91	7.57±1.06	12.87±1.21	-39.24±2.19
ORCO		42.50±0.00	11.40±2.32	18.78±1.38	17.67±3.44	22.59±2.19	-29.52±2.85
UNISA(Ours)		**67.50±0.00**	**55.47±2.62**	**45.61±0.89**	**39.86±2.41**	**52.11±1.83**	**0.00±0.00**
Baseline	3	18.33±0.00	17.87±2.16	10.06±2.10	9.29±1.34	13.89±1.65	-39.08±2.01
F2M		25.00±0.00	16.93±0.98	12.56±1.47	10.81±0.82	16.33±0.97	-36.64±1.5
FSLL		22.50±0.00	10.93±1.05	9.61±1.01	7.81±0.24	12.71±0.74	-40.26±1.37
ICARL		23.50±0.50	6.67±0.00	5.56±0.00	4.76±0.00	10.12±0.25	-42.85±1.18
Rebalance		12.50±0.23	9.80±0.44	8.56±0.18	7.14±0.19	9.50±0.28	-43.47±1.18
S3C		50.00±0.00	42.13±0.68	34.78±0.60	28.95±0.86	38.97±0.62	-14.00±1.31
SCN		23.75±0.00	10.80±1.01	7.89±1.34	6.19±1.79	12.16±1.23	-40.81±1.68
ORCO		42.50±0.00	12.87±0.88	17.44±2.64	15.57±2.83	22.1±1.98	-30.87±2.29
UNISA(Ours)		**67.50±0.00**	**56.47±1.85**	**48.28±1.16**	**39.62±0.72**	**52.97±1.15**	**0.00±0.00**
Baseline	2	18.33±0.00	15.00±2.76	9.67±0.98	7.57±0.99	12.64±1.55	-39.42±1.88
F2M		25.00±0.00	17.33±0.69	11.94±0.22	10.71±0.71	16.25±0.51	-35.81±1.19
FSLL		22.50±0.00	9.87±0.54	9.56±0.73	7.90±0.82	12.46±0.61	-39.6±1.23
ICARL		25.42±1.41	6.67±0.00	5.56±0.00	4.76±0.00	10.60±0.71	-41.46±1.28
Rebalance		12.50±0.57	10.13±0.23	8.50±0.19	7.14±0.13	9.57±0.33	-42.49±1.12
S3C		50.00±0.00	40.60±1.19	35.22±1.00	28.62±1.65	38.61±1.13	-13.45±1.56
SCN		23.75±0.00	8.20±1.45	7.39±1.68	6.67±1.02	11.50±1.22	-40.56±1.62
ORCO		42.50±0.00	11.07±0.91	16.22±1.51	15.81±2.19	21.40±1.41	-30.66±1.77
UNISA(Ours)		**67.50±0.00**	**55.00±1.21**	**46.39±1.38**	**39.33±1.11**	**52.06±1.07**	**0.00±0.00**
Baseline	1	18.33±0.00	12.33±2.89	9.67±0.93	8.19±0.67	12.13±1.55	-38.39±2.32
F2M		25.00±0.00	18.07±3.34	12.28±1.04	10.90±1.21	16.56±1.85	-33.96±2.53
FSLL		22.50±0.00	10.40±1.02	8.06±0.74	7.05±0.74	12.00±0.73	-38.52±1.87
ICARL		23.92±0.85	6.67±0.00	5.56±0.00	4.76±0.00	10.23±0.43	-40.29±1.77
Rebalance		12.58±0.15	9.87±0.30	8.50±0.12	7.29±0.21	9.56±0.21	-40.96±1.73
S3C		50.00±0.00	40.33±1.00	34.89±1.26	27.67±1.24	38.22±1.02	-12.30±2.00
SCN		23.75±0.00	7.40±1.58	6.61±1.70	5.76±0.79	10.88±1.23	-39.64±2.11
ORCO		42.50±0.00	12.87±1.04	14.61±1.91	8.86±1.63	19.71±1.36	-30.81±2.19
UNISA(Ours)		**67.50±0.00**	**55.33±0.58**	**43.00±2.56**	**36.24±2.22**	**50.52±1.72**	**0.00±0.00**

Table 6.2: Classification accuracy on split AID dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot.

Method	#S	Performance on Split AID					
		Session				Avg	Gap to UNISA
		1	2	3	4		
Baseline	5	18.06±0.00	12.87±0.69	10.01±0.29	7.90±0.43	12.21±0.43	-49.21±0.71
F2M	5	16.49±0.00	13.38±0.25	10.10±0.17	9.67±0.37	12.41±0.24	-49.01±0.61
FSLL	5	25.78±0.00	12.13±2.02	8.34±0.71	7.15±0.39	13.35±1.09	-48.07±1.23
ICARL	5	16.41±0.31	14.58±0.73	12.47±0.64	10.15±0.71	13.40±0.62	-48.02±0.84
Rebalance	5	7.52±0.00	5.41±0.00	4.34±0.00	3.60±0.00	5.22±0.00	-56.20±0.56
S3C	5	49.27±0.00	34.08±1.09	27.94±1.03	23.92±0.47	33.80±0.79	-27.62±0.97
SCN	5	36.33±0.00	13.50±1.28	11.77±1.27	12.15±0.62	18.44±0.95	-42.98±1.10
ORCO	5	31.94±0.00	14.85±1.88	10.61±1.80	12.73±1.58	17.53±1.52	-43.89±1.62
UNISA(Ours)	5	**85.91±0.00**	**63.83±0.76**	**52.23±0.52**	**43.71±0.63**	**61.42±0.56**	**0.00±0.00**
Baseline	4	18.06±0.00	13.00±0.84	9.49±0.27	8.20±0.09	12.19±0.44	-49.26±0.70
F2M	4	16.49±0.00	13.51±0.80	9.98±0.56	9.08±0.61	12.27±0.58	-49.18±0.80
FSLL	4	25.78±0.00	12.21±1.73	9.58±1.86	7.85±0.61	13.86±1.31	-47.59±1.42
ICARL	4	16.56±0.43	5.41±0.00	4.34±0.00	3.60±0.00	7.48±0.22	-53.97±0.59
Rebalance	4	7.52±0.00	5.41±0.00	4.34±0.00	3.60±0.00	5.22±0.00	-56.23±0.55
S3C	4	49.27±0.00	33.87±0.72	27.29±1.14	23.19±0.85	33.41±0.80	-28.04±0.97
SCN	4	36.33±0.00	11.92±1.24	11.14±0.97	11.70±0.89	17.77±0.90	-43.68±1.05
ORCO	4	31.94±0.00	15.36±1.39	10.04±1.96	10.16±1.94	16.88±1.54	-44.57±1.64
UNISA(Ours)	4	**85.91±0.00**	**63.42±0.80**	**52.93±0.74**	**43.54±0.21**	**61.45±0.55**	**0.00±0.00**
Baseline	3	18.06±0.00	12.01±0.80	9.31±0.43	7.91±0.71	11.82±0.58	-48.71±0.69
F2M	3	16.49±0.00	13.17±1.13	10.17±1.22	9.28±0.72	12.28±0.91	-48.25±0.99
FSLL	3	25.78±0.00	13.08±1.07	9.80±1.76	9.04±1.09	14.43±1.17	-46.10±1.23
ICARL	3	16.62±0.37	5.41±0.00	4.34±0.00	3.60±0.00	7.49±0.19	-53.04±0.42
Rebalance	3	7.52±0.00	5.41±0.00	4.34±0.00	3.60±0.00	5.22±0.00	-55.31±0.38
S3C	3	49.27±0.00	32.79±1.20	26.45±0.86	22.56±0.59	32.77±0.79	-27.76±0.88
SCN	3	36.33±0.00	11.38±1.66	9.93±1.16	10.04±1.18	16.92±1.17	-43.61±1.23
ORCO	3	31.94±0.00	12.22±1.62	9.41±1.40	10.62±0.69	16.05±1.12	-44.48±1.18
UNISA(Ours)	3	**85.91±0.00**	**62.46±0.34**	**51.06±0.28**	**42.68±0.61**	**60.53±0.38**	**0.00±0.00**
Baseline	2	18.06±0.00	10.75±0.85	9.30±1.12	8.12±0.50	11.56±0.75	-49.28±0.87
F2M	2	16.49±0.00	13.33±0.79	10.33±0.38	9.04±0.41	12.3±0.48	-48.54±0.66
FSLL	2	25.78±0.00	11.83±0.90	9.78±0.88	7.69±1.70	13.77±1.06	-47.07±1.15
ICARL	2	16.58±0.46	5.41±0.00	4.34±0.00	3.60±0.00	7.48±0.23	-53.36±0.51
Rebalance	2	7.52±0.00	5.41±0.00	4.34±0.00	3.60±0.00	5.22±0.00	-55.62±0.45
S3C	2	49.27±0.00	33.69±0.30	26.20±0.98	22.81±0.65	32.99±0.61	-27.85±0.76
SCN	2	36.33±0.00	9.49±1.41	7.10±0.92	8.79±1.10	15.43±1.01	-45.41±1.11
ORCO	2	31.94±0.00	11.31±2.32	8.61±1.50	10.52±3.37	15.6±2.18	-45.24±2.23
UNISA(Ours)	2	**85.91±0.00**	**63.20±0.39**	**51.05±0.69**	**43.20±0.41**	**60.84±0.45**	**0.00±0.00**
Baseline	1	18.06±0.00	11.53±0.80	9.59±0.35	8.13±0.15	11.83±0.44	-48.10±0.79
F2M	1	16.49±0.00	13.68±0.94	9.95±0.45	8.88±0.48	12.25±0.57	-47.68±0.87
FSLL	1	25.78±0.00	11.04±1.00	9.84±0.68	8.90±1.19	13.89±0.85	-46.04±1.08
ICARL	1	16.47±0.33	5.41±0.00	4.34±0.00	3.60±0.00	7.46±0.17	-52.47±0.68
Rebalance	1	7.52±0.00	5.41±0.00	4.34±0.00	3.60±0.00	5.22±0.00	-54.71±0.66
S3C	1	49.27±0.00	32.00±1.26	24.80±0.29	20.56±0.70	31.66±0.74	-28.27±0.99
SCN	1	36.33±0.00	8.48±1.02	6.43±0.51	7.41±0.61	14.66±0.65	-45.27±0.93
ORCO	1	31.94±0.00	13.50±1.16	8.53±0.64	7.58±0.82	15.39±0.78	-44.14±1.02
UNISA(Ours)	1	**85.91±0.00**	**62.31±0.47**	**50.19±0.32**	**41.31±1.20**	**59.93±0.66**	**0.00±0.00**

Table 6.3: Classification accuracy on split RSICB256 dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot.

Method	#S	Performance on Split RSICB256					
		Session				Avg	Gap to UNISA
		1	2	3	4		
Baseline	5	41.71±0.00	33.18±0.47	28.39±1.07	28.50±0.61	32.95±0.66	-44.12±0.98
F2M	5	21.78±0.00	18.19±0.73	16.25±0.63	14.64±1.16	17.72±0.75	-59.35±1.04
FSLL	5	42.21±0.00	24.18±2.05	21.02±0.55	23.28±0.90	27.67±1.15	-49.40±1.36
ICARL	5	31.91±0.62	2.23±0.00	1.76±0.00	1.43±0.00	9.33±0.31	-67.74±0.78
Rebalance	5	6.54±0.06	5.33±0.12	4.25±0.03	3.45±0.03	4.89±0.07	-72.18±0.72
S3C	5	82.74±0.00	69.33±0.41	62.10±0.71	57.23±0.21	67.85±0.42	-9.22±0.83
SCN	5	33.60±0.00	8.32±0.42	17.42±0.59	14.18±1.30	18.38±0.74	-58.69±1.03
ORCO	5	40.83±0.00	31.67±0.00	15.66±0.00	21.12±0.00	27.10±0.00	-49.97±1.16
UNISA(Ours)	5	**95.41±0.00**	**80.36±0.83**	**70.08±0.69**	**62.43±0.94**	**77.07±0.72**	**0.00±0.00**
Baseline	4	41.71±0.00	33.00±0.12	28.41±0.75	28.13±0.68	32.81±0.51	-43.41±1.05
F2M	4	21.78±0.00	18.19±0.65	16.52±0.36	15.32±1.16	17.95±0.69	-58.27±1.15
FSLL	4	42.21±0.00	24.18±2.05	21.02±0.55	23.28±0.90	27.67±1.15	-48.55±1.47
ICARL	4	32.42±0.49	2.23±0.00	1.76±0.00	1.43±0.00	9.46±0.25	-66.76±0.95
Rebalance	4	6.54±0.07	5.42±0.08	4.20±0.07	3.43±0.04	4.90±0.07	-71.32±0.92
S3C	4	82.74±0.00	68.81±0.32	61.30±0.77	56.00±0.42	67.21±0.47	-9.01±1.03
SCN	4	33.60±0.00	8.74±0.66	16.33±1.08	13.31±1.03	18.00±0.82	-58.22±1.23
ORCO	4	40.83±0.00	31.57±0.00	26.42±0.00	25.45±0.00	28.21±0.00	-48.01±1.38
UNISA(Ours)	4	**95.41±0.00**	**80.15±0.60**	**68.69±1.08**	**60.63±1.36**	**76.22±0.92**	**0.00±0.00**
Baseline	3	41.71±0.00	32.78±0.33	27.89±1.26	27.35±1.46	32.43±0.98	-42.34±1.23
F2M	3	21.78±0.00	18.26±0.94	16.68±0.75	14.84±1.23	17.89±0.86	-56.88±1.13
FSLL	3	42.21±0.00	22.56±2.15	19.92±1.13	22.14±1.30	26.71±1.38	-48.06±1.57
ICARL	3	32.52±0.47	2.23±0.00	1.76±0.00	1.43±0.00	9.49±0.24	-65.28±0.78
Rebalance	3	6.58±0.13	5.35±0.04	4.22±0.05	3.51±0.03	4.92±0.07	-69.85±0.74
S3C	3	82.74±0.00	69.35±0.49	61.02±0.90	55.13±0.72	67.06±0.63	-7.71±0.97
SCN	3	33.60±0.00	8.65±0.30	15.10±1.18	12.09±0.87	17.36±0.75	-57.41±1.05
ORCO	3	40.83±0.00	27.70±0.00	25.21±0.00	27.39±0.00	27.61±0.00	-47.16±1.35
UNISA(Ours)	3	**95.41±0.00**	**78.77±0.79**	**66.31±0.61**	**58.58±1.09**	**74.77±0.74**	**0.00±0.00**
Baseline	2	41.71±0.00	32.49±0.86	26.67±0.68	25.69±1.36	31.64±0.87	-42.43±1.26
F2M	2	21.78±0.00	17.65±0.66	15.82±1.05	14.85±1.62	17.53±1.02	-56.54±1.37
FSLL	2	42.21±0.00	22.03±1.86	19.23±2.01	21.73±1.14	26.3±1.48	-47.77±1.74
ICARL	2	32.28±0.60	2.23±0.00	1.76±0.00	1.43±0.00	9.43±0.30	-64.64±0.96
Rebalance	2	6.57±0.11	5.40±0.06	4.20±0.08	3.44±0.05	4.90±0.08	-69.17±0.91
S3C	2	82.74±0.00	68.73±0.71	58.97±1.43	53.32±1.02	65.94±0.95	-8.13±1.32
SCN	2	33.60±0.00	7.15±1.61	13.62±1.20	10.77±1.22	16.29±1.17	-57.78±1.48
ORCO	2	40.83±0.00	24.84±0.00	23.58±0.00	5.38±0.00	25.99±0.,00	-48.08±1.83
UNISA(Ours)	2	**95.41±0.00**	**79.19±1.00**	**65.19±0.91**	**56.49±1.21**	**74.07±0.91**	**0.00±0.00**
Baseline	1	41.71±0.00	32.50±0.62	26.13±1.18	24.40±0.73	31.19±0.76	-40.68±1.21
F2M	1	21.78±0.00	16.33±1.44	15.54±0.79	13.51±0.74	16.79±0.90	-55.08±1.30
FSLL	1	42.21±0.00	19.69±1.78	18.07±1.96	19.78±2.47	24.94±1.81	-46.93±2.04
ICARL	1	32.76±0.36	2.23±0.00	1.76±0.00	1.43±0.00	9.55±0.18	-62.32±0.96
Rebalance	1	6.56±0.08	5.38±0.04	4.21±0.05	3.47±0.05	4.91±0.06	-66.96±0.94
S3C	1	82.74±0.00	67.37±0.77	58.27±0.99	51.83±0.71	65.05±0.72	-6.82±1.18
SCN	1	33.60±0.00	5.43±0.80	8.56±2.04	8.31±1.83	13.98±1.43	-57.89±1.71
ORCO	1	40.83±0.00	10.72±0.00	13.61±0.00	14.79±0.00	20.55±0.00	-51.32±1.22
UNISA(Ours)	1	**95.41±0.00**	**77.68±0.71**	**62.37±1.12**	**52.02±1.32**	**71.87±0.94**	**0.00±0.00**

Table 6.4: Classification accuracy on split RESISC45 dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot.

Method	#S	Performance on Split RESISC45					
		Session				Avg	Gap to UNISA
		1	2	3	4		
Baseline	5	16.93±0.00	11.24±0.00	10.21±0.00	8.63±0.00	11.75±0.00	-54.66±0.00
F2M	5	16.52±0.00	13.00±0.00	10.92±0.00	9.33±0.00	12.44±0.00	-53.97±0.00
FSLL	5	16.70±0.00	11.43±0.00	8.46±0.00	7.05±0.00	10.91±0.00	-55.50±0.00
ICARL	5	16.92±0.18	3.23±0.00	2.63±0.00	2.22±0.00	6.25±0.09	-60.16±0.09
Rebalance	5	4.32±0.02	3.31±0.01	2.70±0.01	2.28±0.01	3.15±0.01	-63.26±0.01
S3C	5	47.11±0.00	36.71±0.64	30.41±0.21	26.04±0.24	35.07±0.36	-31.34±0.36
SCN	5	14.61±0.00	5.37±0.46	6.48±0.38	6.63±0.56	8.27±0.41	-58.14±0.41
ORCO	5	34.35±0.00	8.86±0.94	10.67±1.09	10.06±1.12	15.99±0.91	-50.43±0.91
UNISA(Ours)	5	**88.51±0.00**	**69.56±0.21**	**57.93±0.18**	**49.62±0.71**	**66.41±0.38**	**0.00±0.00**
Baseline	4	16.93±0.00	11.89±0.00	10.98±0.00	9.10±0.00	12.23±0.00	-53.83±0.00
F2M	4	16.01±0.00	13.29±0.00	11.45±0.00	9.08±0.00	12.46±0.00	-53.59±0.00
FSLL	4	16.55±0.00	11.29±0.00	7.80±0.00	6.63±0.00	10.57±0.00	-55.48±0.00
ICARL	4	17.05±0.17	3.23±0.00	2.63±0.00	2.22±0.00	6.28±0.09	-59.77±0.09
Rebalance	4	4.29±0.02	3.32±0.01	2.71±0.00	2.29±0.00	3.15±0.01	-62.90±0.01
S3C	4	47.11±0.00	36.82±0.47	30.50±0.54	25.90±0.52	35.08±0.44	-30.97±0.44
SCN	4	14.61±0.00	5.29±0.43	6.64±0.62	6.04±0.48	8.15±0.45	-57.91±0.45
ORCO	4	34.35±0.00	10.12±0.76	8.89±1.82	9.38±0.57	15.69±1.03	-50.37±1.03
UNISA(Ours)	4	**88.51±0.00**	**69.36±0.40**	**57.42±0.46**	**48.90±0.45**	**66.05±0.38**	**0.00±0.00**
Baseline	3	17.05±0.00	11.96±0.00	10.60±0.00	8.86±0.00	12.12±0.00	-53.29±0.00
F2M	3	16.07±0.00	12.10±0.00	11.13±0.00	9.79±0.00	12.27±0.00	-53.14±0.00
FSLL	3	16.34±0.00	10.51±0.00	8.89±0.00	7.44±0.00	10.80±0.00	-54.62±0.00
ICARL	3	16.98±0.09	3.23±0.00	2.63±0.00	2.22±0.00	6.27±0.05	-59.15±0.05
Rebalance	3	4.29±0.02	3.33±0.01	2.70±0.01	2.29±0.01	3.15±0.01	-62.26±0.01
S3C	3	47.11±0.00	36.13±0.62	29.92±0.55	25.06±0.40	34.56±0.46	-30.86±0.46
SCN	3	14.61±0.00	5.40±0.13	5.89±0.52	6.02±0.94	7.98±0.54	-57.43±0.54
ORCO	3	34.35±0.00	8.69±1.02	9.00±1.19	7.99±1.62	15.01±1.13	-50.40±1.13
UNISA(Ours)	3	**88.51±0.00**	**68.82±0.56**	**56.29±0.81**	**48.03±0.74**	**65.41±0.62**	**0.00±0.00**
Baseline	2	16.73±0.00	11.24±0.00	9.94±0.00	7.98±0.00	11.47±0.00	-53.95±0.00
F2M	2	15.48±0.00	11.77±0.00	10.28±0.00	8.94±0.00	11.62±0.00	-53.80±0.00
FSLL	2	16.93±0.00	9.65±0.00	8.14±0.00	6.13±0.00	10.21±0.00	-55.21±0.00
ICARL	2	17.10±0.12	3.23±0.00	2.63±0.00	2.22±0.00	6.30±0.06	-59.13±0.06
Rebalance	2	4.30±0.01	3.32±0.01	2.71±0.01	2.29±0.01	3.16±0.01	-62.27±0.01
S3C	2	47.11±0.00	36.09±0.58	29.11±0.16	24.63±0.47	34.24±0.38	-31.19±0.38
SCN	2	14.61±0.00	5.25±0.55	4.83±0.63	5.03±0.57	7.43±0.51	-57.99±0.51
ORCO	2	34.35±0.00	8.58±1.27	7.88±2.35	8.20±1.74	14.75±1.59	-50.67±1.59
UNISA(Ours)	2	**88.51±0.00**	**68.53±0.66**	**56.65±0.39**	**47.99±0.65**	**65.42±0.50**	**0.00±0.00**
Baseline	1	16.61±0.00	12.47±0.00	10.58±0.00	7.94±0.00	11.90±0.00	-52.99±0.00
F2M	1	16.76±0.00	11.82±0.00	9.42±0.00	8.16±0.00	11.54±0.00	-53.35±0.00
FSLL	1	16.37±0.00	8.80±0.00	7.78±0.00	4.86±0.00	9.45±0.00	-55.44±0.00
ICARL	1	17.17±0.28	3.23±0.00	2.63±0.00	2.22±0.00	6.31±0.14	-58.58±0.14
Rebalance	1	4.30±0.01	3.33±0.02	2.72±0.01	2.29±0.01	3.16±0.01	-61.73±0.01
S3C	1	47.11±0.00	35.65±1.07	27.97±0.75	22.93±0.82	33.42±0.77	-31.48±0.77
SCN	1	14.61±0.00	4.63±0.45	4.17±0.98	3.85±0.42	6.82±0.58	-58.08±0.58
ORCO	1	34.35±0.00	7.81±1.29	5.97±0.76	5.95±0.34	13.52±0.77	-51.37±0.77
UNISA(Ours)	1	**88.51±0.00**	**68.26±0.47**	**55.68±0.57**	**47.09±0.57**	**64.89±0.47**	**0.00±0.00**

Table 6.5: Classification accuracy on split EuroSAT dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot.

Method	#S	Performance on Split EuroSAT					
		Session				Avg	Gap to UNISA
		1	2	3	4		
Baseline	5	53.39±0.00	46.77±0.85	34.66±0.86	27.61±0.97	40.61±0.78	-33.04±1.95
F2M	5	56.22±0.00	43.12±1.10	33.91±1.61	27.53±1.52	40.20±1.24	-33.45±2.18
FSLL	5	57.78±0.00	30.73±2.26	21.82±2.22	23.14±2.44	33.37±2.00	-40.28±2.68
ICARL	5	36.48±0.15	30.58±1.99	23.70±2.61	20.54±1.72	27.83±1.85	-45.82±2.57
Rebalance	5	26.09±0.00	18.75±0.00	13.95±0.00	11.11±0.00	17.48±0.00	-56.17±1.79
S3C	5	84.26±0.00	65.87±0.28	52.31±1.62	43.53±1.16	61.49±1.01	-12.16±2.06
SCN	5	67.35±0.00	27.52±1.07	18.21±3.61	16.66±2.15	32.44±2.17	-41.21±2.81
ORCO	5	71.70±0.00	27.28±0.23	20.66±3.41	20.57±3.53	35.05±2.46	-38.60±3.04
UNISA(Ours)	5	**96.70±0.00**	**79.06±1.71**	**64.95±2.45**	**53.89±1.98**	**73.65±1.79**	**0.00±0.00**
Baseline	4	53.39±0.00	46.99±0.45	34.20±0.67	27.46±0.64	40.51±0.52	-31.76±1.74
F2M	4	56.22±0.00	42.66±1.15	32.45±1.09	27.06±0.80	39.60±0.89	-32.67±1.88
FSLL	4	57.78±0.00	39.43±0.65	26.43±2.44	26.16±1.39	37.45±1.44	-34.82±2.20
ICARL	4	36.54±0.12	18.75±0.00	13.95±0.00	11.11±0.00	20.09±0.06	-52.18±1.66
Rebalance	4	26.09±0.00	18.75±0.00	13.95±0.00	11.11±0.00	17.48±0.00	-54.79±1.66
S3C	4	84.26±0.00	66.11±0.93	53.20±1.40	44.13±0.89	61.93±0.95	-10.34±1.91
SCN	4	67.35±0.00	27.20±0.72	18.40±2.98	13.10±1.67	31.51±1.75	-40.76±2.41
ORCO	4	71.70±0.00	26.41±1.73	26.27±4.56	19.08±1.61	35.87±2.57	-36.40±3.06
UNISA(Ours)	4	**96.70±0.00**	**78.42±0.95**	**62.93±2.06**	**51.01±2.43**	**72.27±1.66**	**0.00±0.00**
Baseline	3	53.39±0.00	46.48±0.98	33.54±1.60	27.80±0.73	40.30±1.01	-30.92±2.69
F2M	3	56.22±0.00	42.56±1.07	32.61±0.67	26.51±0.63	39.48±0.71	-31.74±2.59
FSLL	3	57.78±0.00	39.48±1.37	27.23±0.94	25.77±1.14	37.57±1.01	-33.65±2.69
ICARL	3	36.63±0.20	18.75±0.00	13.95±0.00	11.11±0.00	20.11±0.10	-51.11±2.49
Rebalance	3	26.09±0.00	18.75±0.00	13.95±0.00	11.11±0.00	17.48±0.00	-53.74±2.49
S3C	3	84.26±0.00	66.03±0.40	52.15±1.30	43.81±1.51	61.56±1.02	-9.66±2.69
SCN	3	67.35±0.00	26.39±0.92	19.57±2.17	17.13±1.87	32.61±1.50	-38.61±2.91
ORCO	3	71.70±0.00	24.26±2.64	23.63±5.92	19.44±0.41	34.76±3.25	-36.46±4.09
UNISA(Ours)	3	**96.70±0.00**	**78.88±1.01**	**60.47±2.44**	**48.81±4.22**	**71.22±2.49**	**0.00±0.00**
Baseline	2	53.39±0.00	45.09±0.96	32.75±2.52	26.40±1.54	39.41±1.55	-31.88±2.43
F2M	2	56.22±0.00	40.83±1.39	32.21±0.74	27.11±1.04	39.09±0.94	-32.20±2.09
FSLL	2	57.78±0.00	39.88±1.22	26.98±1.01	24.21±1.29	37.21±1.02	-34.08±2.13
ICARL	2	36.61±0.13	18.75±0.00	13.95±0.00	11.11±0.00	20.11±0.07	-51.18±1.87
Rebalance	2	26.09±0.00	18.75±0.00	13.95±0.00	11.11±0.00	17.48±0.00	-53.81±1.87
S3C	2	84.26±0.00	65.30±1.17	51.85±2.18	42.75±2.46	61.04±1.74	-10.25±2.55
SCN	2	67.35±0.00	24.76±2.23	13.67±1.36	16.29±2.64	30.52±1.86	-40.77±2.64
ORCO	2	71.70±0.00	22.42±2.46	20.08±2.32	19.20±5.55	33.35±3.25	-37.94±3.75
UNISA(Ours)	2	**96.70±0.00**	**76.84±0.91**	**61.54±1.73**	**50.08±3.19**	**71.29±1.87**	**0.00±0.00**
Baseline	1	53.39±0.00	41.69±3.05	31.10±1.80	25.56±1.34	37.94±1.89	-31.92±2.52
F2M	1	56.22±0.00	40.55±1.66	31.21±1.02	24.71±1.50	38.17±1.23	-31.69±2.07
FSLL	1	57.78±0.00	38.79±1.68	26.51±1.91	22.16±1.71	36.31±1.53	-33.55±2.26
ICARL	1	36.49±0.14	18.75±0.00	13.95±0.00	11.11±0.00	20.08±0.07	-49.78±1.66
Rebalance	1	26.09±0.00	18.75±0.00	13.95±0.00	11.11±0.00	17.48±0.00	-52.38±1.66
S3C	1	84.26±0.00	63.38±2.29	49.35±2.04	39.71±1.96	59.18±1.82	-10.68±2.46
SCN	1	67.35±0.00	24.39±1.15	14.48±1.21	10.61±1.98	29.21±1.29	-40.65±2.10
ORCO	1	71.70±0.00	24.67±0.99	19.03±1.25	18.32±0.77	33.43±0.89	-36.43±1.88
UNISA(Ours)	1	**96.70±0.00**	**75.06±1.56**	**59.69±1.50**	**48.00±2.52**	**69.86±1.66**	**0.00±0.00**

Table 6.6: Classification accuracy on split HyRank dataset with 1-5 shot setting averaged across 5 times run, symbol #S denote the number of shot.

Method	#S	Performance on Split HyRank (Loukia)					
		Session				Avg	Gap to UNISA
		1	2	3	4		
Baseline	5	33.42±0.00	26.56±3.59	21.51±2.33	23.37±2.03	26.22±2.37	-17.48±2.65
F2M	5	16.79±0.00	10.27±2.44	9.26±2.36	5.43±1.15	10.44±1.79	-33.26±2.14
FSLL	5	35.27±0.00	26.12±1.46	21.54±2.10	23.91±0.53	26.71±1.31	-16.99±1.76
ICARL	5	26.78±1.03	31.29±0.00	29.44±0.00	29.06±0.00	29.14±0.52	-14.56±1.29
Rebalance	5	34.31±0.00	31.29±0.00	29.44±0.00	29.06±0.00	31.03±0.00	-12.67±1.18
S3C	5	49.86±0.00	34.31±0.83	30.71±0.60	28.76±1.03	35.91±0.73	-7.79±1.39
SCN	5	47.48±0.00	8.74±0.06	8.16±1.11	5.10±1.56	17.37±0.96	-26.33±1.52
ORCO	5	36.37±0.00	7.86±0.59	4.65±1.36	1.70±0.84	12.65±0.85	-31.05±1.45
UNISA(Ours)	5	**52.75±0.00**	**44.56±0.51**	**39.84±1.39**	**37.64±1.83**	**43.70±1.18**	**0.00±0.00**
Baseline	4	33.42±0.00	27.62±2.36	23.13±1.91	19.72±3.02	25.97±2.14	-17.75±2.48
F2M	4	16.79±0.00	11.05±1.12	7.30±1.90	4.43±1.51	9.89±1.34	-33.83±1.84
FSLL	4	35.27±0.00	25.39±0.60	23.62±1.52	22.65±1.77	26.73±1.20	-16.99±1.74
ICARL	4	25.77±0.67	31.29±0.00	29.44±0.00	29.06±0.00	28.89±0.34	-14.83±1.31
Rebalance	4	34.31±0.00	31.29±0.00	29.44±0.00	29.06±0.00	31.03±0.00	-12.69±1.26
S3C	4	49.86±0.00	34.49±0.86	29.74±1.20	28.62±1.14	35.68±0.93	-8.04±1.57
SCN	4	47.39±0.00	8.79±0.00	6.62±1.46	3.13±0.80	16.48±0.83	-27.24±1.51
ORCO	4	36.37±0.00	7.86±0.59	4.65±1.36	1.70±0.84	12.65±0.85	-31.07±1.50
UNISA(Ours)	4	**52.75±0.00**	**45.15±0.97**	**39.85±1.26**	**37.14±1.96**	**43.72±1.26**	**0.00±0.00**
Baseline	3	33.42±0.00	29.85±0.58	23.95±2.22	23.01±2.67	27.56±1.76	-16.46±1.91
F2M	3	16.79±0.00	9.57±2.46	4.95±1.28	5.32±1.24	9.16±1.52	-34.86±1.69
FSLL	3	35.27±0.00	26.05±1.71	23.88±0.37	23.81±0.82	27.25±0.97	-16.77±1.22
ICARL	3	26.98±1.73	31.29±0.00	29.44±0.00	29.06±0.00	29.19±0.87	-14.83±1.14
Rebalance	3	34.31±0.00	31.29±0.00	29.44±0.00	29.06±0.00	31.03±0.00	-12.99±0.74
S3C	3	49.86±0.00	35.05±0.64	30.50±0.94	28.23±1.83	35.91±1.08	-8.11±1.31
SCN	3	47.39±0.00	8.75±0.02	7.21±1.30	4.62±1.05	16.99±0.84	-27.03±1.12
ORCO	3	36.37±0.00	7.86±0.59	4.65±1.36	1.70±0.84	12.65±0.85	-31.37±1.13
UNISA(Ours)	3	**52.75±0.00**	**44.90±0.65**	**40.49±1.08**	**37.93±0.76**	**44.02±0.74**	**0.00±0.00**
Baseline	2	33.42±0.00	30.40±0.42	24.87±2.27	23.43±1.55	28.03±1.39	-15.61±1.55
F2M	2	16.79±0.00	9.87±1.83	6.46±1.15	5.53±1.19	9.66±1.23	-33.98±1.41
FSLL	2	35.27±0.00	25.87±2.71	24.89±1.04	23.10±1.31	27.28±1.59	-16.36±1.73
ICARL	2	24.09±5.44	31.29±0.00	29.44±0.00	29.06±0.00	28.47±2.72	-15.17±2.80
Rebalance	2	34.31±0.00	31.29±0.00	29.44±0.00	29.06±0.00	31.03±0.00	-12.61±0.68
S3C	2	49.86±0.00	34.54±0.99	29.61±1.20	27.58±1.15	35.4±0.97	-8.24±1.18
SCN	2	47.48±0.00	8.54±0.37	5.26±2.51	3.07±1.42	16.09±1.45	-27.55±1.60
ORCO	2	36.37±0.00	7.72±0.45	4.55±1.56	1.79±1.16	12.61±1.00	-31.03±1.21
UNISA(Ours)	2	**52.75±0.00**	**44.32±1.05**	**39.78±0.77**	**37.72±0.36**	**43.64±0.68**	**0.00±0.00**
Baseline	1	33.42±0.00	30.45±0.39	27.97±0.29	27.46±0.53	29.83±0.36	-15.03±1.03
F2M	1	16.79±0.00	10.98±1.65	8.35±1.70	6.30±0.98	10.61±1.28	-34.25±1.60
FSLL	1	35.27±0.00	24.75±1.99	25.20±1.56	23.97±1.01	27.30±1.36	-17.56±1.66
ICARL	1	21.72±7.02	31.29±0.00	29.44±0.00	29.06±0.00	27.88±3.51	-16.98±3.64
Rebalance	1	34.31±0.00	31.29±0.00	29.44±0.00	29.06±0.00	31.03±0.00	-13.83±0.96
S3C	1	49.86±0.00	33.57±3.28	27.51±2.31	26.41±1.77	34.34±2.19	-10.52±2.39
SCN	1	47.48±0.00	7.99±1.24	3.51±1.14	2.60±1.26	15.4±1.05	-29.46±1.42
ORCO	1	36.37±0.00	7.36±0.61	3.60±0.55	0.93±0.08	12.07±0.41	-32.79±1.04
UNISA(Ours)	1	**52.75±0.00**	**44.80±0.70**	**41.49±1.15**	**40.38±1.37**	**44.86±0.96**	**0.00±0.00**

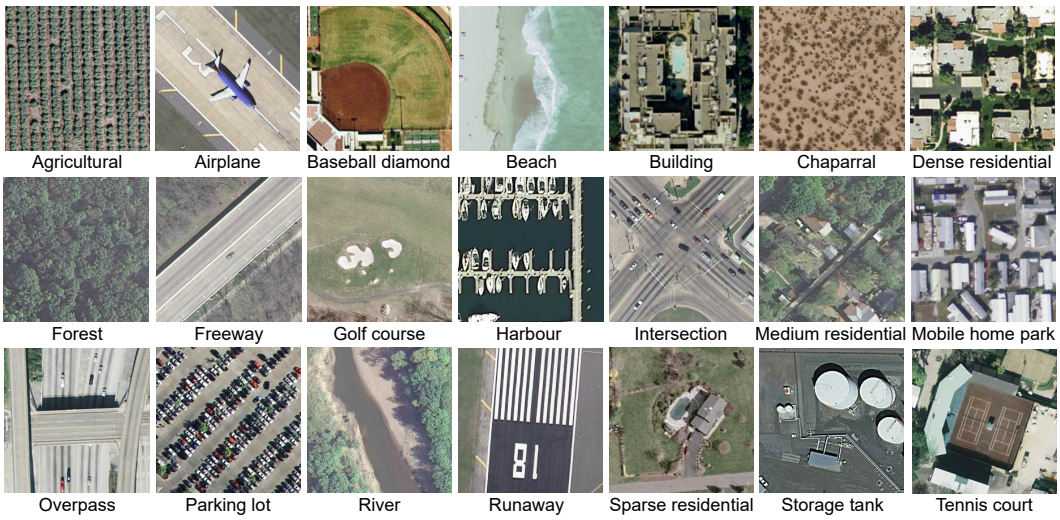


Figure 6.4: Visualization of UC Merced dataset.

Table 6.7: Ablation study of the proposed method with various configurations on the 5-shot UC Merced dataset, FM and WM stand for Flat Minima and Wide Minima, respectively.

Code	Configuration						Session				Avg
	FM	WM	($\mathcal{L}_{cpr}$)	$\mathcal{L}_{psl}$	$\mathcal{L}_{psa}$	$\mathcal{L}_{ball}$	1	2	3	4	
A			✓	✓	✓	✓	65.00±0.00	55.80±1.35	48.83±1.19	40.95±1.05	52.65±1.04
B	✓			✓	✓	✓	62.92±0.00	56.27±1.98	50.22±1.48	43.33±0.66	53.19±1.28
C	✓	✓			✓	✓	65.83±0.00	55.20±0.86	46.17±0.97	38.43±1.01	51.41±0.82
D	✓	✓	✓			✓	51.67±0.00	45.80±1.74	41.33±1.56	35.10±2.62	43.48±1.76
E	✓	✓	✓	✓			66.67±0.00	56.93±0.34	47.00±1.89	40.05±1.45	52.66±1.20
UNISA	✓	✓	✓	✓	✓	✓	**67.50±0.00**	**58.27±1.51**	**51.67±0.77**	**43.76±1.50**	**55.30±1.13**

Table 6.8: Extended ablation study of the proposed method with various loss coefficients, i.e., $\lambda_i, i \in [1-5]$ on 5-shot UC Merced dataset, * denotes the result of our default configuration.

Code	λ_1	λ_2	λ_3	λ_4	λ_5	Avg. Accuracy
U1	**1.0**	**1.0**	**1.0**	**0.1**	**1.0**	***55.30+-1.13**
U2	1.0	1.0	0.1	0.1	1.0	54.35+-1.47
U3	1.0	0.1	1.0	0.1	1.0	54.35+-1.47
U4	1.0	0.1	0.1	0.1	1.0	54.35+-1.47
U5	0.1	1.0	1.0	0.1	1.0	49.16+-1.02
U6	0.1	1.0	0.1	0.1	1.0	49.16+-1.02
U7	0.1	0.1	1.0	0.1	1.0	49.16+-1.02
U8	0.1	0.1	0.1	0.1	1.0	49.16+-1.02
U9	1.0	1.0	1.0	0.1	0.1	55.31+-1.14
U10	1.0	1.0	1.0	0.01	1.0	55.31+-1.14
U11	1.0	1.0	1.0	0.01	0.1	55.31+-1.14

Table 6.9: Robustness analysis of the proposed method against various noises on the 5-shot UC Merced dataset.

Method	Noise	Session				Avg
		1	2	3	4	
UNISA(Ours)	-	**67.50±0.00**	**58.27±1.51**	**51.67±0.77**	**43.76±1.50**	**55.30±1.13**
UNISA(Ours)	Brightness	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
UNISA(Ours)	Contrass	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
UNISA(Ours)	Hue	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
UNISA(Ours)	Saturation	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
UNISA(Ours)	Blur	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
UNISA(Ours)	Gaussian	66.25±0.00	57.67±0.92	49.72±1.85	42.86±1.11	54.13±1.17
Baseline	Gaussian	19.58±0.00	17.33±0.87	14.22±1.44	11.71±0.95	15.71±0.97
F2M	Gaussian	25.42±0.00	16.27±0.39	13.39±1.27	10.81±1.13	16.47±0.87
FSLL	Gaussian	24.17±0.00	11.73±1.29	9.39±1.26	7.90±1.01	13.30±1.03
ICARL	Gaussian	26.00±2.55	20.40±0.50	17.28±1.39	14.43±0.86	19.53±1.53
Rebalance	Gaussian	13.33±0.40	10.80±0.40	9.00±0.12	7.57±0.16	10.18±0.30
S3C	Gaussian	53.75±0.00	41.40±1.79	33.11±0.56	30.14±1.45	39.60±1.19
SCN	Gaussian	21.67±0.00	10.67±1.57	8.61±1.90	8.76±1.11	12.43±1.35
ORCO	Gaussian	35.83±0.00	10.67±2.49	14.89±3.32	15.62±2.74	19.25±2.49

Table 6.10: Robustness analysis of the proposed method in various hyperparameter settings on the 5-shot UC Merced dataset. Min.LR and BS denote the minimum learning rate and batch size, respectively.

Hyper Params.		Session				Avg
Min.LR	BS	1	2	3	4	
0.001	128	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.0001	128	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.00001	128	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.0	128	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.0	64	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.0	32	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.0	16	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.0	8	67.50±0.00	57.20±1.74	49.61±1.79	43.38±0.96	54.42±1.00
0.0	4	67.50±0.00	55.40±2.00	47.56±2.16	40.10±1.47	52.64±1.64

Table 6.11: Sensitivity analysis of the proposed method with various bound (b) value on 5-shot UC Merced dataset.

Bound (b)	Session				Average
	1	2	3	4	
0.001	69.58±0.00	57.73±1.85	51.11±1.35	43.48±1.23	55.48±1.30
0.005	69.58±0.00	58.00±1.81	51.11±0.92	43.86±1.36	55.64±1.22
0.01	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
0.05	69.58±0.00	58.00±1.81	51.11±0.92	43.86±1.36	55.64±1.22
0.1	68.33±0.00	58.07±1.20	50.61±0.42	42.43±0.52	54.86±0.69

Table 6.12: Impact of the number of perturbation loops (M) on the proposed method performance on the 5-shot UC Merced dataset.

Perturbation loop (M)	Session				Average
	1	2	3	4	
1	64.58±0.00	55.87±1.43	48.94±1.09	40.90±0.95	52.57±1.02
2	67.50±0.00	58.27±1.51	51.67±0.77	43.76±1.50	55.30±1.13
3	65.42±0.00	57.07±1.40	49.33±1.75	41.57±1.63	53.35±1.39
4	67.50±0.00	60.27±1.29	53.39±0.81	46.00±1.34	56.79±1.01
5	72.50±0.00	58.93±2.04	50.22±1.01	43.43±0.41	56.27±1.16
6	75.83±0.00	63.20±1.59	54.28±1.22	45.57±0.99	59.72±1.12
7	79.17±0.00	66.07±1.11	56.78±1.33	47.86±0.48	62.47±0.90

Table 6.13: Running time of the consolidated algorithms.

Method	Training time per epoch (s)					
	UC Merced		AID		RSICB256	
	Base task	Few-shot task	Base task	Few-shot task	Base task	Few-shot task
Baseline	0.54±0.01	0.04±0.00	1.33±0.02	0.09±0.02	0.73±0.02	0.15±0.01
F2M	0.64±0.01	0.73±0.05	1.38±0.06	1.11±0.08	1.54±0.01	0.57±0.03
FSLL	0.54±0.01	0.68±0.07	1.33±0.02	1.36±0.04	0.73±0.02	1.02±0.05
ICARL	0.59±0.02	0.30±0.01	1.27±0.04	0.62±0.10	0.71±0.03	1.64±0.16
Rebalance	0.58±0.02	0.34±0.00	1.15±0.01	1.67±0.42	0.81±0.04	1.64±0.16
S3C	1.18±0.05	0.62±0.01	3.14±0.04	0.95±0.06	2.11±0.35	0.84±0.08
SCN	0.84±0.01	2.40±0.17	6.53±0.20	7.05±0.94	3.60±0.92	7.10±0.54
ORCO	2.20±0.13	1.15±0.01	4.31±0.22	1.04±0.03	2.59±0.14	0.83±0.02
UNISA	2.60±0.55	1.88±0.18	11.67±0.58	3.52±0.56	20.67±0.58	2.34±0.50
	RESISC45		EuroSAT		HyRank	
	Base task	Few-shot task	Base task	Few-shot task	Base task	Few-shot task
Baseline	0.96±0.09	0.11±0.02	0.65±0.01	0.15±0.01	0.62±0.01	0.10±0.00
F2M	1.13±0.06	0.41±0.02	0.84±0.01	0.68±0.05	0.65±0.00	0.78±0.02
FSLL	0.96±0.09	0.36±0.05	0.65±0.01	0.7±0.08	0.62±0.01	0.67±0.02
ICARL	0.67±0.07	1.77±0.15	0.79±0.10	1.37±0.06	0.60±0.00	1.35±0.04
Rebalance	0.65±0.10	1.68±0.13	0.63±0.01	1.26±0.02	0.79±0.00	1.27±0.01
S3C	2.90±0.05	1.28±0.02	2.41±0.72	0.86±0.02	1.51±0.09	0.76±0.02
SCN	4.03±0.00	8.20±0.30	2.17±0.56	5.97±0.89	1.64±0.14	7.63±0.21
ORCO	3.62±0.75	1.38±0.06	2.28±0.22	1.18±0.11	2.71±0.36	1.25±0.02
UNISA	29.33±0.58	1.67±0.58	17.33±0.58	1.58±0.37	4.33±0.58	2.11±0.36

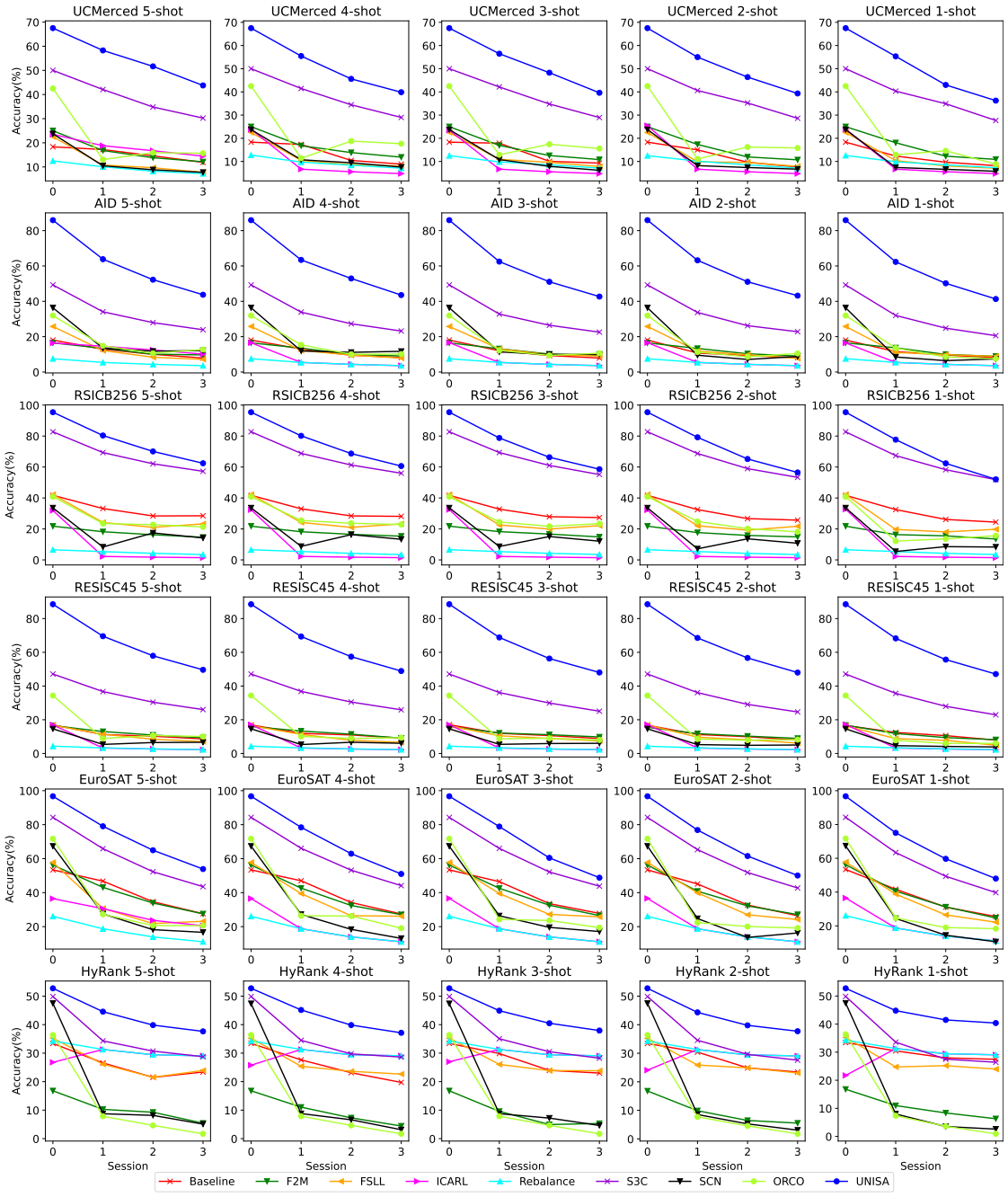


Figure 6.5: Visualization of Performances of Consolidated Algorithms across 6 datasets in 1-5 shot settings.

6.5 Experiment Results and Analysis

This section presents numerical validations of UNISA, including experiments on remote sensing problems, comparisons against prior arts, ablation studies to demonstrate the advantage of each learning module and sensitivity analysis. Source codes of UNISA are shared publicly in https://github.com/anwarmaxsum/UNISA.

Dataset:

6.5.1 Dataset

All consolidated algorithms are benchmarked in six RS datasets, i.e., UC-Merced [216], the AID [217], the RSICB-256 [218], the RESISC45 [219], the EuroSAT [220], and the HyRANK [199]. The UC Merced dataset comprises 21 classes, where each class carries 100 images. It is split into 4 tasks where the base task possesses 12 classe,s while each few-shot task includes 3 classes. The spatial resolution of this dataset is 0.3 m. Visualization of the UC Merced problem is offered in Fig. 6.4. The AID dataset presents 30 categories where each category has around 300 images with the spatial resolution between 0.5 m and 8 m. This dataset is divided into 4 tasks where the base task encompasses 15 classes, while each few-shot task consists of 5 classes. The dataset, RSICB256, has 35 classes where each class comprises 800 images with spatial resolution between 0.22 m and 3 m. This dataset is set to 4 tasks, where the base task consists of 20 classes, and each few-shot task carries 5 classes. RESISC45 dataset carries 45 classes, where each class contains 700 images with spatial resolution between 0.2 m and more than 30m/px. This dataset is set to 4 tasks, where the base task consists of 24 classes, and each few-shot task carries 7 classes. The EuroSAT dataset portrays the land-use classification problem of the Sentinel 2 satellite. This dataset comprises 10 classes where the base task covers 4 classes, while each few-shot task carries 2 classes. The HyRANK dataset is a hyperspectral dataset developed by the International Society for Photogrammetry and Remote Sensing scientific initiatives [199]. The dataset is collected by the Hyperion sensor which has 176 spectral bands. It consists of two scenes, Dioni and Loukia, where we focus only on the Loukia scene here, having 249×945 pixels. This dataset comprises 12 classes, where the first 6 classes are

reserved for the base class while the few-shot tasks carry 2 classes, totaling 3 few-shot tasks. We only apply a maximum 10% (25 samples) label proportion of the base task, while the few-shot tasks exploit the full label proportions due to the data scarcity problems. 1-5 shot cases are attempted in our experiments. Note that labeled samples are only used for inferences in UNISA, while the training process itself is free of labeled samples.

6.5.2 Baseline

UNISA is compared against iCaRL [31], Rebalance [34], FSLL [56], F2M [42], S3C [192], and ORCO [45] representing prior arts of the few-shot continual learning algorithms. In addition, a baseline is added and reflects the case where a network is frozen while leaving only the prototype construction in the few-shot learning phase. we also compare our approach with state-of-the-art continual RS learning, namely asymmetric collaborative network (SCN) [195]. Since these algorithms are fully supervised in nature, they are only trained with 10% labeled samples of the base task. There are no existing algorithms designed specifically for unsupervised few-shot continual learning problems. All consolidated algorithms are executed under the same computational environments, a single NVDIA A100 GPU with 40 GB RAM, using their official implementations. Note that the competitor methods are trained in a supervised way, utilizing all available labels both in the base task and the few-shot tasks, while our proposed method (UNISA) is trained in an unsupervised way using unlabeled samples in both tasks. UNISA only utilizes labels for inference (testing).

6.5.3 Experiment Setup

All the methods are evaluated in the mentioned six datasets under 1-shot, 2-shot, 3-shot, 4-shot, and 5-shot configurations. The hyperparameter setting follows the setting from F2M [42] for competitor methods and UNISA. UNISA also follows the same hyperparameters as [204]. **Base Task**: F2M is run with 1000 epochs, 0.1 initial learning rate, 0.01 bound value, and 2 random times. Baseline and FSLL are run with 240 epochs and 0.1 initial learning rate. ICARL and Rebalance are run with 130 epochs and 0.1 initial learning rate. UNISA is run with 1000

epochs, 0.05 initial learning rate, 0.01 bound value, and 2 random times. **Few-Shot Task**: F2M is run with 6 epochs, and 0.02 initial learning rate. Baseline model only generates prototypes (not updated) in the few-shot task. FSLL is run with 16 epochs and 0.001 initial learning rate. ICARL and Rebalance are run with 90 epochs and 0.01 initial learning rate. UNISA is run with a maximum of 5 epochs and a predicted learning rate by cosine annealing. All methods use 0.9 momentum, 5e-4 weight decay, 128 batch size, SGD optimizer, learning rate scheduler, and ResNet18 as base network. These hyperparameters are found using the grid search to ensure fair comparisons. λ_i for $i \in [1, 5]$ is set to 1.0 except λ_4 that is set to 0.1. The values for the coefficient are chosen so that each term in equation (9) is in the range [0.0, 1.0].

6.5.4 Numerical Results

Table 6.1 reports consolidated numerical results of the UC merced problem. UNISA outperforms its counterparts with large margins in 1-5 shots settings, i.e., over 30%. This applies for both per task accuracy and average accuracy. The same pattern is observed in the AID dataset shown in Table 6.2. UNISA beats other algorithms with over 50% gaps for all cases. This finding is consistent also for the RSICB256 problem, RESISC45, and the EuroSAT problem as reported in Table 6.3, 6.4 and 6.5 where UNISA delivers the most encouraging performances with large gaps, over 30% differences. In addition, UNISA performs well in the hyperspectral dataset, the HyRank dataset, with over 8% margins compared to its competitors as reported in Table 6.6. Large margins confirm the challenging characteristics of the unsupervised few-shot continual learning which cannot be handled by existing few-shot continual learning algorithms, calling for fully labeled samples. There are no direct competitors of UNISA specifically devised for unsupervised few-shot continual learning problems. In addition, RS images are also difficult to deal with because of their spatio-temporal natures. It is seen by the low base task accuracy of consolidated algorithms except UNISA. On the other hand, UNISA copes with the RS data and low labels equally well, where its base task accuracy is significantly better than prior arts. Fig. 6.5 visualizes the trace of classification accuracy per task across all four datasets. It is seen that UNISA

is superior to the other seven algorithms with large margins. Its performances are consistent across all tasks, and the accuracy of the base task is significantly higher than that of other algorithms. This fact indicates the success of UNISA in handling low-label situations and spatio-temporal natures of RS images.

6.5.5 Ablation Study

This subsection analyzes UNISA performances under five different configurations, where numerical results are reported in Table 6.7. Our ablation study is run under the UC Merced problem. Configuration A presents the absence of flat minima concepts, while configuration B portrays UNISA's performances without the wide minima concept. Configuration C and D, respectively, stand for the absence of prototype scattering loss and the positive alignment loss. Last but not least, configuration E offers the case without the ball generator loss. It is confirmed that each learning configuration plays a key role in UNISA's performance, where the absence of one loss function leads to degraded performance. The most substantial deterioration is observed in configuration D without the positive alignment loss, where the performance is compromised by over 12%. This finding demonstrates the presence of a class collision issue. The absence of prototype scattering loss in configuration C also leads to dramatic performance losses. This result signifies a lack of uniformity in terms of bringing down generalization performances. Configurations A and B also exhibit lower performances than UNISA attributed to the catastrophic forgetting problem. Configuration E is worse by about 3% than UNISA.

6.5.6 The Balance of Loss Coefficients

We extend our ablation study to investigate the impact of loss coefficient $\lambda_i, i \in [1, 5]$ on the average performance of UNISA. Table 6.8 shows the average performance of UNISA in variations of λ_i. The upper part of the table (U1-U8) shows the impact of λ_1, λ_2, and λ_3 that are applied since the base task, while the lower part of the table (U9-U11) shows the impact of λ_4 and λ_5 that are applied in few-shot tasks only. The table shows that the lower value of λ_1 costs the highest performance drop, i.e., about 5%. While the lower value of λ_2 or λ_3 decreases the

average performance by about 1%. Please note that the zero value of λ_1 and λ_2 (the absence of $\mathcal{L}_{psa}$ and $\mathcal{L}_{cpr}$) drop the average accuracy by 12% and 2% respectively. It means that the lower coefficients still achieve a higher performance than zero. The table shows that the proposed method maintains its performance in a lower value of λ_4 or λ_5.

6.5.7 Robustness Analysis

This subsection discusses the robustness analysis of the proposed method against noise attacks on testing data and different hyperparameter settings.

(1)Against noises: Table 6.9 shows that the proposed method consistently maintains its performance against brightness, contrast, hue, saturation, blur, and Gaussian [221] noises. The Gaussian noise attack reduces its performance by about 1% on average, while the other noise attacks don't impact its performance. In addition, the table shows that the proposed method consistently achieves the highest performance with a significant gap ($\geq$ 14%) compared to the competitor methods under the same noise attack, i.e., Gaussian noise.

(2)In Various Hyperparameter settings: Table 6.10 shows the robustness of our proposed method under different hyperparameter settings, i.e., minimum learning rate and batch size. Note that UNISA utilizes a learning rate predictor by cosine annealing that takes the minimum learning rate as its lower bound. The table shows that the performance of our proposed method decreases only on the smallest batch-size settings, i.e., 4 and 8.

6.5.8 Sensitivity Analysis

This subsection offers our sensitivity analysis done with the UC Merced problem. Specifically, different flat minima bounds b are attempted, as well as the number of perturbation loops M. Table 6.11 and 6.12 report our numerical results. Different flat minima bounds do not result in significant differences in performance. Nevertheless, a too-large bound causes performance deterioration because of inaccurate approximations of flat-wide minima regions contributing to the catastrophic forgetting problem. An interesting observation is found in respect to the perturbation

loop M, where the least performance is attained with $M = 1$ because there does not exist any flat-wide minima estimations. UNISA's performances constantly increase with the increase of the perturbation loop M, i.e., the highest performance is attained with $M = 7$ since this step result in fine-grained approximations of flat-wide minima regions. Note that the extra perturbation loop M incurs additional computational burdens.

6.5.9 Complexity Analysis and Execution Times

Suppose that $\mathcal{T}_k$ is the training data for task $k \in [1, K]$, N_k is the size of $\mathcal{T}_k$, and $N_{k\mathcal{B}} = |\mathcal{B}|$ is the size of a batch sample of $\mathcal{T}_k$. Following the UNISA algorithm as presented in algorithm 5, UNISA has several atomic operations, i.e., generating pseudo label (line 4 and 24) that costs $O(N_k)$, noise sampling and perturbation (line 8-9) that costs $O(1)$, noise line 8) that costs $O(1)$, augmentation (line 11 and 28) that costs $O(1)$, computing cluster center (line 12 and 29) that costs $O(1)$, computing $\mathcal{L}_{psa}$ and $\mathcal{L}_{psl}$ (line 13 and 30) that costs $O(1)$, generating, transforming, and merging synthetic samples (line 34-36) that costs $O(1)$, computing $\mathcal{L}_{ball}$ and $\mathcal{L}_{KL}$ (line 38-39) that costs $O(N_{k\mathcal{B}})$, computing total loss (line 15 and 40) that costs $O(1)$, and finally updating, clamping and resetting network parameters (line 16-18 and 42-44) that costs $O(1)$. Knowing the cost of these atomic operations, the complexity of our proposed method can be expressed as:

$$O(UNISA) = O(BaseTask) + O(FewShotTask) \tag{6.10}$$

$$\begin{aligned} O(BaseTask) =& E.(O(N_1) + B.M.(N_{1\mathcal{B}}.O(1) \\ &+ N_{1\mathcal{B}}.O(1) + O(1) + O(1))) \end{aligned} \tag{6.11}$$

$$O(BaseTask) = E.(O(N_1) + B.M.(N_{1\mathcal{B}}.O(1))) \tag{6.12}$$

$$O(BaseTask) = E.(O(N_1) + O(B.M.N_{1\mathcal{B}})) \tag{6.13}$$

Please note that $B.N_{1\mathcal{B}} = N_1$. Therefore, the equation above can be derived into

$$O(BaseTask) = E.(O(N_1) + O(M.N_1)) \tag{6.14}$$

$$O(BaseTask) = E.O(M.N_1) = O(E.M.N_1) \tag{6.15}$$

Please note that the number of perturbation loops (M) is a small integer. Therefore, the complexity of the Base task can be derived into:

$$O(BaseTask) = O(E.M.N_1) = O(E.N_1) \tag{6.16}$$

Similarly, the complexity of the few shot tasks can be presented by the following equation:

$$\begin{aligned} O(FewShotTask) =&(K-1).E.(O(N_k) + B.(N_{k\mathcal{B}}.O(1) \\ &+ S.N_{k\mathcal{B}}.O(1) + O(1) + O(1))) \end{aligned} \tag{6.17}$$

$$O(FewShotTask) =(K-1).E.(O(N_k) + B.S.(N_{k\mathcal{B}}.O(1))) \tag{6.18}$$

$$O(FewShotTask) =(K-1).E.(O(N_k) + O(B.S.N_{k\mathcal{B}})) \tag{6.19}$$

Similar to the base task, please note that $B.N_{k\mathcal{B}} = N_k$. Therefore, the equation above can be derived into

$$O(FewShotTask) =(K-1).E.(O(N_k) + O(S.N_k)) \tag{6.20}$$

$$O(FewShotTask) =(K-1).E.O(S.N_k) \tag{6.21}$$

$$O(FewShotTask) = O(E.S.(K-1).N_k) \tag{6.22}$$

The number of augmented features (S) is a small integer, then the complexity of few-shot tasks can be derived as:

$$O(FewShotTask) = O(E.S.(K-1).N_k) = O(E.(K-1).N_k) \tag{6.23}$$

Then the complexity of UNISA can be expressed as

$$O(UNISA) = O(BaseTask) + O(FewShotTask) \tag{6.24}$$

$$O(UNISA) = O(E.N_1) + O(E.(K-1).N_k) \tag{6.25}$$

$$O(UNISA) = O(E.(N_1 + (K-1).N_k)) \tag{6.26}$$

Please note that $N_1 + (K-1).N_k = N$, where N is the size of the accumulated dataset across all tasks. Then the complexity of UNISA can be derived into:

$$O(UNISA) = O(E.N) \tag{6.27}$$

Following the standard training of deep learning, where the number of epochs E is a constant, the complexity of UNISA can be simplified as:

$$O(UNISA) = O(N) \tag{6.28}$$

where N is the size of the accumulated dataset across all tasks.

Table 6.13 reports the training time per epoch for all consolidated algorithms. UNISA imposes higher execution times than other consolidated algorithms in the base task. The reason for this is found in the perturbation mechanism of UNISA, where M perturbation loops are required to discover flat-wide minima regions. Note that other algorithms utilize 10% of base samples due to the absence of true class labels, thus expediting their running times. On the other side, the execution times of few-shot tasks of UNISA are comparable to other algorithms and always lower than SCN.

6.6 Limitations

UNISA has demonstrated encouraging performances compared to prior arts while addressing an unexplored problem, unsupervised few-shot continual learning (UFSCL). Nonetheless, there exists some room for further studies.

- Although the UFSCL problem is highly challenging and concerns the problem of label scarcities in the FSCL, all tasks are drawn from the same domain. This opens a future research issue to address the problem of cross-domain FSCL involving several domains.
- Although UNISA is memory-free and unlike [42], it relies on the data augmentation procedure to address a sequence of few-shot learning tasks. Such a mechanism simplifies the FSCL problem to some extent. Another research issue is how to devise an augmentation-free FSCL approach. Note that [40], [42], [56], [188] usually bias toward the base task, notwithstanding that they do not apply any data augmentation procedures. That is, they are only capable of classifying the base classes while performing poorly for few-shot classes.

6.7 Summary

This chapter proposes an extension of few-shot continual learning, namely unsupervised few-shot continual learning, where no labels are exploited for inducing the training process, including its solution, the unsupervised flat-wide learning approach (UNISA). UNISA is built upon the flat-wide learning approach to address the CF problem, while the concept of prototype scattering and positive sampling is implemented as a representation learning strategy. The efficacy of UNISA has been validated in the remote sensing image scene classification problems, where it produces the most encouraging numerical results over its prior arts.

The next chapter is devoted to addressing federated continual learning problems to handle catastrophic forgetting in a distributed system under data privacy constraints. The problems simulate **the presence of many agents/clients**, e.g., institutions that conduct continual learning in their respective environments to collaborate with each other without sharing their data, then aggregate the local models into an optimal global model. Each client carries only partial data and partial class labels for each task. The goal is to achieve an accurate global model that recognizes all learned classes for all collaborating clients.

Chapter 7

PIP: Prototypes-Injected Prompt for Federated Class Incremental Learning

Previous chapters address continual learning in a stand-alone system where the learning process is conducted by a single agent that carries all the observed data. From the perspective of a privacy-constrained distributed system, where the data is distributed among all clients, CL can not be executed in the normal way. Federated Class Incremental Learning (FCIL) is a new direction in continual learning (CL) for addressing catastrophic forgetting and non-IID data distribution simultaneously. Existing FCIL methods call for high communication costs and exemplars from previous classes that may breach data privacy. We propose a novel rehearsal-free method for FCIL named prototypes-injected prompt (PIP) that involves 3 main ideas: (1) prototype injection on prompt learning, (2) prototype augmentation, and (3) weighted Gaussian aggregation on the server side. The experiment result shows that the proposed method outperforms the current state of the art (SOTAs) with a significant improvement in CIFAR100, MiniImageNet, and TinyImageNet datasets. The extensive analysis demonstrates the robustness of PIP in different task sizes and the advantage of requiring smaller participating local clients and smaller global rounds.

Part of this chapter is published as: MA Ma'sum, M Pratama, S Ramasamy,

L Liu, H Habibullah, R Kowalczyk. PIP: Prototypes-Injected Prompt for Federated Class Incremental Learning. Proceedings of the 33rd ACM International Conference on Information and Management (CIKM). 2024: 1670–1679. https://doi.org/10.1145/3627673.3679794 [**CORE A**]

7.1 Introduction

Federated learning (FL) offers a collaborative approach for many clients to produce a shared global model while protecting their data privacy [145][222] [223]. FL has recently sparked a great deal of academic interest and achieved outstanding success in various application areas, including medical diagnosis [224], autonomous vehicle [225], and wearable technology [226]. However, the majority of FL methods are designed for a static application scenario, assuming the target classes are fixed and known in advance. In real-world applications, the data are dynamic, allowing local clients to access unseen target classes online.

Existing studies have addressed dynamic data challenges in FL through Federated Class Incremental Learning (FCIL), where each local client continually gathers training data according to their continuous observation of the environment. In contrast, new clients with unforeseen classes are always welcome to join the FL training [59] [47] [46]. The clients have to cooperatively train a global model that continually learns new classes while maintaining its capability to recognize the previous classes. Despite its capability to learn new classes, the global model tends to lose its knowledge of previous classes, known as catastrophic forgetting. Therefore, FCIL calls for answers on how to handle catastrophic forgetting in a dynamic, collaborative, continual learning environment while preserving data privacy. In addition, the clients may carry non-independently and identically distributed (non-i.i.d.) data with imbalanced classes.

The current FCIL state-of-the-art (SOTA), i.e., GLFC[47] and LGA[59], train and share whole backbone parameters, resulting in a large number of parameters (11.3M) to train that requires a longer training time, and a high communication cost (61.7MB) as shown in Table 7.1. In addition, in the case of client-server limited communication costs, e.g., due to limited bandwidth, this approach is forced

to utilize a smaller (more shallow) model to reduce the shared parameter size. Second, in the current SOTAs, the clients privately share perturbed images with the central server for the aggregation process. Besides its inefficient size (16.5MB), sharing perturbed images may violate data privacy principles since perturbation does not guarantee information leakage. Furthermore, this mechanism can not answer the data openness problem, where the data is only open for a client at a specific moment. These intriguing gaps motivate us to develop a more efficient yet more effective approach to the FCIL problem, where a client trains and shares as few parameters as possible but produces a highly accurate global model without sharing any (perturbed) samples.

Table 7.1: The difference between our proposed approach and current SOTA in Federated Class Incremental Learning (FCIL), * indicates sharable parameters, while ^{x} indicates unsharable parameters.

Method	B.Bone	Trainable Params			Prototype		Com.
		Type	#N	Size(B)	Type	Size(B)	Cost(B)
GLFC	ResNet18*	ResNet18*	11.3M	45.2 M	Perturbed Images*	16.5 M	61.7 M
LGA	ResNet18*	ResNet18*	11.3M	45.2 M	Perturbed Images*	16.5 M	61.7 M
Baseline (Ours)	ViT^x	Prompt*	0.33M	1.32 M	-	-	1.32 M
PIP (Ours)	ViT^x	Prompt*	0.33M	1.32 M	Deep Features*	0.03 M	1.35 M

This chapter proposes a new approach named federated prompt for the FCIL problem that is implemented in our proposed baselines and our proposed method, named prototype-injected prompt (PIP). In our proposed approach, a client trains and shares only a small (0.33M) set of parameters called prompts, while the backbone model remains frozen and unsharable. In our approach, the bound for communication cost is only in the prompt size, not in the backbone size. As an advantage, we could utilize any backbone with any complexity, number of parameters, and size. Our approach utilizes deep features as a shareable prototype to handle the non-i.i.d. data between clients. As the second advantage, our approach offers far smaller prototypes (0.03MB) and guarantees data privacy as deep features represent a totally different structure and value compared to raw or perturbed samples. The key points of difference between our approach and the current SOTA

are described in Table 7.1. **The contributions** of this chapter are summarized as follows:

1. This chapter proposes a new approach, prompt-based federated continual learning to solve the FCIL problem, where clients and central servers exchange prompts and head layer units instead of a whole model. This approach significantly reduces the size of the exchanged parameters between clients and the central server in each round.
2. This chapter proposes a new baseline method for the FCIL problem named Federated DualPrompt (Fed-DualP) that already outperforms the current state-of-the-art (SOTA) methods in three benchmark datasets.
3. This chapter proposes a novel method for the FCIL problem named Prototype-Injected Prompt (PIP) that consists of three main ideas, i.e., a) prototype injection on prompt learning, b) prototype augmentation, and c) weighted Gaussian aggregation on the server side. The proposed method outperforms the baseline and current SOTAs with a significant gap.
4. This chapter provides a comprehensive analysis of three benchmark datasets as well as the robustness of the proposed method in different task sizes, smaller participating clients, and smaller rounds per task.

7.2 Preliminaries

7.2.1 Problem Formulation

Class incremental learning (CIL) problem is defined as a learning problem of a sequence of fully supervised learning tasks $\{\mathcal{T}^t\}_{t=1}^{T}$ where T represents the number of consecutive tasks, the value of T is unknown to the model before CIL process, and can be infinite. Each task carries N_t pairs of training samples $\mathcal{T}^t = \{x_i^t, y_i^t\}_{i=1}^{N_t}$ where $x_i \in \mathcal{X}^t$ denotes input image and $y_i \in \mathcal{Y}^t$ denotes its corresponding class label. Each task carries the same image size but possesses disjoint target classes from the other tasks. Suppose that $\mathcal{Y}^t$ is the labels set of task t and $\mathcal{C}^t$ denotes the unique class labels in $\mathcal{Y}^t$, $\mathcal{C}^t = \{y_i^t, y_j^t \in \mathcal{Y}^t, i, j \in [1..N^t]\}_{\neq}$, then we have $\forall t, t' \in [1..T], t \neq t' \Rightarrow (\mathcal{Y}^t \cap \mathcal{Y}^{t'} = \emptyset) \wedge (\mathcal{C}^t \cap \mathcal{C}^{t'} = \emptyset)$.

Federated Class-Incremental Learning (FCIL) is defined as: for $r-th$ global round $r \in [1..R]$, a set of selected local clients $\{S_l\}_{l=1}^{L}$ are conducting class incremental learning for the task $t \in [1..T]$ with their local training set $\{\mathcal{T}_l{}^t\}_{l=1}^{L}$ coordinated by a central server S_G in a federated manner, where R is the maximum global round. A client S_l may carry a unique set of a training set $\mathcal{T}_l^t = \{x_{li}^t, y_{li}^t\}_{i=1}^{N_l^t}$, which satisfies $\forall l, l' \in [1..L], l \neq l' \Rightarrow (\mathcal{T}_l^t \neq \mathcal{T}_{l'}^t) \wedge (\mathcal{C}_l^t \neq \mathcal{C}_{l'}^t)$ where, N_l^t denotes number of training samples of client l for task t. FCIL satisfies $\mathcal{T}^t = \cup_{l=1}^{L} \mathcal{T}_l^t$ and $\mathcal{C}^t = \cup_{l=1}^{L} \mathcal{C}_l^t$, therefore it satisfies $\mathcal{C}_l^t \subseteq \mathcal{C}^t$ and $|\mathcal{C}_l^t| \leq |\mathcal{C}^t|$. Please note that $\mathcal{T}_l^t$ is not shareable to another client $S_{l'}(l \neq l')$ nor central server S_G. Clients may carry non-identically and distributed data (non-i.i.d), suppose that $\mathcal{D}_l$ is the class distribution of $l-th$ client, then it satisfies $\forall l, l' \in [1..L], l \neq l' \Rightarrow (\mathcal{D}_l^t \neq \mathcal{D}_{l'}^t)$. In each global round $r-th$ for task t, each client S_l conducts local CIL training using its training samples $\mathcal{T}_l^t$, optimizing its local model $\Theta_l^{r,t}$, then the clients send their local models to the central server S_G. The central server aggregates all participating local updated parameters $\{\Theta_l^{r,t}\}_{l=1}^{L}$, producing the latest global model parameter $\Theta_G^{r,t}$, then distributes it to all clients $\{S_l\}_{l=1}^{L}$ for the next global round process. $\Theta_l^{r,t}$ and $\Theta_g^{r,t}$ denote the latest updated version of local parameter Θ_l and global parameter Θ_g respectively after $r-th$ round of $t-th$ task of FCIL training. Please note that a set of local clients $\{S_l\}_{l=1}^{L}$ is randomly selected from all possibly participating clients, simulating real-world conditions where only a small percentage of registered clients join in a round of federated learning.

7.2.2 Proposed Approach and Baselines: Federated Prompt-Based Continual Learning for FCIL

In prompt-based class incremental learning, a client uses a frozen backbone model, e.g., Vision Transformer (ViT), as a feature extractor. The ViT embedding layer transforms an input image $x \in R^{w \times h \times c}$, where w, h, c are the image's width, height, and channel respectively, into sequence-like output feature $h \in R^{L \times D}$, where L is the sequence length and D is the embedding dimension. A client only needs to update a small-sized trainable parameter called prompt $p \in R^{L_p \times D}$ and its head layer parameter ϕ to deal with the sequence of tasks $\{\mathcal{T}_t\}_{t=1}^{T}$, L_p denotes the prompt length. Inspired by its effectiveness and efficiency in CIL problem,

We proposed a federated prompt-based continual learning for the FCIL problem, where in a global round $r-th$ for the $t-th$ task a client S_l only updates its learnable parameters, i.e., prompt $p_l^{r,t} \in R^{L_p \times D}$, and its classification (head) layer $\phi_l^{r,t}$, then send them to central server S_G. The central S_G aggregates accumulated parameters $\{(p_l^{r,t}, \phi_l^{r,t})\}_{l=1}^{L}$ producing global model parameters $(p_G^{r,t}, \phi_G^{r,t})$ and send it back to all clients. The size of $(p_l^{r,t}, \phi_l^{r,t})$ is far smaller than the whole model $\Theta_l = (\theta_l, \phi_l)$. Therefore, our proposed approach is more efficient both in terms of the local training process and communication costs. The main distinction between our proposed approach and the existing state-of-the-art (SOTA) methods is that the SOTA methods train the whole model parameters $\Theta = (\theta, \phi)$ where a model is defined as $F_\Theta(x) = f_\phi(g_\theta(x))$, $g_\theta(.)$ is feature extractor and $f_\phi(.)$ is classifier, while our approach trains only small-sized parameters, i.e., prompt p and head layer ϕ, and freezes the backbone θ.

In this study, we propose two baseline methods, i.e., Federated Learning to Prompt (Fed-L2P), and Federated DualPrompt (Fed-DualP) by customizing L2P [37] and DualP [38], for federated class incremental learning problems. The main role of the baselines is to demonstrate that our new approach is more effective and efficient than current SOTAs for FCIL problem. Second, the baselines show that our approach applies to any prompt structure, as demonstrated by two different prompt structures, i.e., L2P and DualP. In addition, we proposed those two methods as new baselines for further study on FCIL problem. Following L2P and DualP structure, in our baselines, the local client's prompt p_l represented as $p_l = \{(e_l^t, k_l^t)\}_{t=1}^{T}$ in Fed-L2P $p_l = (g_l, \{(e_l^t, k_l^t)\}_{t=1}^{T})$ in Fed-DualP, where $p_l^t \in R^{L_p \times D}$ and $e_l^t \in R^{L_p \times D}$ represent a task-specific prompt, $g_l \in R^{L_p \times D}$ represents global prompt, k_l^t is the key associated with a task-specific prompt p_l^t or e_l^t, and T denotes the number of tasks of client S_l. The central server aggregates participating clients' parameters by performing FedAvg [145].

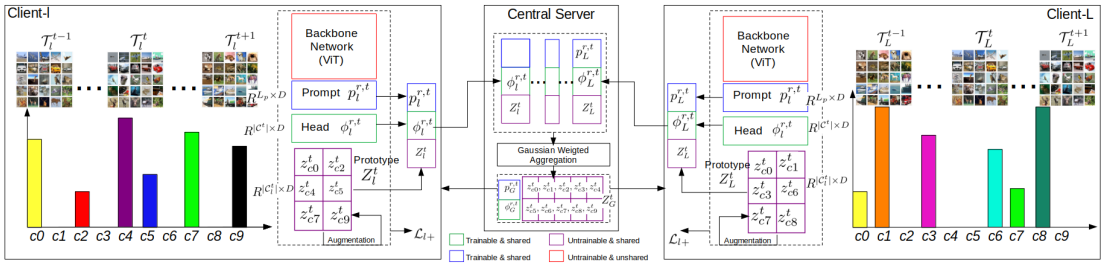


Figure 7.1: Visualization of our proposed method (PIP) that includes prototypes-injected-prompt learning with $\mathcal{L}_{l+}$ to handle local catastrophic forgetting, shared prototypes to handle non-i.i.d distributions between clients, prototypes augmentation to handle class-imbalance, and server Gaussian weighted aggregation to improve global model generalizations.

7.3 Proposed Method: Prototype-Injected Prompt (PIP)

Our proposed method, as visualized in Figure 7.1 performs prompt learning on each client to handle local catastrophic forgetting (section 7.3.1), handling non-i.i.d distribution by applying shared prototype injection(section 7.3.2), handling class-imbalance by performing prototypes augmentation (section 7.3.3), and finally improving global model generalization by using weighted aggregation (section 7.3.4). Figure 1 shows, in the proposed method, the clients and server exchange prompts, prototypes, and the head layer instead of the whole model.

The uniqueness of our method is that we utilize a prompt and a prototype as the shared knowledge between clients and the central server. Second, we propose a prototype injection mechanism in local training to improve local model generalization. Third, we propose a weighted aggregation on the server side. Our proposed method is distinct from prompt-based methods, e.g., Fed-CPrompt[152] and FCILPT[153], which share and train prompts only. Our proposed method is also distinguished from prototype-based FL methods, e.g., FedProto [227] and CCVR [228], which shared only prototypes to refine the local model or use the prototype as classifiers. Besides, FedProto and CCVR are developed for federated learning only, meaning the methods didn't address the catastrophic forgetting issue.

7.3.1 Handling Local Catastrophic Forgetting via Prompt Learning

On a round $r-th$ of $t-th$ task, each client S_l optimizes its local prompt p_l and head layer parameter ϕ_l using its available training samples $\mathcal{T}_l^{r,t}$. Please note that p_l can be any prompt structure as mentioned in our baselines, e.g., e.g.in L2P, DualPrompt, or another prompt structure. Given a pre-trained ViT backbone f with M Multi head Self-Attention (MSA), where $h^{(i)}, i = 1, 2..., M$ represents the input for $i-th$ MSA layer, suppose that a local client wants to attach a latest updated prompt $p_l^{r,t}$ into the $i-th$ MSA layer, the prompt instance $p_l^{r,t}$ transform feature $h^{(i)}$ via prompt function as defined in equation 7.1.

$$h^{(i)}_{p_l^{r,t}} = f_{prompt}(p_l^{r,t}, h^{(i)}) \tag{7.1}$$

Note that h_i is the extension of h, a sequence-like parameter produced by the ViT embedding layer for the $i-th$ MSA layer. Prompt function f_{prompt} can be implemented by using prompt tuning (f^{Pro-T}_{prompt}) [82] or prefix (f^{Pre-T}_{prompt}) tuning [83] approaches as defined in equation 7.2 and 7.3.

$$f^{Pro-T}_{prompt}(p_l^{r,t}, h^{(i)}) = MSA([p_l^{r,t} \oplus h_Q^{(i)}], [p_l^{r,t} \oplus h_K^{(i)}], [p_l^{r,t} \oplus h_V^{(i)}]) \tag{7.2}$$

$$f^{Pre-T}_{prompt}(p_l^{r,t}, h^{(i)}) = MSA(h_Q^{(i)}, [p_{lK}^{r,t} \oplus h_K^{(i)}], [p_{lV}^{r,t} \oplus h_V^{(i)}]) \tag{7.3}$$

$$\begin{aligned} &MSA(h_Q^{(i)}, h_K^{(i)}, h_V^{(i)}) = (h_1^{(i)} \oplus h_2^{(i)} \oplus ... h_{m-1}^{(i)} \oplus h_m^{(i)}) W^O \\ &where\ h_j^{(i)} = Attention(h_Q^{(i)} W_j^Q, h_K^{(i)} W_j^K, h_V^{(i)} W_j^V) \end{aligned} \tag{7.4}$$

MSA function is defined as in equation 7.4 following [229] where $h_Q^{(i)}$, $h_K^{(i)}$, and $h_V^{(i)}$ are input query, key, and value, for $i-th$ MSA layer, $p_{lK}^{r,t} \in R^{L_{p/2} \times D}$ and $p_{lV}^{r,t} \in R^{L_{p/2} \times D}$ are splits of $p_l^{r,t}$, W^O, W_j^Q, W_j^K, and W_j^V are the projection matrices, and m is number of head, $\oplus$ represents concatenation function, and in ViT $h_Q^{(i)} = h_K^{(i)} = h_V^{(i)} = h^{(i)} \in R^{L \times D}$. Given a pair of samples $(x_{li}^t, y_{li}^t) \in \mathcal{T}_l^t$, each client S_l trains its latest updated parameters $p_l^{r,t}$ and $\phi_l^{r,t}$ using $\mathcal{L}_l$ as defined in equation 7.5, where $\mathcal{L}$ represents cross-entropy loss, $\mathcal{L}_m$ represents matching loss between sample x_{li}^t and key x_{li}^t, λ represents a constant factor, f_{p_l} represents prompt function, and f_{ϕ_l} represents of $l-th$ client.

$$\mathcal{L}_l = \mathcal{L}(f_{\phi_l^{r,t}}(f_{p_l^{r,t}}(x_{li}^t), y_{li}^t)) + \lambda \mathcal{L}_m(x_l^t, k_l^{r,t}), (x_{li}^t, y_{li}^t) \in \mathcal{T}_l^t \tag{7.5}$$

$$
\begin{aligned}
p_l^{r,t} &\leftarrow p_l^{r,t} - \alpha \nabla \mathcal{L}_l \\
\phi_l^{r,t} &\leftarrow \phi_l^{r,t} - \alpha \nabla \mathcal{L}_l
\end{aligned}
\tag{7.6}
$$

The client's parameters are updated using equation 7.6 where α represents the learning rate and $\nabla \mathcal{L}_l$ represents the gradient with respect to $\mathcal{L}_l$.

7.3.2 Shared Prototype Injection: What, Why and How?

We define a prototype set on the $t-th$ task of a client S_l as $Z_l^t = \{z_{lc}^t\}, z_{lc}^t \in R^{1\times D}$ is the prototype for class $c \in \mathcal{C}_l^t$, $\mathcal{C}_l^t$ is the available classes in $\mathcal{T}_l^t$, and D is the embedding dimension. Assuming that the prototype follows a Gaussian distribution $z_{lc}^t \sim \mathcal{N}(\mu_{lc}^t, \Sigma_{lc}^t)$ and the prototype is considered as D disjoint univariate distribution, then we have $z_{lc}^t \sim \mathcal{N}(\mu_{lc}^t, {\sigma_{lc}^t}^2)$ where ${\sigma_{lc}^t}^2 = I_D.{\sigma_{lc,i}^t}^2, i \in \{1, 2..D\}$, I_D is identity matrix. Suppose that $\mathcal{T}_{lc}^t = \{(x_{li}^t, y_{li}^t) \in \mathcal{T}_l^t, y_{li}^t = c\}$ is the samples of class-c in $\mathcal{T}_l^t$ and $|\mathcal{T}_{lc}^t|$ is the number of samples in $\mathcal{T}_{lc}^t$, we compute the prototype z_{lc}^t properties by Eq. 7.7 and 7.8.

$$
\mu_{lc}^t = \frac{1}{|\mathcal{T}_{lc}^t|} \sum_{li=1}^{|\mathcal{T}_{lc}^t|} f_{p_l^{r,t}}(x_{lci}^t)\,, x_{lci} \in \mathcal{T}_{lc}^t \tag{7.7}
$$

$$
{\sigma_{lc}^t}^2 = \frac{1}{|\mathcal{T}_{lc}^t|} \sum_{li=1}^{|\mathcal{T}_{lc}^t|} (\mu_{lc}^t - f_{p_l^{r,t}}(x_{lci}))^2\,, x_{lci} \in \mathcal{T}_{lc}^t \tag{7.8}
$$

Due to non-i.i.d data between clients, FCIL satisfies $\mathcal{T}_l^t \subseteq (\mathcal{T}^t = \cup_{l=1}^L \mathcal{T}_l^t)$ and $\forall l, l' \in [1..L], l \neq l' \Rightarrow (\mathcal{T}_l^t \neq \mathcal{T}_{l'}^t) \wedge (\mathcal{C}_l^t \neq \mathcal{C}_{l'}^t)$. It implies a client only optimize p_l and ϕ_l with regard to $\mathcal{T}_l^t$ but not yet to $\mathcal{T}^t$. The client S_l only learns C_l^t but not $\overline{\mathcal{C}_l^t} = \mathcal{C}^t - \mathcal{C}_l^t$, classes that exist in $\mathcal{T}^t$ but not in $\mathcal{T}_l^t$. Meanwhile, we can not afford to share any exemplar sample (x_{li}^t, y_{li}^t) due to data privacy. Aggregating the prompts and head does not guarantee optimal global model $(p_G^{r,t}, \phi_G^{r,t})$ for $\mathcal{T}^t$ since in the next round, the participating clients and their available data may be different from the current rounds' $\forall r, r' \in [1..R], \forall l \in [1..L], r \neq r' \Rightarrow (S_l^{r,t} \neq S_l^{r',t}) \wedge (\mathcal{T}_l^{r,t} \neq \mathcal{T}_l^{r',t})$. Furthermore, training in too many rounds leads to catastrophic forgetting.

To handle this challenge, we propose prototype sharing between clients and the central server, where the server collects all clients' prototype $\{Z_l^t\}_{l=1}^L$ then distributes global prototype $Z_G^t = Z^t = \cup_{l=1}^L Z_l^t$. The shared prototype set Z_l^t ensures each client learns $\overline{\mathcal{C}_l^t}$ via $\overline{Z_l^t} = Z^t - Z_l^t$, therefore each client has chances to learn

all classes $\mathcal{C}^t$ in $\mathcal{T}^t$. Besides, the prototype set Z^t has a small size in comparison to the prompt p_l or the head layer ϕ_l, where $z_c^t \in Z^t$ has the size of $1 \times D$, where D is the embedding dimension, and it does not contain any raw information as contained in an exemplar sample.

Now each client can enhance its local training by **injecting** the shareable prototype set Z^t into its training process by considering $\mathcal{T}_l^t \cup (Z^t, \mathcal{C}^t) \approx \mathcal{T}^t$, where Z^t is the prototype feature set, and $\mathcal{C}^t$ acts as the label for the prototypes of all classes $c \in \mathcal{C}^t$. Therefore, the loss function and parameters update of each client can be defined as in Eq. 7.9 and 7.10. In case the prototype is not yet available, e.g., first round ($r = 1$), a local model is updated using Eq. 7.5 and 7.6.

$$\begin{aligned} \mathcal{L}_{l+} =& \mathcal{L}(f_{\phi_l^{r,t}}(f_{p_l^{r,t}}(x_{li}^t) \cup Z^t), y_{li}^t \cup \mathcal{C}^t)) \\ &+ \lambda \mathcal{L}_m(x_l^t, k_l^{r,t}), (x_{li}^t, y_{li}^t) \in \mathcal{T}_l^t \end{aligned} \tag{7.9}$$

$$\begin{aligned} p_l^{r,t} &\leftarrow p_l^{r,t} - \alpha \nabla \mathcal{L}_{l+} \\ \phi_l^{r,t} &\leftarrow \phi_l^{r,t} - \alpha \nabla \mathcal{L}_{l+} \end{aligned} \tag{7.10}$$

7.3.3 Handling Class Imbalance via Prototype Augmentation

We can simply assign $z_c^t \in Z^t$ by μ_c^t to generate a single prototype for class-c. However, to handle class imbalance between locally available classes and unavailable classes that are represented by the shared prototypes. We enrich the prototypes by using an augmentation. Suppose that x_{c_1} is the sample for class $c_1 \in \mathcal{C}_l^t$, $z_{c_2}^t \in Z^t$ is the prototype of class $c_2 \in \overline{\mathcal{C}_l^t} = \mathcal{C}^t - \mathcal{C}_l^t$, c_1 is locally available class, and c_2 is locally the unavailable class in client-l, then we have $|x_{c_1}| \gg 1$, while $|z_{c_2}^t| = 1$. The prototype augmentations create artificial prototypes that satisfy $|z_{c_2}^t| \approx |x_{c_1}|$ as defined in equation 7.11 to generate m augmented prototypes for class-c based on Gaussian distribution $z_c^t \sim \mathcal{N}(\mu_c^t, \sigma_c^{t2})$, and β is a random value in $(0, 1)$ range.

$$\begin{aligned} z_c^t &= \{\mu_c^t\} \cup \{u_{ci}^t\}_{i=1}^m, \; where \\ u_{ci}^t &= \mu_c^t + \beta \sigma_c^t, m \geq 1 \end{aligned} \tag{7.11}$$

7.3.4 Server Weighted Aggregation

We proposed weighted Gaussian aggregation on the server side to improve global model generalization. The aggregation treats clients' contributions to the global

aggregation proportionally based on their participation and their training sample size, following best practice, where a model that learns more produces a more convergent weight. We consider clients to learn their local data and then produce Gaussian-distributed local parameters. We define $\omega_l = \rho_l^t|\mathcal{T}_l^t|$ as the weight of a client-l on the t-th task, where ρ_t is the total of client-l participation until the $t-th$ task and $|\mathcal{T}_l^t|$ is the number of samples in $\mathcal{T}_l^t$. Given $\{(p_l^{r,t}, \phi_l^{r,t}, Z_l^t)\}_{l=1}^L$ is a set of locally optimized parameters by selected local clients $\{S_l\}_{l=1}^L$ on the $r-th$ of $t-th$ task, then the global parameters ($p_G^{r,t}$, $\phi_G^{r,t}$ and Z_G^t are computed by Gaussian-based weighted aggregation as We define in Eq. 7.12-7.15. The pseudo-code of PIP is presented in algorithm 6.

$$p_G^{r,t} = \frac{1}{\sum_{l=1}^L \omega_l} \sum_{l=1}^{L} (p_l^{r,t} \omega_l) \tag{7.12}$$

$$\phi_G^{r,t} = \frac{1}{\sum_{l=1}^L \omega_l} \sum_{l=1}^{L} (\phi_l^{r,t} \omega_l) \tag{7.13}$$

$$\begin{aligned} z_{Gc}^t &= \mathcal{N}(\mu_{Gc}^t, {\sigma_{Gc}^t}^2), \; where \\ \mu_{Gc}^t &= \frac{1}{\sum_{l=1}^L a_{lc}^t \omega_l} \sum_{l=1}^{L} a_{lc}^t \mu_{lc}^t \omega_l \end{aligned} \tag{7.14}$$

$$\begin{aligned} {\sigma_{Gc}^t}^2 &= \frac{1}{\sum_{l=1}^L a_{lc}^t \omega_l} \sum_{l=1}^{L} a_{lc}^t ({\mu_{lc}^t}^2 + {\sigma_{lc}^t}^2) \omega_l - {\mu_{Gc}^t}^2 \\ where \; a_{lc}^t &= 1, \; if (\mu_{lc}^t \; exists) \; else \; = 0 \end{aligned} \tag{7.15}$$

7.4 Theoretical Analysis

7.4.1 Derivation of Weighted Aggregation

Suppose that we have n samples of an observation x_i with the weight of w_i. Then, we measure mean and variance as:

$$\mu = \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^{n} (x_i . w_i) \tag{7.16}$$

$$\begin{aligned} \sigma^2 &= \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^{n} w_i (x_i - \mu)^2 \\ &\approx \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^{n} (w_i {x_i}^2 - w_i \mu^2) \end{aligned} \tag{7.17}$$

Algorithm 6 PIP

Input: Number of clients N, number of selected local clients L, total number of rounds R, number of task T, local epochs E, batch size B.
Distribute frozen ViT backbone f to all clients $\{S_l\}_{l=1}^{N}$ and central server S_G
Initiate prompt, key, and head layer for all clients and central server $p_G = p_l$, $\phi_G = \phi_l$, $l \in \{1..N\}$
$R_T \leftarrow R/T$, R_T represents round per task
for $t = 1 : T$ **do**
    Init global and local prototypes $Z_G^t = Z_l^t, = Z^t = \emptyset$
    for $r = 1 : R_T$ **do**
        $S_l \leftarrow$ randomly select L local clients from N total clients
        Clients execute:
        Receive global parameters i.e (p_G, Z_G^t, ϕ_G)
        Assign local parameters $(p_l, \phi_l, Z_l^t) \leftarrow (p_G, \phi_G, Z_G^t)$
        $\mathcal{B} \leftarrow$ Split $\mathcal{T}_l^t$ into B sized batches
        for $e = 1 : E$ **do**
            for $b = 1 : \mathcal{B}$ **do**
                Compute feature $f_{p_l}(x)$ as in Eq. (7.1) to (7.4)
                Compute logits with prototypes $f_{\phi_l}(f_{p_l}(x) \cup Z^t)$
                Compute loss $\mathcal{L}_{t+}$ as in Eq. (7.9)
                Update local parameters (p_l, ϕ_l) w.r.t $\mathcal{L}_{t+}$ as in Eq. (7.10)
            end for
            if $Z_l^t = \emptyset$ **then**
                Compute prototype $z_{lc}^t \sim \mathcal{N}(\mu_{lc}^t, {\sigma_{lc}^t}^2)$ as in Eq. (7.7)-(7.8)
            end if
            Augment and unify the prototypes as in Eq. (7.11)
        end for
        Compute prototype $z_{lc}^t \sim \mathcal{N}(\mu_{lc}^t, {\sigma_{lc}^t}^2)$ as in Eq. (7.7)-(7.8)
        Set $Z_l^t = \{z_{lc}^t\} = \{(\mu_{lc}^t, {\sigma_{lc}^t}^2)\}$, for class-$c$ available in $\mathcal{T}_l^t$
        Store local parameters (p_l, ϕ_l, Z_l^t)
        Compute clients' weight ω_l^t
        Send local parameters (p_l, ϕ_l, Z_l^t) and weight ω_l^t to server
        Server executes:
        Receives selected local clients parameters $\{(p_l, \phi_l, Z_l^t)\}_{l=1}^{l=L}$
        Aggregates parameters $(p_G, \phi_G) \leftarrow (p_l, \phi_l)$ as in Eq. (7.12)-(7.13)
        Aggregates prototypes $Z_G^t \leftarrow Z_l^t$ as in Eq. (7.14)-(7.15)
        Send global parameters (p_G, ϕ_G, Z_G^t) to all clients
    end for
end for
Output: Optimal Global parameters (p_G, ϕ_G) and local parameters $(p_l, \phi_l), l \in 1..N$

Or we have

$$\sum_{i=1}^{n} w_i\sigma^2 = \sum_{i=1}^{n}(w_i {x_i}^2 - w_i\mu^2) = \sum_{i=1}^{n} w_i {x_i}^2 - \sum_{i=1}^{n} w_i\mu^2 \tag{7.18}$$

that equal

$$\sum_{i=1}^{n} w_i\sigma^2 + \sum_{i=1}^{n} w_i\mu^2 = \sum_{i=1}^{n} w_i {x_i}^2 \tag{7.19}$$

If we have another m observation, then we have

$$\sum_{i=1}^{n+m} w_i\sigma^2 + \sum_{i=1}^{n+m} w_i\mu^2 = \sum_{i=1}^{n+m} w_i {x_i}^2 \tag{7.20}$$

$$\sum_{i=1}^{n+m} w_i\sigma^2 + \sum_{i=1}^{n+m} w_i\mu^2 = \sum_{i=1}^{n} w_i {x_i}^2 + \sum_{i=n+1}^{n+m} w_i {x_i}^2 \tag{7.21}$$

$$\begin{aligned}\sum_{i=1}^{n+m} w_i\sigma^2 + \sum_{i=1}^{n+m} w_i\mu^2 = (\sum_{i=1}^{n} w_i {\sigma_{1:n}}^2 + \sum_{i=1}^{n} w_i {\mu_{1:n}}^2) + \\ (\sum_{i=n+1}^{n+m} w_i {\sigma_{n+1:m}}^2 + \sum_{i=n+1}^{n+m} w_i {\mu_{n+1:m}}^2)\end{aligned} \tag{7.22}$$

$$\begin{aligned}\sum_{i=1}^{n+m} w_i\sigma^2 + \sum_{i=1}^{n+m} w_i\mu^2 = \sum_{i=1}^{n} w_i(\sigma^2_{1:n} + {\mu_{1:n}}^2) + \\ \sum_{i=n+1}^{n+m} w_i(\sigma^2_{n+1:m} + {\mu_{n+1:m}}^2)\end{aligned} \tag{7.23}$$

Therefore:

$$\begin{aligned}\sigma^2 = &\frac{\sum_{i=1}^{n} w_i(\sigma^2_{1:n} + {\mu_{1:n}}^2)}{\sum_{i=1}^{n+m} w_i} + \\ &\frac{\sum_{i=n+1}^{n+m} w_i(\sigma^2_{n+1:m} + {\mu_{n+1:m}}^2) - \sum_{i=1}^{n+m} w_i\mu^2}{\sum_{i=1}^{n+m} w_i}\end{aligned} \tag{7.24}$$

$$\begin{aligned}\sigma^2 = &\frac{\sum_{i=1}^{n} w_i(\sigma^2_{1:n} + {\mu_{1:n}}^2)}{\sum_{i=1}^{n+m} w_i} + \\ &\frac{\sum_{i=n+1}^{n+m} w_i(\sigma^2_{n+1:m} + {\mu_{n+1:m}}^2)}{\sum_{i=1}^{n+m} w_i} - \mu^2\end{aligned} \tag{7.25}$$

The derivation above shows that if we have two weighted Gaussian distributions, e.g., $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$ with total weight $W_1 = \sum_{i=1}^{|X_1|} w_i$ and $W_2 = \sum_{j=1}^{|X_2|} w_j$ respectively, then the aggregated distribution will be:

$$\mu* = \frac{(\mu_1.W_1 + \mu_2.W_2)}{W_1 + W_2} \tag{7.26}$$

$$\sigma*^2 = \frac{((\mu_1{}^2 + \sigma_1{}^2).W_1 + (\mu_2{}^2 + \sigma_2{}^2).W_2)}{W_1 + W_2} - \mu*^2 \tag{7.27}$$

Generalizing equations above into N observations, i.e., $X_1 \sim \mathcal{N}(\mu_1, \sigma_1{}^2)$, $X_2 \sim \mathcal{N}(\mu_2, \sigma_2{}^2)$, ... $X_N \sim \mathcal{N}(\mu_N, \sigma_N{}^2)$ with total weight W_1, W_2, ... W_N respectively. Then the aggregated distribution will be:

$$\mu* = \frac{\sum_{i=1}^{N}(\mu_i.W_i)}{\sum_{i=1}^{N} W_i} \tag{7.28}$$

$$\sigma*^2 = \frac{\sum_{i=1}^{N}(\mu_i{}^2 + \sigma_i{}^2).W_i}{\sum_{i=1}^{N} W_i} - \mu*^2 \tag{7.29}$$

7.4.2 Theorems

Let $\theta = (p, \phi)$ is trainable parameters and $F(\theta) = \mathbb{E}[\mathcal{L}_{l+}(\mathcal{T}; \theta)] = \mathbb{E}[\mathcal{L}_{l+}(\mathcal{T}; (p, \phi))]$ is the expected loss function, k, E, R, and L_S is local iteration, local epoch, global round, and number of selected local clients respectively. We follow L-smooth and μ-strongly convex F, random uniformly distributed batches, G-bounded uniformly gradient assumptions, and decreasing learning rate as in [230],[231] as detailed below:

Assumption 1: $F_1, ...F_l, ..., F_{L_S}$ are all $L-$smooth: for all θ and θ', $F_l(\theta) \leq F_l(\theta') + (\theta - \theta')^T \nabla F_l(\theta) + \frac{L}{2}||\theta - \theta'||_2^2$.

Assumption 2: $F_1, ...F_l, ..., F_{L_S}$ are all $\mu-$strongly convex: for all θ and θ', $F_l(\theta) \leq F_l(\theta') + (\theta - \theta')^T \nabla F_l(\theta) + \frac{\mu}{2}||\theta - \theta'||_2^2$.

Assumption 3: Let ξ_l^k be the random uniformly sampled from l-th local data at $k - th$ iteration . The variance of stochastic gradients in each client is bounded by: $\mathbb{E}||\nabla F_l(\theta_l^k, \xi_l^k) - \nabla F_l(\theta_l^k)|| \leq \sigma_l^2$ for $l = 1, 2, ..., L_S$

Assumption 4: The expected squared norm of stochastic gradients in each client is bounded by: $\mathbb{E}||\nabla F_l(\theta_l^k, \xi_l^k)|| \leq G^2$ for all $l = 1, 2, ..., L_S$ and $k = 1, 2, ..., K$ where $K \in \mathbb{N}$.

Assumption 5: $\sum_{k=1}^{\infty} \alpha_l^k = \infty$ and $\sum_{k=1}^{\infty} \alpha_l^{k^2} < \infty$ where α_l^k is the learning rate of $l - th$ client in k-th step training.

Assumption 6: The objective function F_l and SG satisfy the following conditions.

(a). The sequence of $\{\theta_l^k\}$ is contained in an open space where F_l is bounded below by a scalar F_{inf}

(b) Exist scalars $\nu_G \geq \nu > 0$ so that for all $k \in \mathbb{N}$ satisfy:

$$\begin{aligned} &\nabla F_l(\theta_l^k)^T \mathbb{E}_{\xi_l^k}[g(\theta_l^k, \xi_l^k)] \geq \nu ||\nabla F_l(\theta_l^k)T||_2^2, and \\ &||\mathbb{E}_{\xi_l^k}[g(\theta_l^k, \xi_l^k)]||_2 \leq \nu_G ||\nabla F_l(\theta_l^k)||_2. \end{aligned} \tag{7.30}$$

We state the following theorems:

Theorem 1: $\liminf_{k\to\infty} \mathbb{E}[||\nabla F(\theta_k)||_2^2] = 0$

Theorem 2: Let Chose $\kappa = L/\mu$, $\delta = \max(8\kappa, E)$, $\alpha_r = 2/(\mu(\delta + r))$, $C = 4/L_S E^2 G^2$, $\theta^1, \theta^R, \theta^*$ is the initial, last updated (R-th), and optimum parameter respectively, F^* is minimum value of F then:

$\mathbb{E}[F(\theta^R)] - F^* \leq \frac{\kappa}{(\delta+R-1)}(\frac{2(B+C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\theta^1 - \theta^*||)$.

Theorem 3: Given θ^* and θ are optimal parameter in $\mathcal{T}_l^t \cup Z^t$ and $\mathcal{T}_l^t$ respectively, where $\mathcal{T}_l^t \subset \mathcal{T}^t$, where $|\mathcal{T}_l^t|/|\mathcal{T}^t| = \eta \in (0, 1)$, then at least there's $\epsilon > 0$ that satisfy $F(\theta; \mathcal{T}^t) - F(\theta^*; \mathcal{T}^t) \geq \epsilon$. Theorems 1 and 2 prove PIP local training and federated convergence, respectively, while Theorem 3 proves PIP generalization.

Proof of Theorem 1: Suppose that a client S_l is trained locally with its local data $\mathcal{T}_l^t \cup Z^t$, where $\mathcal{T}_l^t$ is local training samples for t-th task and Z^t is shared prototype for t-th task respectively. We already stated that Z^t contains augmented prototype so that $|z_{c_2}^t| \approx |x_{c_1}|$ for $z_{c_2}^t \in Z^t$ and $x_{c_1}^t \in \mathcal{T}_{l_c}^t, T_{l_c}^t$ is subset of $\mathcal{T}_l^t$ where $y_i = c$ so the number of prototype of unavailable classes in $\mathcal{T}_l^k$ and the samples of available classes in $\mathcal{T}_l^k$ are balanced. Then the local model $\theta_l = (p_l, \phi_l)$ is updated in K iteration based on sample ξ_l^k that sampled (e.g. minibatch) from $\mathcal{T}_l^t \cup Z^t$. Since the backbone (feature extractor) is frozen, and $\mathcal{T}_l^t \cup Z^t$ has balanced samples for all classes, then ξ_l^k approximates ξ^k, which is a sample from $\mathcal{T}^t$. The local model is updated by the stochastic gradient (SG) approach as presented in equations (6) and (10). Suppose that $g(\theta_l, \xi_l^k)$ is the stochastic gradient function, then the update process can be simplified as:

$$\theta_l^{k+1} \leftarrow \theta_l^k - \alpha_l^k g(\theta_l^k, \xi_l^k) \tag{7.31}$$

Under assumption 1, and local training updates θ by iterating SG with sample ξ_l^k, then we have:

$$\begin{aligned} F_l(\theta_l^{k+1}) - F_l(\theta_l^k) &\leq (\theta_l^{k+1} - \theta_l^k)^T \nabla F_l(\theta_l^k) + \frac{L}{2}||\theta_l^{k+1} - \theta_l^k||_2^2 \\ &\leq \nabla F_l(\theta_l^k)^T(\theta_l^{k+1} - \theta_l^k) + \frac{L}{2}||\theta_l^{k+1} - \theta_l^k||_2^2 \\ &\leq -\alpha_l^k \nabla F_l(\theta_l^k)^T g(\theta_l^k, \xi_l^k) + {\alpha_l^k}^2 \frac{L}{2}||g(\theta_l^k, \xi_l^k)||_2^2 \end{aligned} \tag{7.32}$$

The equation above can be derived into:

$$\begin{aligned} \mathbb{E}_{\xi_l^k}[F_l(\theta_l^{k+1})] - F_l(\theta_l^k) \leq &- \alpha_l^k \nabla F_l(\theta_l^k)^T \mathbb{E}[g(\theta_l^k, \xi_l^k)] \\ &+ {\alpha_l^k}^2 \frac{L}{2} \mathbb{E}_{\xi_l^k}[||g(\theta_l^k, \xi_l^k)||_2^2] \end{aligned} \tag{7.33}$$

The inequality above shows how SG optimizes θ_l^k at a step k, and we can see clearly that the decreasing of F_l (left side) is bounded by a quantity in the right side involving ∇F_l which is directional derivative of F_l at θ_l^k along with $-g(\theta_l^k, \xi_l^k)$ (first term) and second moment of $g(\theta_l^k, \xi_l^k)$ (second term). In the case where $g(\theta_l^k, \xi_l^k)$ is unbiased estimator of ∇F_l, then the inequality above can be derived as:

$$\mathbb{E}_{\xi_l^k}[F_l(\theta_l^{k+1})] - F_l(\theta_l^k) \leq -\alpha_l^k \nabla ||F_l(\theta_l^k)||_2^2 + {\alpha_l^k}^2 \frac{L}{2} \mathbb{E}_{\xi_l^k}[||g(\theta_l^k, \xi_l^k)||_2^2] \tag{7.34}$$

The inequality above shows that as long as the stochastic directions and step size are chosen, then the SGD convergence is guaranteed. To avoid the harm of the second term of the right side in the inequality above, we apply the restriction below

$$\mathbb{V}[g(\theta_l^k, \xi_l^k)] = \mathbb{E}[||g(\theta_l^k, \xi_l^k)||_2^2] - ||\mathbb{E}[g(\theta_l^k, \xi_l^k)]||_2^2. \tag{7.35}$$

Adopting first and second-moment limit as in [231], then we add the following assumption.

Assumption 6: The objective function F_l and SG satisfy the following conditions.

(a). The sequence of $\{\theta_l^k\}$ is contained in an open space where F_l is bounded below by a scalar F_{inf}

(b) Exist scalars $\nu_G \geq \nu > 0$ so that for all $k \in \mathbb{N}$ satisfy:

$$\begin{aligned} &\nabla F_l(\theta_l^k)^T \mathbb{E}_{\xi_l^k}[g(\theta_l^k, \xi_l^k)] \geq \nu ||\nabla F_l(\theta_l^k) T||_2^2, and \\ &||\mathbb{E}_{\xi_l^k}[g(\theta_l^k, \xi_l^k)]||_2 \leq \nu_G ||\nabla F_l(\theta_l^k)||_2. \end{aligned} \tag{7.36}$$

(c) Exist scalars $m_1 \geq 0$ and $m_2 \geq 0$ so that for all $k \in \mathbb{N}$ satisfy:

$$\mathbb{V}[g(\theta_l^k, \xi_l^k)] \leq m_1 + m_2 ||\nabla F_l(\theta_l^k)||_2^2 \tag{7.37}$$

Combining assumption 6 and restriction criteria as presented in equation (A19), then we have:

$$\begin{aligned} \mathbb{E}_{\xi_l^k}[||g(\theta_l^k, \xi_l^k)||_2^2] \leq m_1 + m_G ||\nabla F_l(\theta_l^k)||_2^2, with \\ m_G = m_2 + \nu_G^2 \geq \nu^2 > 0 \end{aligned} \tag{7.38}$$

Then by substituting $\mathbb{E}_{\xi_l^k}[||g(\theta_l^k, \xi_l^k)||_2^2]$ from equation (7.38) into equation (7.33), we have:

$$\begin{aligned} \mathbb{E}_{\xi_l^k}[F_l(\theta_l^{k+1})] - F_l(\theta_l^k) \leq &- \alpha_l^k \nabla F_l(\theta_l^k)^T \mathbb{E}[g(\theta_l^k, \xi_l^k)] \\ &+ \alpha_l^{k\,2} \frac{L}{2} (m_1 + m_G ||\nabla F_l(\theta_l^k)||_2^2) \end{aligned} \tag{7.39}$$

Assumption 5 ensures that $\{\alpha_l^k\} \to 0$ practically can be achieved by using a learning rate decay that reduces the learning rate in every step of local training. Then, by choosing $\alpha_l^k L m_G \leq \nu$ and substituting $\nabla F_l(\theta_l^k)^T \mathbb{E}[g(\theta_l^k, \xi_l^k)]$ in equation (7.39) with the condition in assumption 6.b, we have

$$\begin{aligned} \mathbb{E}_{\xi_l^k}[F_l(\theta_l^{k+1})] - F_l(\theta_l^k) \leq &- \alpha_l^k \nu ||\nabla F_l(\theta_l^k)||_2^2 \\ &+ \alpha_l^{k\,2} \frac{L}{2} (m_1 + m_G ||\nabla F_l(\theta_l^k)||_2^2) \end{aligned} \tag{7.40}$$

Applying expectation to the equation above yields

$$\begin{aligned}
\mathbb{E}_{\xi_l^k}[F_l(\theta_l^{k+1})] - \mathbb{E}[F_l(\theta_l^k)] \leq & -\alpha_l^k \nu \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2] \\
& + {\alpha_l^k}^2 \frac{1}{2}(m_1 + m_G \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2]) \\
\mathbb{E}_{\xi_l^k}[F_l(\theta_l^{k+1})] - \mathbb{E}[F_l(\theta_l^k)] \leq & -(\nu - \frac{1}{2}\alpha_l^k L m_G)\alpha_l^k \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2] \\
& + \frac{1}{2}{\alpha_l^k}^2 L m_1 \\
\mathbb{E}_{\xi_l^k}[F_l(\theta_l^{k+1})] - \mathbb{E}[F_l(\theta_l^k)] \leq & -\frac{1}{2}\nu \alpha_l^k \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2] \\
& + \frac{1}{2}{\alpha_l^k}^2 L m_1
\end{aligned} \tag{7.41}$$

Sum both sides for $k \in \{1, ..., K\}$ we get

$$\begin{aligned}
F_{inf} - \mathbb{E}[F(\theta_l^1)] &\leq \mathbb{E}[F_l(\theta_l^{K+1})] - \mathbb{E}[F_l(\theta_l^1)] \\
F_{inf} - \mathbb{E}[F(\theta_l^1)] &\leq -\frac{1}{2}\nu \sum_{k=1}^{K} \alpha_l^k \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2] + \frac{1}{2} L m_1 \sum_{k=1}^{K} {\alpha_l^k}^2
\end{aligned} \tag{7.42}$$

Dividing by ν for both sides, then we get

$$\sum_{k=1}^{K} \alpha_l^k \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2] \leq \frac{2(\mathbb{E}[F(\theta_l^1)] - F_{inf})}{\nu} + \frac{L m_1}{\nu} \sum_{k=1}^{K} {\alpha_l^k}^2 \tag{7.43}$$

Applying $\lim_{K\to\infty}$ and assumption 5 to the equation above we get

$$\begin{aligned}
\lim_{K\to\infty} \sum_{k=1}^{K} \alpha_l^k \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2] \leq & \frac{2(\mathbb{E}[F(\theta_l^1)] - F_{inf})}{\nu} \\
& + \frac{L m_1}{\nu} \lim_{K\to\infty} \sum_{k=1}^{K} {\alpha_l^k}^2 < \infty
\end{aligned} \tag{7.44}$$

Dividing both sides with $\sum_{k=1}^{K} \alpha_l^k$, and following assumption 5 where $\lim_{K\to\infty} \sum_{k=1}^{K} \alpha_l^k = \infty$ and $\lim_{K\to\infty} \sum_{k=1}^{K} {\alpha_l^k}^2 < \infty$, then the right side will return 0. Therefore, we have

$$\lim_{K\to\infty} \frac{\sum_{k=1}^{K} \mathbb{E}[\alpha_l^k ||\nabla F_l(\theta_l^k)||_2^2]}{\sum_{k=1}^{K} \alpha_l^k} = 0 \tag{7.45}$$

$$\lim_{K\to\infty} \mathbb{E}[\frac{\sum_{k=1}^{K} \alpha_l^k ||\nabla F_l(\theta_l^k)||_2^2}{\sum_{k=1}^{K} \alpha_l^k}] = 0 \tag{7.46}$$

$$\lim_{k\to\infty} \mathbb{E}[||\nabla F_l(\theta_l^k)||_2^2] = 0 \tag{7.47}$$

The equation above presents the convergence for local training in the l-th client, where the gradient of loss F converges to 0 along with the increase of training step/iteration k.

Proof of Theorem 2: Suppose that set of local client $\{S_l\}_{l=1}^{l=L_S}$ is trained locally with its local data $\{\mathcal{T}_l^t \cup Z^t\}_{l=1}^{l=L_S}$, where $\mathcal{T}_l^t$ is local training sample for l-th client. Local training is conducted in k steps/iterations using a sample, i.e., a minibatch of the local training set $\xi_l^k \in \mathcal{T}_l^t$ on each step. Global synchronization is executed in each round $r = \{1, 2, ..., R\}$. We global synchronization step as $\mathcal{I}_E = \{rE | r = 1, 2, ...R\}$. Following [230], we define θ_l^{k+1} as the local parameter of l-client after communication steps, while ϑ_l^{k+1} represents the local parameter after an immediate result of one step SGD. Therefore, the definition satisfies:

$$\vartheta_l^{k+1} = \theta_l^k - \alpha_l^k \nabla F_l(\theta_l^k, \xi_l^k) \tag{7.48}$$

$$\theta_l^{k+1} = \begin{cases} \vartheta_l^{k+1} \text{ if } k+1 \notin \mathcal{I}_E \\ \sum_{l=1}^{L_S} w_l^k \vartheta_l^{k+1} \text{ if } k+1 \in \mathcal{I}_E \end{cases} \tag{7.49}$$

Where $w_l = \omega_l / \sum_{l=1}^{L_S} \omega_l$, where ω_l is the weight of l-th client. We define $\bar{\vartheta}_l^{k+1} = \sum_{l=1}^{L_S} w_l \vartheta_l^{k+1}$ and $\bar{\theta}_l^{k+1} = \sum_{l=1}^{L_S} w_l \theta_l^{k+1}$, $\bar{\vartheta}_l^{k+1}$ is the result of single step SGD iteration from $\bar{\theta}_l^{k+1}$. We also define $\bar{g}^k = \sum_{l=1}^{L_S} w_l \nabla F_l(\theta_l^k)$ and $g^k = \sum_{l=1}^{L_S} w_l \nabla F_l(\theta_l^k, \xi_l^k)$. We adopt the following lemmas from [230], which were derived from fully participating clients in federated learning.

Lemma 1: By applying assumptions 1 and 2, in one step SGD training and choosing $\alpha \leq \frac{1}{4L}$, we have

$\mathbb{E}[||\bar{\vartheta}^{k+1} - \theta^*||^2] \leq (1 - \alpha^k \mu)\mathbb{E}[||\bar{\theta}^k - \theta^*||^2] - (\alpha^k)^2 \mathbb{E}[||g^k - \bar{g}^k||^2] + 6L(\alpha^k)^2 \Gamma + 2\mathbb{E}[\sum_{l=1}^{L_S} w_l ||\bar{\theta}^k - \theta_l^k||^2]$ where $\Gamma = F^* - \sum_{l=1}^{L_S} w_l F_l^* \geq 0$.

Lemma 2: By applying assumption 3, the gradient function follows:

$\mathbb{E}[||\bar{g}^k - \bar{g}^k||^2] \leq \sum_{l=1}^{L_S} w_l^2 \sigma_l^2$, where σ_l^2 is the variance of θ_l

Lemma 3: By applying assumption 4, where α^k is non-increasing and it satisfies $\alpha^k \leq \alpha^{k+E}$for all$k \geq 0$, then we have $\mathbb{E}[\sum_{l=1}^{L_S} ||\bar{\theta}^{k+1} - \theta_l^k||^2] \leq 4(\alpha^k)^2 (E-1)^2 G^2$

In fully participating clients we always have $\bar{\theta}^{k+1} = \bar{\vartheta}^{k+1}$. However, in partially participating clients, we use a random sampling mechanism so that it satisfies $\mathbb{E}_{S_L}[\bar{\theta}^{k+1}] = \bar{\vartheta}^{k+1}$. We also adopt the bounding condition from [230] as shown in Lemma 4.

Lemma 4: The expected different between $\bar{\theta}^{k+1}$and $\bar{\vartheta}^{k+1}$ bounded by : $\mathbb{E}_{S_L}[||\bar{\vartheta}^{k+1} - \bar{\theta}^{k+1}||^2] \leq \frac{4}{L_S}(\alpha^k)^2E^2G^2$ and in the case of w_l is uniform for all l-th client, then $\mathbb{E}_{S_L}[||\bar{\vartheta}^{k+1} - \bar{\theta}^{k+1}||^2] \leq \frac{4(N_S-L_S)}{N_S-1}(\alpha^k)^2E^2G^2$, where N_S is total clients and L_S is number of selected clients.

Please note that

$$||\bar{\theta}^{k+1} - \theta^*||^2 = ||\bar{\theta}^{k+1} - \theta^*||^2 \tag{7.50}$$

$$||\bar{\theta}^{k+1} - \theta^*||^2 = ||\bar{\theta}^{k+1} - \bar{\vartheta}^{k+1} + -\bar{\vartheta}^{k+1} - \theta^*||^2 \tag{7.51}$$

$$\begin{aligned} ||\bar{\theta}^{k+1} - \theta^*||^2 =& ||\bar{\theta}^{k+1} - \bar{\vartheta}^{k+1}||^2 + ||\bar{\vartheta}^{k+1} - \theta^*||^2 \\ &+ 2||\bar{\theta}^{k+1} - \bar{\vartheta}^{k+1}||.||\bar{\vartheta}^{k+1} - \theta^*|| \end{aligned} \tag{7.52}$$

$$\begin{aligned} ||\bar{\theta}^{k+1} - \theta^*||^2 =& ||\bar{\theta}^{k+1} - \bar{\vartheta}^{k+1}||^2 + ||\bar{\vartheta}^{k+1} - \theta^*||^2 \\ &+ 2\langle\bar{\theta}^{k+1} - \bar{\vartheta}^{k+1}, \bar{\vartheta}^{k+1} - \theta^*\rangle \end{aligned} \tag{7.53}$$

In the case of $k+1 \notin \mathcal{I}_E$, then the term $||\bar{\theta}^{k+1} - \bar{\vartheta}^{k+1}||^2$ vanishes. Then, by applying Lemma 4, we get

$$\mathbb{E}[||\bar{\theta}^{k+1} - \theta^*||^2] \leq (1 - \alpha^k\mu)\mathbb{E}[||\bar{\theta}^{k+1} - \theta^*||^2] + (\alpha^k)B \tag{7.54}$$

In the case of $k + 1 \in \mathcal{I}_E$, then by applying lemma 4, we get

$$\mathbb{E}[||\bar{\theta}^{k+1} - \theta^*||^2] \leq (1 - \alpha^k\mu)\mathbb{E}[||\bar{\theta}^{k+1} - \theta^*||^2] + (\alpha^k)(B + C) \tag{7.55}$$

where $B = \sum_{l=1}^{L_S} w_l\sigma_l^2 + 6L\Sigma + 8(E-1)^2G^2$ and $C = \frac{4(N_S-L_S)}{N_S-1}(E^2G^2)$ if w_l is uniform and $C = \frac{4}{L_S}(E^2G^2)$ otherwise.

By choosing $\alpha^k = \frac{\beta}{k+\delta}$ for some $\beta > 1/\mu$ and $\delta > 0$ so that $\alpha^1 \leq \min\{1/\mu, 1/4L\} = 1/4L$ and $\alpha^k \leq 2\alpha^{k+E}$ then we have $\mathbb{E}[||\bar{\theta}^{k+1} - \theta^*||^2] \leq \frac{v}{\delta+k}$ where $v = \max\{\frac{\beta^2(B+C)}{\beta\mu-1}, (\delta+1)||\bar{\theta}^{k+1} - \theta^*||^2\}$

Then, by applying a strong convexity assumption of F, we have

$$\mathbb{E}[\bar{\theta}^k] - F^* \leq \frac{L}{2}\Delta^k \leq \frac{L}{2}\frac{v}{\delta + k} \tag{7.56}$$

where F^* is the minimum value of F where optimum parameter θ^* is achieved. Later on, if we choose $\beta = 2/\mu, \delta = \max\{8L/\mu, E\}$and denote $\kappa = L/\mu, \alpha^k = 2/u(1/(\delta + k))$ then we have

$$\mathbb{E}[F(\bar{\theta}^k)] - F^* \leq \frac{\kappa}{(\delta + k - 1)}(\frac{2(B + C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\theta^1 - \theta^*||) \tag{7.57}$$

The equation generalizes federated learning, where the model is trained in a total of k steps/iterations, where practical implementation satisfies $k = b.E.R, b$ is the number of batches. here we know that $k > R$ as $E$and$b$ are positive integers. Therefore, substituting k with R in the inequality above will produce a higher amount on the right side. Therefore, the inequality above can be generalized into:

$$\mathbb{E}[F(\theta^R)] - F^* \leq \frac{\kappa}{(\delta + R - 1)}(\frac{2(B + C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\theta^1 - \theta^*||) \tag{7.58}$$

The inequality above guarantees that to achieve a convergence condition in federated learning with weighted aggregation is upper bounded by the amount on the right side. The inequality above also shows the convergence guarantee by the method in the case of $b = 1$ or $E = 1$ or both, which simulates a small number of samples so that local training is only executed with 1 batch, or in an urgent (need to be fast) condition where local training is only executed with 1 epoch, or those two conditions occur in the same time.

Proof of Theorem 3: Given θ^* and θ are optimal parameter in $\mathcal{T}_l^t \cup Z^t$ and $\mathcal{T}_l^t$ respectively, where $\mathcal{T}_l^t \subset \mathcal{T}^t$, where $|\mathcal{T}_l^t|/|\mathcal{T}^t| = \eta \in (0, 1)$, then we have

$$F(\theta; \mathcal{T}^t) = \eta F(\theta; \mathcal{T}_l^t) + (1 - \eta)F(\theta; (\mathcal{T}^t - \mathcal{T}_l^t)) \tag{7.59}$$

$$F(\theta^*; \mathcal{T}^t) = \eta F(\theta^*; \mathcal{T}_l^t) + (1 - \eta)F(\theta^*; (\mathcal{T}^t - \mathcal{T}_l^t)) \tag{7.60}$$

Suppose that θ^o is the initial value both for $\theta$and θ^* that is set by a random uniform initiation method. Therefore for all class $c \in \mathcal{T}_c^t = \mathcal{T}_{y=c}^t$ It satisfy $F(\theta^o; \mathcal{T}_c^t) = e^o$. After optimally learning on $\mathcal{T}_l^t$ and $\mathcal{T}_l^t \cup Z^t$ then θ^o become to θ and θ^* respectively. Please note that θ learns only available class in $\mathcal{T}_l^t$, while θ^* learns classes that

available in $\mathcal{T}_l^t$ and classes in $\mathcal{T}^t - \mathcal{T}_l^t$ via Z^t. Suppose that the loss for predicting classes in $\mathcal{T}_l^t$ is defined as $e^a < e^o$ then we have $F(\theta; \mathcal{T}_l^t) = F(\theta^*; \mathcal{T}_l^t) = e^a < e^o$. Since the backbone is frozen, θ^* learn Z^t then we have $F(\theta; (\mathcal{T}^t - \mathcal{T}_l^t)) = e^o$, while $F(\theta^*; (\mathcal{T}^t - \mathcal{T}_l^t)) = e^b$, where $e^a \geq e^b \geq e^o$.

Then equations (7.59) and (7.60) can be derived to

$$F(\theta; \mathcal{T}^t) = \eta e^a + (1-\eta)e^o \tag{7.61}$$

$$F(\theta^*; \mathcal{T}^t) = \eta e^a + (1-\eta)e^b \tag{7.62}$$

Substracting the equations above, then we have

$$F(\theta; \mathcal{T}^t) - F(\theta^*; \mathcal{T}^t) = \eta e^a + (1-\eta)e^o - (\eta e^a + (1-\eta)e^b) \tag{7.63}$$

$$F(\theta; \mathcal{T}^t) - F(\theta^*; \mathcal{T}^t) = \eta e^a + (1-\eta)e^o - \eta e^a - (1-\eta)e^b \tag{7.64}$$

$$F(\theta; \mathcal{T}^t) - F(\theta^*; \mathcal{T}^t) = (1-\eta)e^o - (1-\eta)e^b \tag{7.65}$$

$$F(\theta; \mathcal{T}^t) - F(\theta^*; \mathcal{T}^t) = (1-\eta)(e^o - e^b) \tag{7.66}$$

Since $0 < \eta < 1$ and $e^o > e^b$ the right side of the inequality above has a positive value. By choosing a small positive value $\epsilon > 0$ where $(1-\eta)(e^o - e^b) \geq \epsilon$ then we have.

$$F(\theta; \mathcal{T}^t) - F(\theta^*; \mathcal{T}^t) \geq \epsilon \tag{7.67}$$

inequality above proves that θ^* is more generalized to $\mathcal{T}^t$ than θ. This shows that our idea, i.e., injecting the shared prototypes improves model generalization.

7.5 Experiment Results and Analysis

7.5.1 Datasets

Our experiment is conducted using three main benchmarks in FCIL, i.e., split CIFAR100, split MiniImageNet, and split TinyImageNet. The CIFAR100 and miniImageNet datasets each contain 100 classes, while TinyImagenet is a dataset with 200 classes. We follow settings from [47] and [59] where the dataset is split equally into all tasks. In our main numerical result, each dataset is split into 10 tasks, i.e., 10 classes per task for CIFAR100 and MiniImageNet, and 20 classes per task for the TinyImageNet dataset. In our further analysis, we investigate the performance of the proposed methods in different task sizes, e.g., T=5 and T=20.

7.5.2 Benchmark Algorithms

PIP is implemented in two version algorithms, i.e., PIP-L2P and PIP-DualP with L2P and DualP prompt structure respectively, PIP-L2P and PIP-DualP are compared with 10 state-of-the-art algorithms: LGA [59], TARGET [63], GLFC[47], AFC[232]+FL, DyTox[233]+FL, SS-IL[234]+FL, GeoDL[235]+iCaRL[31]+FL, DDE[236]+iCaRL[31]+FL, PODNet[237]+FL, BiC[35]+FL, iCaRL[31]+FL, and the proposed baseline models (FedL2P and Fed-DualP.). We compare with Fed-CPrompt[152] in the CIFAR100 dataset since it reported evaluation in the CIFAR100 dataset only. We didn't include a performance report from FCILPT[153] since they use memory in their experiment, which is different from our setting. Please note that we can not find the complete and runnable released code from both Fed-CPrompt and FCILPT to produce the results in our simulation.

7.5.3 Experiment Setup

Our numerical study is executed under a single NVIDIA A100 GPU with 40 GB memory across 3 runs with different random seeds {2021,2022,2023}. Fed-L2P, Fed-DualP, PIP-L2P, and PIP-DualP train T number of prompts $p \in R^{5\times 768}$ and $\phi \in R^{|\mathcal{C}\times 768|}$ head layer, while the competitors train whole CNN models following their original implementation. Following [59] and [47], each experiment is simulated by 30 total clients and 1 global server, where in each round, 10 (33.33%) local clients are selected randomly. Each client randomly receives 60% ($\eta = 0.6$) class label space to simulate non-i.i.d data between clients. The total global round is set to 100, the local clients' epoch is set to 2, and the learning rate is set by choosing the best value from {0.02,0.002}. On each task, we evaluate average accuracy and forgetting from all learned tasks.

7.5.4 Numerical Results

The numerical result of the consolidated algorithms is shown in tables 7.2 and 7.3. Except from Fed-CPrompt, the baseline method (Fed-DualP) already achieves higher average accuracy (Avg) than the SOTA methods with $1-13\%$ improvement in accuracy. Fed-DualP also experiences a lower performance drop (PD) ($19-27\%$)

Table 7.2: Numerical results on CIFAR100 (T=10), "BL" denotes our proposed baseline, "Avg" denotes the average accuracy of all tasks, denotes performance drop, and "Imp" denotes improvement/gap of PIP-DualP compared to the respective method.

	Accuracy in each task (T) on CIFAR100 (T=10, @10classes/task)												
Method	1	2	3	4	4	6	7	8	9	10	Avg	PD	Imp
iCaRL+FL	89.0	55.0	57.0	52.3	50.3	49.3	46.3	41.7	40.3	36.7	51.8	52.3	36.2
BiC+FL	88.7	63.3	61.3	56.7	53.0	51.7	48.0	44.0	42.7	40.7	55.0	48.0	33.0
PODNet+FL	89.0	71.3	69.0	63.3	59.0	55.3	50.7	48.7	45.3	45.0	59.7	44.0	28.3
DDE+iCaRL+FL	88.0	70.0	67.3	62.0	57.3	54.7	50.3	48.3	45.7	44.3	58.8	43.7	29.2
GeoDL+iCaRL+FL	87.0	76.0	70.3	64.3	60.7	57.3	54.7	50.3	48.3	46.3	61.5	40.7	26.5
SS-IL+FL	88.3	66.3	54.0	54.0	44.7	54.7	50.0	47.7	45.3	44.0	54.9	44.3	33.1
DyTox+FL	86.2	76.9	73.3	69.5	62.1	62.7	58.1	57.2	55.4	52.1	65.4	34.1	22.7
AFC+FL	85.6	73.0	65.1	62.4	54.0	53.1	51.9	47.0	46.1	43.6	58.2	42.0	29.8
GLFC	90.0	82.3	77.0	72.3	65.0	66.3	59.7	56.3	50.3	50.0	66.9	40.0	21.1
LGA	89.6	83.2	79.3	76.1	72.9	71.7	68.4	65.7	64.7	62.9	73.5	26.7	14.6
TARGET	70.6	39.0	27.5	21.6	18.8	16.4	14.2	12.2	10.7	9.1	24.0	61.4	64.0
Fed-CPrompt	97.3	90.5	86.7	83.4	83.3	82.2	81.6	79.7	79.0	79.4	84.3	17.9	3.7
Fed-L2P (BL)	94.4	73.9	68.6	67.5	64.7	64.8	66.6	66.6	66.2	65.9	69.9	28.6	18.1
Fed-DualP (BL)	96.8	81.6	75.7	72.6	69.5	67.3	69.9	69.2	69.9	69.7	74.2	27.1	13.8
PIP-L2P (Ours)	95.4	90.1	86.7	84.7	82.9	80.9	80.1	79.5	78.5	77.5	83.6	17.9	4.4
PIP-DualP (Ours)	**98.7**	**92.9**	**89.4**	**87.6**	**87.0**	**85.3**	**85.2**	**84.8**	**84.8**	**84.3**	**88.0**	**14.4**	**-**

Table 7.3: Numerical results on MiniImageNet (T=10), "BL" denotes our proposed baseline, "Avg" denotes the average accuracy of all tasks, "PD" denotes performance drop, and "Imp" denotes improvement/gap of PIP-DualP compared to the respective method.

	Accuracy in each task (T) on MiniImageNet (T=10, @10classes/task)												
Method	1	2	3	4	5	6	7	8	9	10	Avg	PD	Imp
iCaRL+FL	74.0	62.3	56.3	47.7	46.0	40.3	37.7	34.3	33.3	32.7	46.5	41.3	44.6
BiC+FL	74.3	63.0	57.7	51.3	48.3	46.0	42.7	37.7	35.3	34.0	49.0	40.3	42.0
PODNet+FL	74.3	64.0	59.0	56.7	52.7	50.3	47.0	43.3	40.0	38.3	52.6	36.0	38.5
DDE+iCaRL+FL	76.0	57.7	58.0	56.3	53.3	50.7	47.3	44.0	40.7	39.0	52.3	37.0	38.7
GeoDL+iCaRL+FL	74.0	63.3	54.7	53.3	50.7	46.7	41.3	39.7	38.3	37.0	49.9	37.0	41.1
SS-IL+FL	69.7	60.0	50.3	45.7	41.7	44.3	39.0	38.3	38.0	37.3	46.4	32.4	44.6
DyTox+FL	76.3	68.3	64.8	58.6	45.4	41.3	39.7	37.1	36.2	35.3	50.3	41.0	40.7
AFC+FL	82.5	74.1	66.8	60.0	48.0	44.3	42.5	40.9	39.0	36.1	53.4	46.4	37.6
GLFC	73.0	69.3	68.0	61.0	58.3	54.0	51.3	48.0	44.3	42.7	57.0	30.3	34.0
LGA	83.0	74.2	72.3	72.2	68.1	65.8	64.0	59.6	58.4	57.5	67.5	25.5	23.5
TARGET	56.1	27.5	20.0	15.9	11.9	8.8	6.9	5.0	2.7	2.3	15.7	53.8	75.3
Fed-L2P (BL)	95.2	80.9	79.1	77.7	76.3	75.7	76.0	75.8	74.1	76.7	78.7	18.6	12.3
Fed-DualP (BL)	97.6	83.6	82.5	79.2	76.6	76.2	75.8	75.8	75.6	78.5	80.1	19.1	10.9
PIP-L2P (Ours)	96.1	91.9	89.6	88.9	88.2	88.2	87.4	87.2	85.5	85.7	88.9	10.5	2.2
PIP-DualP (Ours)	**98.4**	**90.9**	**90.4**	**90.7**	**90.2**	**90.7**	**90.3**	**90.4**	**88.9**	**89.2**	**91.0**	**9.2**	-

Table 7.4: Numerical results on TinyImageNet (T=10), "BL" denotes our proposed baseline, "Avg" denotes the average accuracy of all tasks, "PD" denotes performance drop, and "Imp" denotes improvement/gap of PIP-DualP compared to the respective method.

	Accuracy in each task (T) on TinyImageNet (T=10, @10classes/task)												
Method	1	2	3	4	5	6	7	8	9	10	Avg	PD	Imp
iCaRL+FL	63.0	53.0	48.0	41.7	38.0	36.0	33.3	30.7	29.7	28.0	40.1	35.0	46.1
BiC+FL	65.3	52.7	49.3	46.0	40.3	38.3	35.7	33.0	31.7	29.0	42.1	36.3	44.1
PODNet+FL	66.7	53.3	50.0	47.3	43.7	42.7	40.0	37.3	33.7	31.3	44.6	35.4	41.7
DDE+iCaRL+FL	69.0	52.0	50.7	47.0	43.3	42.0	39.3	37.0	33.0	31.3	44.5	37.7	41.8
GeoDL+iCaRL+FL	66.3	54.3	52.0	48.7	45.0	42.0	39.3	36.0	32.7	30.0	44.6	36.3	41.6
SS-IL+FL	62.0	48.7	40.0	38.0	37.0	35.0	32.3	30.3	28.7	27.0	37.9	35.0	48.4
DyTox+FL	73.2	66.6	48.0	47.1	41.6	40.8	37.4	36.2	32.8	30.6	45.4	42.6	40.8
AFC+FL	73.7	59.1	50.8	43.1	37.0	35.2	32.6	32.0	28.9	27.1	42.0	46.6	44.3
GLFC	66.0	58.3	55.3	51.0	47.7	45.3	43.0	40.0	37.3	35.0	47.9	31.0	38.4
LGA	70.3	64.0	60.3	58.0	55.8	53.1	47.9	45.3	39.8	37.3	53.2	33.0	33.1
TARGET	57.1	32.7	20.9	14.7	11.6	8.3	7.1	5.2	3.8	3.0	16.4	54.1	69.8
Fed-L2P (BL)	80.6	71.5	70.6	67.5	65.2	66.7	66.2	65.9	64.2	64.3	68.3	16.3	18.0
Fed-DualP (BL)	86.3	74.6	71.2	65.9	63.3	62.0	61.3	59.9	58.2	61.3	66.4	25.0	19.9
PIP-L2P (Ours)	85.4	84.0	83.5	84.5	83.3	83.7	82.3	81.3	79.8	77.4	82.5	**7.9**	3.7
PIP-DualP (Ours)	**92.8**	**86.4**	**86.6**	**87.5**	**86.7**	**87.0**	**85.3**	**84.9**	**83.9**	**81.4**	**86.3**	**11.4**	-

compared to the SOTA methods ($\geq$ 26%) except in the CIFAR100 dataset vs. LGA. Fed-L2P achieves higher performance than SOTAs in MiniImageNet and TinyImageNet datasets with $\geq$ 11% gap, but lower performance in the CIFAR100 dataset. The proposed method (PIP-DualP) achieves the highest accuracy with $\geq$ 10% gap compared to the baseline method, and $\geq$ 14% gap compared to the competitor methods (except Fed-CPrompt). The proposed method also achieves the lowest performance drop with $(10 - 13\%)$ gap compared to the baseline and

Table 7.5: Detailed average forgetting of the consolidated methods in CIFAR100 (T=10).

Method	Average Forgetting on each Task										AVG
	1	2	3	4	5	6	7	8	9	10	
LGA	-	4.0	4.4	8.9	13.7	12.6	13.9	14.7	15.7	16.8	11.6
TARGET	-	-0.6	0.4	1.7	1.7	2.0	2.5	3.8	4.7	4.7	2.3
Fed-CPrompt	-	3.8	5.0	5.3	5.2	5.2	5.1	5.4	5.3	4.8	5.0
Fed-L2P	-	0.9	1.9	0.3	-0.4	-3.4	-5.0	-4.0	-3.2	-4.0	-1.9
Fed-DualP	-	-0.4	2.7	2.3	2.4	2.8	-0.9	0.6	0.4	0.6	1.2
PIP-L2P	-	2.3	4.7	5.1	6.0	6.5	6.0	6.4	6.4	8.4	5.8
PIP-DualP	-	1.3	1.5	1.9	2.3	3.1	2.9	2.4	2.5	2.8	2.3

Table 7.6: Ablation study on CIFAR100 dataset in one-seeded run, i.e., 2021, using DualP prompt structure. "Avg" denotes the average accuracy of all tasks, "PD" denotes performance drop, and "Imp" denotes improvement/gap of PIP compared to the respective configuration.

Conf.	Prompt	Proto +Aug	W.Agg	Head	T1 (C=10)	T10 (C=100)	Avg	PD	Imp
A	✓				82.15	60.66	65.69	21.49	22.59
B	✓	✓			90.99	77.56	81.59	**13.43**	6.69
C	✓		✓		82.13	61.35	66.10	20.78	22.18
D	✓	✓	✓		91.37	78.47	82.41	**12.90**	5.87
E	✓			✓	96.60	70.68	75.38	25.92	12.90
F	✓	✓		✓	**98.90**	84.17	87.82	14.73	0.46
G	✓		✓	✓	98.60	69.94	72.57	28.66	15.71
PIP	✓	✓	✓	✓	**98.60**	**84.60**	**88.28**	**14.00**	**-**

$\geq$ 12% gap compared to the competitor methods. The table shows that PIP achieves a higher gap in TinyImageNet, which is a relatively more complex problem than in MiniImageNet and CIFAR100. Compared to Fed-CPrompt, our proposed method achieves a higher accuracy and lower performance drop with a significant margin, i.e., $\geq 3\%$

Looking at performance on each task, tables 7.2 and 7.3 show that PIP-DualP achieves the highest accuracy in all tasks in those three datasets. In the first task, the proposed method achieves higher performance with a small gap, i.e., $1-4\%$ in comparison to the baseline method. However, with the increasing number of tasks, the gap gets higher, e.g., $5-12\%$ in task-2, $5-20\%$ in task-3, and $6-20\%$ in task-10 (last task). It shows that the proposed method handles catastrophic forgetting better than the baseline method. PIP-L2P achieves second-best performance in all tasks except in the first task of the CIFAR100 dataset.

7.5.5 Forgetting Analysis

Table 7.5 shows the average forgetting of the consolidated methods in each task T. The table shows that our proposed method (PIP-DualP) achieves a smaller average forgetting than existing SOTAs, i.e., LGA and Fed-CPrompt, by a significant gap, i.e., 2.7% and 9.3% margin, respectively. Fed-DualP achieves a comparable average forgetting with TARGET, but looking at the trend, TARGET forgetting is

increasing along with the number of tasks, while PIP-DualP forgetting is relatively stable. Please note that TARGET achieves lower average accuracy than PIP-DualP with a 64% gap. Our baselines, i.e., Fed-DualP and Fed-L2P, achieve lower average forgetting than PIP-DualP despite achieving lower accuracy than PIP-DualP with a huge gap, i.e., ($>$ 13%). This shows that the baselines achieve better stability (old task accuracy) but, in exchange, struggle to achieve plasticity (new task accuracy).

7.5.6 Ablation Study

We conducted an ablation study to investigate the contribution of each component of the proposed method. The result is summarized in Table 7.6. The result shows that the prototype and augmentation contribute the most to the improvement of the performance as shown by the performance difference of configurations E vs. F (12%), and G vs. H (16%). The weighted aggregation improves performance up to 1%, as shown by the performance difference of PIP and configuration F. The head layer aggregation also plays an important role in the proposed method, as shown that the absence of this component decreases performance by $\geq$ 6% margin. The absence of two components, e.g., prototype and head aggregation (configuration C) or prototype and weighted aggregation (configuration E), impacts its performance significantly, with 22% and 13% accuracy drop, respectively. The absence of three components, i.e., prototype, weighted aggregation, and head aggregation (configuration A), costs a 23% performance drop in the method.

7.5.7 Performance Analysis on Diferent Task Size

We evaluate the performance of the proposed method compared to the competitor methods in different task sizes, i.e., T=5 and T=20 to further investigate the robustness of the proposed method. Figure 7.2 summarizes the performance of the consolidated methods in CIFAR100, MiniImageNet, and TinyImageNet with T=5 (upper figures) and T=20 (bottom figures). Both in T=5 and T=20 settings, PIP-DualP achieves the highest performance, almost in every task in all configurations. It achieves a slightly lower performance than PIP-L2P in the last 3 tasks of CIFAR100 with T=5. Besides, all figures show that the proposed method has

gentle slopes compared to the baseline and competitor methods. It shows that the proposed method experiences the lowest performance drop from t_{th} to $t+1_{th}$ task. It confirmed the robustness of the proposed method in different task-size settings. The baseline (Fed-DualP) method achieves better accuracy than the competitor methods in all 6 settings, except in CIFAR100 with the T=5 setting. In CIFAR100 with the T=5 setting, the baseline method achieves higher performance in task-1 and task-5, but lower performance in task-3, and comparable performance in task-2 and task-4. It shows the promising idea of the federated prompt-based approach for the FCIL problem.

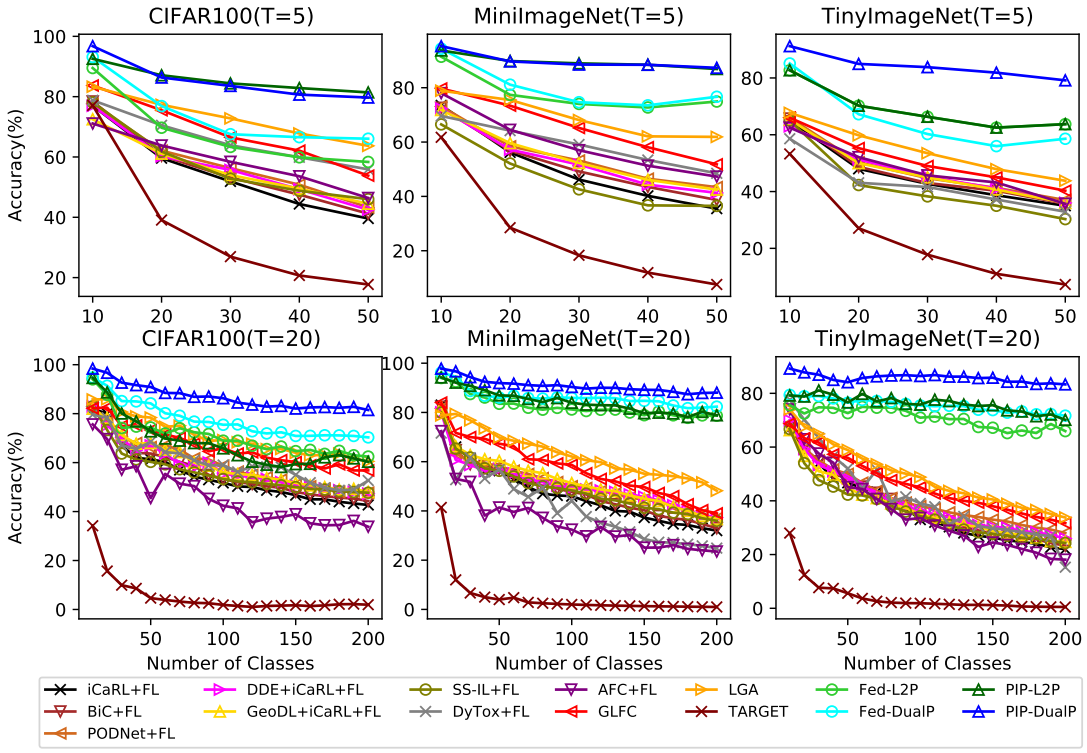


Figure 7.2: Performance of the consolidated methods in CIFAR100, MiniImageNet and TinyImageNet with T=5 and T=20.

7.5.8 Performance Analysis on Smaller Local Clients

We evaluate the performance of the proposed method on a smaller selected local clients to advance our investigation of the robustness of the proposed method

on smaller selected local clients. This simulates cross-device federated learning, where in each round, only a small portion of registered clients participate in the federated learning process. In our experiment with 30 total clients, we investigated the performance of the consolidated methods with 3 (10%) and 2 (6.67%) selected clients compared with the default setting with 10 clients (33.33%). Table 7.7 shows the performance of the consolidated methods in smaller local clients scenarios. The figure shows that the proposed methods experience the lowest performance degradation in fewer local clients. The figures show that both in CIFAR100 and TinyImageNet datasets, the proposed method still achieves higher accuracy than the baseline and the competitor methods with 10 participating local clients.

7.5.9 Performance Analysis on Smaller Global Rounds

We continue the previous experiment in smaller selected local clients, into smaller local clients with smaller global rounds, to study the robustness of the proposed method on a more extreme condition. This scenario simulates a condition where the model is urgently needed by the clients. In our experiment, we use 30 total clients and 3 (10%) local clients. Figure 7.3 summarizes our investigation on the performance of the proposed methods with 20 to 100 (default) global rounds in CIFAR100 and TinyImageNet datasets. The figure shows a common trend that the performance of the method is decreasing with the decrease in global rounds. However, our proposed methods (PIP-DualP and PIP-L2P) experience a relatively small amount of performance drop with the number of global rounds. Meanwhile, the baseline method experiences a significant drop in the CIFAR100 dataset when the number of global rounds is reduced to 20. Besides, it may experience overfitting when the number of global rounds is too high, as shown by the result on the TinyImageNet dataset. Compared to the current SOTA (GLFC and LGA) methods with the default number of global rounds (100), PIP-DualP still achieves higher performance, even though the number of global rounds is reduced to 20 (20% default setting).

Table 7.7: Performance of the consolidated methods in CIFAR100 and TinyImageNet with smaller selected local clients (L), ^{o} and * denote the results obtained from the first 4 and 5 tasks due to system crash.

Method	CIFAR100 (T=10)			TinyImageNet (T=10)		
	L=10 (33.3%)	L=3 (10%)	L=2 (6.67%)	L=10 (33.3%)	L=3 (10%)	L=2 (6.67%)
GLFC	66.92	62.42	57.39	47.89	34.89^o	32.77^o
LGA	73.45	68.87	66.68	53.18	60.61*	58.54*
TARGET	24.00	14.19	14.18	16.43	9.34	7.56
Fed-L2P	70.86	71.94	71.09	70.02	69.16	65.25
Fed-DualP	83.10	75.08	64.68	67.31	66.11	67.03
PIP-L2P	75.38	76.96	72.17	82.85	79.23	76.17
PIP-DualP	88.28	87.06	83.78	86.79	85	82.13

7.5.10 Complexity and Running Time Analysis

Following the pseudo-code in Algorithm 6, PIP generates prototypes when its prototype set is empty after a local epoch of a round-r (line 21), augments the prototypes in each local epoch (23), and updates the prototypes after local epochs (line 26-27). Knowing that generating prototypes from $\mathcal{T}_l^{r,t}$ costs $O(N_l^t)$, augmenting the prototypes costs $O(1)$ since it runs $m \in \{1..5\}$ times, then the PIP complexity will be:

$$O(PIP) = O(1) + T.R_T.(O(\text{clients}) + O(\text{server})) \tag{7.68}$$

$$O(PIP) = O(1) + T.R_T.(L.O(\text{1-client}) + O(\text{server})) \tag{7.69}$$

$$O(PIP) = O(1) + T.R_T.(L.O(\text{1-client}) + O(1)) \tag{7.70}$$

$$\begin{aligned} O(PIP) = O(1) + T.R_T.L.O(E.(\sum_{b=1}^{\beta} N_{bl}^t + N_l^t) + O(N_l^t) \\ + O(1)) + O(T.R_T) \end{aligned} \tag{7.71}$$

Since we have $\sum_{b=1}^{\beta} N_{bl}^t = N_l^t$, then we have

$$\begin{aligned} O(PIP) = O(1) + T.R_T.L.O(E.(N_l^t + N_l^t) + O(N_l^t) \\ + O(1)) + O(T.R_T) \end{aligned} \tag{7.72}$$

$$\begin{aligned} O(PIP) = O(1) + T.R_T.L.O(E.(N_l^t) + O(N_l^t) \\ + O(1)) + O(T.R_T) \end{aligned} \tag{7.73}$$

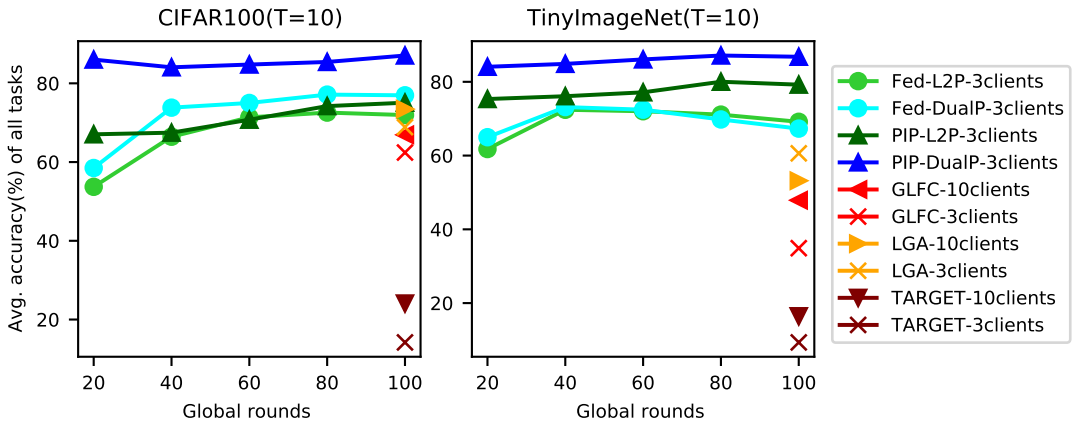


Figure 7.3: Performance of the consolidated methods in CIFAR100 and TinyImageNet with smaller local clients (L) and smaller rounds.

$$O(PIP) = O(1) + O(T.R_T.L.E.N_l^t) + O(T.R_T.L.N_l^t) + O(T.R_T.L) + O(T.R_T) \tag{7.74}$$

$$O(PIP) = O(T.R_T.L.E.N_l^t) + O(T.R_T.L.N_l^t) + O(T.R_T.L) + O(T.R_T) \tag{7.75}$$

Substituting the equalities in the previous definition that $R_T = R/T$ and $N_l = T.N_l^t = T$ the complexity of PIP will be

$$O(PIP) = O(T.R/T.L.E.N_l^t) + O(T.R/T.L.N_l^t) + O(T.R/T.L) + O(T.R/T) \tag{7.76}$$

$$O(PIP) = O(R.L.E.N_l^t) + O(R.L.N_l^t) + O(R.L) + O(R) \tag{7.77}$$

$$O(PIP) = O(R.L.E.N_l^t) \tag{7.78}$$

Since $N_l^t < N_l$ and E is set as a small constant in our method, i.e., 2, then the PIP complexity will be:

$$O(PIP) = O(R.L.N_l) \tag{7.79}$$

Our complexity analysis shows that both the baseline method and our proposed method have the same complexity, i.e., $O(R.L.N_l)$ where R is the number of global rounds, L is the number of participating local clients in each round, and N_l is the size of the training data in each client. Table 7.8 summarizes the training time of the consolidated method in three datasets with T=10. The table shows that the

proposed method requires lower training time and far lower communication cost than the current competing SOTAs, i.e., LGA and GLFC, in all datasets. The training time is higher than the baseline training time because, in our proposed method, there are additional processes that do not exist in the baseline, i.e., prototype generation, augmentation, injection (concatenation with prompt features), aggregation, and feedback. TARGET may achieve the smallest running time, but It has poor performance with more than 60% margin compared to our proposed method in all datasets.

Table 7.8: Training time of the consolidated algorithm in CIFARR100, MiniImagenet and TinyImageNet.

Method	Com.Cost	CIFAR100	MiniImageNet	TinyImagenet
GLFC	61.7 MB	24.3h	36.6h	46.4h
LGA	61.7 MB	23.2h	35.7h	45.9h
TARGET	69.8 MB	2.4h	2.3h	3.4h
Fed-L2P	0.49 MB	5.1h	6.6h	15.3h
Fed-DualP	1.32 MB	11.3h	11.1h	19.5h
PIP-L2P	0.52 MB	7.5h	8.0h	17.8h
PIP-DualP	1.35 MB	13.0h	14.5h	25.5h

7.6 Summary

This chapter proposes a new approach named prompt-based federated learning, new baselines named Fed-L2P and Fed-DualP, and a novel method named prototype-injected prompt (PIP) for the FCIL problem. PIP consists of three main ideas: a) prototype injection on prompt, b) prototype augmentation, and c) weighted Gaussian aggregation on the server side. Our experimental result shows that the proposed method outperforms the current SOTAs with a significant gap (up to 33%) in CIFAR100, MiniImageNet, and TinyImageNet datasets. Our extensive analysis demonstrates the robustness of our proposed method in different task sizes, smaller participating local clients, and smaller global rounds. Our proposed method has the same complexity as the baseline method and experimentally requires shorter training time than the current SOTAs. In practice, our proposed method can be applied in both cross-silo and cross-domain federated class incremental learning.

Our future work is directed to federated few-shot class incremental learning, where each client only holds a few samples for each task.

The next chapter is devoted to address federated few-shot class incremental learning problems to handle catastrophic forgetting in a distributed system under data privacy and data scarcity constraints. Similar to FCIL, the problem simulates collaborative agents that conduct continual learning in their respective environments without sharing their data, then aggregate the local models into an optimal global model. However, the problem is more challenging than FCIL since the local clients carry only a few samples for each class.

Chapter 8

FFSCIL: Federated Few-Shot Class Incremental Learning

In the previous chapter, PIP demonstrates its effectiveness in addressing federated class incremental learning, but it has not been proven yet in a data scarcity constraint where each class holds only a few samples. On the other hand, FLOWER proves its capability in addressing few-shot class incremental learning, but it has not been proven yet in a data privacy-constrained distributed system where each client holds only partial data. This chapter proposes a new challenging yet practical Federated Few-Shot Class-Incremental Learning (FFSCIL) problem, where continual learning is conducted by collaborative clients that only hold very few samples for new classes. In addition, this chapter proposes a novel Unified Optimized Prototype Prompt (UOPP) model to simultaneously handle catastrophic forgetting, over-fitting, and prototype bias in FFSCIL. UOPP utilizes task-wise prompt learning to mitigate task interference and over-fitting, unified static-dynamic prototypes to achieve a stability-plasticity balance, and adaptive dual heads for enhanced inferences. Dynamic prototypes represent new classes in the current few-shot task and are rectified to deal with prototype bias. Our comprehensive experimental results show that UOPP significantly outperforms state-of-the-art (SOTA) methods on three datasets in terms of both average accuracy and harmonic mean accuracy. The extensive analysis shows UOPP robustness in various numbers of local clients and global rounds, low communication costs, and

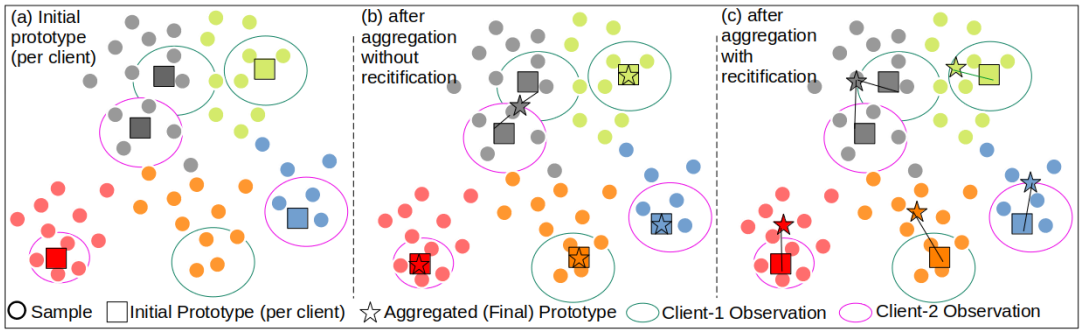


Figure 8.1: The importance of prototype rectification to handle prototype bias (a) Initial prototype per client (b) Aggregation without rectification can not handle prototype-bias (c) Aggregation with rectification overcomes prototype-bias.

moderate running time.

Part of this chapter is published as: MA Ma'sum, M Pratama, L Liu, H Habibullah, R Kowalczyk. Federated Few-Shot Class-Incremental Learning. The Thirteenth International Conference on Learning Representations (ICLR 2025). https://openreview.net/forum?id=ZiPoAlKf9Y **[CORE A*]**

8.1 Introduction

The previous studies on Federated Class Incremental Learning (FCIL) address catastrophic forgetting challenges in a dynamic environment with data privacy constraints. Coordinated by a central server, a collection of clients continually develops a global recognition model without sharing their local data. The first issue of FCIL is that the existing works, i.e., LGA [59], TARGET [63], and LANDER [64], assume the clients carry abundant training data, which is impractical in resource-constrained environments. They are data-hungry, such that they face the issues of prototype bias and over-fitting in the realm of the data scarcity constraint. As in stand-alone few-shot learning [238], federated learning with few samples leads to prototype bias problems. Figure 8.1 visualizes how prototypes of the observed classes are generated and aggregated from the few samples carried by the clients. In FCIL simulation [47], [59], where a client holds only partial classes, e.g., 60% of the total classes, it shows that the observed few samples by a client

lead to prototype bias, where a prototype does not represent the true population, but rather represents the few locally gathered samples. To overcome the prototype bias problem, each prototype should be refined to the correct location. As shown in Figure 8.1, we also emphasize that aggregating the prototype can not handle the prototype bias. Few-shot learning (FSL), e.g., MetaNode [238] or few-shot class incremental learning (FSCIL), e.g., S3C [239] methods can not be expected since the methods require the presence of all classes, whereas, in a federated setting, a client carries only a subset of all classes.

Second, the current FCIL methods train and share whole backbone parameters, resulting in a large number of parameters during optimization processes, which implies a long training time and a high communication cost. Third, in the current SOTAs, clients generate and share synthetic images for global model aggregation on the server side. Aside from overloading the communication costs, this mechanism may violate data privacy principles since synthetic data may reveal partial information about the private data of a client to another party due to its similarity to the original data. Last but not least, some FCIL methods, e.g., LGA [59] GLFC [47], save several exemplars from previous tasks for rehearsal that may breach data openness policy, where data are only open for a client at a specific moment.

These gaps motivate us to address a new direction of FCIL, i.e., Federated Few-Shot Class Incremental Learning (FFSCIL), where a client participating in the federated learning process only possesses very few samples. Second, we develop a novel, efficient, but effective approach to the FFSCIL problem, where a client trains and shares as few parameters as possible but produces a highly accurate global model without sharing any synthetic samples or saving exemplars from previous samples. Therefore, the proposed method handles catastrophic forgetting in dynamic collaborative learning with data privacy and data scarcity constraints. **The contributions of this chapter are**:

1. This chapter emphasizes the data scarcity issue leading to the prototype-bias and over-fitting problems in FCIL and defines a new problem, namely Federated Few-Shot Class Incremental Learning (FFSCIL);
2. This chapter proposes a **novel** method for the FFSCIL problem termed

Unified Optimized Prototype Prompt (UOPP) built upon a prompt learning framework coupled with static and dynamic prototypes optimized by Neural Ordinary Differential Equation (ODE). The proposed method utilizes an adaptive dual-head to enhance its predictive accuracy. To our knowledge, UOPP is the **first** prompt-based approach that integrates prompt tuning, prototype rectification by a trainable network, and adaptive dual classifiers in a single pipeline;

3. This chapter offers theoretical studies for the convergence and generalization of the proposed method;

4. This chapter provides a comprehensive analysis in three benchmark datasets that show the proposed method outperforms the baseline and current SOTAs with significant gaps and achieves improved stability-plasticity balances. The analysis emphasizes the robustness of the proposed method in various participating clients and small rounds per task.

8.2 Problem Formulation

8.2.1 Federated Few-Shot Class-Incremental Learning (FFSCIL)

FFSCIL is defined as: Given a sequence of tasks $[0, 1, 2, ..., T]$ where each task t carries a labeled training set $\mathcal{T}^t = \{(x_i^t, y_i^t)\}_{i=1}^{|\mathcal{T}^t|}$, where $x_i^t \in X$ denotes an input image and $y_i^t \in Y$ denotes its label, and $|.|$ denotes the cardinality. Each task t is disjoint with another task t'., i.e., $\forall_{t,t'} \mathcal{T}^t \cap \mathcal{T}^{t'} = \emptyset$. On each task-$t$, a set of clients $\{l\}_{l=1}^{L_{all}}$ coordinated by a central server G are deployed to learn $\mathcal{T}^t$. In the first task ($t = 0$), each client l carries abundant training samples while the remaining tasks ($t > 0$), it carries far smaller samples than the first task i.e $|\mathcal{T}_l^0| >> |\mathcal{T}_l^{t>0}|$. For a convenient way, task-0 is called the base task, while the rest are called the few-shot task (FS task). A client l holds only a subset of the current task training set, i.e., $\mathcal{T}_l^t \subset \mathcal{T}^t$. Each client-$l$ carries non-identically and distributed data (non-i.i.d) to another client l', i.e., $\forall_{l,l'} \mathcal{D}_l^t \neq \mathcal{D}_{l'}^t$ where $\mathcal{D}_l^t$ and $\mathcal{D}_{l'}^t$ are distribution of $\mathcal{T}_l^t$ and $\mathcal{T}_{l'}^t$ respectively. Following FCIL, the non-i.i.d. distribution is represented

by the percentage of available classes η. Each task-t is learned in a federated way that is repeated in R_T rounds where in each round $r \in [1..R_T]$, a set of local clients is randomly selected from all available clients, i.e., $\{l\}_{l=1}^{L} \subset \{l\}_{l=1}^{L_{all}}$. Due to data privacy constraints, a client-l is not allowed to share any training sample $(x_i, y_i) \in \mathcal{T}_l^t$ to another client or server, but permitted to share its parameters.

8.2.2 FFSCIL Objectives

Let a deep neural network $g_\Phi(f_\Theta(.))$ be parameterized by Θ and Φ where $f(.)$ and $g(.)$ are the feature extractor and classifier respectively. In each round r of task t, a central server G coordinates selected local clients $\{l\}_{l=1}^{L}$ to conduct local CIL training using its training samples $\{\mathcal{T}_l^t\}$. Each client-l optimizes its local parameters (Θ_l, Φ_l), then sends its locally optimized parameters to the central server G to be aggregated. The central server G aggregates all received local parameters into optimum global parameters, i.e., $(\Theta_G, \Phi_G) = Agg(\{(\Theta_G, \Phi_G)\}_{l=1}^{L})$ and communicates them back to all clients for the next round process. The objective of FFSCIL is to achieve an optimum global model $g_{\Phi_G}(f_{\Theta_G}(.))$ to recognize the learned classes from the first task (task-0) until the current task (task-t), i.e., $\{\mathcal{T}^0, ..., \mathcal{T}^t\}$.

8.3 Proposed Method: Unified Optimized Prototype Prompt(UOPP)

8.3.1 Overview

This chapter proposes a new method named Unified Optimized Prototype Prompt (UOPP) to address challenges in FFSCIL, i.e., catastrophic forgetting, over-fitting, and prototype bias simultaneously under data privacy constraints. Figure 8.2 exhibits the flow of UOPP in both the local training view (a) and federated training view (b). Looking at Figure 8.2 (a), we utilize a prompt-based approach on top of the frozen ViT backbone as it minimizes task interference that leads to better handling of catastrophic forgetting with lightweight learnable parameters (prompts). Then we add a rectification block to handle prototype bias in the few-shot tasks $(t > 0)$ by iteratively rectifying the prototypes. Note that each client only holds a few samples for any class in the few-shot tasks. Then we add a unification block

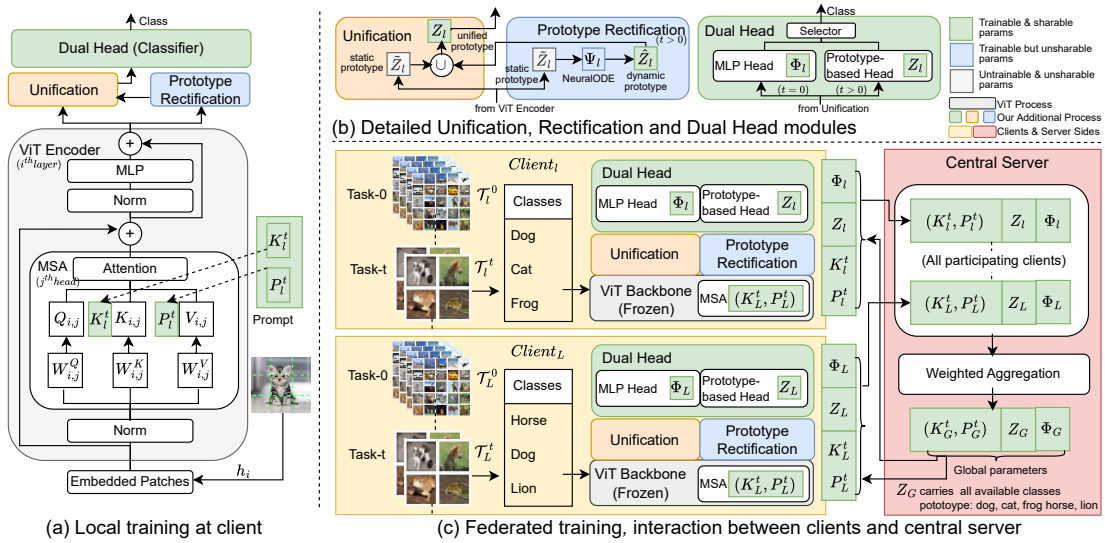


Figure 8.2: The visualization of UOPP, includes task-wise prompt learning empowered by shared unified static-dynamic prototypes, dynamic prototypes rectification, adaptive dual heads, and weighted aggregation. The gray-colored parameters are frozen and unshareable parameters, the green-colored parameters are trainable and shareable parameters, while the blue-colored parameters are trainable but unshareable parameters.

to unify the rectified prototypes and the feature produced by the ViT encoder. Since the prototypes are shareable to/from the server, this mechanism handles a non-i.i.d challenge where a client-l is accommodated to learn knowledge representation from the classes not available in $\mathcal{T}_l^t$. Last, we design an adaptive dual head that leverages the strength of both MLP and prototype-based classifiers. On the federated view (b), the Figure shows that in UOPP, a client shares only small-sized parameters, i.e., prompts, prototypes, and MLP parameters rather than the whole backbone parameters. Thus, it minimizes the communication cost between clients and the central server. The details of UOPP are presented in the following subsections. The **uniqueness** of UOPP from the existing prototype-based methods, i.e., [118], [44], [45], [240], and [154] is that we utilize a trainable Neural ODE that works by support and query samples drawn from different distribution, while the existing method utilizes similarity ratio for rectification process. Second, alongside prototype rectification, we adjust task-wise key and prompt parameters to improve inter-task separability. Different from PILoRA [154], the prompt is prepended to

$W^K x$ and $W^V x$ of ViT model, while PILoRA appends matrices $A.B$ into W^Q and W^V and does not use task-wise keys. Third, we utilize dual-head classifiers to leverage the strength of both combined with task prediction.

8.3.2 Minimizing Task Interference via Prompt Learning

UOPP define a prompt $\mathbf{P}_l$ for each client-l to learn a sequence of task $\{t\}_{t=0}^{T}$ as: $\mathbf{P}_l = [(K_l^0, P_l^0), (K_l^t, P_l^t), .., (K_l^T, P_l^T)], t \in [0..T]$ where (K_l^t, P_l^t) is a pair of prompt key-and-value corresponded to task-t. UOPP utilizes the prefix tuning technique [83] because it usually outperforms the prompt tuning method. Following the definition of prefix tuning, and the attention mechanism in ViT, the output of a ViT backbone given an input x and prompt (K_l^t, P_l^t) is defined as:

$$f_{(K_l^t, P_l^t)}(x) = A(Q_{ij}, [K_l^t + K_{i,j}], [P_l^t + V_{i,j}]) \tag{8.1}$$

where $Q_{i,j}, K_{i,j}, V_{i,j}$ are query, key and value of jth head MSA in i^{th} layer of ViT encoder, $+$ denotes concatenation, and A denotes attention function. Note that the function is applied for all MSA heads $j \in [1..J]$ and all encoder layers $i \in [1..I]$. At task-t, a client-l optimizes only (K_l^t, P_l^t) and not the other prompt key-value pair, and (K_l^t, P_l^t) is adjusted only in task t, not in the previous or upcoming tasks. Therefore, once optimal to $\mathcal{T}_l^t$, the pair (K_l^t, P_l^t) will remain robust against forgetting since its value is not adjusted afterward. In addition, K_l^t is adjusted to match sample $x_i \in \mathcal{T}_l^t$ by using a matching loss $\mathcal{L}_m$ during training. This mechanism is designed to make (K_l^t, P_l^t) exclusive to task-t and not the samples from other tasks. In other words, it minimizes task interference. In addition, since only prompts rather than the whole network parameters are adjusted, this strategy alleviates the over-fitting problem due to the few samples.

8.3.3 Unified Static-Dynamic Prototype and Its Usability

8.3.3.1 Static Prototype

UOPP defines a vector of D-dimension $\tilde{z}_c$ as the prototype for class $c \in \mathcal{T}^t$ that is produced from ViT encoder, where D is the embedding dimension. A static prototype set $\tilde{Z}_l = \{\tilde{z}_{lc}\}$ is a collection of static prototypes of class-c that are available at client-l. Assuming that a prototype follows a Gaussian distribution,

i.e., $z_c \sim \mathcal{N}(\mu_c, \Sigma_c)$ and forms D disjoint uni-variate distribution, then a prototype of class c is represented as $z_c \sim \mathcal{N}(\mu_c, {\sigma_c}^2)$ where ${\sigma_c}^2 = I_D.{\sigma_{c,i}}^2, i \in \{1, 2, ..., D\}$, I_D is identity matrix. Note that at task-t, a selected local client-l holds its local training set $\mathcal{T}_l^t$. Suppose that $\mathcal{T}_{lc}^t = \{(x_{li}^t, y_{li}^t) \in \mathcal{T}_l^t, y_{li}^t = c\}$ is the samples of class-c in $\mathcal{T}_l^t$ and $|.|$ denotes the number of samples. Then, a static prototype $\tilde{z}_{lc} \sim \mathcal{N}(\mu_{lc}, {\sigma_{lc}}^2)$ for a class-c available in $\mathcal{T}_l^t$ is computed by Eq. 8.2 and 8.3.

$$\mu_{lc} = \frac{1}{|\mathcal{T}_{lc}^t|} \sum_{i=1}^{|\mathcal{T}_{lc}^t|} f_{(K_l^t, P_l^t)}(x_i) \, , x_i \in \mathcal{T}_{lc}^t \tag{8.2}$$

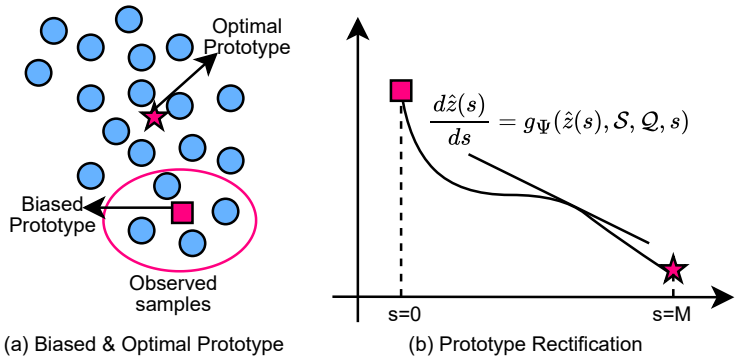
$${\sigma_{lc}}^2 = \frac{1}{|\mathcal{T}_{lc}^t|} \sum_{i=1}^{|\mathcal{T}_{lc}^t|} (\mu_{lc}^t - f_{(K_l^t, P_l^t)}(x_i))^2 \, , x_i \in \mathcal{T}_{lc}^t \tag{8.3}$$

8.3.3.2 Dynamic Prototype

UOPP defines a vector $\hat{z}_c$ of D dimension as the prototype for class $c \in \mathcal{T}^{t>0}$ iteratively rectified during local training on task-t, where D is the embedding dimension. A dynamic prototype set $\hat{Z}_l = \{\hat{z}_{lc}\}$ is a collection of dynamic prototypes of class-c available at client-l. At the beginning of local training at task $t > 0$ on client-l, a dynamic prototype $\hat{z}_{lc}$ is set by its corresponding static prototype value $\tilde{z}_{lc}$, then iteratively rectified by GradNet g_{Ψ_l} during local training, i.e., $\hat{z}_{lc} = Rectification(g_{\Psi_l}, \tilde{z}_{lc})$. After finishing the training of task-t, the dynamic prototypes of all classes $c \in \mathcal{T}^t$ are not updated anymore and are stored as static prototypes. Therefore, the dynamic prototype set $\hat{Z}_l$ includes prototypes from currently learned classes only, while the static prototype set $\tilde{Z}_l$ includes both currently learned classes and previously learned classes. The details of the rectification process are explained in the following sub-section.

8.3.3.3 Unified Prototype

UOPP defines the unified prototype $Z_l = \tilde{Z}_l \cup \hat{Z}$ as the union of the static and dynamic prototype of client-l. Following the definition of static and dynamic prototypes, Z_l contains static prototypes of previously learned classes and dynamic prototypes of currently learned classes that are still updated during the local training at client-l. Unified prototype Z_l plays an important role in UOPP, both in the base task ($t = 0$) and the few-shot task ($t > 0$). As illustrated by the unification



(a) Biased & Optimal Prototype (b) Prototype Rectification

Figure 8.3: Visualization of Biased and Optimal Prototype (a), and Rectification (b).

block in Figure 8.2, in the base task, Z_l is unified with the output of ViT, i.e., $f_{(K_l^t, P_l^t)}(x)$. Since Z_l is shareable, then Z_l contains prototypes of classes unavailable in $\mathcal{T}_l^t$ shared by the central server. Therefore, each client-l can afford to learn all classes in $\mathcal{T}^t$. In the few-shot task, each client-l rectifies $\hat{z}_{lc}$ for all classes $c \in \mathcal{T}_l^t$. Similarly, the shareable Z_l enhances the separability of $\hat{z}_{lc}$ since it is contrasted to all other prototypes of all learned classes.

8.3.4 Handling Prototype-Bias via Dynamic Prototype Rectification

Previous studies [238], [241] emphasize that generations of prototypes by averaging a few samples leads to prototype bias, as the prototypes only represent the observed samples but not the whole population, as illustrated in Figure 8.3(a). Therefore, UOPP rectifies a prototype $\hat{z}_c$ to become as close as possible to the population mean. We follow the framework of episodic training in few-shot learning [175], [238], a rectification process of a prototype $\hat{z}_c$ can be defined as: $\hat{z}_c(M) = Rectification(g_\Psi(), \hat{z}_c(0), \mathcal{S}, \mathcal{Q}, s = M)$ where s is the number of rectification steps, $\hat{z}_c(0)$ and $\hat{z}_c(M)$ are the initial prototype and final prototype after M-steps rectification process, $\mathcal{S}$ is the support set that contains few observed samples per class, $\mathcal{Q}$ is the query set comprising unlabeled samples, and $g_\Psi()$ is gradient networks (GradNet) parameterized by Ψ. Now looking into a more detailed view, suppose that $L(\hat{z}_c(s))$ as a differentiable loss function with prototype $\hat{z}_c(s)$ and $\nabla L(\hat{z}_c(s))$ as its gradient during the optimization process, one step pro-

totype rectification can be defined as an iterative process of Gradient Descent algorithm as described in equation 8.4. Symbol α denotes the learning rate, while ω denotes l_2-norm regularizer.

$$\hat{z}_c(s+1) = \hat{z}_c(s) - \alpha(\nabla L(\hat{z}_c(s)) + \omega(\hat{z}_c(s)) \tag{8.4}$$

The previous study [241] discovered that the iterative process above can be viewed as an Euler discretization of an Ordinary Differential Equation (ODE). Therefore, the term $\nabla L(\hat{z}_c(s))$ in the process above can be derived into equation 8.5, where s represents a continuous variable such as time, and $\frac{d\hat{z}_c(s)}{ds}$ represents a continuous gradient flow of prototype $\hat{z}_c^t(s)$ over s. Since the optimization process is executed by a neural network model $g_\Psi(.)$ (last part of equation 8.5), then ODE becomes Neural ODE.

$$-\nabla L(\hat{z}_c(s)) = \frac{d\hat{z}_c(s)}{ds} = g_{\Psi_l}((\hat{z}_c(s), \mathcal{S}, \mathcal{Q}, s)) \tag{8.5}$$

We follow the implementation of GradNet [238] as the neural network model $g_\psi(.)$ that executes the rectification process. Following the implementation on GradNet, the optimum prototype is produced in the last step of rectification, i.e., $\hat{z}_c(M) = \hat{z}_c(0) + \int_{s=0}^{M} g_{\Psi_l}((\hat{z}_c(0), \mathcal{S}, \mathcal{Q}, s))$. We follow the implementation of the previous study [238], [241], where the last term is solved by ODESolver based on Runge-Kutta method [242]. Therefore, the optimum prototype $\hat{z}_c(M)$ can be obtained by executing equation 8.6.

$$\hat{z}_c(M) = ODESolver(g_{\Psi_l}, \hat{z}_c(0), \mathcal{S}, \mathcal{Q}, s = M) \tag{8.6}$$

In previous studies [238], [241], $\mathcal{Q}$ contains images from base task. Now dealing with limitations in FFSCIL, we can not afford to save exemplars from previous tasks or gather images from other clients. However, we have a shareable unified prototype $Z_l = \tilde{Z}_l \cup \hat{Z}_l$ containing all learned class prototypes. Therefore, the **uniqueness** of our rectification is that we generate support set $\mathcal{S}$ is constructed by drawing prototypes from $\mathcal{N}(\mu_{lc}, {\sigma_{lc}}^2)$ where $(\mu_{lc}, {\sigma_{lc}}^2)$ is computed by eq. 8.2-8.3 for all class c available in $\mathcal{T}_l^t$, and generate query set $\mathcal{Q}$ by drawing prototypes from $\mathcal{N}(\hat{\mu}_l, \hat{\sigma}_l^2)$ where $(\hat{\mu}_l, \hat{\sigma}_l^2)$ is the property of $\hat{z}_l \in \hat{Z}_l$. As an implications, our rectification occurs fully in the embedding space and pseudo-rehearsal free. Note that Z_l is assigned by aggregated prototypes, i.e., $Z_l = Z_G$ in each federated round.

8.3.5 Adaptive Dual-Head Classifier for A Better Prediction

We design an adaptive dual head classifier by leveraging MLP classifier $g_\Phi(.)$ that works effectively in the highly labeled task ($\mathcal{T}^0$) and prototype-based (PB) classifier $g_Z(.)$ that has been proven robust in few shot tasks ($\mathcal{T}^{t>0}$). We combine both MLP and PB classifiers as one united head layer. MLP parameter Φ is optimized in the base task, while the unified prototype set Z is optimized in all few-shot tasks. In the testing phase, we deploy a head selector to select which classifier to use for an input x. The head selector works by predicting the task ID based on the input-key matching. Given an input image x, and a sequence of prompt-key $K^0, K^1, ..., K^T$, we can find the task-id where x belong, by finding the highest similarity of x and $K^t, t \in [0, ..T]$. The predicted label $\hat{y}$ of an input x is computed by equation 8.7.

$$\hat{y} = \begin{cases} g_\Phi(f_{K^{(0},P^0)}(x)), & \text{if } \hat{t} = 0 \\ g_Z(f_{(K^{\hat{t}},P^{\hat{t}})}(x), & \text{otherwise,} \end{cases}, where,\ \hat{t} = arg\min_t \mathcal{L}_m(x, K^t) \tag{8.7}$$

where $\mathcal{L}_m$ is matching loss between input x and prompt-key K^t. Note that equation 8.7 is applicable both for the client and server side. The united classifier is expected to elevate the prediction results, rather than employing one of the FC or PB classifiers alone. This hypothesis will be proven in our ablation study.

8.3.6 Federated Learning and Server-Side Aggregation

Figure 8.2 shows that on a task-t, each client-l optimizes its local parameters based on its local training data $\mathcal{T}_l^t$. Then the optimized parameters, i.e., $((K_l^t, P_l^t), \Phi_l, Z_l)$ to the central server G are aggregated. We propose a simple weighted aggregation applying the principle "the more you learn, the better your knowledge". We consider the total participation of a client S_l on the task t as the client's weight w_l. In a round r of a task t, given clients that carry locally optimized parameter $\{((K_l^t, P_l^t), \Phi_l, Z_l)\}_{l=1}^L$ and their weights $\{w_l\}_{l=1}^L$, then the aggregated parameter $((K_G^t, P_G^t), \Phi_l, Z_G)$ is computed using equation 8.8.

$$((K_G^t, P_G^t), \Phi_G, Z_G) = \frac{1}{\sum_{l=1}^L w_l} \sum_{l=1}^{L} ((K_l^t, P_l^t), \Phi_l, Z_l).w_l \tag{8.8}$$

In the base task ($t = 0$), the server aggregates and distributes prompt parameter (K_G^t, P_G^t), MLP head parameter Φ_G, and unified prototype set Z_G^t to all parti-

cipating clients. while in the few-shot tasks ($t > 0$), the server aggregates and distributes prompt parameters and the unified prototype set only. The GradNet parameter Ψ_l is unshared and utilized only for local prototype rectification. Rather, a client sends the rectified dynamic prototype $\hat{Z}_l \subseteq Z_l$ to the central server.

8.3.7 Wrap Up and Final Objective

In the base task, each client-l unifies $f_{(K_l^0, P_l^0)}(x)$ with Z_l then updates prompt and MLP head parameters, i.e., (K_l^0, P_l^0) and Φ_l. In the few-shot tasks, client-l performs $f_{(K_l^t, P_l^t)}(x)$ to form static prorotype $\tilde{Z}_{lc}$, then construct support sample ($\mathcal{S}$) and query samples ($\mathcal{Q}$), and perform rectification for all $\hat{z}_{lc} \in \hat{Z}_{lc}$. Afterward, it updates prompt parameters (K_l^t, P_l^t) and GradNet parameter Ψ_l. Therefore, the objectives of local training are defined in the equations below:

(i). Base Task ($t = 0$) objective :

$$\mathcal{L}_{t=0} = \mathcal{L}_{ce}(g_{\Phi_l}(f_{(K_l^0, P_l^0)}(x_i) \cup Z_l), y_i \cup \mathcal{C}_l) + \lambda \mathcal{L}_m(x_i, K_l^0), (x_i, y_i) \in \mathcal{T}_l^0 \tag{8.9}$$

(ii). Few Shot Tasks ($t \geq 1$) objective:

$$\mathcal{L}_{t>0} = \mathcal{L}_{pce}(g_{Z_l}(g_{\Psi_l}(\hat{Z}_{lc}(s), \mathcal{S}, \mathcal{Q}, s), \mathcal{C}_l) + \lambda \mathcal{L}_m(x_i, K_l^t), (x_i, y_i) \in \mathcal{T}_l^t \tag{8.10}$$

where $\mathcal{L}_{ce}$ denotes cross-entropy loss, $\mathcal{L}_{pce}$ denotes prototype cross-entropy loss utilizing cosine distance as similarity measurement, $\mathcal{L}_m$ denotes matching loss, and $\mathcal{C}_l$ is the label set of the unified prototype set Z_l, $(f_{(K_l^t, P_l^t)})$ denotes prompting output as in eq. 8.1, and g_{Ψ_l} denotes rectification function as in eq. 8.5.

8.3.8 Algorithm

This subsection presents the detailed algorithm of UOPP as shown in algorithm 7.

8.4 Theoretical Analysis

Let $\Theta = (P, \Phi, \Psi)$ be trainable parameters, $F(\Theta) = \mathbb{E}[\mathcal{L}(\mathcal{T}; \Theta)] = \mathbb{E}[\mathcal{L}(\mathcal{T}; (P, \Phi, \Psi))]$ is the expected loss function, k, E, R, and L_S is local iteration, local epoch, global round, and number of selected local clients respectively. We follow L-smooth and

Algorithm 7 UOPP

Input: Number of clients L_{all}, number of selected local clients L, total number of rounds R, number of task $T+1$, local epochs E, batch size B.
Distribute frozen ViT backbone f to all clients $\{l\}_{l=1}^{L_{all}}$ and central server G
Initiate prompt, key, and head layer for all clients and central server $P_G = P_l$, $\Phi_G = \Phi_l$, $\Psi_l = init(), l \in \{1..L_{all}\}$
$R_T \leftarrow R/(T+1)$, R_T represents round per task
Init global and local unified prototypes $Z_G = Z_l, = Z = \emptyset$
for $t = 0 : T$ **do**
  for $r = 1 : R_T$ **do**
    $\{l\}_{l=1}^{L} \leftarrow$ randomly select L local clients from L_{all} total clients
    Clients execute:
    if $R_T = 1$ **then**
      Compute static prototype $\tilde{Z}_l$ as in Eq. (8.3)-(8.4), then send it to server
    end if
    Receive global parameters, i.e., prompt, FC layer, and prototypes set P_G, Φ_G, and Z_G
    Assign local parameters $(P_l, \Phi_l, Z_l) \leftarrow (P_G, \Phi_G, Z_G)$
    $\mathcal{B} \leftarrow$ Split $\mathcal{T}_l^t$ into B sized batches
    for $e = 1 : E$ **do**
      for $b = 1 : \mathcal{B}$ **do**
        if $(t = 0)$ **then** // Base Task Update
          Compute prompt-generated feature $f_{(K_l^t, P_l^t)}(x)$ as in Eq. (2)
          Compute logits with FC classifier $g_{\Phi_l}(f_{(K_l^t, P_l^t)}(x) \cup Z_G)$
          Compute loss $\mathcal{L}_{t=0}$ as in Eq. (8.10)
          Update local parameters (K_l^t, P_l^t, Φ_l) based on $\mathcal{L}_{t=0}$
        else $(t > 0)$ // Few-shot Task Update
          Compute static prototype $\tilde{Z}_l$ using feature $f_{(K_l^t, P_l^t)}(x)$ as in Eq. (8.2)
          Draw $\mathcal{S}$ from $\tilde{Z}_l$ and draw $\mathcal{Q}$ from $Z_l = Z_G$
          Rectify dynamic prototype $\hat{Z}_l$ using $g_\Psi(.)$as in Eq. (8.5) to (8.7)
          Form unified prototype $Z_l = Z_G \cup \hat{Z}_l$
          Compute logits with PB classifier $g_{Z_l}(f_{(K_l^t, P_l^t)}(x) \cup \mathcal{S})$
          Compute loss $\mathcal{L}_{t>0}$ as in Eq. (8.11)
          Update local parameters (K_l^t, P_l^t, Ψ_l) based on $\mathcal{L}_{t>0}$
        end if
      end for
      if $t = 0$ **then**
        Update local static prototype $\tilde{Z}_l$ as Eq. (8.3)-(8.4) for all class $c \in \mathcal{T}_l^t$
      end if
    end for
    if $t = 0$ **then**
      Set unified prototype $Z_l = \tilde{Z}_G \cup \tilde{Z}_l$
    else
      Set unified prototype $Z_l = \tilde{Z}_G \cup \hat{Z}_l$
    end if
    Store local parameters $(K_l^t, P_l^t, \Phi_l, \Psi_l, Z_l)$
    Compute clients' weight ω_l
    Send local parameters $(K_l^t, P_l^t, \Phi_l, Z_l)$ and weight ω_l^t to server
    Server executes:
    if $R_T = 1$ **then**
      Receive clients initial static prototype $\tilde{Z}_l$ for $l \in [1..L]$
      Generate $Z_G = Z_G \cup Agg(\tilde{Z}_l$ for $l \in [1..L])$ and send Z_G to clients
    end if
    Receives selected clients parameters $(K_l^t, P_l^t, \Phi_l, Z_l)$ and weight ω_l for $l \in [1..L]$
    Do weighted aggregation as in Eq. (8.9)
    Send global parameters $(K_G^t, P_G^t, \Phi_G, Z_G)$ to clients for the next round
  end for
end for
Output: Optimal Global parameters $(\mathbf{P}_G, \Phi_G, Z_G)$

μ-strongly convex F, G-bounded uniformly gradient assumptions, random uniformly distributed batches, and decreasing learning rate as in [230],[231] We state the following theorems.

Theorem 1: $\liminf_{k\to\infty} \mathbb{E}[||\nabla F(\Theta_k)||_2^2] = 0$

Theorem 2: Let $\Theta^1, \Theta^R, \Theta^*$ is the initial, last updated (R-th), and optimum parameter respectively, F^* is minimum of F, exist $A, B, C, \delta > 0$ so that: $\mathbb{E}[F(\Theta^R)] - F^* \leq \frac{A}{R}(\frac{2(B+C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\Theta^1 - \Theta^*||)$.

Theorem 3: Given Θ^* and Θ are optimal parameter in $\mathcal{T}_l^t \cup Z$ and $\mathcal{T}_l^t$ respectively, where $\mathcal{T}_l^t \subset \mathcal{T}^t$, where $|\mathcal{T}_l^t|/|\mathcal{T}^t| = \eta \in (0, 1)$, then at least there's $\epsilon > 0$ that satisfy $F(\Theta; \mathcal{T}^t) - F(\Theta^*; \mathcal{T}^t) \geq \epsilon$. Theorems 1 and 2 prove UOPP local training and federated convergence, respectively, while Theorem 3 proves UOPP generalization.

Let $\Theta = (P, \Phi, \Psi)$ be the trainable parameters, $F(\Theta) = \mathbb{E}[\mathcal{L}(\mathcal{T}; \Theta)] = \mathbb{E}[\mathcal{L}(\mathcal{T}; (P, \Phi, \Psi))]$ is the expected loss function, k, E, R, and L is local iteration, local epoch, global round, and number of selected local clients respectively. Please note that in this analysis, L denotes the number of selected local clients, while $l \geq 1$ denotes a constant for the l-smooth coefficient. Following the update rule in section 4.3, the expression of $F(\Theta)$ above can be detailed as follows:

(i) Base Task ($t = 0$): $\Theta = (P, \Phi)$, and $F(\Theta) = \mathbb{E}[\mathcal{L}(\mathcal{T}; \Theta)] = \mathbb{E}[\mathcal{L}_{l+}(\mathcal{T}; (P, \Phi))]$ as local clients update (P, Φ) using $\mathcal{L}_{l+}$ following equations 7 and 8.

(i) FS Task ($t \geq 1$): $\Theta = (P, \Psi)$, and $F(\Theta) = \mathbb{E}[\mathcal{L}(\mathcal{T}; \Theta)] = \mathbb{E}[\mathcal{L}_{lfs+}(\mathcal{T}; (P, \Psi))]$ as local clients update (P, Ψ) using $\mathcal{L}_{lfs+}$ following equations 9 and 10.

We adopt the SGD optimization convergence analysis [231] and FedAvg convergence analysis [230] assumptions as follows:

Assumption 1: $F_1, ...F_l, ..., F_{L_S}$ are all $L-$smooth: for all Θ and Θ', $F_l(\Theta) \leq F_l(\Theta') + (\Theta - \Theta')^T \nabla F_l(\Theta) + \frac{L}{2}||\Theta - \Theta'||_2^2$.

Assumption 2: $F_1, ...F_l, ..., F_{L_S}$ are all $\mu-$strongly convex: for all Θ and Θ', $F_l(\Theta) \leq F_l(\Theta') + (\Theta - \Theta')^T \nabla F_l(\Theta) + \frac{\mu}{2}||\Theta - \Theta'||_2^2$.

Assumption 3: Let ξ_l^k be the random uniformly sampled from l-th local data at

$k-th$ iteration . The variance of stochastic gradients in each client is bounded by the following criteria: $\mathbb{E}||\nabla F_l(\Theta_l^k, \xi_l^k) - \nabla F_l(\Theta_l^k)|| \leq \sigma_l^2$ for $l = 1, 2, ..., L_S$

Assumption 4: The expected squared norm of stochastic gradients in each client is bounded by: $\mathbb{E}||\nabla F_l(\Theta_l^k, \xi_l^k)|| \leq G^2$ for all $l = 1, 2, ..., L_S$, and $k = 1, 2, ..., K$ where $K \in \mathbb{N}$.

Assumption 5: $\sum_{k=1}^{\infty} \alpha_l^k = \infty$ and $\sum_{k=1}^{\infty} \alpha_l^{k^2} < \infty$ where α_l^k is the learning rate of $l-th$ client in k-th step training.

We follow the theoretical analysis in the federated class incremental learning method [243] as it has a similar characteristic, i.e., a prompt-based method supported by shared prototypes.

Proof of Theorem 1: Let a client-l be trained locally with its local data $\mathcal{T}_l^t \cup Z$, where $\mathcal{T}_l^t$ is local;y observed training samples for t-th task and $Z = Z_l = Z_G$ is aggregated unified prototype for task t shared by server respectively. We assume that Z is augmented so that $|z_{c_b}| \approx |x_{c_a}|$ for $z_{c_b} \in Z$ and $x_{c_a}^t \in \mathcal{T}_l^t \subseteq \mathcal{T}^t$. As an implication, the number of prototypes of unavailable classes in $\mathcal{T}_l^t$ and the samples of available classes in $\mathcal{T}_l^t$ are balanced. Then the local model $\Theta_l = (\mathbf{P}_l, \Phi_l)$ or $\Theta_l = (\mathbf{P}_l, \Psi_l)$ is updated in K iterations based on minibatches drawn from $\mathcal{T}_l^t \cup Z$. Since the feature extractor parameters are frozen, and $\mathcal{T}_l^t \cup Z$ has balanced samples for all classes, then ξ_l^k approximates ξ^k, which is a sample from $\mathcal{T}^t$. The local model parameters are optimized by a stochastic gradient (SG) approach. Suppose that $g(\Theta_l, \xi_l^k)$ is an SG function, then the parameter update can be simplified as:

$$\Theta_l^{k+1} \leftarrow \Theta_l^k - \alpha_l^k g(\Theta_l^k, \xi_l^k) \tag{8.11}$$

Applying assumption 1, and local parameter updates Θ by iterating stochastic gradient with sample ξ_l^k, then we get:

$$\begin{aligned} F_l(\Theta_l^{k+1}) - F_l(\Theta_l^k) &\leq (\Theta_l^{k+1} - \Theta_l^k)^T \nabla F_l(\Theta_l^k) + \frac{L}{2}||\Theta_l^{k+1} - \Theta_l^k||_2^2 \\ &\leq -\alpha_l^k \nabla F_l(\Theta_l^k)^T g(\Theta_l^k, \xi_l^k) + \alpha_l^{k^2} \frac{L}{2}||g(\Theta_l^k, \xi_l^k)||_2^2 \end{aligned} \tag{8.12}$$

The equation above can be derived into:

$$\begin{aligned}\mathbb{E}_{\xi_l^k}[F_l(\Theta_l^{k+1})] - F_l(\Theta_l^k) \leq & -\alpha_l^k \nabla F_l(\Theta_l^k)^T \mathbb{E}[g(\Theta_l^k, \xi_l^k)] \\ & + {\alpha_l^k}^2 \frac{L}{2} \mathbb{E}_{\xi_l^k}[||g(\Theta_l^k, \xi_l^k)||_2^2]\end{aligned} \tag{8.13}$$

The inequality above shows Θ_l^k optimization by SG method at a step k, and it shows the reduction of F_l (left side) is bounded by the value in the right side involving ∇F_l which is derivative of F_l at Θ_l^k along with $-g(\Theta_l^k, \xi_l^k)$ (first term) and second moment of $g(\Theta_l^k, \xi_l^k)$ (second term). Let $g(\Theta_l^k, \xi_l^k)$ be the unbiased estimator of ∇F_l, then we derive inequality above into:

$$\mathbb{E}_{\xi_l^k}[F_l(\Theta_l^{k+1})] - F_l(\Theta_l^k) \leq -\alpha_l^k \nabla ||F_l(\Theta_l^k)||_2^2 + {\alpha_l^k}^2 \frac{L}{2} \mathbb{E}_{\xi_l^k}[||g(\Theta_l^k, \xi_l^k)||_2^2] \tag{8.14}$$

The inequality above guarantees SGD convergence as long as the stochastic directions and stepsize are chosen. We apply the restriction below to avoid the harm of the second term of the right side in the inequality above,

$$\mathbb{V}[g(\Theta_l^k, \xi_l^k)] = \mathbb{E}[||g(\Theta_l^k, \xi_l^k)||_2^2] - ||\mathbb{E}[g(\Theta_l^k, \xi_l^k)]||_2^2. \tag{8.15}$$

Adopting first and second-moment limit as in [231], then we add the following assumption.

Assumption 6: The objective function F_l and SG satisfy the following conditions.

(a). The sequence of $\{\Theta_l^k\}$ is contained in an open space where F_l is bounded below by a scalar F_{inf}

(b) Exist scalars $\nu_G \geq \nu > 0$ so that for all $k \in \mathbb{N}$ satisfy:

$$\begin{aligned}&\nabla F_l(\Theta_l^k)^T \mathbb{E}_{\xi_l^k}[g(\Theta_l^k, \xi_l^k)] \geq \nu ||\nabla F_l(\Theta_l^k) T||_2^2, and \\ &||\mathbb{E}_{\xi_l^k}[g(\Theta_l^k, \xi_l^k)]||_2 \leq \nu_G ||\nabla F_l(\Theta_l^k)||_2.\end{aligned} \tag{8.16}$$

(c) Exist scalars $m_1 \geq 0$ and $m_2 \geq 0$ so that for all $k \in \mathbb{N}$ satisfy:

$$\mathbb{V}[g(\Theta_l^k, \xi_l^k)] \leq m_1 + m_2 ||\nabla F_l(\Theta_l^k)||_2^2 \tag{8.17}$$

Combining assumption 6 and restriction criteria as presented in equation (8.13), then we have:

$$\begin{aligned}\mathbb{E}_{\xi_l^k}[||g(\Theta_l^k,\xi_l^k)||_2^2] \leq m_1 + m_G||\nabla F_l(\Theta_l^k)||_2^2, with \\ m_G = m_2 + \nu_G^2 \geq \nu^2 > 0\end{aligned} \tag{8.18}$$

Then by substituting $\mathbb{E}_{\xi_l^k}[||g(\Theta_l^k,\xi_l^k)||_2^2]$ from equation (8.18) into equation (8.13), we have:

$$\begin{aligned}\mathbb{E}_{\xi_l^k}[F_l(\Theta_l^{k+1})] - F_l(\Theta_l^k) \leq & -\alpha_l^k \nabla F_l(\Theta_l^k)^T \mathbb{E}[g(\Theta_l^k,\xi_l^k)] \\ & + {\alpha_l^k}^2 \frac{L}{2}(m_1 + m_G||\nabla F_l(\Theta_l^k)||_2^2)\end{aligned} \tag{8.19}$$

Assumption 5 ensures that $\{\alpha_l^k\} \to 0$ is practically achievable by applying a learning rate scheduler (with decay) that reduces the learning rate in each step of local training. Then by choosing $\alpha_l^k L m_G \leq \nu$ and substituting $\nabla F_l(\Theta_l^k)^T \mathbb{E}[g(\Theta_l^k,\xi_l^k)]$ in equation (8.19) with the condition in assumption 6.b, we have

$$\begin{aligned}\mathbb{E}_{\xi_l^k}[F_l(\Theta_l^{k+1})] - F_l(\Theta_l^k) \leq & -\alpha_l^k \nu ||\nabla F_l(\Theta_l^k)||_2^2 \\ & + {\alpha_l^k}^2 \frac{L}{2}(m_1 + m_G||\nabla F_l(\Theta_l^k)||_2^2)\end{aligned} \tag{8.20}$$

Applying expectation into the equation above we get

$$\begin{aligned}\mathbb{E}_{\xi_l^k}[F_l(\Theta_l^{k+1})] - \mathbb{E}[F_l(\Theta_l^k)] \leq & -\alpha_l^k \nu \mathbb{E}[||\nabla F_l(\Theta_l^k)||_2^2] \\ & + {\alpha_l^k}^2 \frac{1}{2}(m_1 + m_G \mathbb{E}[||\nabla F_l(\Theta_l^k)||_2^2]) \\ \mathbb{E}_{\xi_l^k}[F_l(\Theta_l^{k+1})] - \mathbb{E}[F_l(\Theta_l^k)] \leq & -\frac{1}{2}\nu\alpha_l^k \mathbb{E}[||\nabla F_l(\Theta_l^k)||_2^2] \\ & + \frac{1}{2}{\alpha_l^k}^2 L m_1\end{aligned} \tag{8.21}$$

Sum both sides for $k \in \{1, ..., K\}$ we get

$$\begin{aligned}F_{inf} - \mathbb{E}[F(\Theta_l^1)] &\leq \mathbb{E}[F_l(\Theta_l^{K+1})] - \mathbb{E}[F_l(\Theta_l^1)] \\ F_{inf} - \mathbb{E}[F(\Theta_l^1)] &\leq -\frac{1}{2}\nu \sum_{k=1}^{K} \alpha_l^k \mathbb{E}[||\nabla F_l(\Theta_l^k)||_2^2] + \frac{1}{2} L m_1 \sum_{k=1}^{K} {\alpha_l^k}^2\end{aligned} \tag{8.22}$$

Dividing by ν for both sides, then we get

$$\sum_{k=1}^{K} \alpha_l^k \mathbb{E}[||\nabla F_l(\Theta_l^k)||_2^2] \leq \frac{2(\mathbb{E}[F(\Theta_l^1)] - F_{inf})}{\nu} + \frac{L m_1}{\nu} \sum_{k=1}^{K} {\alpha_l^k}^2 \tag{8.23}$$

Applying $\lim_{K\to\infty}$ and assumption 5 to the equation above we get

$$\lim_{K\to\infty}\sum_{k=1}^{K}\alpha_l^k\mathbb{E}[||\nabla F_l(\Theta_l^k)||_2^2] \leq \frac{2(\mathbb{E}[F(\Theta_l^1)]-F_{inf})}{\nu} + \frac{Lm_1}{\nu}\lim_{K\to\infty}\sum_{k=1}^{K}{\alpha_l^k}^2 < \infty \tag{8.24}$$

Dividing both sides with $\sum_{k=1}^{K}\alpha_l^k$, and following assumption 5 where $\lim_{K\to\infty}\sum_{k=1}^{K}\alpha_l^k = \infty$ and $\lim_{K\to\infty}\sum_{k=1}^{K}{\alpha_l^k}^2 < \infty$, then the right side will return 0. Therefore, we have

$$\lim_{K\to\infty}\frac{\sum_{k=1}^{K}\mathbb{E}[\alpha_l^k||\nabla F_l(\Theta_l^k)||_2^2]}{\sum_{k=1}^{K}\alpha_l^k} = 0 \tag{8.25}$$

$$\lim_{K\to\infty}\mathbb{E}[\frac{\sum_{k=1}^{K}\alpha_l^k||\nabla F_l(\Theta_l^k)||_2^2}{\sum_{k=1}^{K}\alpha_l^k}] = 0 \tag{8.26}$$

$$\lim_{k\to\infty}\mathbb{E}[||\nabla F_l(\Theta_l^k)||_2^2] = 0 \tag{8.27}$$

The equation (8.27) proves the convergence for local training in the l-th client, where the gradient of loss F converges to 0 along with the increase of training step/iteration k and the decreasing learning rate α.

Proof of Theorem 2: Let the selected local clients $\{l\}_{l=1}^{l=L_S}$ are conduct local optimization with its local training data $\{\mathcal{T}_l^t \cup Z\}_{l=1}^{l=L_S}$ coordinated by central server G, where $\mathcal{T}_l^t$ is local training sample for client l for task t. Client local update is conducted in k iterations using minibatch sampling on local training data set $\xi_l^k \in \mathcal{T}_l^t$. Global model aggregation is executed in each round $r = \{1, 2, ..., R\}$. We define global aggregation step as $\mathcal{I}_E = \{rE|r = 1, 2, ...R\}$. Following [230], symbol Θ_l^{k+1} denotes the local parameter of client l after communication steps, while φ_l^{k+1} denotes the local parameter after an immediate result of one step of stochastic gradient descent. Then the definition satisfies the following expressions:

$$\varphi_l^{k+1} = \Theta_l^k - \alpha_l^k\nabla F_l(\Theta_l^k, \xi_l^k) \tag{8.28}$$

$$\Theta_l^{k+1} = \begin{cases}\varphi_l^{k+1} \text{ if } k+1 \notin \mathcal{I}_E \\ \sum_{l=1}^{L_S} w_l^k\varphi_l^{k+1} \text{ if } k+1 \in \mathcal{I}_E\end{cases} \tag{8.29}$$

Where $w_l = \omega_l / \sum_{l=1}^{L_S}\omega_l$, where ω_l is the weight of client l. We state $\bar{\varphi}_l^{k+1} = \sum_{l=1}^{L_S} w_l\varphi_l^{k+1}$ and $\bar{\Theta}_l^{k+1} = \sum_{l=1}^{L_S} w_l\Theta_l^{k+1}$, $\bar{\varphi}_l^{k+1}$ is the result of single step of stochastic

gradient descent iteration from $\bar{\Theta}_l^{k+1}$. Then we define $\bar{g}^k = \sum_{l=1}^{L_S} w_l \nabla F_l(\Theta_l^k)$ and $g^k = \sum_{l=1}^{L_S} w_l \nabla F_l(\Theta_l^k, \xi_l^k)$. We adopt the Lemmas below from [230], where their derivation is obtained from fully participating clients in the FL setting.

Lemma 1: By applying assumptions 1 and 2, in one step SGD training and choosing $\alpha \leq \frac{1}{4L}$, we have

$\mathbb{E}[||\bar{\varphi}^{k+1} - \Theta^*||^2] \leq (1 - \alpha^k \mu)\mathbb{E}[||\bar{\Theta}^k - \Theta^*||^2] - (\alpha^k)^2 \mathbb{E}[||g^k - \bar{g}^k||^2] + 6L(\alpha^k)^2\Gamma + 2\mathbb{E}[\sum_{l=1}^{L_S} w_l ||\bar{\Theta}^k - \Theta_l^k||^2]$ where $\Gamma = F^* - \sum_{l=1}^{L_S} w_l F_l^* \geq 0$.

Lemma 2: By applying assumption 3, the gradient function follows:

$\mathbb{E}[||\bar{g}^k - \bar{g}^k||^2] \leq \sum_{l=1}^{L_S} w_l^2 \sigma_l^2$, where σ_l^2 is the variance of Θ_l

Lemma 3: By applying assumption 4, where α^k is non-increasing and it satisfies $\alpha^k \leq \alpha^{k+E}$for all$k \geq 0$, then we have $\mathbb{E}[\sum_{l=1}^{L_S} ||\bar{\Theta}^{k+1} - \Theta_l^k||^2] \leq 4(\alpha^k)^2(E-1)^2G^2$

In FL setting with all clients participating, we get $\bar{\Theta}^{k+1} = \bar{\varphi}^{k+1}$. However, in a setting with only partial clients participating, we utilize a random sampling mechanism that satisfies $\mathbb{E}_{S_L}[\bar{\Theta}^{k+1}] = \bar{\varphi}^{k+1}$. We also adopt the bounding condition from [230] as shown in Lemma 4.

Lemma 4: The expected difference between $\bar{\Theta}^{k+1}$and $\bar{\varphi}^{k+1}$ bounded by : $\mathbb{E}_{S_L}[||\bar{\varphi}^{k+1} - \bar{\Theta}^{k+1}||^2] \leq \frac{4}{L_S}(\alpha^k)^2E^2G^2$ and in the case of w_l is uniform for all l-th client, then $\mathbb{E}_{S_L}[||\bar{\varphi}^{k+1} - \bar{\Theta}^{k+1}||^2] \leq \frac{4(N_S - L_S)}{N_S - 1}(\alpha^k)^2E^2G^2$, where N_S is total clients and L_S is number of selected clients.

Please note that

$$||\bar{\Theta}^{k+1} - \Theta^*||^2 = ||\bar{\Theta}^{k+1} - \Theta^*||^2 \tag{8.30}$$

$$||\bar{\Theta}^{k+1} - \Theta^*||^2 = ||\bar{\Theta}^{k+1} - \bar{\varphi}^{k+1} + -\bar{\varphi}^{k+1} - \Theta^*||^2 \tag{8.31}$$

$$\begin{aligned} ||\bar{\Theta}^{k+1} - \Theta^*||^2 =& ||\bar{\Theta}^{k+1} - \bar{\varphi}^{k+1}||^2 + ||\bar{\varphi}^{k+1} - \Theta^*||^2 \\ &+ 2||\bar{\Theta}^{k+1} - \bar{\varphi}^{k+1}||.||\bar{\varphi}^{k+1} - \Theta^*|| \end{aligned} \tag{8.32}$$

$$\begin{aligned} ||\bar{\Theta}^{k+1} - \Theta^*||^2 =& ||\bar{\Theta}^{k+1} - \bar{\varphi}^{k+1}||^2 + ||\bar{\varphi}^{k+1} - \Theta^*||^2 \\ &+ 2\langle \bar{\Theta}^{k+1} - \bar{\varphi}^{k+1}, \bar{\varphi}^{k+1} - \Theta^* \rangle \end{aligned} \tag{8.33}$$

In the case of $k+1 \notin \mathcal{I}_E$, then the term $||\bar{\Theta}^{k+1} - \bar{\varphi}^{k+1}||^2$ vanishes. Then by applying Lemma 4, we get

$$\mathbb{E}[||\bar{\Theta}^{k+1} - \Theta^*||^2] \leq (1-\alpha^k\mu)\mathbb{E}[||\bar{\Theta}^{k+1} - \Theta^*||^2] + (\alpha^k)B \tag{8.34}$$

In the case of $k+1 \in \mathcal{I}_E$, then by applying Lemma 4, we get

$$\mathbb{E}[||\bar{\Theta}^{k+1} - \Theta^*||^2] \leq (1-\alpha^k\mu)\mathbb{E}[||\bar{\Theta}^{k+1} - \Theta^*||^2] + (\alpha^k)(B+C) \tag{8.35}$$

where $B = \sum_{l=1}^{L_S} w_l\sigma_l^2 + 6L\Sigma + 8(E-1)^2G^2$ and $C = \frac{4(N_S-L_S)}{N_S-1}(E^2G^2)$ if w_l is uniform and $C = \frac{4}{L_S}(E^2G^2)$ otherwise.

By choosing $\alpha^k = \frac{\beta}{k+\delta}$ for some $\beta > 1/\mu$ and $\delta > 0$ so that $\alpha^1 \leq \min\{1/\mu, 1/4L\} = 1/4L$ and $\alpha^k \leq 2\alpha^{k+E}$ then we have $\mathbb{E}[||\bar{\Theta}^{k+1} - \Theta^*||^2] \leq \frac{v}{\delta+k}$ where $v = \max\{\frac{\beta^2(B+C)}{\beta\mu-1}, (\delta+1)||\bar{\Theta}^{k+1} - \Theta^*||^2\}$

Then, by applying a strong convexity assumption of F, we have

$$\mathbb{E}[\bar{\Theta}^k] - F^* \leq \frac{L}{2}\Delta^k \leq \frac{L}{2}\frac{v}{\delta+k} \tag{8.36}$$

where F^* is the minimum value of F where optimum parameter Θ^* is achieved. Later on, if we choose $\beta = 2/\mu, \delta = \max\{8L/\mu, E\}$and denote $\kappa = L/\mu, \alpha^k = 2/u(1/(\delta+k))$ then we have

$$\mathbb{E}[F(\bar{\Theta}^k)] - F^* \leq \frac{\kappa}{(\delta+k-1)}(\frac{2(B+C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\Theta^1 - \Theta^*||) \tag{8.37}$$

The equation above generalizes FL where the model is optimized in a total of k iterations, where in a practical implementation, $k = b.E.R, b$ is the number of batches. Then, we get $k > R$ as E and b are positive integers. Therefore, substituting k with R in the inequality above produces the higher amount on the right side. On that basis, the inequality above can be derived to:

$$\mathbb{E}[F(\Theta^R)] - F^* \leq \frac{\kappa}{(\delta+R-1)}(\frac{2(B+C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\Theta^1 - \Theta^*||) \tag{8.38}$$

The equation above can be derived into:

$$\mathbb{E}[F(\Theta^R)] - F^* \leq \frac{1}{R}\frac{\kappa}{(\delta/R+1-1/R)}(\frac{2(B+C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\Theta^1 - \Theta^*||) \tag{8.39}$$

Let $A = \frac{\kappa}{(\delta/R+1-1/R)}$ is a positive number. Then the equation above can be derived into:

$$\mathbb{E}[F(\Theta^R)] - F^* \leq \frac{A}{R}(\frac{2(B+C)}{\mu} + \frac{\mu\delta}{2}\mathbb{E}||\Theta^1 - \Theta^*||) \tag{8.40}$$

The inequality (8.430) guarantees that the proposed weighted federated learning achieves a convergence condition that is upper bounded by the amount on the right side.

Proof of Theorem 3: Let Θ^* and Θ are optimal parameter in $\mathcal{T}_l^t \cup Z$ and $\mathcal{T}_l^t$ respectively, where $\mathcal{T}_l^t \subset \mathcal{T}^t$, where $|\mathcal{T}_l^t|/|\mathcal{T}^t| = \eta \in (0,1)$, then we have

$$F(\Theta; \mathcal{T}^t) = \eta F(\Theta; \mathcal{T}_l^t) + (1-\eta)F(\Theta; (\mathcal{T}^t - \mathcal{T}_l^t)) \tag{8.41}$$

$$F(\Theta^*; \mathcal{T}^t) = \eta F(\Theta^*; \mathcal{T}_l^t) + (1-\eta)F(\Theta^*; (\mathcal{T}^t - \mathcal{T}_l^t)) \tag{8.42}$$

Let Θ^o is the initial value of $\Theta$and Θ^* assigned by a random uniform initiation. Thus or all class $c \in \mathcal{T}_c^t = \mathcal{T}_{y=c}^t$ It satisfy $F(\Theta^o; \mathcal{T}_c^t) = e^o$. After optimally learning on $\mathcal{T}_l^t$ and $\mathcal{T}_l^t \cup Z$ then Θ^o become Θ and Θ^* respectively. Note that Θ knows only available classes in $\mathcal{T}_l^t$, while Θ^* knows classes that available both in $\mathcal{T}_l^t$ and classes in $\mathcal{T}^t - \mathcal{T}_l^t$ via Z. Assuming that the loss for predicting classes in $\mathcal{T}_l^t$ is $e^a < e^o$, then we have $F(\Theta; \mathcal{T}_l^t) = F(\Theta^*; \mathcal{T}_l^t) = e^a < e^o$. Since all the backbone parameters are frozen and Θ^* learn Z, then we get $F(\Theta; (\mathcal{T}^t - \mathcal{T}_l^t)) = e^o$, while $F(\Theta^*; (\mathcal{T}^t - \mathcal{T}_l^t)) = e^b$, where $e^a \geq e^b \geq e^o$.

Then, the equations (8.41) and (8.42) can be derived to

$$F(\Theta; \mathcal{T}^t) = \eta e^a + (1-\eta)e^o \tag{8.43}$$

$$F(\Theta^*; \mathcal{T}^t) = \eta e^a + (1-\eta)e^b \tag{8.44}$$

Subtracting the equations above, we have

$$F(\Theta; \mathcal{T}^t) - F(\Theta^*; \mathcal{T}^t) = \eta e^a + (1-\eta)e^o - (\eta e^a + (1-\eta)e^b) \tag{8.45}$$

$$F(\Theta; \mathcal{T}^t) - F(\Theta^*; \mathcal{T}^t) = \eta e^a + (1-\eta)e^o - \eta e^a - (1-\eta)e^b \tag{8.46}$$

$$F(\Theta; \mathcal{T}^t) - F(\Theta^*; \mathcal{T}^t) = (1-\eta)e^o - (1-\eta)e^b \tag{8.47}$$

$$F(\Theta; \mathcal{T}^t) - F(\Theta^*; \mathcal{T}^t) = (1-\eta)(e^o - e^b) \tag{8.48}$$

As we have $0 < \eta < 1$ and $e^o > e^b$, the right side of the inequality above has a positive value. Then, by choosing a small positive number $\epsilon > 0$ where $(1-\eta)(e^o - e^b) \geq \epsilon$, then we have.

$$F(\Theta; \mathcal{T}^t) - F(\Theta^*; \mathcal{T}^t) \geq \epsilon \tag{8.49}$$

inequality above proves that Θ^* is more generalized to $\mathcal{T}^t$ than Θ. This shows that our idea, i.e., empowering prompt learning with shared unified prototypes improves model generalization.

8.5 Experimental Results and Analysis

8.5.1 Dataset

The experiment is done using three benchmarks, i.e., split CIFAR100, split MiniImageNet, and split CUB200. The CIFAR100 and miniImageNet datasets contain 100 classes, while CUB200 is a dataset of 200 classes. We follow the settings from [40] for tasks and classes per task split, and [59] for the federated setting. For CIFAR100 and MiniImageNet, we split the dataset into 9 tasks, i.e., 60 classes for the base task ($t = 0$), and 5 classes for each few-shot task ($t \geq 1$). We split the CUB200 dataset into 11 tasks, i.e., 100 classes for the base task, and 10 classes for each few-shot task. Few shot tasks are measured in 5-shot and 1-shot settings.

8.5.2 Baseline

UOPP are compared with 9 benchmark algorithms, i.e., LGA [59], TARGET [63], LANDER [64], and Fed-CPrompt [152] that represent the SOTA in federated class incremental learning, Fed-S3C[239] that represents SOTA in few shot class incremental learning, Fed-L2P[37], Fed-DualP[38], Fed-CODAP[39], and PILoRA[154] that represent SOTA in FCIL. Except for Fed-CPrompt, "Fed-" denotes that the method is customized in a federated manner from its original (stand-alone) mode, by using FedAvg as the aggregation function. We only run LANDER and PILoRA for CIFAR100 since their official code, settings, and pre-trained embedding that can be executed in our setting is the embedding of CIFAR100 dataset. We also evaluated UOPP in standalone FSCIL and compared it to FSCIL SOTAs,

i.e., TEEN[118], NC-FSCIL[44], OrCo[45], and PriViLege[129], Please see section 8.5.14.

8.5.3 Experiment Setup

Details: The experiment is executed under a single NVIDIA A100 GPU with 40 GB memory across 3 different random seeds. Adapted from [59], the simulation is run by 20 total clients and 1 global server, where in each round, 6 (30%) local clients are selected randomly. Each client randomly receives 60% ($\eta = 0.6$) classes. The total global round is set to 90 (10 rounds/task) for CIFAR100 and MiniImageNet and 110 for CUB200. Our task split setting is different from the recent study [244], since it follows FCIL setting, while our setting follows FSCIL setting.

Metrics: The consolidated algorithms are evaluated for all learned classes with accuracy metrics (Acc.) and performance drop (PD). Besides, the methods are evaluated by the accuracy of base classes, novel classes, and the harmonic mean accuracy, which represents stability-plasticity performance.

CNN-Based Methods Implementation: The competitor methods, i.e., Fed-S3C, LGA, TARGET, and LANDER run with 2-20 local epochs on each client. The learning rate is set with the best result from 0.001 to 5.0 by a 5 or 10 increment factor. The other hyperparameters, such as weight decay, momentum, and dropout rate, follow their original setting. The methods utilize ResNet18 as the backbone model. LGA utilizes LeNet as the perturbation model. TARGET and LANDER use CNN as their synthesizer model. TARGET and LANDER generate 10000-50000 synthetic images on each task.

Prompt-Based Methods Implementation: PILoRA and the prompt-based methods, i.e., Fed-L2P, Fed-DualP, Fed-CODAP, Fed-CPrompt, and UOPP, are run with a ViT backbone. The base task is run with 1-2 epochs, while the few-shot task is run with 2-20 local epochs. For UOPP, the rectification step M is set to 40 steps per iteration. The initial learning rate is set with the best result from 0.001 to 0.2 by a 2 or 5 increment factor. The learning rate for the FS task and base task may be different. The prompt length is set to 5. The dual-head selection is executed in a batch-wise manner for convenience in implementation. The other

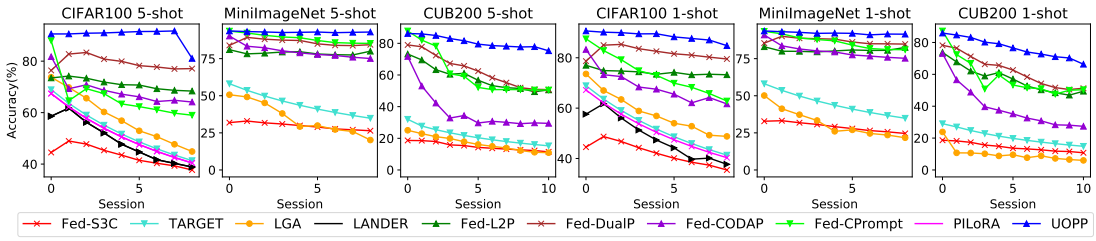


Figure 8.4: Visualization of the performance of consolidated algorithms in MiniImageNet, CIFAR100 and CUB200.

parameters are set with the default settings.

8.5.4 General Performance

The numerical result of the consolidated algorithms is shown in table 8.1. The proposed method (UOPP) achieves the highest accuracy with a significant gap $5-76\%$ compared to the competitor methods, both in 5-shot and 1-shot settings. Fed-S3C, TARGET, LGA, and LANDER achieve relatively low performance with $30-76\%$ gap in 3 benchmark datasets in both 5-shot and 1-shot settings compared to UOPP. The results confirm that the FCIL and FSCIL methods can not be applied directly in FFSCIL. Meanwhile, prompt-based methods, i.e., Fed-L2P, Fed-DualP, Fed-CODAP, and Fed-CPrompt, achieve a relatively better performance than those 4 methods. Compared to UOPP, the methods have a relatively smaller gap, i.e., $5-36\%$. The results prove that prompt-based methods are more promising than the SOTAs of FCIL and FSCIL, The proposed method also achieves the lowest performance drop with $(0.7-19\%)$, followed by Fed-L2P with $(0.4-22\%)$ PD and Fed-DualP with $(-0.4-28\%)$ PD. Despite utilizing ViT and the prototype approach, PILoRA achieves lower performance than the prompt-based method, i.e., $< 52\%$ on average. Looking at per-task performance, Figure 8.4 shows that UOPP achieves the highest accuracy in all tasks in those three datasets. In the first task (base task), the proposed method achieves higher performance with a small gap, i.e., $1-2\%$, than the competitor methods. However, with the increasing number of tasks, the gap gets higher, e.g., $\geq 3\%$ in task-1, $\geq 4\%$ in task-2, and $\geq 6\%$ in task-9 (last task). It shows that the proposed method handles catastrophic forgetting better than the competitor methods.

Table 8.1: Numerical result of the consolidated algorithms in CIFAR100, MiniImageNet, and CUB200 dataset in 5-shot and 1-shot setting across 3 different seeded runs.

(a) Complete numerical result on CIFAR100 dataset with 5-shot setting

Method	**Trainable Params.**	**Accuracy in each session (%) ↑**									**Avg↑**	**PD ↓**	**Gap↓**
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**			
Fed-S3C	CNN	44.51	48.97	47.77	45.35	43.48	41.47	40.33	39.32	37.71	43.21	6.80	46.80
TARGET	CNN	68.90	63.61	59.06	55.12	51.68	48.64	45.94	43.52	41.34	53.09	27.56	36.92
LGA	CNN	73.76	69.80	65.59	60.26	56.87	52.94	50.66	47.69	44.89	58.05	28.87	31.96
LANDER	CNN	58.60	61.75	56.26	52.11	47.71	44.71	41.69	40.28	38.87	49.11	19.73	40.90
Fed-L2P	Prompt	73.47	74.20	73.37	71.88	70.85	70.72	69.28	68.66	68.37	71.20	5.10	18.81
Fed-DualP	Prompt	76.39	82.75	83.37	80.80	79.93	78.26	77.73	76.98	77.11	79.26	-0.72	10.75
Fed-CODAP	Prompt	81.73	69.29	70.81	68.67	67.17	66.14	64.32	64.79	64.12	68.56	17.62	21.45
Fed-CPrompt	Prompt	88.00	64.63	69.30	67.39	63.39	62.33	61.11	59.78	59.00	66.10	29.00	23.91
PILoRA	Params (A,B)	67.40	62.22	57.77	53.92	50.55	47.58	44.93	42.57	40.44	51.93	26.96	38.08
UOPP (Ours)	**Prompt**	**90.57**	**90.58**	**90.85**	**90.96**	**91.23**	**91.51**	**91.56**	**91.74**	**81.05**	**90.01**	**9.52**	**0.00**

(b) Summarized numerical result on CIFAR100 dataset with 1-shot setting, MiniImagenet, and CUB200 dataset with 5-shot and 1-shot settings

Method	**CIFAR100**			**MiniImageNet**			**CUB200**								
	1-shot			**5-shot**			**1-shot**			**5-shot**			**1-shot**		
	Avg	**PD**	**Gap**	**Avg**	**PD**	**Gap**	**Avg**	**PD**	**Gap**	**Avg**	**PD**	**Gap**	**Avg**	**PD**	**Gap**
Fed-S3C	41.97	9.07	46.65	29.73	5.48	63.19	29.16	8.24	63.05	14.91	7.22	65.89	14.40	7.85	62.33
TARGET	53.09	27.56	35.53	44.77	23.24	48.15	44.77	23.24	47.44	21.47	16.77	59.33	20.70	14.17	56.03
LGA	58.29	24.80	30.33	35.07	30.51	57.85	31.65	28.44	60.56	16.92	14.16	63.88	10.02	17.86	66.71
LANDER	48.43	19.90	40.19	-	-	-	-	-	-	-	-	-	-	-	-
Fed-L2P	74.29	3.75	14.33	78.92	1.01	14.00	80.80	0.45	11.41	58.26	22.67	22.54	57.12	24.34	19.61
Fed-DualP	81.95	-0.87	6.67	85.91	-0.45	7.01	86.88	0.67	5.33	62.89	28.21	17.91	62.41	26.94	14.32
Fed-CODAP	68.72	21.62	19.90	80.11	15.12	12.81	80.13	15.70	12.08	37.55	42.26	43.25	39.80	45.68	36.93
Fed-CPrompt	73.82	24.69	14.80	88.77	8.29	4.15	86.34	12.38	5.87	61.23	37.33	19.57	58.26	36.86	18.47
PILoRA	51.78	26.88	36.84	-	-	-	-	-	-	-	-	-	-	-	-
UOPP (Ours)	**88.62**	**5.83**	**0.00**	**92.92**	**0.73**	**0.00**	**92.21**	**2.18**	**0.00**	**80.80**	**10.90**	**0.00**	**76.73**	**19.54**	**0.00**

Table 8.2: Harmonic mean accuracy of the consolidated algorithms in CIFAR100 dataset with 5-shot setting.

Method	**Harmonic Mean Acc. by Session (%)**								**Avg**	**PD**	**Gap**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**			
Fed-S3C	49.3	42.8	38.0	35.4	33.9	33.4	32.9	32.1	37.2	17.3	53.3
TARGET	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	90.5
LGA	44.7	31.5	19.3	16.2	12.5	12.3	9.9	7.8	19.3	36.9	71.2
LANDER	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	90.5
Fed-L2P	46.0	50.2	49.2	50.0	53.5	54.8	56.1	57.7	52.2	-11.7	38.3
Fed-DualP	67.4	71.4	66.8	68.4	68.2	69.3	70.1	71.8	69.2	-4.5	21.3
Fed-CODAP	69.8	69.3	66.1	65.2	63.3	61.8	62.0	61.3	64.9	8.5	25.6
Fed-CPrompt	73.3	74.4	67.6	63.5	62.3	60.4	59.4	58.9	65.0	14.3	25.5
PILoRA	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	90.5
UOPP	**90.6**	**91.6**	**91.6**	**91.9**	**92.1**	**92.0**	**92.1**	**81.8**	**90.5**	**8.8**	**0.0**

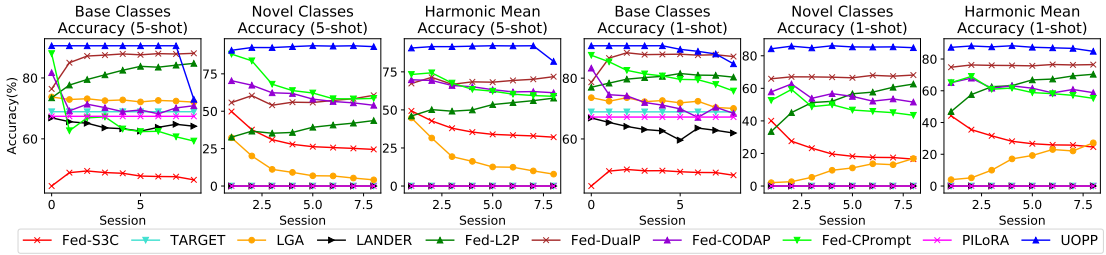


Figure 8.5: Visualization of the performance of consolidated methods for base classes and novel classes in CIFAR100 Dataset.

8.5.5 Stability-Plasticity Analysis

This subsection evaluates UOPP stability-plasticity performance by evaluating the harmonic mean accuracy on each few-shot task. Table 8.2 shows the harmonic mean accuracy of consolidated methods in CIFAR100 with the 5-shot setting, while Figure 8.5 visualizes the accuracy for base classes, novel classes, and harmonic mean accuracy. Both show that UOPP achieves the best harmonic mean accuracy with $\geq 15\%$ gap on each task and $18-58\%$ on average of all tasks. The results prove that UOPP handles stability-plasticity dilemmas better than its competitors. Looking at the base classes and novel classes, performance, Figure 8.5 shows that UOPP achieves the highest accuracy of base classes and novel classes that are consistent through all tasks. Fed-L2P has increased base classes and novel classes' accuracy, but the performance is still below UOPP with a significant gap. Fed-Dual, Fed-CODAP, and Fed-CPrompt have relatively stable base class accuracy but decreasing novel class accuracy. TARGET, LANDER, and PILoRA can not achieve plasticity since their novel classes' accuracy is close to 0. Fed-S3C maintains stability-plasticity dilemma with relative balances, but the performance of both components is low ($< 40\%$).

8.5.6 Performance on Different k-Shot

We extend our investigation of the methods' performance on different k-shot values, i.e., 7-shot, 5-shot, 3-shot, and 1-shot. Table 8.3 presents the detailed numerical results on CIFAR100 dataset with 1-7 shot settings. Table 8.3 shows that UOPP consistently achieves the highest performance in different k-shot values. The performance gap is consistently significant, i.e., 7-46%. Regarding average

performance, UOPP tends to achieve lower performance in lower k-shot values, i.e., 1 and 3. UOPP performances in 5-shot and 7-setting are comparable on average. However, in terms of the performance in the last (final) session, UOPP achieves a better final performance in a 7-shot setting. This indicates that more samples for each client improve the global model performance.

This trend also applies to other methods, where their performance on the higher shots is better than their performance on the lower shots. However, Fed-CPrompt shows the anomaly, where its performance in lower shots is better than its performance in higher shots. This condition indicates that the increase in novel classes' accuracy is lower than the drop in base classes' accuracy. Thus, the higher sample is contra-productive for few-shot task training. Looking at the performance drop (PD), the table shows that UOPP achieves a lower (better) performance drop in the higher k-shot value. This fact is in line with the mentioned trend of higher performance in the higher k-shot value before.

8.5.7 Performance Analysis on Different Non-i.i.d. Level

Table 8.4 (a) shows the performance of the methods w.r.t. non-i.i.d. level represented by the percentage of available class η (lower is harder). The table shows that UOPP outperforms the existing methods in all non-i.i.d. levels with a significant margin, i.e., up to 50%. The table shows that the small available class ($\eta = 20\%$) remains challenging since all the methods achieve less than 70% on average. In contrast, in other cases, i.e., ($\eta > 40\%$), UOPP achieves $> 86\%$ accuracy on average.

8.5.8 Performance Analysis on Different Participating Local Clients

This subsection evaluates UOPP robustness in various selected local clients, simulating fluctuations of participating clients in real-life applications. Table 8.4 (b) shows the performance of consolidated methods with 4 (20%), 6 (30%), and 8 (40%) selected local clients from 20 total clients. The table shows that the UOPP achieves the highest performance in all combinations with a significant gap, i.e., $\geq 10\%$, compared to the competitor methods. Table 8.4 also shows that UOPP

Table 8.3: Numerical result of the FFSCIL methods in CIFAR100, 7-shot, 5-shot, 3-shot, and 1-shot setting, S indicates the number of k-shot, PD indicates the performance drop, and Gap indicates the gap between the respective method and UOPP (UOPP).

Method	**S**	**Accuracy in each session (%)**									**Avg**	**PD**	**Gap**
		0	1	2	3	4	5	6	7	8			
Fed-S3C	7	43.98	49.65	48.01	45.83	43.21	41.68	40.39	39.69	38.18	43.40	5.80	46.36
TARGET	7	68.65	63.40	58.36	52.68	50.04	46.58	43.96	39.37	34.04	50.78	34.61	38.98
LGA	7	73.15	67.08	64.56	59.87	56.15	52.04	50.00	47.29	44.39	57.17	28.76	32.59
LANDER	7	66.30	60.65	55.89	51.05	48.15	44.53	42.66	40.83	38.26	49.81	28.04	39.95
Fed-DualP	7	78.45	83.71	84.61	83.40	83.29	81.86	81.84	81.51	81.80	82.27	-3.35	7.49
Fed-Cprompt	7	88.13	77.9385	69.6857	67.13	66.7125	64.8824	64.28	62.51	61.51	69.20	26.62	20.56
PILoRA	7	66.48	61.37	56.99	53.19	49.86	46.93	44.32	41.99	39.89	51.22	26.59	38.54
UOPP	**7**	**90.77**	**90.86**	**91.04**	**90.75**	**91.08**	**89.35**	**88.84**	**87.87**	**87.32**	**89.76**	**3.45**	**0.00**
Fed-S3C	5	44.51	48.97	47.77	45.35	43.48	41.47	40.33	39.32	37.71	43.21	6.80	46.80
TARGET	5	68.90	63.61	59.06	55.12	51.68	48.64	45.94	43.52	41.34	53.09	27.56	36.92
LGA	5	73.76	69.80	65.59	60.26	56.87	52.94	50.66	47.69	44.89	58.05	28.87	31.96
LANDER	5	58.60	61.75	56.26	52.11	47.71	44.71	41.69	40.28	38.87	49.11	19.73	40.90
Fed-L2P	5	73.47	74.20	73.37	71.88	70.85	70.72	69.28	68.66	68.37	71.20	5.10	18.81
Fed-DualP	5	76.39	82.75	83.37	80.80	79.93	78.26	77.73	76.98	77.11	79.26	-0.72	10.75
Fed-CODAP	5	81.73	69.29	70.81	68.67	67.17	66.14	64.32	64.79	64.12	68.56	17.62	21.45
Fed-Cprompt	5	88.00	64.63	69.30	67.39	63.39	62.33	61.11	59.78	59.00	66.10	29.00	23.91
PILoRA	5	67.40	62.22	57.77	53.92	50.55	47.58	44.93	42.57	40.44	51.93	26.96	38.08
UOPP	**5**	**90.57**	**90.58**	**90.85**	**90.96**	**91.23**	**91.51**	**91.56**	**91.74**	**81.05**	**90.01**	**9.52**	**0.00**
Fed-S3C	3	43.98	49.60	48.03	45.19	42.69	41.12	39.87	38.82	37.47	42.97	6.51	45.64
TARGET	3	68.65	63.03	58.10	52.64	49.00	44.92	42.09	37.18	32.20	49.76	36.45	38.85
LGA	3	73.52	67.37	63.57	59.07	56.01	52.35	49.50	46.89	44.15	56.94	29.37	31.67
LANDER	3	66.30	59.77	55.56	50.56	47.40	44.04	41.72	40.72	38.43	49.39	27.87	39.22
Fed-DualP	3	73.20	79.31	82.73	81.53	82.06	81.16	81.56	81.21	80.34	80.34	-7.14	8.27
Fed-Cprompt	3	88.32	83.62	81.46	77.68	77.18	75.40	74.32	70.63	68.51	77.46	19.81	11.15
PILoRA	3	66.43	61.32	56.94	53.15	49.83	46.89	44.29	41.96	39.86	51.19	26.57	37.42
UOPP	**3**	**90.62**	**90.57**	**90.84**	**89.84**	**90.18**	**89.69**	**88.20**	**88.24**	**79.31**	**88.61**	**11.31**	**0.00**
Fed-S3C	1	44.51	48.70	46.76	44.26	42.09	40.11	38.51	37.35	35.44	41.97	9.07	46.65
TARGET	1	68.90	63.61	59.06	55.12	51.68	48.64	45.94	43.52	41.34	53.09	27.56	35.53
LGA	1	73.58	67.00	63.44	58.88	56.90	54.00	52.82	49.25	48.78	58.29	24.80	30.33
LANDER	1	57.60	61.75	56.09	51.29	47.33	44.22	39.73	40.15	37.70	48.43	19.90	40.19
Fed-L2P	1	77.00	74.91	74.73	74.40	73.45	74.20	73.22	73.46	73.25	74.29	3.75	14.33
Fed-DualP	1	73.20	79.31	82.73	81.53	82.06	81.16	81.56	81.21	80.34	80.34	-7.14	8.28
Fed-CODAP	1	83.29	73.27	72.54	68.37	67.52	65.54	62.11	64.19	61.67	68.72	21.62	19.90
Fed-Cprompt	1	87.53	82.86	79.24	74.87	73.04	69.92	68.28	65.81	62.84	73.82	24.69	14.80
PILoRA	1	67.20	62.03	57.60	53.76	50.40	47.44	44.80	42.44	40.32	51.78	26.88	36.84
UOPP	**1**	**90.65**	**90.16**	**89.97**	**89.49**	**89.53**	**88.29**	**87.66**	**87.04**	**84.82**	**88.62**	**5.83**	**0.00**

Table 8.4: Summary of the numerical result of the consolidated algorithms in CIFAR100 dataset with the variation of Non-i.i.d. level (a), selected local clients (b), and variation of total global round (c).

	(a) Non-i.i.d level (η)						(b) Selected Local Client (L)						(c) Total Global Round (R)					
Method	**η=0.6 (60%)**		**η=0.4 (40%)**		**η=0.2 (20%)**		**L=4 (20%)**		**L=6 (30%)**		**L=8 (40%)**		**R=54 (6 r/task)**		**R=72 (8 r/task)**		**R=90 (10 r/task)**	
	Avg	**PD**	**Avg**	**PD**	**Avg**	**PD**	**Avg**	**PD**	**Avg**	**PD**	**Avg**	**PD**	**Avg**	**PD**	**Avg**	**PD**	**Avg**	**PD**
Fed-S3C	43.2	6.8	34.2	0.5	18.9	1.4	43.9	4.2	43.2	6.8	42.7	6.2	43.4	5.3	47.4	10.1	42.6	6.5
TARGET	53.1	27.6	41.3	27.5	23.2	13.5	51.4	26.7	53.1	27.6	56.7	29.4	44.4	23.0	51.8	26.9	53.5	27.8
LGA	58.1	28.9	56.4	29.5	56.4	29.7	56.3	30.4	58.1	28.9	57.9	29.1	57.1	22.1	55.2	26.3	58.0	29.3
LANDER	49.1	19.7	37.6	18.9	17.5	8.9	47.4	22.3	49.1	19.7	51.6	20.8	37.9	31.4	47.2	26.0	49.1	19.7
Fed-DualP	79.3	-0.7	60.2	-41.9	6.6	-21.5	76.2	-11.2	79.3	-0.7	79.2	11.1	81.5	-0.9	80.2	1.8	79.3	-0.7
Fed-Cpompt	66.1	29.0	66.2	28.1	66.2	28.1	62.5	36.9	66.1	29.0	59.6	38.1	62.5	36.9	59.6	38.1	66.1	29.0
PILoRA	51.9	27.0	43.4	22.5	35.3	18.3	50.6	26.3	51.9	27.0	51.1	26.5	55.9	29.0	53.9	28.0	51.9	27.0
UOPP	**90.0**	**9.5**	**86.3**	**4.3**	**68.8**	**29.4**	**89.1**	**4.2**	**90.0**	**9.5**	**91.3**	**0.0**	**90.7**	**-1.5**	**90.8**	**0.2**	**90.0**	**9.5**

Table 8.5: Summary of the numerical result of our ablation study in CIFAR100 dataset, MLP, and PB and Rect. denote the MLP-based, prototypes-based, and rectification, respectively.

Conf.	**Stiatic Proto.**	**Dynamic Proto.**	**MLP Head**	**PB. Head**	**Rect.**	**Avg**	**PD**	**Gap**	**Time**
A	-	✓	✓	✓	✓	80.62	7.76	9.39	6.01h
B	✓	-	✓	✓	-	85.32	10.06	4.69	4.17h
C	✓	✓	-	✓	✓	88.72	5.49	1.29	8.00h
D	-	-	✓	-	-	69.23	37.76	20.78	3.50h
UOPP	✓	✓	✓	✓	✓	90.01	9.52	0.00	6.05h

achieves higher performance on a higher percentage of participating local clients. Besides, UOPP achieves a relatively lower performance drop (PD) than Fed-S3C, TARGET, LGA, LANDER, and Fed-CPrompt. In 8 (40%) local client cases, UOPP achieves the lowest PD, while in 4 (20%) and 6 (30%) local client cases, UOPP experiences a higher PD than in 8 and 4 local client cases due to an accuracy drop on the last task. The accuracy drop is caused by a mismatch between the samples and prompt keys, resulting in inaccurate feature extraction and classifier selection.

8.5.9 Performance Analysis on Different Global Rounds

This subsection evaluates UOPP robustness in smaller global rounds simulating real-world conditions where the global model is urgently needed, thereby requiring smaller rounds. Table 8.4(b) summarizes our investigation on 54 (6 r/task) to 90 (10 r/task) global rounds in CIFAR100 datasets. The table shows that UOPP achieves the highest performance with a significant gap, i.e., ($\geq$ 9%), compared

to the competitors. In the lower global rounds, the UOPP achieves even better performance than in normal global rounds, as it does not experience the accuracy drop aforementioned. UOPP also achieves the smallest PD compared to the competitor methods in smaller rounds. Prompt-based method, i.e., Fed-DualP is proven to be more robust than Fed-S3C, TARGET, LGA, LANDER, and Fed-CPrompt. Furthermore, those 3 methods achieve lower accuracy in the smaller global rounds. This finding confirms the robustness of UOPP in the case of low global rounds.

8.5.10 Ablation Study

This subsection discusses an ablation study to investigate the contribution of each component of the proposed method. The result is summarized in Table 8.5. The result shows that the absence of the static prototype (Conf. A) and dynamic prototype (Conf. B) drops the average performance with 9.4% and 4.7% gap, respectively. This result proves the importance of unified prototypes for prompt learning to deal with FFSCIL problems. The absence of the MLP head drops the performance with 1.3% gap. This result shows that the presence of MLP classifiers (Conf. C) contributes to the model prediction performance. Last, the absence of the prototype-based head (Conf. D) drops the performance with the most significant magnitude, e.g., 20.8%. It proves that the PB classifier is a must to deal with few-shot tasks. Note that the presence of rectification (Rect.) follows the presence of the dynamic prototype (e.g., in configurations B and D, where the dynamic prototype is absent, the rectification is absent in those configurations).

e) The Importance of Prototype Rectification: We analyze the impact of prototype rectification in UOPP. Figure 8.6 shows the validation loss of novel classes with rectification and without rectification on CIFAR100 dataset with 5-shot and 1-shot settings. The figure shows that, without prototype rectification (red line), UOPP produces a far higher validation loss for novel classes, as there are many misclassified samples. On the contrary, with prototype rectification (blue line), UOPP produces far smaller and more stable validation loss. The figure also shows that the difference in loss magnitude between the two variants is even higher in the 1-shot setting. This finding emphasizes the importance of prototype

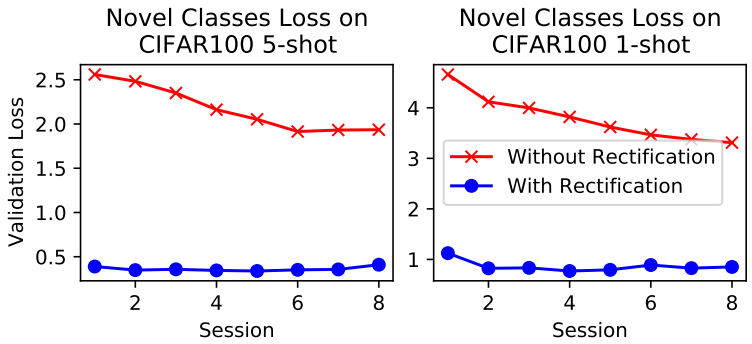


Figure 8.6: Visualization of Novel Classes Validation Loss with vs without Rectification.

Table 8.6: Comparison of Parameters, Communication Cost, and Training time in CIFAR100 dataset.

Method	Number of Parameters (M)				Comm. Cost (MB)		Running Time (h)
	Trainable		Sharable				
	Base Task	FS Task	Base Task	FS Task	Base Task	FS Task	
Fed-S3C	11.7	11.7	11.7	11.7	46.8	46.8	4.1
TARGET	11.3	11.3	17.45	17.45	61.7	61.7	2.02
LGA	11.3	11.3	15.42	15.42	69.8	69.8	8.22
LANDER	11.3	11.3	17.45	17.45	61.7	61.7	4.78
Fed-L2P	0.29	0.29	0.29	0.29	1.16	1.16	4.83
Fed-DualP	0.33	0.33	0.33	0.33	1.32	1.32	3.2
Fed-CODAP	0.33	0.33	0.33	0.33	1.32	1.32	1.42
Fed-Cprompt	0.33	0.33	0.33	0.33	1.32	1.32	2.03
PILoRA	0.30	0.30	0.34	0.37	1.37	1.50	6.24
UOPP	**0.33**	**34.1**	**0.38**	**0.41**	**1.5**	**1.63**	**6.05**

rectification in UOPP.

8.5.11 Complexity, Running Time, Parameters, and Communication Cost

Following the pseudo-code in Algorithm 7, UOPP have several operations, e.g., generate static prototype (line 11, 24, 34), drawing $\mathcal{S}$, $\mathcal{Q}$ from prototypes (line 25), Rectify prototype (line 26), updating model parameters (line 20-22, 28-30), forming unified prototype (line 27, 38, 40) data exchange between clients and server. Knowing that accumulating on all batches, generating prototype or compute features from $\mathcal{T}_l^t$ cost $O(N_l^t)$, drawing (augment) samples from feature costs $O(N_l^t)$, rectifying prototypes cost costs $O(N_l^t)$, parameters update cost costs $O(N_l^t)$, form-

ing uniform prototype cost $O(1)$, and parameters exchange include aggregation costs $O(1)$, and we have 1 base task and T few-shot tasks (total task is (T+1)) then the UOPP complexity will be:

$$O(UPPP) = O(BaseTask) + O(FewStotTask) \tag{8.50}$$

$$\begin{aligned} O(UOPP) =& O(1) + R_T(O(clients_{base}) + O(server_{base}) \\ &+ O(1) + T.R_T.(O(clients_{fs}) + O(server_{fs})) \end{aligned} \tag{8.51}$$

$$\begin{aligned} O(UOPP) =& O(1) + R_T.(L.O(1client_{base}) + O(server_{base})) \\ &+ T.R_T.(L.O(1client_{fs}) + O(server_{fs})) \end{aligned} \tag{8.52}$$

$$\begin{aligned} O(UOPP) =& O(1) + R_T.(L(O(N_l^0) + O(E.N_l^0) \\ &+ O(E.N_l^0)) + O(1) \\ &+ T.R_T.(L(O(N_l^t) + O(E.N_l^t) + O(E.N_l^t) \\ &+ O(E.1) + O(E.N_l^t)) + O(1) \end{aligned} \tag{8.53}$$

$$O(UOPP) = O(1) + R_T.L.O(E.N_l^0) + T.R_T.L.O(E.N_l^t) \tag{8.54}$$

$$O(UOPP) = O(1) + R_T.L.E.O(N_l^0) + T.R_T.L.E.O(N_l^t) \tag{8.55}$$

$$O(UOPP) = O(1) + R_T.L.E(O(N_l^0) + T.O(N_l^t)) \tag{8.56}$$

Please note that $N_l = N_l^0 + N_l^1 + ... + N_l^T = N_l^0 + T(N_l^t), t \in [1..T]$. Therefore, the equation above can be derived into:

$$O(UOPP) = O(1) + R_T.L.E(O(N_l^0 + T.ON_l^t)) \tag{8.57}$$

$$O(UOPP) = R_T.L.E.O(N_l) \tag{8.58}$$

$$O(UOPP) = O(R_T.L.E.N_l) \tag{8.59}$$

Since E is set as a small constant in UOPP, i.e., 1-20 and $R_T < R$, then the UOPP complexity will be:

$$O(UOPP) = O(R.L.N_l) \tag{8.60}$$

The complexity analysis above shows that UOPP has the same complexity, i.e., $O(R.L.N)$ where R is the number of global rounds, L is the number of participating local clients in each round, and N is the size of the training data in each client.

Table 8.6 shows the training time of the consolidated method in CIFAR100 dataset. The table shows that the proposed method requires a moderate training time since it is lower than LGA and higher than the other methods. In the base task, UOPP trains only a small amount (0.33 M) of parameters, since it trains only its prompts and MLP classifier. However, in a few-shot tasks, it trains GradNet for prototype rectification, which contributes to the high amount of trainable parameters. However, UOPP keeps its low communication cost both in the base task and few-shot tasks since it only shares prompt+MLP or prompt+prototypes in both tasks. On the contrary, Fed-S3C, TARGET, LGA, and LANDER train and share a relatively high amount of parameters, as they train the whole model. In addition, LGA and TARGET generate synthetic data and share it with the server for the aggregation process. On the other hand, the prompt-based methods, i.e., Fed-L2P, Fed-DualP, Fed-CODAP, and Fed-CPrompt, train and share only a small amount of parameters.

8.5.12 Memory Consumption Analysis

We extended our resource analysis by evaluating the memory consumption by client and server. Table 8.7 shows the memory consumption by a single client and central server. Please note that each client trains its local model utilizing its local dataset, while the central server coordinates the clients, aggregates the locally optimized models into a global model, and redistributes it to the clients.

Table 8.7 shows the memory consumption for UOPP, Fed-DualP, and PILoRA, where these 3 methods have comparable communication costs and similar aggregation methods. As is commonly known, Table 8.7 shows that the memory consumption of deep learning training is highly affected by the batch size (BS) of the data loader, while the memory consumption for model aggregation is affected by the number of participating clients. Second, the table shows that the memory consumption for a central server is relatively lower than the client's memory consumption. Third, the memory consumption of UOPP is comparable with Fed-DualP, and PILoRa, even though UOPP utilizes more operations for prototype rectification and has more trainable parameters during the few-shot tasks. This fact indicates that UOPP has comparable scalability and resource requirements to

Table 8.7: Analysis of Memory Consumptions by Client and Server.

Method	Single Client Memory Consumption (GB)					
	BS=128	**BS=64**	**BS=32**	**BS=16**	**BS=8**	**BS=4**
Fed-DualP	14.88	9.15	5.82	4.2	3.25	2.82
PILoRA	15.13	9.97	7.02	4.33	3.6	3.2
UOPP	14.89	8.68	5.59	3.98	3.26	2.83
Method	**Server Memory Consumption (GB)**					
	L=1000	**L=100**	**L=50**	**L=10**	**L=5**	**L=2**
Fed-DualP	1.32	0.1320	0.0660	0.0013	0.0066	0.0026
PILoRA	1.50	0.1495	0.0748	0.0015	0.0075	0.0030
UOPP	1.63	0.1627	0.0814	0.0016	0.0081	0.0033

the existing methods. Last but not least, the table shows the high possibility of practical deployment of UOPP both in cross-silo and cross-device scenarios. The local training setting can be adjusted by end device specification. For example, in the case of cross-device settings where local training is conducted by small edge devices such as laptops or IoT nodes, the batch size can be set to a smaller value, i.e., 16 or less. In the case of a cross-silo setting, where local training is conducted by more powerful end devices such as a corporate server, the batch size can be chosen by the larger value, i.e., 128 or more.

8.5.13 Running Time Analysis w.r.t Data Size and Selected Local Clients

We extend our scalability analysis by investigating the simulation time w.r.t. the training data size, e.g., indicated by the available classes percentage (η), and the number of participating local clients (L). table 8.8 shows the running time of our simulation w.r.t. those 2 factors. Please note that our simulation is run on a single GPU device. Thus, each local training is executed sequentially. Thus, If the local training is conducted in a parallel way, the difference in the simulation time will be smaller.

The table shows that the ratio of the training data does not imply the same running time ratio. For example, in the case of $\eta = 0.6$, where each client carries 3x the number of samples of $\eta = 0.2$, the running time in those $\eta = 0.6$ is not equal to 3x the training time of $\eta = 0.2$. In UOPP, the increase of running time in those case is less than 20% (5.35 to 6.05). Thus, we found that the scale-up ratio of

Table 8.8: Running Time w.r.t. Percentage of Available Classes Number of Selected Local Client in Each Round.

Method	**Run time (h) w.r.t. available classes percentage (η)**			**Run time (h) w.r.t. number of Selected local clients (L)**		
	η=0.6(60%)	**η=0.4(40%)**	**η=0.2(20%)**	**L=4**	**L=6**	**L=8**
Fed-S3C	4.10	2.07	2.08	2.18	4.10	6.23
TARGET	2.02	2.15	1.99	1.50	2.02	2.50
LGA	8.22	5.35	5.03	3.51	8.22	15.23
LANDER	4.78	1.90	1.85	3.90	4.78	4.60
Fed-DualP	3.20	2.17	1.90	2.07	3.20	4.07
Fed-CPompt	2.03	2.37	2.45	1.50	2.03	2.77
PILoRA	6.17	4.78	3.13	4.12	6.17	8.62
UOPP	6.05	5.87	5.35	4.70	6.05	8.81

training samples with r factor will impact far less than r training time. Second, as we mentioned earlier regarding the sequential process of local training, table 8.8 shows a linear trend, which is a logical result. For example, for UOPP and LORA, the total simulation time can be formulated by L x 1h. In the real application, local training should be conducted parallel, since each client runs on its local device. The bottleneck will be shifted to the server that receives the locally trained models, and the delay for round-trip communication (model transfer) between clients and server.

8.5.14 Evaluation on Standalone FSCIL

We measure the performance of UOPP in standalone FSCIL setting to evaluate our idea on prompting with a unified static-dynamic prototype and dual-head classifiers. We compare UOPP with existing SOTAs, i.e., TEEN(NeurIPS, 2023)[118], NC-FSCIL(ICLR, 2023)[44], OrCo(CVPR, 2024)[45], and PriViLege(CVPR, 2024)[129]. The evaluation is conducted in 3 datasets, i.e., CIFAR100, MiniImageNet, and CUB200, following common settings in FSCIL.

Table 8.9 shows the detailed numerical results of our experiment. The table shows that UOPP achieves a better performance in general. In comparison to TEEN, NC-FSCIL, and ORCO, UOPP archives a significantly better performance, i.e., with more than 20%, 25%, and 14% margin in CIFAR100, MiniImagenet, and CUB200 dataset, respectively. In comparison to PriViLedge, UOPP achieves higher per-

Table 8.9: Numerical result of the FSCIL methods in CIFAR100, MiniImageNet, and CUB200 dataset with 5-shot setting, PD indicates the performance drop, and Gap indicates the gap between the respective method and UOPP (UOPP).

Method	**Accuracy in each session (%)**											**Avg**	**PD**	**Gap**
CIFAR100: 60 base classes @5 classes in few-shot task														
	0	1	2	3	4	5	6	7	8	9	10			
TEEN	74.92	72.65	68.74	65.01	62.01	59.29	57.90	54.76	52.64	-	-	63.10	22.28	26.63
NC-FSCIL	89.51	82.62	76.72	71.61	67.13	63.18	59.67	56.53	53.70	-	-	68.96	35.81	20.77
OrCO	80.08	68.16	66.99	60.97	59.78	58.60	57.04	55.13	52.19	-	-	62.10	27.89	27.63
PriViLege	90.88	89.39	88.97	87.55	87.83	87.35	87.53	87.15	86.06	-	-	88.08	4.82	1.65
UOPP (Ours)	**90.27**	**90.45**	**90.14**	**89.40**	**89.26**	**89.52**	**89.70**	**89.48**	**89.35**	**-**	**-**	**89.73**	**0.92**	**0.00**
MiniImageNet: 60 base classes @5 classes in few-shot task														
TEEN	73.53	70.55	66.37	63.23	60.53	57.95	55.24	53.44	52.08	-	-	61.44	21.45	31.58
NC-FSCIL	77.25	71.30	66.21	61.80	57.94	54.53	51.50	48.79	46.35	-	-	59.52	30.90	33.50
OrCO	83.30	75.32	71.53	68.16	65.63	63.12	60.20	58.82	58.08	-	-	67.13	25.22	25.89
PriViLege	96.68	96.49	95.65	95.54	95.54	94.91	94.33	94.19	94.10	-	-	95.27	2.58	-2.25
UOPP (Ours)	**93.57**	**93.66**	**93.51**	**93.51**	**93.13**	**92.24**	**92.23**	**92.66**	**92.71**	-	-	93.02	0.86	0.00
CUB200: 100 base classes @10 classes in few-shot task														
TEEN	77.26	76.13	72.81	68.16	67.77	64.40	63.25	62.29	61.19	60.32	59.31	68.14	17.95	14.82
NC-FSCIL	78.49	71.52	65.54	60.30	55.81	51.96	48.72	45.78	43.18	40.92	38.80	57.92	39.69	25.04
OrCO	75.59	66.85	64.05	63.69	62.20	60.38	60.18	59.20	58.00	58.42	57.94	63.35	17.66	19.61
PriViLege	82.21	81.25	80.45	77.76	77.78	75.95	75.69	76.00	75.19	75.19	75.08	78.03	7.13	4.93
UOPP (Ours)	**86.63**	**86.99**	**85.19**	**83.37**	**82.42**	**80.19**	**80.21**	**80.76**	**80.93**	**81.16**	**81.17**	**82.96**	**5.46**	**0.00**

formance in CIFAR100 and CUB200 dataset with 1.65% and 4.9% margins, respectively. UOPP achieves lower performance than PriViLedge in MiniImageNet dataset. Looking at a more detailed view, that lower performance is basically due to UOPP achieving lower performance in the base task, i.e., 3% lower accuracy. Thus, It affects the later (few-shot) tasks and average performance. Please note that PriviLedge utilize vision and language modality, while the rest of the methods only utilize vision modality. This factor should be considered into account in the performance comparison.

Looking from the performance drop (PD) metrics, UOPP archives the lowest performance drop in all dataset by a significant margin, i.e., $2 - 28\%$. This indicates that UOPP handle the catastrophic forgetting better than the existing methods.

8.5.15 Limitation and Potential Solution

This study has several limitations that can be improved in future studies.

a) Same η for each client : First, in our simulation, each client has the same non-i.i.d level represented by the same percentage of available classes (η). In the real application, each client may have a different degree of class availability from

the other clients. Thus, in the future study, the simulation can be extended into a variation percentage of available classes where each client is assigned with a random (picked from a min-max) η range. This variable η raises a new challenge for FFSCIL aside from simulating a more realistic setting.

b) Simulation on a single device (GPU) : Second, our simulation is conducted on a single GPU, where each local training is executed sequentially one by one. This limitation will produce a linearly increasing training time with the number of selected local clients. It is less realistic, especially when we want to measure the training time. This limitation can be solved by utilizing a server/workstation that has multiple GPUs, such as an Nvidia DGX server. The other solution is utilizing multiple cloud devices/servers as clients.

c) Fixed-size of Prompt : In this study, UOPP utilized a fixed-size prompt for all clients. Related to the randomly selected available class and the new challenge of different η (point a), It will be more realistic if a client decides its prompt size following the condition of its local data. The evolving prompt approaches such as ConvPrompt[65] and EvoPrompt[87] may be suitable for those case. However, It raises a new challenge in the aggregation process, i.e., how to produce an optimum global model from the different-sized local models that are optimum for the current task as well as previously learned tasks.

d) Overfitting Handling : The current version of UOPP does not utilize advanced overfitting handling. Thus, in some cases, e.g., in the CIFAR100 dataset, the model may suffer from overfitting, indicated by the performance drop in the last task. One of the potential solutions is to apply early stopping during the training process. The other potential solution is applying learning decay to reduce the learning rate in line with the increasing training epochs.

e) Multi-Modality : The current version of UOPP utilizes vision modality only. In the advance of Vision-Language Models, a language-guided approach may become a prospective approach to improve the model performance. Alternatively, language embedding can be utilized for prototype rectification instead of the prototypes generated by ViT.

8.6 Summary

This chapter defines a new Federated Few-Shot Class-Incremental Learning (FFS-CIL) problem and proposes a novel Unified Optimized Prototype Prompt (UOPP) model that utilizes task-wise prompt learning to mitigate task interference empowered by shared static-dynamic prototypes, adaptive dual heads, and weighted aggregation. The dynamic prototype tackles prototype bias by iterative rectifications. The comprehensive experimental results show that UOPP significantly outperforms existing SOTA methods of FCIL, FSCIL, and CIL, on three datasets with a significant gap, i.e., up to 76%. Our deeper analysis confirms that the proposed method achieves better stability-plasticity trade-off and robustness in different local clients and small global rounds. Our analysis shows that UOPP requires moderate training time but a lower communication cost than the SOTAs.

The next chapter is devoted to address rehearsal-free online continual learning (OCL) problems where CL is conducted in streaming data and there is no exemplar memory from previously learned tasks. In real-world application, streaming data is a common occurrence, such as streaming media, e.g., audio and video, or streaming signals from end-device sensors. The streaming data in OCL complicates the standard CL since the data can be seen only once. Learning in a single epoch, the CL methods barely handle catastrophic forgetting and maintain stability-plasticity.

Chapter 9

PROL : Rehearsal Free Continual Learning in Streaming Data via Prompt Online Learning

The previous chapters address CL and its sub problems with the assumption that there is data storage for the data, so the training process can be repeated (epochs) as many times as needed. This chapter address online continual learning (OCL), i.e., CL in streaming data, where the data is not stored and can be seen only once. The OCL limitation, where the data can be seen only once, complicates the catastrophic forgetting problem in streaming data, as CL models suffer from insufficient learning. A common approach applied by the current SOTAs in OCL is the use of memory-saving exemplars or features from previous classes to be replayed in the current task. On the other hand, the prompt-based approach performs excellently in rehearsal-free continual learning, but at the cost of a growing number of trainable parameters. The first approach may not be applicable in practice due to the data openness policy, while the second approach has the issue of throughput associated with the streaming data. This chapter proposes a novel prompt-based method for online continual learning that includes 3 main components: (1) Single light-weight prompt generator as a general knowledge, (2) trainable scalers-and-shifters as specific knowledge, and (3) a hard-soft updates mechanism. Our proposed method achieves significantly higher performance than the current SOTAs in CIFAR100, ImageNet-R, ImageNet-A, and CUB datasets,

i.e., up to 64% margin. Our complexity analysis shows that our method requires a relatively smaller number of parameters and achieves moderate training time, inference time, and throughput.

Part of this chapter is submitted for publication as: MA Ma'sum, M. Pratama, S. Ramasamy, L. Liu, H. Habibullah, and R Kowalczyk. "PROL : Rehearsal Free Continual Learning in Streaming Data via Prompt Online Learning". Submitted to International Conference on Computer Vision 2025. https://openreview.net/forum?id=mKvOmPHnhO

9.1 Introduction

Continual learning (CL) becomes a spotlight in the world of artificial intelligence (AI) for dynamic environments, as it tackles catastrophic forgetting for a continually evolving sequence of tasks [54], [55]. CL methods are proven to be promising in various domains, e.g., computer vision, automation, NLP, and graph [177], [178], [179], [180]. Online continual learning (OCL) is an advanced sub-problem of CL where CL is conducted in streaming data. The limitation of seeing data only once in OCL leads to insufficient learning of a deep learning model that results in difficulty in gaining knowledge of a new task while maintaining knowledge of previously learned tasks. In other words, it complicates how to resolve catastrophic forgetting problem. To handle such a limitation, most of the OCL states of the arts (SOTAs) save previous task exemplars for a rehearsal process to maintain the knowledge from previously learned tasks [134], [135], [136]. Practically, this approach is not reliable since, due to privacy constraints or data openness policy, previously learned data can be unavailable anymore. In addition, the rehearsal process increases memory consumption and computational costs.

Leveraging the presence of the foundation model, recent SOTAs utilized a pre-trained model (PTM) that carries a generalization attribute as initial weight [142], [143], [144]. However, the SOTAs still store and replay previously learned exemplars or features in their memory. The insight tells us that rehearsal-free online continual learning remains an open problem. On the other hand, the prompt-based approach has been proven to be effective in rehearsal-free continual learning. Spe-

cifically, the growing prompt components methods [39], [65], [87] outperform the prompt-pool method [37] and task-wise prompt methods [38], [84], [85], [86] both in terms of accuracy and forgetting measure. However, from the perspective of OCL, the growing components prompt methods violates the throughput aspects, as the model parameters keep growing with the increasing number of classes. This growth incurs more executed operations and longer training time, which leads to more skipped (unprocessed) data streams. Thus, the approach is not favorable for dealing with catastrophic forgetting in streaming data.

These intriguing insights motivate us to develop a novel method that addresses the OCL problem. This chapter develops the proposed method based on two compulsory aspects in OCL, i.e., (1) achieving stability-plasticity and (2) being efficient both in training and inference phase. To achieve those 2 mandatory components, we design our method with the following components: (1) a single lightweight prompt generator for all tasks, (2) a set of scalers and shifters associated with the class-wise learnable keys, (3) PTM generalization preservation via cross-correlation matrices, and (4) hard-soft learning updates for a better parameter tuning. The generator is trained only on the first task to avoid forgetting and achieve stability, while scalers and shifters are trained on every task to achieve plasticity. To achieve high throughput, our lightweight generator has fewer parameters than a single generator in [65] and fewer than a single task-specific prompt in [38].

This chapter proposes four main contributions as follows:

1. This chapter highlights rehearsal-free constraint in online continual learning and proposes a novel prompt online learning (PROL) that applies the aforementioned components.

2. A design of a new mechanism to generate prompts by a single generator with a few smaller-shifter parameters, and a joint loss function for the proposed method.

3. Rigorous experiments and analysis proving the superiority of our method compared to the existing SOTAs in general and historical performance. The analysis shows that our proposed method has relatively low parameters, mod-

erate training, inference time, and throughput compared to the SOTAs. The entended analysis proves the robustness of the proposed method in various settings, i.e., different prompt lengths and numbers of ViT layers.

4. Extensive analysis and discussion both theoretically and numerically, and we provide and complete numerical results for further study.

9.2 Preliminary

9.2.1 Problem Formulation

Class incremental learning (CIL) problem is defined as the problem of learning a sequence of fully supervised tasks $\{\mathcal{T}^t\}_{t=1}^{T}$, where after finishing learning a task $\mathcal{T}^t$, a model must recognize all learned tasks, i.e., $\mathcal{T}^1$,$\mathcal{T}^2$,...$\mathcal{T}^{t-1}$,$\mathcal{T}^t$. Symbol T represents the number of consecutive tasks. Each task carries pairs of training samples, i.e., $\mathcal{T}^t = \{(x_i^t, y_i^t)\}_{i=1}^{|\mathcal{T}^t|}$ where $x_i \in \mathcal{X}^t$ and $y_i \in \mathcal{Y}^t$ denotes input image and corresponding label, while $|.|$ denotes cardinality. $\mathcal{C}^t$ denotes the unique class labels in $\mathcal{Y}^t$, i.e., $\mathcal{C}^t = unique(\mathcal{Y}^t)$, and $|\mathcal{C}^t|$ denotes the number of classes in $\mathcal{T}^t$.

Online Class incremental learning (OCIL) is similar to CIL but the data comes in streams $s_1, s_2, s_3, ...$ where $s_j = \{(x_i^t, y_i^t)\}_{i=1}^{|s_j|}$ rather than in a storable $\mathcal{T}^t = \{(x_i^t, y_i^t)\}_{i=1}^{|\mathcal{T}^t|}$ manner. $|s_j|$ is relatively small, e.g., 10. The data can be seen only once, as the data will not be available after arriving. Following [50], there are three main settings in OCL, i.e., disjoint task where each task t is disjoint from another task t' i.e., $\forall t, t' \neq t, (\mathcal{T}^t \cap \mathcal{T}^{t'} = \emptyset)$, blurry task setting where similar to single task setting with data sampling, and iBlurry setting where consist of $N\%$ of the classes are put into disjoint tasks and the other $(100 - N)\%$ classes appear in every task. In this study, we focus on the disjoint task setting that is the most common in OCL and CL.

9.2.2 The Weakness of Existing OCL Methods

Figure 2.6 in chapter 2 shows a topology that the OCL methods can be divided into 2 main categories, i.e., OCL methods without PTM (Non-PTM) and with PTM. The main weakness of this category is that it tunes the whole backbone

$f_\theta(.)$ and classifier $g_\phi(.)$ from scratch in the case of streaming data where the data can be seen only once. In addition, the backbone training in every task increases the chance of model forgetting. Even with the presence of sample memory to avoid forgetting, the methods can not perform well since its weight is not yet convergent. A customized backbone tuning technique such as the proximal point method, real-synthetic similarity maximization, dual view, or adaptive layer freezing [132], [133], [134], [135], [136], or customized optimization method as in [52], [53] can not compensate the insufficient training and avoid model forgetting. The other approach trains dual models, i.e., student ($f_\theta^{student}$) and teacher ($f_\theta^{teacher}$) backbones. Even though one of the model is frozen, this approach requires higher training complexity (longer training time) and reduces model throughput. The prototype approach replaces the MLP classifier (g_θ) with a prototype-based classifier (g_Z). Despite reducing model forgetting, the prototypes do not optimally discriminate the learned classes since the prototypes are calibrated with once-seen data.

On the other side, the PTM-based method leverages PTM generalized weight as its initial values. NSCE[142] transfers the PTM weight and then fine-tunes its weight along with the classifier to adapt to the streaming data. Despite achieving better performance than non-PTM methods, it still requires memory samples that may not be applicable in real-world applications due to data privacy or openness policy. Feature projection network, e.g., RanPAC[144] and RanDumb[143] transform PTM features into a more discriminative form by a projection network Prj_ψ. Similarly, this approach requires memory (in feature space) from a previous task to be replayed in the currently learned class. The adapter methods [91], [92] that were originally for offline CL are considered as baselines in OCL. Aside from utilizing sample memory, this method has a growing number of adapters ($A_1, A_2, ..., A_t$) along with the increasing of learned classes. This mechanism is bound to increase model complexity and reduce model throughput. F-OAL[245] forms a feature fusion from its PTM layers then classify it with analytical learning (AL). Despite the method working on rehearsal-free setting, the method requires more operation for AL since the least square solver in AL requires higher computation than standard MLP or prototype-based classifier.

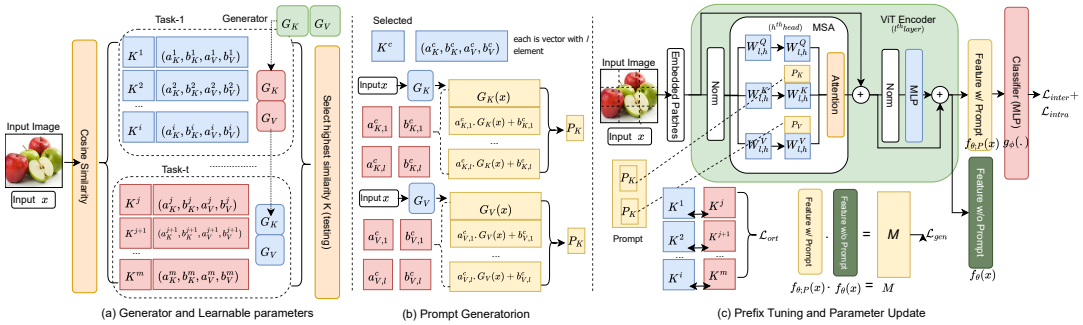


Figure 9.1: The proposed PROL diagram. PROL utilizes a single generator (G_K, G_V) for all classes in all tasks, learnable scalers (a_K^i, a_V^i), and shifters (b_K^i, b_V^i) parameters asocciated to the class-wise key K^i. The generator is trained only in the first task and frozen afterward, while the scalers, shifters, and class-wise keys are trained in every task. The prompt is generated by scaling and shifting $G(x)$ by the selected scalers and shifters associated with the highest similarity key. The model is updated by joining the loss function consisting of intra-task loss $\mathcal{L}_{intra}$, inter-task loss $\mathcal{L}_{intra}$, orthogonality loss $\mathcal{L}_{ort}$, input-to-key similarity loss $\mathcal{L}_{sim}$ and generalization loss $\mathcal{L}_{gen}$.

9.3 Proposed Method: Prompt Online Learning (PROL)

In this study, we propose prompt online learning (PROL) to handle catastrophic forgetting in streaming data. The goal of the method is to achieve stability-and-plasticity and high throughput. Thus, we utilize only a single lightweight generator for all classes in all tasks. The generator is trained only in the base task, then frozen afterward to maintain stability, and assisted by learnable scalers and shifters to achieve plasticity. Figure 9.1 shows the architecture and flow of PROL, i.e., fig.9.1(a) shows the generator and learnable parameters, part fig.9.1(b) shows the prompt generation mechanism, and fig.9.1(c) shows the model forward, loss and parameter updates. The following sub-sections explain the details of the components and learning mechanism.

9.3.1 Learnable Parameters:

9.3.1.1 Single Lightweight Generator for All

We design a lightweight generator (G_K, G_V) to achieve the highest possible throughput in online continual learning that trained only in the first task yet can be utilized

in all tasks. We design it in a pair following the prompt structure in prefix tuning[83] that divides prompt P into two parts, i.e., P_K and P_V that prepend into K and V of the ViT layer's MSA, respectively. Following ConvPrompt[65], the generator is distributed into all MSA heads of the ViT layers. Thus, the generator can be noted as $(G_K, G_V) = \{(G_{K,l,h}, G_{V,l,h})\}, l \in [1..L]h \in [1..H]$, where L and H are the number of layers and MSA heads per layer respectively. Each $G_{K,l,h}$ or $G_{V,l,h}$ is a 1DCNN kernel with the size of 3, thus only saving 3 parameters. Therefore, the total parameters of our generator are only $2 \times H \times L \times 3$, which is generally less than 1K.

9.3.1.2 Learnable Scalers-and-Shifters and Class-wise Key

To assist the generator in achieving plasticity, we train a learnable scalers-and-shifters $(a_K^c, b_K^c, a_V^c, b_V^c)$ associated with learnable class-wise key K^c (fig9.1(a)), where c is the class label meant to be represented by K^c. Following the generator pair, we can specify that (a_K^c, b_K^c) and (a_V^c, b_V^c) are the scallers-and-shifters for G_K and G_V respectively. However, the scaller and shifters are not distributed into all MSA heads and layers but are expanded based-on prompt length (l). Thus, each element of the tuple, e.g., a_K^c is a vector with $l-1$ elements. Note that the first length ($l = 1$) is not scaled or shifted, but left original $G(x)$. In this study, we utilized small prompt length as in DualPrompt[38], i.e., $l = 5$ to maximize the model throughput. Based on that setting, the total shifter for each class is $4 \times 4 = 16$ parameters. To avoid uncontrollable values of $(a_K^c, b_K^c, a_V^c, b_V^c)$ we deploy an upper and lower bounds both for K and V scalers and shifters formulated by the expression below:

$$(1 - \epsilon_a) \leq a^c \leq (1 + \epsilon_a), \text{ and } -\epsilon_b \leq b^c \leq \epsilon_b \tag{9.1}$$

where ϵ_a and ϵ_b are small positive real numbers. The boundaries are deployed to avoid too much perturbation of $G_K(x)$ and $G_V(x)$ that leads to degraded performances. K^c is a $\mathbb{R}^D$ learnable vector that represent class c, where D is the embedding size. By representing its respective class, K^c plays as a reference to select proper scallers-and-shifters during inference time based on its similarity to input x.

9.3.2 Prompt Generation

We propose a novel prompt generation mechanism i.e., by single lightweight generators and scalers-and-shifters. The mechanism is unique to existing prompt-generation methods [39], [65], [87]. Given prompt generator (G_K, G_V) and scaler-and-shifters $(a_K^c, b_K^c, a_V^c, b_V^c)$ where each element as l-elements vector, the prompt is generated by executing the following formula.

$$\begin{aligned} P_K = [P_{K,1}; P_{K,2}; ...; P_{K,l}],\ P_V = [P_{V,1}; P_{V,2}; ...; P_{V,l}] \\ P_{K,i} = a_{K,i-1}^c G_K(x) + b_{K,i-1}^c, \\ P_{V,i} = a_{V,i-1}^c G_V(x) + b_{V,i-1}^c, i \in [2..l] \end{aligned} \tag{9.2}$$

where ";" denotes the concatenation operation. We add similarity factor s to $G(x)$, i.e., $G(x) = s.G(x)$ where s is the cosine similarity between input and closest key, i.e., $s = cosine(x, K^c)$. The addition of the similarity factor acts as a controller if the input chooses the wrong key. Following the previous study[38], [65], the matched input x can be a patched embedding, class token, or class feature that we set as the default in this study.

9.3.3 Current-to-Previous Classes Orthogonality

We apply current-to-previous classes orthogonality to increase the discriminative capability of our method. Let the method currently at task t that learns class j, $j+1$,$j+2$...m. The model must discriminate any currently learned class $c \in [j, j+1, ..m]$ to any previously learned class $c' \in [1, 2, ..j-1]$. Due to rehearsal-free constraints, our model can not replay any samples from class c' with the samples of c. Dealing with the constraint, we utilize K^c and $K^{c'}$ that represent c and c' respectively, thus the model can always improve its discrimination capability. Following the vector principle, that states 2 orthogonal vectors have a zero dot product, our method optimizes K^c so that $K^c.K^{c'} = 0$.

9.3.4 Preserving Generalization Property

We utilize PTM that has generalization capability as the initial weight of our method. Dealing with the sequence of streaming data that may be infinite, the generalization capability must be preserved so that it can be taken benefit by the

upcoming data. Following the previous study[246], we preserve the generalization capability by cross correlating the feature of PTM (only), i.e., $f_\theta(x)$ and the feature of PTM and prompt, i.e., $f_{\theta;P}(x)$ into a matrices M by executing the following equation.

$$M = 1/B[f_\theta(x)]^T[f_{\theta;P}(x)] \tag{9.3}$$

M is $D \times D$ matrix, while $f_\theta(x)$ and $f_{\theta;P}(x)$ are $B \times D$, where B and D is batch and embedding sizes respectively. The total quantity of the diagonal elements of M shows the alignment between the PTM feature ($f_\theta(x)$) and the PTM+prompt tuning feature ($f_{\theta;P}(x)$). The higher the quantity, the higher the generalization preserving by the model.

9.3.5 Learning, Losses, and Final Objectives

In this subsection, we explain the learning objective of our method. For a better understanding, we explain it in per-component way as follows.

9.3.5.1 Tuning Generators and scaler-and-shifters

These 2 components work together to generate prompt, where the prompt is prepended into ViT layers and go through until the last module, i.e., MLP classifier. Along with the MLP classifier, these two components are tuned by using cross entropy loss that derived into two types, i.e., intra-task loss ($\mathcal{L}_{intra}$) and inter-task loss $\mathcal{L}_{inter}$. $\mathcal{L}_{intra}$ improves model discrimination within a task, specifically, currently learned classes, while $\mathcal{L}_{inter}$ improves model discrimination for all learned classes, including current and old classes. The intra and inter-task loss are formulated as in equations 9.4 and 9.5.

$$\mathcal{L}_{intra} = -\sum_{c \in C^t} log \frac{exp(g_\phi(f_{\theta;P}(x))[c])}{\sum_{c' \in C^t} exp(g_\phi(f_{\theta;P}(x))[c'])} \tag{9.4}$$

$$\mathcal{L}_{inter} = -\sum_{t=1}^{t=T} \sum_{c \in C^t} log \frac{exp(g_\phi(f_{\theta;P}(x))[c])}{\sum_{t=1}^{t=T} \sum_{c' \in C^t} exp(g_\phi(f_{\theta;P}(x))[c'])} \tag{9.5}$$

9.3.5.2 Tuning Task-wise learnable key

K^c has two important roles in our method, i.e., as the class representation to select scalers-and-shifters and improving model discrimination through orthogonality. As the class representation, we tune K^c tuned closer to the input sample of class c with similarity loss $\mathcal{L}_{sim}$ as expressed in the equation 9.6.

$$\mathcal{L}_{sim} = -(x.K^c)/(max(||x||_2.||K^c||_2, \epsilon)) \tag{9.6}$$

To improve class discrimination, we tune any K^c from currently learned classes to be orthogonal to any $K^{c'}$ from previously learned classes. let $y = [c_1, c_2, ...c_B]$ is the class labels of current batch-wise input x, $K_{new} = [K^{c_1}, K^{c_2}, ...K^{c_B}]$ is the class-wise keys matched with y, and $K_{old} = [K^{c'_1}, K^{c'_2}, ...K^{c'_B}]$, then the model will be improved by orthogonality loss $\mathcal{L}_{ort}$ as formulated in equation below.

$$\mathcal{L}_{ort} = \frac{1}{B}\sum_{i=1}^{B} K^{ci}.K^{c'i} \tag{9.7}$$

9.3.5.3 Tuning Generalization preserving Matrix

The diagonal elements of matrix M represent the agreement between PTM-only prototype ($f_\theta(x)$) and PTM+Prompt prototype ($f_{\theta;P}(x)$), while the non-diagonal element represents the disagreement between them. Thus we tune the generalization matrix M with generalization loss $\mathcal{L}_{gen}$ as formulated in the equation 9.8. The first term of $\mathcal{L}_{gen}$ push the model towards the agreement, while the second term of $\mathcal{L}_{gen}$ avoids the disagreement of the prototypes.

$$\mathcal{L}_{gen} = \frac{1}{D}\sum_{i=1}^{D}(1 - M_{i,i})^2 + \frac{1}{D(D-1)}\sum_{i=1}^{D}\sum_{j\neq i} M_{i,j}^2 \tag{9.8}$$

9.3.5.4 Joint Loss and Final Objective

Finally, accommodating the losses for all components of our proposed method, i.e., single lightweight generators, scalers-and-shifter, class-wise keys, and generalization matrix, we define a new joint loss function as in equation 9.9. To enhance the

flexibility of our model we add loss coefficients $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$.

$$\begin{aligned}\mathcal{L}_{total} =&\lambda_1\mathcal{L}_{intra} + \lambda_2\mathcal{L}_{inter} + \lambda_3\mathcal{L}_{sim}+\\ &\lambda_4\mathcal{L}_{ort} + \lambda_5\mathcal{L}_{gen}\end{aligned} \tag{9.9}$$

Algorithm 8 PROL Training

Input: A sequence of tasks $\mathcal{T}^1$, $\mathcal{T}^2$,...,$\mathcal{T}^T$, a frozen pre-trained ViT $f_{\theta(.)}$,
Initiate trainable generator (G_K, G_V) and MLP head g_ϕ
for $t = 1 : T$ **do**
    stream data $s = \{(x_i, yi)\}$ from $\mathcal{T}^t$
    initiate (K^c, a^c, b^c) for any all $c \in s$ if not exist before
    if $t > 1$ **then**
        Freeze (G_K, G_V)
    end if
    Generate prompt (P_K, P_V) following eq. (9.2)
    Compute PTM only and PTMP+prompt prototypes: $f_\theta(x)$, and $f_{\theta;P}(x)$
    Compute logits: $g_\phi(f_{\theta;P}(x))$
    Compute $\mathcal{L}_{intra}$ following eq. (9.4)
    Compute $\mathcal{L}_{inter}$ following eq. (9.5)
    Compute $\mathcal{L}_{sim}$ following eq. (9.6)
    Compute $\mathcal{L}_{ort}$ following eq. (9.7)
    Generate M and Compute $\mathcal{L}_{gen}$ as in eq. (9.8)
    Compute $\mathcal{L}_{total}$ following eq. (9.9)
    if $t = 1$ **then**
        Update (G_K, G_V) parameters
    end if
    update (K^c, a^c, b^c) and ϕ
    Clamp (a^c, b^c) following eq.(9.1)
end for
Output: Optimum generator (G_K, G_V), parameters (K^c, a^c, b^c) and ϕ

9.3.5.5 Hard-Soft Update (HSU) Strategy

Dealing with streaming data, the learning cannot be repeated over several epochs, so we devise a learning strategy for updating the model. We utilize Hard-Soft Update (HSU) where the model adaptively switches from-and-into hard update, i.e., update with constant and high learning rate, and soft update, i.e., update with decayed learning rate. The hard update is started when the model see new class

Algorithm 9 PROL Inference

1: **Input:** An input x, a frozen pre-trained ViT $f_{\theta(.)}$, optimized generator (G_K, G_V) parameters (K^c, a^c, b^c) and ϕ
2: Find top-1 K^c where $c \in \mathcal{T}$
3: Generate prompt (P_K, P_V) following eq. (9.2)
4: Compute logits by forwarding the input, i.e., $logits = g_\phi(f_{\theta;P}(x))$
5: Compute Pedicted label $\hat{y} = argmax(logits)$
6: **Output:** Predicted label $\hat{y}$

labels and switch into soft update if the cross entropy loss, i.e., $\mathcal{L}_{ce} = \lambda_1 \mathcal{L}_{intra} + \lambda_2 \mathcal{L}_{inter}$, is less than a loss threshold $\mathcal{L}_{thres}$. The decay is executed by a cosine schedule.

9.3.6 Algorithm

This section presents the PROL training and inference algorithm that is presented in algorithm 8 and 9, respectively.

Table 9.1: Summarized performance of consolidated methods in 4 benchmark datasets, i.e., CIFAR100, Imagnet-R, Imagnet-A, and CUB200.

Method	Rhs	CIFAR100			ImageNet-R			ImageNet-A			CUB200		
		FAA	CAA	FM	FAA	CAA	FM	FAA	CAA	FM	FAA	CAA	FM
RanPAC-J	Yes	79.94	85.91	6.43	52.09	56.21	6.63	45.03	53.30	6.97	85.20	86.31	2.63
RanDumb-J	Yes	80.39	86.15	6.25	52.63	56.13	6.04	44.07	52.30	4.98	85.82	86.84	2.52
RanPAC-R	Yes	59.66	69.35	34.30	28.16	35.37	39.81	28.54	35.16	32.05	75.63	79.37	14.65
RanDumb-R	Yes	50.93	62.14	47.35	21.91	29.90	48.81	24.86	30.82	34.79	73.81	78.70	17.45
RanPAC	No	9.52	27.92	94.52	7.24	18.81	67.27	3.77	14.86	57.76	9.52	27.06	88.98
RanDumb	No	9.56	27.94	94.51	7.32	18.80	67.67	3.69	14.56	55.15	9.53	27.21	89.69
L2P	No	80.99	86.66	6.71	52.08	55.90	5.67	14.64	21.88	5.35	61.98	71.46	9.81
DualPrompt	No	82.53	88.06	4.53	61.17	62.73	3.75	20.05	21.57	1.91	58.06	49.95	6.23
ConvPrompt	No	84.79	88.19	2.38	70.92	74.66	3.39	-	-	-	70.12	77.48	7.21
MOS	No	83.99	89.53	11.34	52.13	67.97	35.72	44.79	55.14	15.50	61.74	77.03	25.25
PROL	No	86.12	91.47	8.14	73.50	78.13	4.82	47.72	58.96	3.29	72.51	79.89	8.92

9.4 Experiment Results and Analysis

9.4.1 Dataset

We evaluate our method in the 4 most popular datasets in CL, i.e., CIFAR100[172], ImageNet-R[247], ImageNet-A[248], and CUB[184]. CIFAR100 contains 100 classes

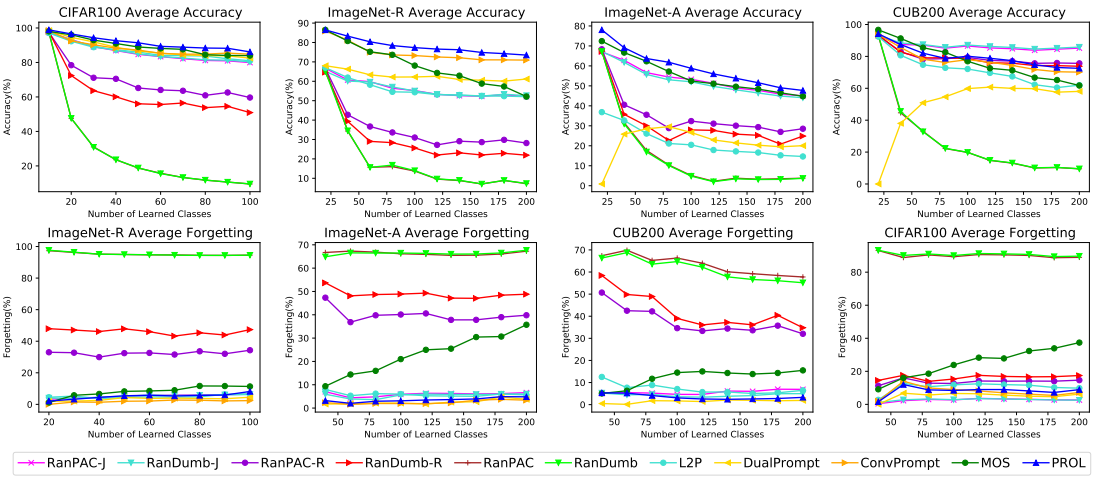


Figure 9.2: Historical performance plot of the consolidated methods in 4 benchmark datasets, i.e., CIFAR100, Imagnet-R, Imagnet-A, and CUB200.

of images, while the other has 200 classes. We uniformly split the dataset into 10 tasks.

9.4.2 Baseline

We compare our proposed method with the current SOTAs of PTM-based OCL, i.e., RanDumb[143], RanPAC[144], and MOS[92], the extension of EASE[91]. We adapt the official code of RanDumb and RanPAC from [143] into our rehearsal-free and disjoint setting. We also adapt the rehearsal version (denoted with -R) with 5 samples per class as memory, and the joint version (denoted with -J) for both of them. We adapted MOS from its official code into rehearsal free setting. We add L2P [37], DualPrompt[38], and ConvPrompt[65] as representations of pool-based, task-specific, and growing components prompting, respectively. We also compare our method to non-PTM methods, i.e., ER [249] DER++[57], ERASE [250], [251], GSA[132], ONPRO [51], and ERSM[135]. The compared result is taken from [135].

9.4.3 Experiment Setup

Implementation: To ensure fairness, all the consolidated methods are run with the same dataset split, backbone, i.e., ViT-B/16 pre-trained on Imagenet-21K. The learning rates for all methods are set by grid search from the range of [0.001,

0.005, 0.01, 0.05, 0.1]. All the methods are run under the same environment, i.e., a single NVIDIA A100 GPU with 40 GB RAM memory, python 3.9, and Pytorch 2.2.0., by three different random seed numbers. We utilize the Adam optimizer for PROL, while the rest follow their official implementation. The λ_1, λ_3 and λ_4 are set to 1.0, while λ_2 is set to 0.001 for CIFAR100, and 0.01 for CUB, and 0.03 for ImageNet-R and ImageNet-A. The $\mathcal{L}_{thres}$ is set to 0.3 for CIFAR100 and CUB, and 0.8 for ImageNet-R and ImageNet-A. The cosine annealing scheduler is set to maxT=20 and min-lr=0.005.

Metrics: Adapted from HidePrompt, we measure both accuracy and forgetting of the methods. Suppose that $A_{i,t}$ denotes the accuracy on the t-th task after learning the t-th task. The average accuracy of all learned task is defined as $AA_t = (1/t)\Sigma_{i=1}^{t} A_{i,t}$. Suppose that T is the number of all tasks, we measure final average accuracy (FAA), cumulative average accuracy(CAA), and final forgetting measure (FM)

$$FAA = AA_T \tag{9.10}$$

$$CAA = \frac{1}{T}\Sigma_{t=1}^{T} AA_t \tag{9.11}$$

$$FFM = \frac{1}{T-1}\Sigma_{i=1}^{T-1} max_{t\in 1,\ldots,T-1}(AA_{i,t} - AA_{i,T}) \tag{9.12}$$

9.4.4 Overall Performance

Table 9.1 shows the summarized performance of the consolidated method in 4 benchmark datasets. In term of accuracy, our method outperforms the existing SOTAs with a significant gap, i.e., 2-76% FAA and 2-64% CAA, respectively, except in the CUB dataset. Surprisingly, our method outperforms the joint version of RanDumb and RunPAC with a 2-20% gap both for FAA and CAA, even though those methods join all the samples' features together. Our method outperforms their replay versions (RanPAC-R and RanDumb-R) with a high margin, i.e., 20-35% FAA and CAA margins. The rehearsal free of those methods, performed extremely poor indicated by the very low FAA and CAA. Our method outperforms the prompt-based methods with a significant margin, i.e., 2-20% margin, both for FAA and CAA. ConvPrompt is the closest competitor to our method, with a 2-3%

difference in all cases. Our method outperforms the adapter method (MOS) with a significant margin, i.e., 2-21% FAA and 2-10% CAA, respectively. In CUB dataset, our method has lower performance than RanPAC-J and RanDumb-J, comparable performance with RanPAC-R and RanDumb-J, and outperforms the other methods with a convincing gap, i.e., 2-63% FAA and 2-52% CAA, respectively. In terms of forgetting, our method achieves a fairly low forgetting rate, i.e., slightly higher than Convprompt methods in many cases but significantly lower than RanPAC, RanDumb, and MOS. Our method even achieves lower forgetting in 2 of 4 datasets than RanPAC-J and RanDumb-J.

9.4.5 Historical Performance

Figure 9.2 plots the detailed performance of the consolidated methods after finishing each task. The figure shows that in CIFAR100, ImageNet-R, and ImageNet-A datasets, our method consistently achieves the highest average accuracy from the first task until the last task. In addition, the figure shows that our method has the smallest slope, which indicates that it suffers from the smaller performance degradation in the learning process. In the CUB dataset, our method consistently achieves higher performance than all rehearsal-free methods and competes with RanPAC-R and RanDumb-R, even though it has lower performance than RanPAC-J and RanDumb-J. Note that both the R and -J methods replay the previous data, and those methods are significantly outperformed by PROL in the other 3 datasets. In terms of the forgeting measure, our method consistently achieves a fairly low forgetting rate along with the prompt-based methods. The plot shows that our method is consistently in the bottom of the group while RanPAC, RanDumb, and MOS are in the middle and upper groups.

9.4.6 Parameter, Throughput and Execution time

Table 9.2 presents the parameter and the summarized throughput and execution time, while the detailed version is presented by fig.9.3. The throughput is computed in the training phase, where the learning process is conducted. The table shows that PROL requires one of the smallest numbers of trainable parameters, i.e., only slightly bigger than RanPAC and L2P, and far smaller than the rest

Table 9.2: Parameter and summarized throughput and execution time of consolidated methods in ImageNet-R dataset.

Method	#Trainable Parameter In million (M)	Average Throughput (sample/s)	Average Training Time (s)	Average Inference Time (s)
RanPAC-J	0.154	82.92	29.60	55.24
RanDumb-J	7.834	77.03	32.00	53.79
RanPAC-R	0.154	89.46	28.30	84.20
RanDumb-R	7.834	116.89	21.20	43.36
RanPAC	0.154	86.83	27.70	45.21
RanDumb	7.834	85.90	29.90	46.81
L2P	0.200	40.94	59.80	54.50
DualPrompt	0.701	46.50	53.20	52.60
ConvPrompt	1.434	11.77	235.70	103.92
MOS	3.197	16.06	165.70	322.20
PROL	0.213	35.42	68.70	55.39

of the methods. RanDumb saves the biggest ($>$ 7M) RBF kernel parameters, while ConvPrompt and MOS require 1.4M and 3.2M parameters, respectively. The table show that PROL has a moderate throughput, i.e., comparable with L2P and DualPrompt, lower than RanDumb and RanPAC. In line with this, PROL achieves moderate training and inference time. RanDumb and RanPAC have higher throughput and lower execution time since the methods update their feature projection and classifier only once, not iteratively per data stream. In the streaming process, it only extracts the input feature, stores it in the memory, and then updates the projection and classifier only once after the current task is finished. The weakness of this mechanism is that it can not handle any time inference where a request for inference comes in the middle of the streaming. Figure 9.3 confirms that PROL has moderate throughput, training time, and testing time from the first task until the last task. In terms of throughput, it competes with DualPrompt and L2P and is far better than ConvPrompt and MOS. ConvPrompt and MOS low throughput and high execution time confirm our preliminary analysis on growing prompt components/adapters.

9.4.7 Robustness Analysis

Table 9.3 shows the robustness of our method in comparison to the closest competitor, i.e., ConvPrompt and DualPrompt, on different prompt lengths and the

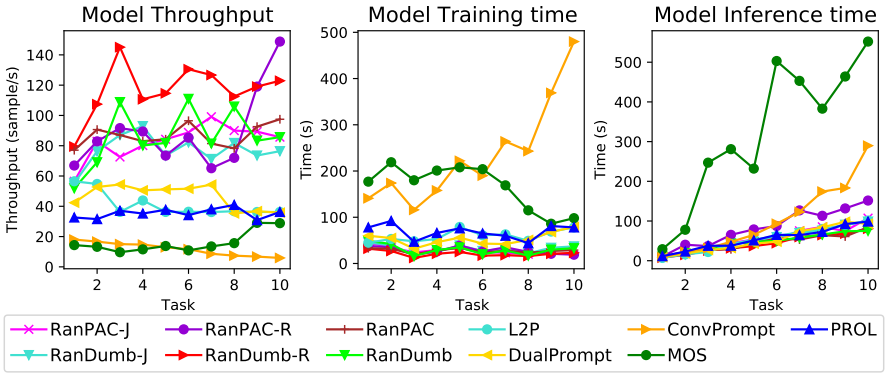


Figure 9.3: Detailed throughput and execution time of consolidated methods in Imagnet-R dataset.

number of ViT layers pre-pended. The table shows that PROL successfully maintains its performance in different prompt lengths and numbers of layers, even in the smaller prompt length or fewer layers. In addition, it maintains the performance gap to the ConvPrompt as the closest competitor with the convincing margin, i.e., $\geq 2\%$. The table shows that PROL achieves a higher CAA in higher numbers of layers. The table also shows that PROL has a better performance stability than competitor methods, where their performance swing is in 2% and 10% different for ConvPrompt and DualPrompt, respectively.

9.4.8 Ablation Study

We conducted an ablation study to evaluate the contribution of each component of our method as presented in Table 9.4. The table shows that fine-tuning (FT) alone can not handle OCL problem since the performance is too low. However, combining both inter-task loss $\mathcal{L}_{inter}$ and intra-task loss $\mathcal{L}_{intra}$ has 1-2% better accuracy than utilizing only one. The generator (G) plays the most important role as it tremendously boosting the accuracy and reduce the forgetting, i.e., up to 57%. The learnable scaler-and-shifters (SS) and class-wise keys (K) and input-to-key similarity loss $\mathcal{L}_{sim}$ improve the performance with a 1% margin. At this point, our method is comparable to ConvPrompt as the closest competitor and even achieves higher CAA. Adding orthogonality loss $\mathcal{L}_{ort}$ on K improves both FAA and FAA even though with a small margin. Applying the hard-soft-update strategy improves the model accuracy with 1.5% FAA and 1.2% CAA improvement, respectively.

Table 9.3: Robustness analysis of the proposed method compared to the competitor methods in ImageNet-R dataset.

Robustness on Different Prompt Length						
Prompt	DualPrompt		ConvPrompt		PROL	
Length	FAA	CAA	FAA	CAA	FAA	CAA
3	49.76	41.13	69.88	72.89	73.42	77.71
5	60.08	61.17	70.92	74.66	73.50	78.13
7	51.19	42.53	70.68	74.11	73.43	77.92
10	50.51	41.84	70.02	74.18	73.45	78.03
15	51.35	43.09	69.45	72.24	73.30	77.72
Sensitivity on Different Number of ViT Layers						
Number of	DualPrompt		ConvPrompt		PROL	
Vit Layers	FAA	CAA	FAA	CAA	FAA	CAA
3	49.33	49.79	71.43	75.95	73.72	77.80
5	48.92	48.69	70.92	74.66	73.50	78.13
7	60.08	61.17	69.38	74.21	73.55	78.14
9	-	-	72.12	77.47	73.75	78.69

Table 9.4: Ablation study of the proposed method in ImageNet-R dataset.

Component	**Loss**	**FAA**	**CFA**	**FM**
FT	$\mathcal{L}_{inter}$	11.98	26.25	75.72
FT	$\mathcal{L}_{inter}$+$\mathcal{L}_{intra}$	12.68	28.13	81.15
FT+G	$\mathcal{L}_{inter}$+$\mathcal{L}_{intra}$	69.77	74.43	3.25
FT+G+SS+K	$\mathcal{L}_{inter}$+$\mathcal{L}_{intra}$+$\mathcal{L}_{sim}$	70.82	75.83	3.96
FT+G+SS+K	$\mathcal{L}_{inter}$+$\mathcal{L}_{intra}$+$\mathcal{L}_{sim}$+$\mathcal{L}_{ort}$	71.13	76.09	4.31
FT+G+SS+K+HSU	$\mathcal{L}_{inter}$+$\mathcal{L}_{intra}$+$\mathcal{L}_{sim}$+$\mathcal{L}_{ort}$	72.53	77.24	4.95
FT+G+SS+K+HSU+M	$\mathcal{L}_{inter}$+$\mathcal{L}_{intra}$+$\mathcal{L}_{sim}$+$\mathcal{L}_{ort}$+$\mathcal{L}_{gen}$	73.50	78.13	4.82

However, it increases the forgetting along with the accuracy. Finally, preserving the PTM generalization property by matrix M and generalization loss $\mathcal{L}_{gen}$ improves the accuracy with 1% margin, and reduces a slight magnitude of forgetting.

9.4.9 Complexity Analysis

This section presents the complexity analysis of our proposed method. Following algorithm 1, the complexity of PROL can be formulated as:

$$O(PROL) = \sum_{t=1}^{T} \sum_{i=1}^{S^t} O(per - stream) \tag{9.13}$$

where S^t is the number of streams in task t. Given a stream s on task t, PROL

executes several operations as presented in lines (5) to (22). All the operations are O(1). Thus PROL complexity can be derived into

$$O(PROL) = \sum_{t=1}^{T} \sum_{i=1}^{S^t} O(|s|) \tag{9.14}$$

Where $|s|$ represents the stream's size. Since $\sum_{i=1}^{S^t} |s| = |\mathcal{T}^t|$ then the PROL complexity becomes

$$O(PROL) = \sum_{t=1}^{T} O(|\mathcal{T}|) \tag{9.15}$$

Since $\sum_{t=1}^{T} |\mathcal{T}| = |\mathcal{T}| = N$ then the PROL complexity summarized as

$$O(PROL) = O(N) \tag{9.16}$$

9.5 Summary

This chapter highlights the rehearsal-free aspect of continual learning in streaming data and proposes a novel rehearsal-free method that incorporates four key concepts: (1) Single lightweight prompt generator for all instances in all classes, (2) learnable scalers-and-shifters parameters associated with learnable class-wise keys, (3) a cross-correlation matrix for preserving PTM generalization, and (4) a hard-soft update for adaptive learning rate assignment. Our experiment and analysis show the superiority of our method to existing SOTAs in CIFAR100, ImageNet-R, ImageNet-A, and CUB dataset with significant margins, i.e., 2-76% and 2-64% margin for FAA and CAA, respectively. Our method excellently outperforms the joint version of existing SOTAs in 3 datasets while running in a rehearsal-free manner. Our historical analysis shows the consistency of our method in maintaining stability-plasticity in every task. Our method achieves moderate training time, and throughput that is significantly better than the existing prompt generator or adapters method. Our robustness analysis shows that our proposed method consistently achieves the highest performance in various prompt lengths and numbers

of ViT layers.

The next chapter is devoted to extend the continual learning concept into a regression domain, i.e., dynamic long-term time-series forecasting. Dynamic long-term time-series forecasting is one of the challenging machine learning problems where the model need to predict far in the future while the trends of the data may change over time. In addition, there is data drift that obstruct the model from forecasting the upcoming series.

Chapter 10

MANTRA : Dynamic Long-Term Time-Series Forecasting via Meta Transformer Networks

The previous chapters discuss CL and its sub-problem, which is applied in the classification domain and image dataset. This chapter proposes a study on extending the continual learning concept into a regression domain, i.e., dynamic long-term time-series forecasting. A reliable long-term time-series forecaster is highly demanded in practice but comes across many challenges, such as low computational and memory footprints, as well as robustness against dynamic learning environments. This chapter proposes Meta-Transformer Networks (MANTRA) to deal with the dynamic long-term time-series forecasting tasks. MANTRA relies on the concept of fast and slow learners where a collection of fast learners learns different aspects of data distributions while adapting quickly to changes. A slow learner tailors suitable representations to fast learners. Fast adaptations to dynamic environments are achieved using the universal representation transformer layers, producing task-adapted representations with a small number of parameters. The experiments in four datasets with different prediction lengths demonstrate the advantage of our approach with significant improvements over the baseline algorithms for both multivariate and univariate settings.

Part of this chapter is published as: MA Ma'sum, MD Sarkar, M Pratama,

S Ramasamy, S Anavatti, L Liu, R Kowlczyk. Dynamic Long-Term Time-Series Forecasting via Meta Transformer Networks. IEEE Transactions on Artificial Intelligence 5 (8), 4258 – 4268. https://doi.org/10.1109/TAI.2024.3365775

10.1 Introduction

Time series forecasting problems play a vital role in many applications where they aim to predict future values given a sequence of current and past observations. This problem becomes even more challenging than that in the long-term setting, where a model has to handle long sequences. In addition, a time-series forecasting problem is inherent with non-stationary behaviors, confirming the case of a dynamic long-term time-series forecasting problem, i.e., the offline time-series forecaster trained in the offline fashion is quickly outdated and requires a retraining process from scratch if a new data pattern is observed. In a nutshell, the long-term time-series forecasting problem calls for a machine learning algorithm to adapt quickly to changes with low computational and memory overheads.

The inception of deep learning technologies has revolutionized the field of machine learning including the time-series forecasting problems. Deep learning approaches include an automatic feature engineering step exploring hidden patterns of data samples, thereby resulting in improved generalization powers. Nevertheless, it is generally accepted that the use of deep learning strategies in the dynamic cases remain challenging. Recent research efforts [252], [253], [254], [255], [256] have been devoted to study the case of dynamic environments but all of them are designed for classification rather than regression. In addition, these approaches are not designed for long-term time-series forecasting cases involving long predictive horizons.

An ensemble approach [257], [258], [259], [260], [261] is one of the most proven techniques for dealing with non-stationary environments because it allows each base learner to master different regions of data space and handles the bias-variance problem better than the single model approach. The influence of base learners can be controlled in such a way as to reflect the current concept. The key lies in the online model selection evaluating each base learner for inferences. Most ensemble approaches are still trained in the traditional manner without fully exploiting the

representational learning power of deep learning technologies. [262] proposes a transformer approach with the self-supervised or contrastive learning approach making it possible to extract meaningful representations for downstream tasks. It also offers suitable augmentation strategies of contrastive learning, usually applied for visual data, for time series data. The transformer offers some potential for time-series forecasting tasks compared to RNNs and LSTMs [263], where it offers a simpler structure than the two approaches and ignores the recurrent structure. However, the contrastive learning strategy is slow and calls for large mini-batch sizes. The transformer approach also does not work well for long-term time-series forecasting problems because of the high computational and memory costs of full attention mechanisms [264]. A sparse version of the self-attention module is used to scale well in the long-term context [263] but suffers from reduced information. [264] proposes the autoformer model going beyond the transformer-based approach where the key lies in the concept of auto-correlation and series decomposition. Despite its power for time-series forecasting, this approach has not been investigated for dynamic settings. Generally speaking, the application of deep learning approaches for dynamic long-term time-series forecasting is scarce and deserves an in-dength study.

Meta-transformer network (MANTRA) is proposed in this chapter for dynamic long-term time-series forecasting problems. MANTRA proposes the concept of the extended dual networks [265] putting forward a slow learner and an array of fast learners working cooperatively. The slow learner functions to produce useful representations for downstream tasks via the self-supervised learning procedure, while an ensemble of fast learners are deployed to explore different regions of data space and to adapt quickly to changing data distributions. That is, the slow training procedure of the self-supervised training approach is addressed by decoupling the array of fast learners from the slow learner. The issue of drifting distributions is overcome by the proposal of a universal representation transformer (URT) concept producing task-adapted representations with few parameters [266] assuring fast adaptations. The original URT layer is generalized for dynamic time-series forecasting problems rather than image classification problems and for task-adapted representations of the ensemble of fast learners. MANTRA is based on the

autoformer [264] approach in this chapter putting forward the auto-correlation and series decomposition modules structured under the encoder-decoder configuration. The series decomposition module enables separation between trend-cyclical and seasonal parts, pinpointing long-term dependencies. Nonetheless, the concepts of MANTRA can be generalized to other network structures, such as Informer [263], FEDformer [267]. It also fits well for recent findings [268] showing the advantage of linear modules over transformer approaches.

This chapter offers three major contributions:

1. The concept of extended dual networks is proposed for dynamic long-term time-series forecasting problems. Ours differ from [265] due to the use of autoformer rather than convolutional learners and the problem of dynamic time-series forecasting. We also replace the slow learner objective, minimizing the controlled reconstruction loss [269], allowing less computations and being less sensitive to augmentations than that of the Barlow Twins approach [270]. In addition, the concept of extended dual networks fits an ensemble of fast learners rather than a single fast learner;

2. The URT concept is proposed here for the dynamic long-term time-series forecasting problems to adapt quickly to the concept changes. Our approach distinguishes itself from [266], where the URT concept dynamically aggregates the final output of the array of fast learners rather than the backbone networks for few-shot learning;

3. Rigorous analysis on 4 popular time-series datasets, both in multivariate and univariate settings, and the significant improvement of the proposed method.

Our numerical results show that MANTRA delivers the most encouraging results compared to baseline algorithms with at least 3% margins.

10.2 Preliminary

10.2.1 Problem Formulation

Given a time-series X with a length of T, $X = [x_1, x_2, ..., x_T]$, where x_t is a component of time-series at a $t-th$ time instant, the goal of time-series forecasting problem is to perform step-ahead predictions $t \geq T$ using a model f parameterized with θ where the model maps the input space to the target space $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$ and $x_t \in \Re^D$ and $y_t \in \Re$ are the input and target of the model. D is the number of features that forms a single observation x_t at the $t-th$ time index and $f_\theta(.)$ is represented by the autoformer here [264] but applicable to other structures. The target attribute $y_t = x_{t+\tau}$ is inherent to future values of the time-series input x_t where τ is the number of time steps. Long-term time-series forecasting is considered here, where a well-ahead prediction is made or τ is large. A fully supervised time-series forecasting problem is considered here where $(x_t, y_t) \in \mathcal{X} \times \mathcal{Y}$. The underlying challenge lies in the concept drift problem [271], presenting the case of changing data distributions $P(X, Y)_t \neq P(X, Y)_{t+1}$ where the speeds and magnitudes of the concept drift are hidden to the model $f_\theta(.)$. The model $f_\theta(.)$ is supposed to detect the concept drifts in a timely fashion and to recover quickly from them. The training process is governed by a minimization of a predictive loss function $\arg\min_\theta \mathcal{L}$ where $\mathcal{L} = -\log P_\theta(Y_{1:T}|X_{1:T}) = \sum_{t=1}^{T}(y_t - f_\theta(x_t))^2$.

10.2.2 Autoformer

MANTRA borrows the Autoformer concept [264] to play roles as the slow learner $g_\psi(.)$ as well as the fast learners $\{f_{\theta_i}(.)\}_{i=1}^{M}$. The Autoformer features an encoder-decoder structure having the auto-correlation block and the series-decomposition block. The series decomposition block separates the cyclical part from the seasonal part using the moving average to capture periodic fluctuations and find the long-term pattern [264]. The moving average is implemented as the average pooling operator with padding to assure consistent time-series length. The auto-correlation block unveils period-based dependencies by means of the calculation of auto-correlation and combines similar sub-series with time-delay aggregations. Similar sub-series are those in the same phase, i.e., the series are rolled according

to a selected time-delay. This mechanism is claimed to replace the self-attention mechanism seamlessly. The period-based dependencies are computed with the Fast Fourier Transform (FFT) method and combined with the output of the period-based aggregation module to obtain the auto-correlations.

The series decomposition block is designed to separate the series into trend-cyclical components and seasonal components [264] pinpointing the long-term progressions and seasonality of the series. This is done by applying the moving average to remove periodic fluctuations and highlight the long-term trends. For series with the length of L $X \in \Re^{L \times D}$, the series decomposition block is expressed:

$$\begin{aligned} X_{cy} &= AvgPool(Padding(X)) \\ X_{se} &= X - X_{cy} \end{aligned} \tag{10.1}$$

where $X_{se}, X_{cy} \in \Re^{L \times D}$ denote the seasonal and trend-cyclical parts respectively. $AvgPool(.)$ with the padding operation is implemented for the moving average, such that the length of the series remains unchanged. The series decomposition block is formalized as $X_{se}, X_{cy} = SD(X)$.

The auto-correlation module unveils the period-based dependencies by computing the series autocorrelation and aggregates similar sub-series by time delay aggregation. The time delay aggregation block rolls the series based on selected time delays $\tau_1, ..., \tau_k$, thus aligning similar sub-series that are at the same phase position of estimated periods. For a single-head situation, after projection, we obtain query Q, key K, and value V, and the auto-correlation module replaces the self-attention mechanism seamlessly.

$$\begin{aligned} \tau_1, ..., \tau_k &= \arg Topk_{\tau \in \{1,...,L\}}(R_{Q,K}(\tau)) \\ \hat{R}_{Q,K}(\tau_1), ..., \hat{R}_{Q,K}(\tau_k) &= SoftMax(R_{Q,K}(\tau_1), ..., R_{Q,K}(\tau_k)) \\ AutoCorrelation(Q, K, V) &= \sum_{i=1}^{k} Roll(V, \tau_i) \hat{R}_{Q,K}(\tau_i) \end{aligned} \tag{10.2}$$

where $\arg Topk(.)$ returns the top k autocorrelations indices while $R_{Q,K}$ denotes the autocorrelation between series Q and K. $Roll(X, \tau)$ stands for the shifting

operation to X with time delay τ. The multi-head version is realized by concatenating all heads and, in turn, producing the final output by projection via W_{out}, where each head is represented by the auto-correlation.

The encoder focuses on the seasonal characteristics of the time-series data and removes the long-term cyclical characteristics, while the decoder combines the extracted information of the encoder and other information, i.e., the seasonal part and the trend-cyclical part, to deliver the final predictions. The encoder receives a mini-batch of time-series data containing the past B records $X_B \in \Re^{B \times D}$ while the input of decoder is both the seasonal part $X_s \in \Re^{(\frac{B}{2}+O) \times D}$ and trend-cyclical part $X_c \in \Re^{(\frac{B}{2}+O) \times D}$ produced by the series decomposition block. Note that O is simply a placeholder filled with zero. The second half of the data batch is fed to the decoder to provide recent information.

$$\begin{aligned} X_{se}, X_{cy} &= SD(X_{\frac{B}{2}:B}) \\ X_s &= concat(X_{se}, X_0) \\ X_c &= concat(X_{cy}, X_{mean}) \end{aligned} \tag{10.3}$$

where $X_{se}, X_{cy} \in \Re^{\frac{B}{2} \times D}$ denote the output of the series decomposition block $SD(.)$ presenting the seasonal information and the trend-cyclical information. X_0, X_{mean} are the placeholder with zero entries and the mean of X_B respectively.

A single encoder $h^l_{enc} = Enc(h^{l-1}_{enc})$ is composed of stacked auto-correlation, series-decomposition, and feed-forward modules:

$$\begin{aligned} S^{l,1}_{en}, _ &= SD(AC(h^{l-1}_{enc}) + h^{l-1}_{enc}) \\ S^{l,2}_{en}, _ &= SD(FF(S^{l,1}_{en}) + S^{l,1}_{en}) \end{aligned} \tag{10.4}$$

where $h^l_{enc} = S^{l,2}_{en}$ is the output of the $l-th$ encoder, and $_$ stands for the eliminated long-term cyclical trend of the series decomposition module. That is, the encoder only focuses on the seasonal part of the time-series data.

As with the encoder, a decoder is configured as a stack of the auto-correlation module, the series decomposition module, and the feed-forward module. Suppose

that there exist L decoders, the $l-th$ decoder $h^l_{dec} = Dec(h^{l-1}_{dec}, h^L_{enc})$ is expressed:

$$\begin{aligned} S^{l,1}_{de}, C^{l,1}_{de} &= SD(AC(h^{l-1}_{de}) + h^{l-1}_{de}) \\ S^{l,2}_{de}, C^{l,2}_{de} &= SD(AC(S^{l,1}_{de}, h^L_{en}) + S^{l,1}_{de}) \\ S^{l,3}_{de}, C^{l,3}_{de} &= SD(FF(S^{l,2}_{de}) + S^{l,2}_{de}) \\ C^l_{de} &= C^{l-1}_{de} + \theta_{l,1} C^{l,1}_{de} + \theta_{l,2} C^{l,2}_{de} + \theta_{l,3} C^{l,3}_{de} \end{aligned} \tag{10.5}$$

where $S^{l,i}_{de}, C^{l,i}_{de}, i \in \{1,2,3\}$ denote the seasonal and trend-cyclical components after applying the $i-th$ series decomposition block $SD(.)$. $\theta_{l,i}, i \in \{1,2,3\}$ is a projection vector of the trend-cyclical components. Given L decoders, the output of autoformer is inferred by a weighted sum operation between the seasonal and trend-cyclical components $f_\theta(x) = \theta_S S^{l,3}_{de} + C^L_{de}, g_\psi(x) = \psi_S S^{l,3}_{de} + C^L_{de}$ where θ_S, ψ_S are the weight vectors to project the seasonal component into the same dimension as the output. The slow learner follows exactly the same steps as the fast learner. Fig. 10.1 exhibits the architecture of MANTRA. It comprises an ensemble of fast learners and a slow learner trained with different objectives. The features of both learners are combined before delivering the final predictions.

10.3 Meta-Transformer Networks (MANTRA)

10.3.1 Fast and Slow Learning Approach

MANTRA is developed from the concept of the extended dual-Net, where an array of fast learners and a slow learner work cooperatively to address the dynamic time-series forecasting problems. This approach addresses the slow characteristic of contrastive learning because the slow learning process is decoupled from the fast learning process. The goal of the slow learner is to extract meaningful or general representations for downstream tasks, while the fast learner adapts quickly to new patterns and combines general representations supplied by the slow learners. Our innovation here is to extend the concept of DualNet for the dynamic time-series forecasting problem, having different characteristics from those continual image classification problems, and apply the controlled time-series reconstruction strategy [269] as the slow learning approach rather than the Barlow Twins (BT) strategy [270]. The controlled reconstruction strategy is preferred over the BT

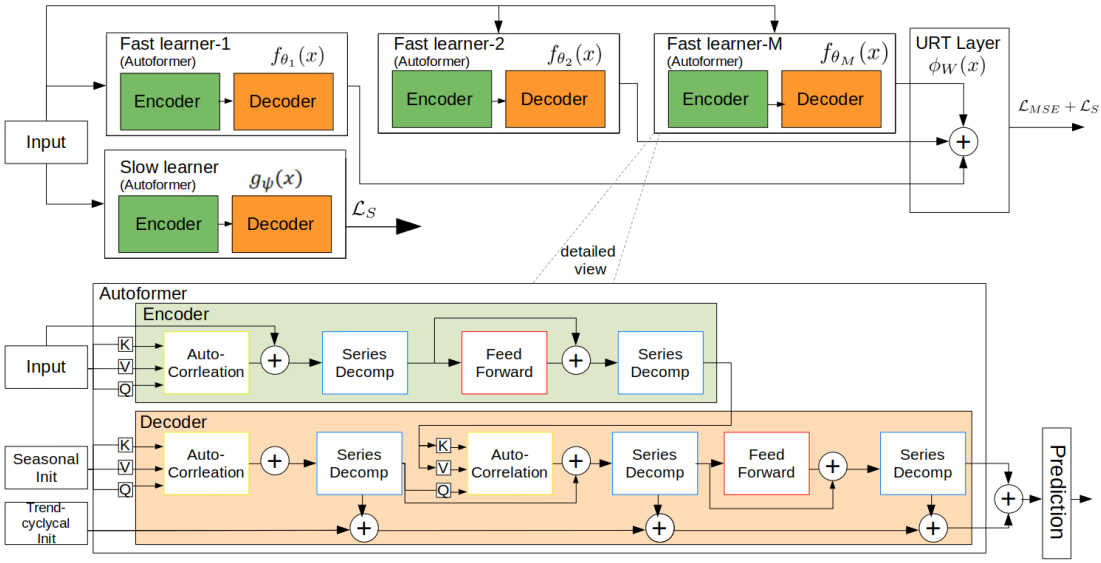


Figure 10.1: MANTRA is consolidated by an ensemble of fast learners and a slow learner, where the final output is aggregated by a URT layer. Fast learners and the URT layer are trained in an end-to-end fashion, minimizing the mean-squared error loss, while the slow learner is learned via the self-supervised learning loss, controlled reconstruction loss.

approach because it incurs fewer computations than the BT approach and is independent of any augmentations. In addition, an ensemble of fast learners is deployed here rather than a single fast learner handling the bias-variance tradeoff, and the concept drifts better than a single fast learner.

The slow learner $g_\psi(.)$ is trained to minimize the masked loss function $\mathcal{L}_m$ without any labeled samples [269]. The controlled masking strategy is implemented where m_t is a binary mask at the $t-th$ time step. A D dimensional time-series input $\tilde{x}_t$ is zeroed if $m_t = 1$. $\mathcal{L}_m$ is realized as the mean-squared error (MSE) loss function and applied to both masked and unmasked time-series samples with the original input x_t as the target variable.

$$\mathcal{L}_m = \frac{1}{D\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} m_t ||g_\psi(\tilde{x}_t) - x_t||_2^2 \tag{10.6}$$

$$\mathcal{L}_{um} = \frac{1}{D(S - \sum_{t=1}^{T} m_t)} \sum_{t=1}^{T} (1 - m_t) ||g_\psi(\tilde{x}_t) - x_t||_2^2 \tag{10.7}$$

Both $\mathcal{L}_m$ and $\mathcal{L}_{um}$ are combined with a tradeoff parameter $0 \leq \lambda \leq 1$ to train the slow learner rather than only the masked components as per [272] since the reconstruction process of the masked input heavily relies on the reconstruction process of the unmasked input. The overall loss function is formalized:

$$\mathcal{L}_S = \lambda \mathcal{L}_m + (1 - \lambda)\mathcal{L}_{um} \tag{10.8}$$

The controlled reconstruction process selects important time-stamps t [269] rather than relying on random selections [272]. The selection strategy is done by checking the attention matrix [269], which measures the matching degree between the keys and the queries. This is done differently here, where important time steps are determined across the top k autocorrelations $\hat{\mathcal{R}}_{\mathcal{Q},\mathcal{K}} \in \Re^{k \times k}$ because of the use of autoformer and the long-term time-series problem involving long sequences. Note that the top k autocorrelations have been extracted in AutoFormer. This strategy risks the selection of the same subset at every epoch, i.e., the over-fitting problem. The regularization strategy [269] is applied to alleviate this problem.

An array of fast learners M is deployed here and created by applying different initialization strategies to capture different aspects of data distributions, where M stands for the array size. Suppose that $\{f_{\theta_i}(.)\}_{i=1}^{M}$ denotes the $i-th$ fast learner and $g_\psi(.)$ labels the slow learner, $S_{de,\theta}^{l,i}, S_{de,\psi}^{i}, C_{de,\theta}^{l,i}, C_{de,\psi}^{i}$ stand for seasonal and cyclical components of the $l-th$ fast learner and the slow learner. Decoder features of the slow learner and the $l-th$ fast learners are concatenated before being projected back to the output dimension [265], [273]:

$$\hat{S}_{de,\theta}^{l,3} = S_{de,\theta}^{l,3} \oplus S_{de,\psi}^{3}; \quad \hat{C}_{de,\theta}^{l,3} = C_{de,\theta}^{l,3} \oplus C_{de,\psi}^{3} \tag{10.9}$$

where $\oplus$ denotes the concatenation operation which occurs at the last layer of Autoformer. The final output of the $l-th$ fast learner is inferred as $f_\theta(x) = \theta_S \hat{S}_{de,\theta}^{l,3} + \theta_C \hat{C}_{de,\theta}^{l,3}$ where θ_S, θ_C are projectors to modify the output dimension. A mean squared error (MSE) loss function $\mathcal{L}_{MSE}$ is applied to train all fast learners as follows:

$$\mathcal{L}_{MSE} = \sum_{t=1}^{T} \frac{1}{2}(y_t - \phi_W(\oplus(\{f_{\theta_i}(x_t)\}_{i=1}^{M})))^2 \tag{10.10}$$

where $\phi_W(\oplus(\{f_{\theta_i}(x)\}_{i=1}^{M}))$ stands for the final output of MANTRA and y_t denotes

the target variable at the $t-th$ time instant. Note that the case of a single-head URT layer is presented in $\phi_W(\oplus(\{f_{\theta_i}(x)\}_{i=1}^M))$ while a generalized version, a multi-head, is elaborated in the next section. Nonetheless, our experiment shows that the use of only the MSE loss function is not sufficient to fully exploit the dual net structure. Hence, all fast learners are trained minimizing the joint loss function $\mathcal{L}_{MSE} + \mathcal{L}_S$.

10.3.2 Drift Handling Mechanism via the URT Layer

MANTRA makes use of an ensemble of fast learners with M components connected to a single-head URT layer having a single attention head delivering the final predictions. An array of fast learners $\{f_{\theta_i}(.)\}_{i=1}^M$ are trained alongside with the URT layer $\phi_W(.)$ minimizing (10.10). MANTRA is sufficiently flexible where if the M fast learners $\{f_{\theta_i}(.)\}_{i=1}^M$ are pre-trained using different but related problems, e.g., different domains [266], one can fix the fast learners and train the URT layer $\phi_W(.)$ only. The URT concept is inspired by the self-attention mechanism of the transformer [274], deriving the attention mechanism via the query-key pairs. Since the original URT layer [266] is designed for few-shot image classification problems, we offer some modifications here. The URT layer concatenates the representations of the M fast learners and converts them into a final output, taking into consideration attentions:

$$\phi_W(x) = \oplus(f_{\theta_1}(x), f_{\theta_2}(x), ..., f_{\theta_M}(x)) \tag{10.11}$$

A universal representation of the current concept is portrayed by an empirical mean of data samples of a mini-batch B.

$$\phi_W(X_B) = \frac{1}{|X_B|} \sum_{x \in X_B} \phi_W(x) \tag{10.12}$$

The query $q(X_B)$ of the current concept is defined as a weighted linear transformation of the query parameters $\{W_q, b_q\}$ and the universal representation $\phi_W(X_B)$ as follows:

$$q(X_B) = W_q \phi_W(X_B) + b_q \tag{10.13}$$

The keys $k_i(X_B)$ are defined for all fast learners and rely on a similar linear transformation as per the queries parameterized by $\{W_k, b_k\}$.

$$k_i(X_B) = W_k f_{\theta_i}(X_B) + b_k \tag{10.14}$$

where $f_{\theta_i}(X_B) = \frac{1}{|X_B|}\sum_{x \in X_B} f_{\theta_i}(x)$. Based on the query-key pairs, the attention score α_i is defined as per the original transformer [274] as follows:

$$\alpha_i = \frac{\beta_i}{\sum_{i'=1}^{M} \beta_{i'}}; \beta_{i'} = \frac{q(X_B)^\top k_i(X_B)}{\sqrt{l}} \tag{10.15}$$

where l is the dimension of the keys and queries. Since we deal with forecasting problems here, there is no aggregation necessary for the full support set. The output of the attention score is produced by mixing the attention score and the fast learner representations as follows:

$$\phi_W(x) = \sum_{i=1}^{M} \alpha_i f_{\theta_i}(x) \tag{10.16}$$

It is obvious that the URT mechanism is capable of selecting a particular fast learner with a maximum attention $\alpha_i = 1$ or combining them with varying attention scores, i.e., the attention score satisfies the partition of unity $\sum_i \alpha_i = 1$ (10.15).

MANTRA can be structured under a multi-head configuration receiving the same input as the first URT layer $\{f_{\theta_i}(.)\}_{i=1}^{M}$. The output of MANTRA under the multi-head configuration is expressed as follows:

$$\phi_W(x) = \oplus(\{\phi_{W_j}(x)\}_{j=1}^{S}) \tag{10.17}$$

where S denotes the number of heads in the URT structure. A regularization approach is implemented to prevent duplication of attention scores as per [266] $\Omega(W) = ||AA^\top - I||_F^2$. A linear transformation $\{W_f, b_f\}$ is associated with (10.17) by following the same concept as the original transformer [274] producing the final output of MANTRA.

$$\phi_W(x) = \sum_{j=1}^{S} \phi_{W_j}(x) W_f^j + b_f^j \tag{10.18}$$

A fast adaptation can be achieved by only updating the most recent URT layer $\phi_{W_S}(x)$ with the query-key parameters $\{W_q^j, b_q^j\}$ and $\{W_k^j, b_k^j\}$ and the multi-head linear layer parameters $\{W_f, b_f\}$ leaving other parameters fixed. This strategy assures fast convergence to a new concept. Pseudo-codes of MANTRA are shown in Algorithm 10.

Algorithm 10 Learning Policy of MANTRA

Input: Fast learners $\{f_{\theta_i}(.)\}_{i=1}^{M}$, slow learner $g_\psi(.)$, URT layer $\phi_W(.)$, number of epochs E, number of batches of train dataset B.
Output: Updated Networks Parameters $\{f_{\theta_i}(.)\}_{i=1}^{M}$, $g_\psi(.)$, $\phi_W(.)$
// Train Fast and Slow Learners
for $e = 1 : E$ **do**
    for $b = 1 : B$ **do**
        $\mathcal{B}$ = minibatch sample of train dataset
        Compute loss $\mathcal{L}_{MSE}$
        Update fast learners parameters $\{f_{\theta_i}(.)\}_{i=1}^{M}$
        Compute m_t and generate $\tilde{x}_t$
        Compute loss $\mathcal{L}_S$ following eq. (10.6)
        Update slow learner parameters $g_\psi(.)$ based on $\mathcal{L}_S$
        Update fast learners parameters $\{f_{\theta_i}(.)\}_{i=1}^{M}$ based on $\mathcal{L}_S + \mathcal{L}_{MSE}$
    end for
end for
Freeze fast learners parameters $\{f_{\theta_i}(.)\}_{i=1}^{M}$,
// Train URT Layer
for $e = 1 : E$ **do**
    for $b = 1 : B$ **do**
        $\mathcal{B}$ = minibatch sample of train dataset
        Compute loss $\mathcal{L}_{MSE}$
        Update URT parameters $\phi_W(.)$
    end for
end for

10.3.3 Complexity Analysis

This sub-section discusses the complexity analysis of the proposed method. Suppose that N is the total samples of the training dataset that are divided into batches B that satisfy $\sum_{b=1}^{B} N_b = N$, E is the number of epochs for networks training, M is the number of fast learners. Please note that the proposed method only

uses 1 slow learner. Following MANTRA learning policy presented in Algorithm 1, there are two main processes, i.e., train fast and slow learners and train the URT layer. Those processes contain atomic operation, i.e., prediction, computing loss, and network update, that are conducted with complexity $O(1)$ for one learner. Therefore, prediction and update by M fast learners have $O(M)$ complexity. Let C denote the complexity of a process. Following Algorithm 1, the complexity of the proposed method can be written in the following equations:

$$\mathcal{C}(MANTRA) = \mathcal{C}(trainFastandSlow) + \mathcal{C}(trainURT) \tag{10.19}$$

$$\begin{aligned}\mathcal{C}(MANTRA) = E.\sum_{b=1}^{B} N_b(O(M) + O(1) + O(M) + \\ O(1) + O(1) + O(1) + O(M)) \\ + E.\sum_{b=1}^{B} N_b(O(M) + O(1) + O(1)) +\end{aligned} \tag{10.20}$$

$$\mathcal{C}(MANTRA) = E.\sum_{b=1}^{B} N_b(O(M)) + E.\sum_{b=1}^{B} N_b(O(M)) \tag{10.21}$$

$$\mathcal{C}(MANTRA) = O(E.M.\sum_{b=1}^{B} N_b) + O(E.M.\sum_{b=1}^{B} N_b) \tag{10.22}$$

Since $\sum_{b=1}^{B} N_b = N$, then the compexity of MANTRA can be written as:

$$\mathcal{C}(MANTRA) = O(E.M.N) + O(E.M.N) \tag{10.23}$$

$$\mathcal{C}(MANTRA) = O(E.M.N) \tag{10.24}$$

The complexity of the proposed method is $O(E.M.N)$ where E is the training epoch, M is the number of fast learners, and N is the size of the training dataset. In the case where the number of fast learners M is a small constant < 10 and the number of epochs is also a constant, then the complexity of the proposed method is $O(N)$. Table 10.1 reports the complexities of MANTRA compared to the ensemble of Autoformer and Autoformer. It is perceived that MANTRA's complexity is in the same order as the ensemble of Autoformer and Autoformer, $O(N)$. It is also seen that execution times per epoch for MANTRA are higher than those of Autoformer and the ensemble of Autoformer in the ETT dataset because it involves additional training procedures, i.e., slow learning procedure and URT layer learning procedure, from those of Autoformer and ensemble of Autoformer.

Table 10.1: Complexities of Different Methods.

Method	Compelxity	Per epoch running time on ETT (s)			
		96	192	336	720
Autoformer	$O(N)$	47.48	57.9	74.04	123.44
Ens.Autoformer	$O(N)$	91.83	101.92	118.45	188.54
MANTRA(Ours)	$O(N)$	140.75	156.42	181.79	299.61

10.4 Experiments

This section discusses our numerical study on four real-world datasets in four different application domains: energy, economics, weather, and disease.

10.4.1 Datasets

The ETT dataset is applied to test the performance of consolidated algorithms, which is formed by electricity transformer data covering load and oil temperature collected every 15 minutes over the period of July 2016 and July 2018. The weather dataset contains 2020 weather data collected every 10 minutes. It consists of 21 meteorological indicators, such as air temperature, humidity, etc. The ILI dataset encompasses weekly recorded influenza-like illness (ILI) patient data obtained from the Centers for Disease Control and Prevention of the United States over the period of 2002 to 2021. Particularly, it presents the ratio of patients seen with ILI and the total number of patients. The exchange dataset consists of the daily exchange rates of eight different countries over the period of 1990 to 2016. We follow [264] for the dataset split, where the ratio of 6:2:2 is implemented for the ETT dataset, while the ratio of 7:1:2 is used for other datasets. These problems are simulated in both univariate and multivariate forecasting problems.

10.4.2 Implementation Details

We follow the same setting as [264], where L_2 loss is applied using the ADAM optimizer, while the initial learning rate of 10^{-2} with an early stopping criterion is applied. The batch size is fixed to 32. Our numerical results are produced using a single NVIDIA A5000 GPU. The fast learner and the slow learner are configured as 2 encoder layers and 1 decoder layer. As with [264], the hyperparameter c of

Table 10.2: Multivariate results with different prediction lengths O E {96, 192, 336, 720}, averaged across 3 times run. We set the input length I as 36 for ILI and 96 for the others.

Models		MANTRA(Ours)		Autoformer[264]		Informer[263]		LogTrans[276]		Reformer[275]		LSTNet[277]		LSTM[278]		TCN[279]	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT	96	**0.212**	**0.295**	0.255	0.326	0.365	0.453	0.768	0.642	0.658	0.619	3.142	1.365	2.041	1.073	3.041	1.330
	192	**0.275**	**0.335**	0.299	0.350	0.533	0.563	0.989	0.757	1.078	0.827	3.154	1.369	2.249	1.112	3.072	1.339
	336	**0.327**	**0.365**	0.337	0.372	1.363	0.887	1.334	0.872	1.549	0.972	3.160	1.369	2.568	1.238	3.105	1.348
	720	**0.440**	**0.435**	0.442	**0.432**	3.379	1.388	3.048	1.328	2.631	1.242	3.171	1.368	2.720	1.287	3.135	1.354
Exchange	96	**0.155**	**0.285**	**0.153**	**0.285**	0.847	0.752	0.968	0.812	1.065	0.829	1.551	1.058	1.453	1.049	3.004	1.432
	192	**0.266**	**0.377**	0.295	0.395	1.204	0.895	1.040	0.851	1.188	0.906	1.477	1.028	1.846	1.179	3.048	1.444
	336	**0.421**	**0.480**	0.446	0.496	1.672	1.036	1.659	1.081	1.357	0.976	1.507	1.031	2.136	1.231	3.113	1.459
	720	**1.168**	**0.847**	1.503	0.919	2.478	1.310	1.941	1.127	1.510	1.016	2.285	1.243	2.984	1.427	3.150	1.458
Weather	96	**0.248**	**0.321**	0.269	0.338	0.300	0.384	0.458	0.490	0.689	0.596	0.594	0.587	0.369	0.406	0.615	0.589
	192	**0.281**	**0.338**	0.297	0.354	0.598	0.544	0.658	0.589	0.752	0.638	0.560	0.565	0.416	0.435	0.629	0.600
	336	**0.329**	**0.369**	0.358	0.392	0.578	0.523	0.797	0.652	0.639	0.596	0.597	0.587	0.455	0.454	0.639	0.608
	720	**0.405**	**0.414**	0.450	0.452	1.059	0.741	0.869	0.675	1.130	0.792	0.618	0.599	0.535	0.520	0.639	0.610
ILI	24	**3.238**	**1.224**	3.680	1.346	5.764	1.677	4.480	1.444	4.400	1.382	6.026	1.770	5.914	1.734	6.624	1.830
	36	**2.964**	**1.176**	3.329	1.260	4.755	1.467	4.799	1.467	4.783	1.448	5.340	1.668	6.631	1.845	6.858	1.879
	48	**2.941**	**1.144**	3.376	1.258	4.763	1.469	4.800	1.468	4.832	1.465	6.080	1.787	6.736	1.857	6.968	1.892
	60	**2.705**	**1.106**	2.917	1.159	5.264	1.564	5.278	1.560	4.882	1.483	5.548	1.720	6.870	1.879	7.127	1.918

auto-correlation lies in the range of 1 to 3 while the number of URT heads and fast learners are set to 1 and 3 respectively.

10.4.3 Baselines

MANTRA is compared with four state-of-the art transformer-based models: Autoformer [264], Informer [263], Reformer [275], LogTrans [276], two recurrent-based model LSTNet [277], LSTM[278] and one convolutional-based model TCN [279]. Comparisons are performed for both multivariate and univariate configurations. Autoformer is simulated in the same computational environments using its official implementations, where its hyperparameters are set using the grid search technique. Other numerical results are taken from [264]. Numerical reports are reported from an average of three independent executions under different random seeds.

10.4.4 Numerical Results

The advantage of MANTRA is demonstrated in Table 10.2 and 10.3 for multivariate and univariate forecasting, respectively. In the realm of multivariate forecasting, MANTRA outperforms other algorithms with noticeable margins across all predictive lengths. MANTRA is only on par with Autoformer in the predictive

Table 10.3: Univariate results with different prediction lengths O E {96, 192, 336, 720}, averaged across 3 times run. We set the input length I as 36 for ILI and 96 for the others.

Models		**MANTRA(Ours)**		Autoformer[264]		Informer[263]		LogTrans[276]		Reformer[275]	
Mettrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT	96	0.1035	0.2447	0.1043	0.2455	**0.0706**	**0.2030**	0.0824	0.2309	0.1740	0.3424
	192	**0.1344**	**0.2844**	0.1574	0.3042	0.2093	0.3797	0.1709	0.3296	0.2409	0.4067
	336	**0.1631**	**0.3145**	0.1608	0.3104	0.3041	0.4743	0.2822	0.4466	0.3538	0.5153
	720	**0.1853**	**0.3359**	0.2007	0.3500	0.3571	0.5110	0.4313	0.5817	0.3759	0.5417
Exchange	96	**0.1427**	**0.2970**	0.1639	0.3179	0.5172	0.5694	0.2737	0.3999	0.5757	0.6101
	192	**0.2815**	**0.4153**	0.3278	0.4413	1.0956	0.8252	1.7398	0.9529	0.9056	0.7751
	336	**0.5099**	**0.5522**	0.6178	0.6097	1.4225	0.9660	2.4667	1.1975	2.0816	1.2192
	720	**1.2448**	**0.8677**	1.3049	0.8960	1.3918	0.9332	2.3174	1.1825	2.3128	1.3229
Weather	96	0.0079	0.0705	0.0159	0.1001	**0.0043**	**0.0478**	0.0041	0.0511	0.0046	0.0542
	192	0.0062	0.0582	0.0086	0.0688	**0.0046**	**0.0521**	0.0048	0.0499	0.0433	0.0927
	336	**0.0040**	**0.0475**	0.0050	0.0543	0.0044	0.0508	0.0088	0.0721	0.0149	0.1019
	720	**0.0040**	**0.0480**	0.0083	0.0707	0.0050	0.0535	0.0064	0.0628	0.0195	0.1168
ILI	24	**0.7315**	**0.6411**	0.7948	0.6827	5.6955	2.1385	3.9573	1.7524	3.7231	1.7053
	36	**0.6986**	**0.6366**	0.7157	0.6848	5.1962	2.0467	3.9925	1.7635	3.6082	1.6725
	48	**0.7427**	**0.7054**	0.8019	0.7482	4.8515	1.9779	3.9118	1.7542	3.8564	1.7636
	60	**0.8327**	**0.7645**	0.9219	0.8205	5.3332	2.0800	3.9626	1.7741	4.3197	1.8932

MANTRA Improvement (%) compared to the competitor methods															
Models		Autoformer[264]		Informer[263]		LogTrans[276]		Reformer[275]		LSTNet[277]		LSTM[278]		TCN[279]	
Metrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT	96	16.63	9.57	41.86	34.85	72.37	54.03	67.75	52.32	93.25	78.38	89.60	72.50	93.02	77.81
	192	7.99	4.31	48.34	40.47	72.16	55.73	74.46	59.47	91.27	75.52	87.76	69.86	91.04	74.97
	336	2.87	1.90	76.02	58.88	75.50	58.18	78.90	62.48	89.66	73.36	87.27	70.54	89.47	72.94
	720	0.41	-0.70	86.97	68.69	85.55	67.28	83.26	65.01	86.11	68.23	83.81	66.24	85.95	67.91
Exchange	96	-0.74	-0.11	81.75	62.08	84.03	64.89	85.48	65.61	90.03	73.05	89.36	72.82	94.85	80.09
	192	9.71	4.43	77.88	57.84	74.39	55.66	77.58	58.35	81.97	63.29	85.57	67.99	91.26	73.87
	336	5.67	3.14	74.81	53.64	74.62	55.57	68.97	50.79	72.06	53.41	80.29	60.98	86.47	67.08
	720	22.30	7.83	52.88	35.34	39.84	24.84	22.67	16.62	48.90	31.85	60.87	40.64	62.93	41.90
Weather	96	7.67	5.01	17.17	16.38	45.74	34.47	63.93	46.12	58.17	45.30	32.66	20.91	59.60	45.48
	192	5.34	4.54	53.01	37.79	57.29	42.55	62.63	46.96	49.82	40.11	32.45	22.21	55.33	43.60
	336	7.93	5.76	43.05	29.42	58.70	43.38	48.49	38.06	44.87	37.11	27.66	18.69	48.49	39.29
	720	9.93	8.33	61.72	44.07	53.35	38.60	64.12	47.67	34.40	30.81	24.22	20.30	36.55	32.06
ILI	24	12.01	9.05	43.83	27.02	27.73	15.24	26.42	11.44	46.27	30.85	45.25	29.42	51.12	33.12
	36	10.95	6.63	37.66	19.83	38.23	19.83	38.02	18.78	44.49	29.49	55.29	36.26	56.77	37.41
	48	12.90	9.04	38.26	22.10	38.74	22.04	39.14	21.88	51.63	35.96	56.34	38.37	57.80	39.51
	60	7.24	4.53	48.61	29.26	48.74	29.08	44.59	25.40	51.24	35.68	60.62	41.12	62.04	42.32

Table 10.4: Improvement of MANTRA to competitor methods on multivariate setting with different prediction lengths $O \in \{96, 192, 336, 720\}$, except for ILI dataset $O \in \{24, 36, 48, 60\}$.

Table 10.5: Improvement of MANTRA to competitor methods on univariate setting with different prediction lengths $O \in \{96, 192, 336, 720\}$, except for ILI dataset $O \in \{24, 36, 48, 60\}$.

		MANTRA Improvement (%) compared to the competitor methods							
Models		Autoformer[264]		Informer[263]		LogTrans[276]		Reformer[275]	
Mettrics		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETT	96	0.75	0.36	-46.63	-20.49	-25.63	-5.96	40.49	28.54
	192	14.61	6.53	35.80	25.10	21.36	13.72	44.22	30.08
	336	-1.48	-1.33	46.35	33.69	42.18	29.59	53.89	38.97
	720	7.67	4.03	48.11	34.27	57.03	42.25	50.70	37.99
Exchange	96	12.88	6.57	72.40	47.84	47.84	25.73	75.20	51.32
	192	14.15	5.90	74.31	49.68	83.82	56.42	68.92	46.42
	336	17.47	9.43	64.16	42.84	79.33	53.89	75.51	54.71
	720	4.60	3.16	10.57	7.02	46.29	26.62	46.18	34.41
Weather	96	50.41	29.51	-82.45	-47.67	-91.70	-38.09	-72.84	-30.14
	192	28.86	15.40	-34.63	-11.73	-28.13	-16.53	85.80	37.22
	336	21.17	12.53	9.97	6.50	54.94	34.09	73.46	53.38
	720	52.16	32.13	20.81	10.29	38.40	23.48	79.76	58.90
ILI	24	7.96	6.10	87.16	70.02	81.51	63.42	80.35	62.41
	36	2.39	7.04	86.56	68.90	82.50	63.90	80.64	61.94
	48	7.38	5.73	84.69	64.34	81.01	59.79	80.74	60.01
	60	9.68	6.83	84.39	63.25	78.99	56.91	80.72	59.62

Table 10.6: T-Test with 0.1 significance level of MANTRA vs. Autoformer on multivariate and univariate setting.

		Multivariate				Univariate			
Mettrics		MSE		MAE		MSE		MAE	
T-Test		p-val	Sig.	P-val	Sig.	p-val	Sig.	P-val	Sig.
ETT	96	0.000662	✓	0.000027	✓	0.466713	✗	0.459712	✗
	192	0.002148	✓	0.017119	✓	0.029832	✓	0.035679	✓
	336	0.000393	✓	0.003745	✓	0.409776	✗	0.323542	✗
	720	0.456497	✗	0.397439	✗	0.082046	✓	0.114526	✗
Exchange	96	0.457713	✗	0.484059	✗	0.029386	✓	0.053399	✓
	192	0.024725	✓	0.029971	✓	0.047900	✓	0.053399	✓
	336	0.101353	✗	0.110363	✗	0.096577	✓	0.080948	✓
	720	0.231046	✗	0.268099	✗	0.277424	✗	0.225318	✗
Weather	96	0.085522	✓	0.033795	✓	0.004570	✓	0.009602	✓
	192	0.066209	✓	0.057598	✓	0.096289	✓	0.035484	✓
	336	0.021258	✓	0.052073	✓	0.002438	✓	0.005528	✓
	720	0.053112	✓	0.035405	✓	0.004743	✓	0.006344	✓
ILI	24	0.069567	✓	0.040147	✓	0.000956	✓	0.00531	✓
	36	0.002188	✓	0.007061	✓	0.256694	✗	0.000204	✓
	48	0.007818	✓	0.012957	✓	0.078750	✓	0.022501	✓
	60	0.000010	✓	0.006843	✓	0.003887	✓	0.003742	✓

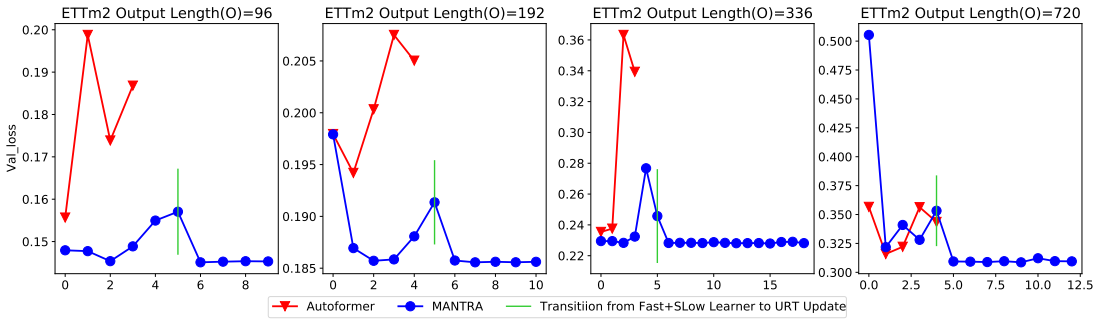


Figure 10.2: MANTRA and Autoformer validation loss on ETTm dataset.

length of 96 of the exchange dataset. As the predictive length increases, MANTRA outperforms Autoformer with notable margins. On the other hand, the same finding is observed in the univariate forecasting setting, where MANTRA beats other algorithms in most cases. MANTRA is only inferior to Informer in the predictive length of 96 of the ETT dataset and in the predictive length of 96 and 192 of the weather dataset. Although MANTRA is behind in the three cases, this fact can be caused by the base learner, where MANTRA applies Autoformer as its base learner. Informer exceeds Autoformer in the three cases. Note that the concept of MANTRA is applicable to any base learners. Percentage improvements of MANTRA w.r.t. other algorithms are reported in Table 10.4 and 10.5. In addition, Table 10.6 reports the t-test between MANTRA and Autoformer. It is seen that MANTRA outperforms Autoformer with statistically significant margins in most cases.

Fig. 10.2 exhibits the validation losses of MANTRA and Autoformer in the ETT dataset. It is perceived that the validation losses of MANTRA are smaller than Autoformer across all predictive lengths. One may wonder about the increase in validation losses in Autoformer and MANTRA. This is caused by the fact that Autoformer adopts early stopping criteria and is trained with a few epochs. MANTRA's validation losses increase later than Autoformer. Importantly, the validation loss of MANTRA decreases further after switching to the URT update phase and fixing the base learners. The advantage of MANTRA compared to Autoformer is also depicted in Fig. 10.3 in which it delivers more accurate predictions than Autoformer across all predictive lengths of the ETT dataset.

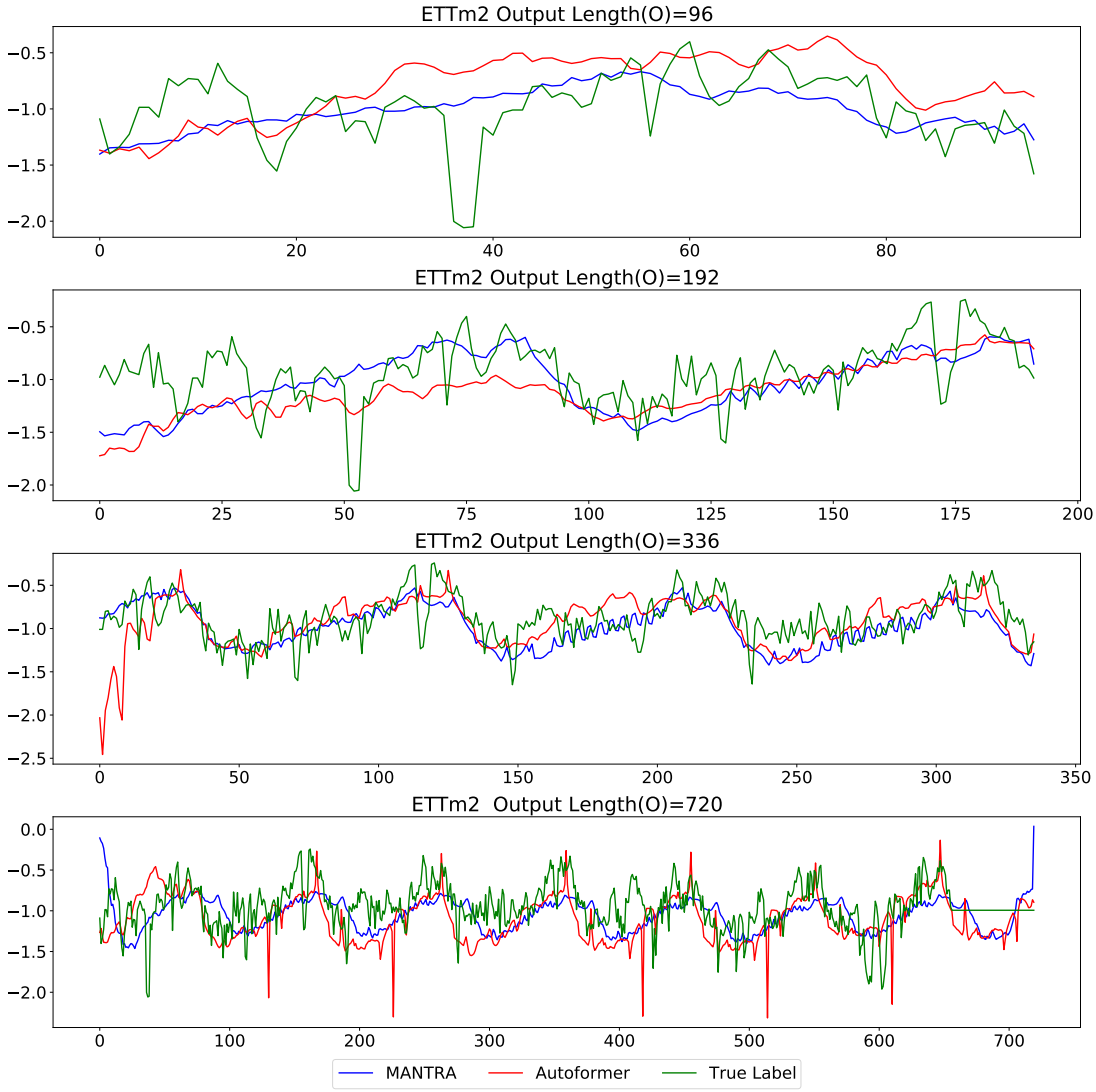


Figure 10.3: Visualization of MANTRA prediction compared to actual label on ETTm dataset.

10.4.5 Ablation Studies

Ablation studies are performed to study the influence of MANTRA's learning modules. Table 10.7 reports our numerical results produced with the ETT and ILI datasets. The URT layer plays a key role for MANTRA, where its absence brings down numerical results significantly in both the ETT dataset and the ILI dataset. The URT layer allows fast adaptations of new patterns because base models can be frozen while only updating a few parameters of the URT layer. On the other hand, the slow learner contributes positively to the overall performance of MANTRA, where its absence deteriorates the numerical results of MANTRA. In the ETT dataset, its effect is not obvious for small predictive lengths but becomes clear with the increase in predictive length. In the ILI dataset, MANTRA's performances are compromised due to the absence of the slow learner for all cases. This fact confirms the efficacy of the extended dual network concept, which decouples the representation learning and the discriminative learning. The slow learner trained in a self-supervised manner is capable of generating useful representations of downstream tasks.

10.4.6 Sensitivity Analysis

Sensitivity Analysis is performed to study the effect of the URT head and fast learners, which is done with the ETT dataset. Numerical results are reported in Table 10.8 and 10.9 respectively. The increase in the fast learners improves the numerical results up to 3 fast learners, but this trend does not continue with over 3 fast learners. This finding is mainly attributed to the random initialization strategy of MANTRA, which does not generate sufficiently diverse fast learners. One might find improved strategies for fast learner creations. On the other hand, the increase of URT heads does not contribute toward the improvement of MANTRA's performance and even negatively affects the computational burdens.

10.5 Summary

This chapter proposes the meta-transformer network (MANTRA) for dynamic long-term time-series forecasting tasks. MANTRA is developed from the concept

Table 10.7: Ablation study on ETT and ILI dataset.

Config.		MSE MAE	MSE MAE	MSE MAE	MSE MAE
		96	192	336	720
MANTRA	ETT	0.2127 0.2951	0.2703 0.3296	0.3255 0.3625	0.4354 0.4286
MANTRA w/o URT		0.2305 0.3095	0.2850 0.3391	0.3387 0.3702	0.4340 0.4202
MANTRA w/o Slow Learner		0.2127 0.2934	0.2677 0.3249	0.3310 0.3682	0.4354 0.4304
		24	36	48	60
MANTRA	ILI	2.7930 1.1274	3.0280 1.1929	3.0008 1.1714	2.6418 1.0855
MANTRA w/o URT		5.2348 1.7377	4.7750 1.6171	4.3002 1.5188	4.3411 1.5185
MANTRA w/o Slow Learner		3.6409 1.3103	3.4154 1.2342	3.3005 1.2076	3.1309 1.1821

Table 10.8: Sensitivity analysis on ETT dataset with the different number of fast learners.

N-Learner	96		192		336		720	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
1	**0.2111**	**0.2936**	0.2706	0.3275	0.3323	0.3718	**0.4233**	**0.4177**
2	0.2114	0.2925	0.2828	0.3425	0.3262	0.3637	0.4261	0.4237
3	0.2127	0.2951	**0.2703**	**0.3296**	**0.3255**	**0.3625**	0.4354	0.4286
4	0.2167	0.2997	0.2738	0.3311	0.3413	0.3789	0.4298	0.4231
5	0.2157	0.2992	0.2721	0.3312	0.3311	0.3687	0.4341	0.4271
6	0.2159	0.2995	0.2723	0.3309	0.3376	0.3783	0.4270	0.4242
7	0.2125	0.2956	0.2726	0.3310	0.3347	0.3715	0.4473	0.4434

Table 10.9: Sensitivity analysis on ETT dataset with the different number of URT heads.

URT-Head	96		192		336		720	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
1	0.2127	0.2951	0.2703	0.3296	0.3255	0.3625	0.4354	0.4286
2	0.2128	0.2952	0.2703	0.3297	0.3255	0.3625	0.4354	0.4286
3	0.2127	0.2952	0.2702	0.3296	0.3256	0.3625	0.4358	0.4292
4	0.2127	0.2952	0.2704	0.3298	0.3255	0.3625	0.4355	0.4287
5	0.2126	0.2951	0.2703	0.3296	0.3256	0.3626	0.4358	0.4291
6	0.2126	0.2951	0.2704	0.3297	0.3258	0.3627	0.4357	0.4290
7	0.2126	0.2951	0.2703	0.3296	0.3256	0.3626	0.4355	0.4286

of extended dual networks where an array of fast learners and a slow learner learn cooperatively with dual objectives, minimizing predictive losses and self-supervised losses. The concept of URT is integrated to adapt quickly to concept drifts involving only a few parameters. That is, the URT layer produces concept-adapted representations enabling fast adaptations to concept changes. The advantage of MANTRA is numerically validated with four datasets and different predictive lengths under both multivariate and univariate settings. MANTRA outperforms other popular algorithms in 29 cases out of 32 cases with at least 3% gaps. Although MANTRA is exemplified with Autoformer as a base learner here, it can be generalized to other base learners as well. Our future work is devoted to study the problem of time-series domain adaptations.

Chapter 11

Conclusion and Future Work

11.1 Conclusion

This thesis is proposed to tackle the catastrophic forgetting problem under various settings and constraints, i.e., rehearsal-based, rehearsal-free, data scarcity, unlabeled data scarcity, non-i.i.d. distribution, and streaming data, applicable both in standalone and distributed systems. Along with proposing novel methods for the mentioned problems, this thesis pushes the continual learning problem into 2 practical advanced sub-problems, i.e., unsupervised few-shot continual learning and federated few-shot class incremental learning. Those two problems are arguably newly defined problems in the state of the writing of this thesis. This thesis contains eight articles, where each of the articles demonstrates an effective new method for continual learning or its subproblem. Generally, the proposed methods are applicable to the image data; one is applied to remote sensing data, and another to time-series data.

In the first article, we propose an assessor-guided learning approach (AGLA) for a rehearsal-based continual learning, where it puts forward a sequence-aware assessor to guide the learning process of the base learner. The base learner forms the meta-weighted combination of the cross-entropy loss function, the DER loss function, and the distillation loss function. While the assessor performs a soft-weighting mechanism controlling the influence of a data sample and the interaction of loss functions. The proposed mechanism is proven to produce a proper tradeoff between

the plasticity and the stability because not only positive samples are selected, but also suitable learning strategies are chosen for every sample, i.e., the assessor balances the stability and the plasticity for every sample. The proposed compensated over-sampling (COS) in AGLA effectively addresses the class-imbalanced problem on continual learning, where corrections are carried out while augmenting data samples to avoid the effect of out-of-distribution augmented samples. AGLA has been compared with ten recently published algorithms under the same computational environments, where AGLA demonstrates improved and stable accuracy across all four problems in both the task-incremental learning problems and the class-incremental learning problems, with noticeable margins. In terms of average forgetting, AGLA delivers comparable performances to those of memory-based approaches, where it mostly occupies the second lowest average forgetting index, and better than those of regularization-based approaches. AGLA also maintains decent performance with tiny memory sizes, as indicated by our memory analysis. However, since it utilizes memory and DER loss, AGLA can't be applied in a rehearsal-free setting. In addition, AGLA, as well as existing replay methods, work with a large number of samples and over-fit quickly in the case of a few samples in each task. Thus, in such limitations, i.e.,rehearsal-free or data scarcity, the practitioners should consider additional solutions.

In the second article, we highlight the importance of privacy-preserving in continual learning and identify the prompt-based approach dilemma and ineffectiveness of the existing language-guided prompt method. We propose a novel prompt-based structure and algorithm, namely LEAPGen, for rehearsal-free continual learning that incorporates 4 key concepts: (1) language as input for prompt generation, (2) task-wise generators, (3) limiting matching descriptors search space via soft task-id prediction, and (4) generated prompt as auxiliary data. Our experimental analysis shows LEAPGEn significantly outperforms the existing SOTAs in CIFAR100, ImageNet-R, and CUB datasets with a significant margin in all measured metrics, i.e., final average accuracy, cumulative average accuracy, final forgetting measure, and cumulative forgetting measure. LEAPGen is historically successful in maintaining its stability-plasticity in every task. The idea of fixed language embedding associated with class-wise learnable keys as input for prompt generation is inten-

ded to achieve better class recognition and reduce forgetting at the same time, rather than training latent parameters trained in every task. In addition, the idea is arguably a novel idea from the perspective of continual learning. The idea of a task-wise generator is proven effective and more efficient than the growing generator approach, both in training and inference phases. Soft task ID prediction is theoretically and numerically proven to be more effective than existing task prediction. The auxiliary data has been proven to improve the recognition accuracy. LEAPGen proves its robustness by achieving higher performance in various prompt lengths, layer depths, and the number of generators per task compared to the SOTAs. Despite successfully handling the catastrophic forgetting in rehearsal-free continual learning, LEAPGen works on large samples and has not been proven in the data scarcity constraint.

In the third article, we present the Flat-to-Wide Approach (FLOWER) to address catastrophic forgetting under data scarcity in few-shot continual learning problems. FLOWER introduces Flat-Wide learning enhanced with Ball augmentation that works in feature space. Our in-depth numerical study confirms the advantage of FLOWER, where it outperforms prior arts with a significant margin in the three benchmark problems for both average and per-session accuracy. This fact is even more obvious with small base tasks, where larger margins to their counterparts are observed than in moderate base tasks, whereas prior arts are compromised in such a setting because they heavily focus on the stability of base tasks. Flat-wide learning has been proven more effective than flat minima learning, that is proposed in a previous study. Ball augmentation has been proven as FLOWER's important component to minimize biased representation. However, FLOWER and the current few-shot continual learning methods rely on a strong assumption that the base task carries abundant labeled samples. The assumption may be inapplicable to a real-world case, since data annotation is expensive.

In the fourth article, we propose an extension of few-shot continual learning (FSCL), namely unsupervised few-shot continual learning (UFSCL), where no labels are exploited for inducing the training process. UFSCL is arguably a newly defined problem at the time of this thesis writing. We propose an unsupervised flat-wide learning approach (UNISA) to handle catastrophic forgetting under un-

labeled data scarcity in UFSCL. UNISA is built upon the unsupervised flat-wide learning approach to address representation enrichment via moving average operation on its feature extractor and positive sampling alignment loss. The pseudo-label by clustering and prototype scattering loss maintains classes' separability in the unsupervised learning process. The efficacy of UNISA has been validated in the remote sensing image scene classification problems, where it produces the most encouraging numerical results over its prior state-of-the-art. Our extended analysis shows the robustness of UNISA in various noise attack and hyperparameter settings. UNISA requires moderate training time in few-shot tasks, but relatively high training time in the base task. It is a logical consequence since UNISA executes more operations, e.g., moving average, generating multiple representations, clustering, and pseudo-labeling. Despite its effectiveness in solving UFSCL, UNISA still experiences a fairly high performance drop from the first task to the last task. It shows that catastrophic forgetting under unlabeled data scarcity is still an open problem for future study.

In the fifth article, we address catastrophic forgetting under non-i.i.d. distribution in federated class incremental learning (FCIL), propose new prompt-based baselines named Fed-L2P and Fed-DualP, and a novel method named prototype-injected prompt (PIP) for the FCIL problem. PIP consists of three main ideas: a) prototype injection on prompt, b) prototype augmentation, and c) weighted Gaussian aggregation on the server side. The shared prototype injection enables each client to learn locally unavailable class representations, while prototype augmentation handles the class imbalance problem between the prototypes and locally available samples. The Gaussian aggregation produces a more converged global model as it considers selected local clients' participation. The experimental result shows that PIP outperforms the current SOTAs with a significant gap in CIFAR100, MiniImageNet, and TinyImageNet datasets, as well as demonstrates its robustness in different task sizes, smaller participating local clients, and smaller global rounds. PIP has the same complexity as our proposed baselines and experimentally requires shorter training time than the previous FCIL SOTAs that train and aggregate the whole model. In practice, our proposed method can be applied in both cross-silo and cross-domain federated class incremental learning.

Despite its effectiveness in FCIL, PIP assumes that each locally available class carries abundant samples. Thus, in the case of data scarcity, it may suffer from a significant performance drop in handling the catastrophic forgetting under a non-i.i.d. distribution.

In the sixth article, we highlight the data scarcity issue in FCIL and define a more challenging yet applied problem named Federated Few-Shot Class-Incremental Learning (FFSCIL). FFSCIL is arguably a new problem at the time of this thesis writing. We propose a novel Unified Optimized Prototype Prompt (UOPP) model to solve FFSCIL problem, which utilizes task-wise prompt learning to mitigate task interference empowered by shared static-dynamic prototypes, adaptive dual heads, and weighted aggregation. The dynamic prototype tackles prototype bias by iterative rectifications executed by a trainable NeuralODE that performs the rectification by generating a query set from the augmentation of shared unified prototypes. The adaptive dual head improves the classification accuracy by adaptively selecting the MLP classifier that performs excellently on the base task, and a prototype-based classifier that is suitable for few-shot tasks. Our comprehensive experimental results show that UOPP significantly outperforms the existing SOTAs of FCIL, FSCIL, and CIL, on three datasets with a significant gap, achieves a better stability-plasticity trade-off, and is robust in different local clients and small global rounds. Our analysis shows that our proposed method requires moderate training time but a lower communication cost than the SOTAs. In addition, UOPP also works on a standalone FSCIL, where all the data is stored in a single client. Despite its effectiveness in FFSCIL, the trainable NeuralODE adds a significant training time for UOPP. Its efficiency can be improved by developing a less time-consuming rectification method.

In the seventh article, we highlight the rehearsal-free aspect of continual learning in streaming data and propose a novel prompt online learning (PROL) for rehearsal-free online continual learning (OCL). PROL incorporates four key concepts: (1) a Single lightweight prompt generator for all instances in all classes, (2) learnable scalers-and-shifters parameters associated with learnable class-wise keys, (3) a cross-correlation matrix for preserving PTM generalization, (4) hard-soft update for adaptive learning rate assignment. The single generator with scalers and

shifters has been proven effective yet efficient for rehearsal-free OCL. The PTM generalization preservation contributes to maintain model stability while achieving plasticity on a new task, while the hard-soft update contributes to achieve model convergence in one-time seen data. Our experiment and analysis show PROL outperforms existing SOTAs in CIFAR100, ImageNet-R, ImageNet-A, and CUB datasets with significant margins for final average accuracy and cumulative average accuracy, and achieves relatively low final forgetting measure. PROL maintains stability-plasticity in every task, and is robust in various prompt lengths and numbers of ViT layers. In addition, PROL outperforms the joint version of existing OCL SOTAs in 3 datasets while running in a rehearsal-free manner. PROL method achieves moderate training time, and throughput that is significantly better than the existing prompt generator or adapters method. PROL achieves lower throughput compared to RanPAC and RanDumb since those two methods collect all streamed features and update their trainable parameters only once, i.e., at the end of a task, while PROL updates its trainable parameters on each streaming chunk. In such a case, PROL supports anytime inference, while those two methods do not. Despite its effectiveness in rehearsal-free OCL, PROL has room for improvement to reduce its forgetting rate.

In the eighth article, we propose the meta-transformer network (MANTRA) for dynamic long-term time-series forecasting tasks. MANTRA is developed from the concept of extended dual networks where an array of fast learners and a slow learner learn cooperatively with dual objectives, minimizing predictive losses and self-supervised losses. The concept of URT is integrated to adapt quickly to concept drifts involving only a few parameters. That is, the URT layer produces concept-adapted representations enabling fast adaptations to concept changes. The advantage of MANTRA is numerically validated with four datasets and different predictive lengths under both multivariate and univariate settings. MANTRA outperforms other popular algorithms in 29 cases out of 32 cases with a significant gap. Although MANTRA is exemplified with Autoformer as a base learner here, it can be generalized to other base learners as well. Our future work is devoted to study the problem of time-series domain adaptations. The drawback of MANTRA requires more training time than Autoformer. However, it is a local

implication since MANTRA ensembles several Autoformers as fast learners, adds a slow learner, and a URT layer to its pipeline.

11.2 Future Work

Although the proposed learning methods, theoretical and numerical analysis deliver an insightful solution for catastrophic forgetting under various settings and constraints, there are several improvements that can be pursued in future studies. The improvements can be directed based on the following considerations.

- All of the chapters assume that the testing is conducted in a closed world system where the model is tested only by the previously learned class labels. In a real-world application, a model may be requested with an unknown class that has never been seen before. Therefore, the ability to recognize unknown classes is an important consideration for future study of continual learning.
- Aligned with the previous point, cross-domain setting is an interesting consideration for future study since all the chapter assumes that the data are drawn from the same domain.
- The data scarcity constraint discussed in this thesis is applied starting from the second task, while the base task is assumed to carry abundant samples. In a real-world application, this assumption may be inapplicable since data scarcity can be encountered from the beginning of the sequences. Thus, in the future study of FSCL, UFSCL, and FFSCIL, the setting should include data scarcity in the first (base) task as well.
- In the study of FCIL and FFSCIL, each client carries the same portion (percentage) of available data. This setting can be improved by setting a randomized percentage for a more realistic simulation in a real-world system, where clients have different sizes of local data, along with different data distribution.
- In the study of OCL, the performance measurement is conducted after finishing a task. For more comprehensive studies, the anytime measurement can be considered to be added. Thus, the performance difference between

the methods that update the model on each streaming chunk and the methods that update only once after storing all streaming features is more clearly visible.

- Last but not least, except for MANTRA, all the proposed methods are evaluated in the image classification task. The future studies can be devoted to measure the methods' performance on more advanced vision tasks, such as object detection and image segmentation, or classification tasks in different data types, such as text and graph.

References

[1] J. P. Pirruccello et al., 'Deep learning enables genetic analysis of the human thoracic aorta,' *Nature genetics*, vol. 54, no. 1, pp. 40–51, 2022.

[2] S. Tasaki, C. Gaiteri, S. Mostafavi and Y. Wang, 'Deep learning decodes the principles of differential gene expression,' *Nature Machine Intelligence*, vol. 2, no. 7, pp. 376–386, 2020.

[3] E. Gawehn, J. A. Hiss and G. Schneider, 'Deep learning in drug discovery,' *Molecular informatics*, vol. 35, no. 1, pp. 3–14, 2016.

[4] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona and T. Blaschke, 'The rise of deep learning in drug discovery,' *Drug discovery today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[5] K. Preuer, G. Klambauer, F. Rippmann, S. Hochreiter and T. Unterthiner, 'Interpretable deep learning in drug discovery,' *Explainable AI: interpreting, explaining and visualizing deep learning*, pp. 331–345, 2019.

[6] V. Shah, R. Keniya, A. Shridharani, M. Punjabi, J. Shah and N. Mehendale, 'Diagnosis of covid-19 using ct scan images and deep learning techniques,' *Emergency radiology*, vol. 28, pp. 497–505, 2021.

[7] H. Panwar, P. Gupta, M. K. Siddiqui, R. Morales-Menendez, P. Bhardwaj and V. Singh, 'A deep learning and grad-cam based color visualization approach for fast detection of covid-19 cases using chest x-ray and ct-scan images,' *Chaos, Solitons & Fractals*, vol. 140, p. 110 190, 2020.

[8] J. Liu et al., 'Applications of deep learning to mri images: A survey,' *Big Data Mining and Analytics*, vol. 1, no. 1, pp. 1–18, 2018.

[9] S. Gassenmaier et al., 'Deep learning applications in magnetic resonance imaging: Has the future become present?' *Diagnostics*, vol. 11, no. 12, p. 2181, 2021.

[10] H.-T. Cheng et al., 'Wide & deep learning for recommender systems,' in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[11] D. Shankar, S. Narumanchi, H. Ananya, P. Kompalli and K. Chaudhury, 'Deep learning based large scale visual recommendation and search for e-commerce,' *arXiv preprint arXiv:1703.02344*, 2017.

[12] W. Yu, Z. Sun, H. Liu, Z. Li and Z. Zheng, 'Multi-level deep learning based e-commerce product categorization.,' in *eCOM@ SIGIR*, 2018.

[13] Z. Liu, Z. Li, K. Wu and M. Li, 'Urban traffic prediction from mobility data using deep learning,' *Ieee network*, vol. 32, no. 4, pp. 40–46, 2018.

[14] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi and B. Yin, 'Deep learning on traffic prediction: Methods, analysis, and future directions,' *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4927–4943, 2021.

[15] H. Nguyen, L.-M. Kieu, T. Wen and C. Cai, 'Deep learning methods in transportation domain: A review,' *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 998–1004, 2018.

[16] X. Liu, W. Liu, T. Mei and H. Ma, 'A deep learning-based approach to progressive vehicle re-identification for urban surveillance,' in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, Springer, 2016, pp. 869–884.

[17] J. Zhu et al., 'Vehicle re-identification using quadruple directional deep learning features,' *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 410–420, 2019.

[18] H. Wang, J. Hou and N. Chen, 'A survey of vehicle re-identification based on deep learning,' *IEEE Access*, vol. 7, pp. 172 443–172 469, 2019.

[19] L. Li, Y. Lv and F.-Y. Wang, 'Traffic signal timing via deep reinforcement learning,' *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[20] D. Ma, B. Zhou, X. Song and H. Dai, 'A deep reinforcement learning approach to traffic signal control with temporal traffic pattern mining,' *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 789–11 800, 2021.

[21] T. Ridnik, E. Ben-Baruch, A. Noy and L. Zelnik-Manor, 'Imagenet-21k pretraining for the masses,' *arXiv preprint arXiv:2104.10972*, 2021.

[22] L. Espeholt et al., 'Deep learning for twelve hour precipitation forecasts,' *Nature communications*, vol. 13, no. 1, p. 5145, 2022.

[23] R. M. French, 'Catastrophic forgetting in connectionist networks,' *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[24] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov and J. van de Weijer, 'Class-incremental learning: Survey and performance evaluation on image classification,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[25] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach and T. Tuytelaars, 'Memory aware synapses: Learning what (not) to forget,' in *ECCV*, 2018.

[26] F. Zenke, B. Poole and S. Ganguli, 'Continual learning through synaptic intelligence,' pp. 3987–3995, 2017.

[27] J. Kirkpatrick et al., 'Overcoming catastrophic forgetting in neural networks,' *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[28] Z. Li and D. Hoiem, 'Learning without forgetting,' *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[29] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu and R. Chellappa, 'Learning without memorizing,' *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5133–5141, 2019.

[30] A. Chaudhry, M. Ranzato, M. Rohrbach and M. Elhoseiny, 'Efficient lifelong learning with a-gem,' *arXiv preprint arXiv:1812.00420*, 2018.

[31] S.-A. Rebuffi, A. Kolesnikov, G. Sperl and C. H. Lampert, 'Icarl: Incremental classifier and representation learning,' pp. 2001–2010, 2017.

[32] L. B. Solum and M. Chung, 'The layers principle: Internet achitecture and the law,' *Notre Dame L. Rev.*, vol. 79, p. 815, 2003.

[33] A. Prabhu, P. H. Torr and P. K. Dokania, 'Gdumb: A simple approach that questions our progress in continual learning,' in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, Springer, 2020, pp. 524–540.

[34] S. Hou, X. Pan, C. C. Loy, Z. Wang and D. Lin, 'Learning a unified classifier incrementally via rebalancing,' *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 831–839, 2019.

[35] Y. Wu et al., 'Large scale incremental learning,' pp. 374–382, 2019.

[36] E. Belouadah and A. Popescu, 'Il2m: Class incremental learning with dual memory,' in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 583–592.

[37] Z. Wang et al., 'Learning to prompt for continual learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 139–149.

[38] Z. Wang et al., 'Dualprompt: Complementary prompting for rehearsal-free continual learning,' in *European Conference on Computer Vision*, Springer, 2022, pp. 631–648.

[39] J. S. Smith et al., 'Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 11 909–11 919.

[40] X. Tao, X. Hong, X. Chang, S. Dong, X. Wei and Y. Gong, 'Few-shot class-incremental learning,' *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 180–12 189, 2020.

[41] J. Liu, L. Song and Y. Qin, 'Prototype rectification for few-shot learning,' in *ECCV*, Springer, 2020, pp. 741–756.

[42] G. Shi, J. Chen, W. Zhang, L.-M. Zhan and X.-M. Wu, 'Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima,' in *NeurIPS*, 2021.

[43] Z. Chi, L. Gu, H. Liu, Y. Wang, Y. Yu and J. Tang, 'Metafscil: A meta-learning approach for few-shot class incremental learning,' in *CVPR*, 2022, pp. 14 166–14 175.

[44] Y. Yang, H. Yuan, X. Li, Z. Lin, P. Torr and D. Tao, 'Neural collapse inspired feature-classifier alignment for few-shot class-incremental learning,' 2023. [Online]. Available: https://openreview.net/forum?id=y5W8tpojhtJ.

[45] N. Ahmed, A. Kukleva and B. Schiele, 'Orco: Towards better generalization via orthogonality and contrast for few-shot class-incremental learning,' pp. 28 762–28 771, 2024.

[46] J. Yoon, W. Jeong, G. Lee, E. Yang and S. J. Hwang, 'Federated continual learning with weighted inter-client transfer,' in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 073–12 086.

[47] J. Dong et al., 'Federated class-incremental learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 164–10 173.

[48] J. Dong, D. Zhang, Y. Cong, W. Cong, H. Ding and D. Dai, 'Federated incremental semantic segmentation,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3934–3943.

[49] Z. Cai, O. Sener and V. Koltun, 'Online continual learning with natural distribution shifts: An empirical study with visual data,' in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 8281–8290.

[50] H. Koh, D. Kim, J.-W. Ha and J. Choi, 'Online continual learning on class incremental blurry task configuration with anytime inference,' *arXiv preprint arXiv:2110.10031*, 2021.

[51] Y. Wei, J. Ye, Z. Huang, J. Zhang and H. Shan, 'Online prototype learning for online continual learning,' in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 764–18 774.

[52] Y. Wu, H. Wang, P. Zhao, Y. Zheng, Y. Wei and L.-K. Huang, 'Mitigating catastrophic forgetting in online continual learning by modeling previous task interrelations via pareto optimization,' in *Forty-first International Conference on Machine Learning*, 2024.

[53] B. Kim, M. Seo and J. Choi, 'Online continual learning for interactive instruction following agents,' *arXiv preprint arXiv:2403.07548*, 2024.

[54] M. De Lange et al., 'A continual learning survey: Defying forgetting in classification tasks,' *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

[55] L. Wang, X. Zhang, H. Su and J. Zhu, 'A comprehensive survey of continual learning: Theory, method and application,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[56] P. Mazumder, P. Singh and P. Rai, 'Few-shot lifelong learning,' in *AAAI*, 2021.
[57] P. Buzzega, M. Boschini, A. Porrello, D. Abati and S. Calderara, 'Dark experience for general continual learning: A strong, simple baseline,' *Advances in neural information processing systems*, vol. 33, pp. 15 920–15 930, 2020.
[58] Y. Cui, W. Xiong, M. Tavakolian and L. Liu, 'Semi-supervised few-shot class-incremental learning,' in *ICIP*, IEEE, 2021, pp. 1239–1243.
[59] J. Dong, H. Li, Y. Cong, G. Sun, Y. Zhang and L. V. Gool, 'No one left behind: Real-world federated class-incremental learning,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–17, 2023. DOI: 10.1109/TPAMI.2023.3334213.
[60] M. G. Z. A. Khan, M. F. Naeem, L. Van Gool, D. Stricker, F. Tombari and M. Z. Afzal, 'Introducing language guidance in prompt-based continual learning,' in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 463–11 473.
[61] J. Snell, K. Swersky and R. S. Zemel, 'Prototypical networks for few-shot learning,' *ArXiv*, vol. abs/1703.05175, 2017.
[62] S. Cha, H. Hsu, F. du Pin Calmon and T. Moon, 'Cpr: Classifier-projection regularization for continual learning,' *ArXiv*, vol. abs/2006.07326, 2021.
[63] J. Zhang, C. Chen, W. Zhuang and L. Lyu, 'Target: Federated class-continual learning via exemplar-free distillation,' in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4782–4793.
[64] M.-T. Tran, T. Le, X.-M. Le, M. Harandi and D. Phung, 'Text-enhanced data-free approach for federated class-incremental learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 870–23 880.
[65] A. Roy, R. Moulick, V. K. Verma, S. Ghosh and A. Das, 'Convolutional prompting meets language models for continual learning,' pp. 23 616–23 626, 2024.
[66] J. Kirkpatrick et al., *Overcoming catastrophic forgetting in neural networks*, 2017.
[67] Z. Li and D. Hoiem, 'Learning without forgetting,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 2935–2947, 2018.
[68] I. Paik, S. Oh, T. Kwak and I. Kim, 'Overcoming catastrophic forgetting by neuron-level plasticity control,' *ArXiv*, vol. abs/1907.13322, 2020.
[69] F. Mao, W. Weng, M. Pratama and E. Yapp, 'Continual learning via inter-task synaptic mapping,' *ArXiv*, vol. abs/2106.13954, 2021.
[70] A. A. Rusu et al., 'Progressive neural networks,' *ArXiv*, vol. abs/1606.04671, 2016.

[71] J. Yoon, E. Yang, J. Lee and S. J. Hwang, 'Lifelong learning with dynamically expandable networks,' *ArXiv*, vol. abs/1708.01547, 2018.

[72] X.-l. Li, Y. Zhou, T. Wu, R. Socher and C. Xiong, 'Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting,' in *ICML*, 2019.

[73] A. Rakaraddi, S.-K. Lam, M. Pratama and M. V. de Carvalho, 'Reinforced continual learning for graphs,' *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022.

[74] J. Xu, J. Ma, X. Gao and Z. Zhu, 'Adaptive progressive continual learning.,' *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2021.

[75] M. Pratama, A. Ashfahani and E. Lughofer, 'Unsupervised continual learning via self-adaptive deep clustering approach,' *ArXiv*, vol. abs/2106.14563, 2021.

[76] A. Ashfahani and M. Pratama, 'Unsupervised continual learning in streaming environments,' *IEEE transactions on neural networks and learning systems*, vol. PP, 2022.

[77] D. Lopez-Paz and M. Ranzato, 'Gradient episodic memory for continual learning,' in *NIPS*, 2017.

[78] A. Chaudhry, M. Ranzato, M. Rohrbach and M. Elhoseiny, 'Efficient lifelong learning with a-gem,' *ArXiv*, vol. abs/1812.00420, 2019.

[79] A. Chaudhry, A. Gordo, P. Dokania, P. H. S. Torr and D. Lopez-Paz, 'Using hindsight to anchor past knowledge in continual learning,' in *AAAI*, 2021.

[80] M. V. de Carvalho, M. Pratama, J. Zhang and Y. San, 'Class-incremental learning via knowledge amalgamation,' *ArXiv*, vol. abs/2209.02112, 2022.

[81] A. Dosovitskiy et al., 'An image is worth 16x16 words: Transformers for image recognition at scale,' *arXiv preprint arXiv:2010.11929*, 2020.

[82] B. Lester, R. Al-Rfou and N. Constant, 'The power of scale for parameter-efficient prompt tuning,' *arXiv preprint arXiv:2104.08691*, 2021.

[83] X. L. Li and P. Liang, 'Prefix-tuning: Optimizing continuous prompts for generation,' *arXiv preprint arXiv:2101.00190*, 2021.

[84] Y. Wang, Z. Huang and X. Hong, 'S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning,' *Advances in Neural Information Processing Systems*, vol. 35, pp. 5682–5695, 2022.

[85] L. Wang, J. Xie, X. Zhang, M. Huang, H. Su and J. Zhu, 'Hierarchical decomposition of prompt-based continual learning: Rethinking obscured sub-optimality,' *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[86] Z. Gao, J. Cen and X. Chang, 'Consistent prompting for rehearsal-free continual learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 28 463–28 473.

[87] M. R. Kurniawan et al., 'Evolving parameterized prompt memory for continual learning,' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 13 301–13 309.

[88] J. S. Smith et al., 'Continual diffusion: Continual customization of text-to-image diffusion with c-lora,' *arXiv preprint arXiv:2304.06027*, 2023.

[89] Y.-S. Liang and W.-J. Li, 'Inflora: Interference-free low-rank adaptation for continual learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 638–23 647.

[90] Q. Gao et al., 'A unified continual learning framework with general parameter-efficient tuning,' in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 483–11 493.

[91] D.-W. Zhou, H.-L. Sun, H.-J. Ye and D.-C. Zhan, 'Expandable subspace ensemble for pre-trained model-based class-incremental learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 554–23 564.

[92] H.-L. Sun, D.-W. Zhou, H. Zhao, L. Gan, D.-C. Zhan and H.-J. Ye, 'Mos: Model surgery for pre-trained model-based class-incremental learning,' *arXiv preprint arXiv:2412.09441*, 2024.

[93] L. Sun, M. Zhang, B. Wang and P. Tiwari, 'Few-shot class-incremental learning for medical time series classification,' *JBHI*, vol. 28, no. 4, pp. 1872–1882, 2024. DOI: 10.1109/JBHI.2023.3247861.

[94] Y. Zhao, L. Ha, H. Wang and X. Ma, 'Few-shot class incremental learning for hyperspectral image classification based on constantly updated classifier,' in *IGARSS*, IEEE, 2022, pp. 1376–1379.

[95] Y. Wang, N. J. Bryan, M. Cartwright, J. P. Bello and J. Salamon, 'Few-shot continual learning for audio classification,' in *ICASSP*, IEEE, 2021, pp. 321–325.

[96] Z. Tan, K. Ding, R. Guo and H. Liu, 'Graph few-shot class-incremental learning,' in *WSDM*, 2022, pp. 987–996.

[97] M. Boschini, L. Bonicelli, P. Buzzega, A. Porrello and S. Calderara, 'Class-incremental continual learning into the extended der-verse,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5497–5512, 2022.

[98] M. A. Ma'Sum, M. Pratama, R. Savitha, L. Liu, Habibullah and R. Kowalczyk, 'Unsupervised few-shot continual learning for remote sensing image scene classification,' *TGRS*, vol. 62, pp. 1–14, 2024. DOI: 10.1109/TGRS.2024.3445887.

[99] M. Hersche, G. Karunaratne, G. Cherubini, L. Benini, A. Sebastian and A. Rahimi, 'Constrained few-shot class-incremental learning,' in *CVPR*, 2022, pp. 9057–9067.
[100] I. Achituve, A. Navon, Y. Yemini, G. Chechik and E. Fetaya, 'Gp-tree: A gaussian process classifier for few-shot incremental learning,' in *ICML*, PMLR, 2021, pp. 54–65.
[101] B. Liu, B. Yang, L. Xie, R. Wang, Q. Tian and Q. Ye, 'Learnable distribution calibration for few-shot class-incremental learning,' *TPAMI*, vol. 45, no. 10, pp. 12 699–12 706, 2023. DOI: 10.1109/TPAMI.2023.3273291.
[102] M. A. Ma'sum, M. Pratama, E. Lughofer, L. Liu, Habibullah and R. Kowalczyk, 'Few-shot continual learning via flat-to-wide approaches,' *TNNLS*, pp. 1–13, 2024. DOI: 10.1109/TNNLS.2024.3415048.
[103] H. Liu et al., 'Few-shot class-incremental learning via entropy-regularized data-free replay,' in *ECCV*, 2022, pp. 146–162.
[104] A. F. Akyürek, E. Akyürek, D. Wijaya and J. Andreas, 'Subspace regularizers for few-shot class incremental learning,' in *ICLR*, 2022. [Online]. Available: https://openreview.net/forum?id=boJy41J-tnQ.
[105] T. Ahmad et al., 'Few-shot class incremental learning leveraging self-supervised features,' in *CVPR*, 2022, pp. 3900–3910.
[106] J. Lin, Z. Wu, W. Lin, J. Huang and R. Luo, 'M2sd: Multiple mixing self-distillation for few-shot class-incremental learning,' in *AAAI*, vol. 38, 2024, pp. 3422–3431.
[107] H. Kang, J. Yoon, S. R. H. Madjid, S. J. Hwang and C. D. Yoo, 'On the soft-subnetwork for few-shot class incremental learning,' *ICLR*, 2022.
[108] D.-Y. Kim, D.-J. Han, J. Seo and J. Moon, 'Warping the space: Weight space rotation for class-incremental few-shot learning,' in *ICLR*, 2023.
[109] C. Finn, P. Abbeel and S. Levine, 'Model-agnostic meta-learning for fast adaptation of deep networks,' *ArXiv*, vol. abs/1703.03400, 2017.
[110] S. W. Yoon, D.-Y. Kim, J. Seo and J. Moon, 'Xtarnet: Learning to extract task-adaptive representation for incremental few-shot learning,' in *ICML*, 2020.
[111] D.-W. Zhou, H.-J. Ye, L. Ma, D. Xie, S. Pu and D.-C. Zhan, 'Few-shot class-incremental learning by sampling multi-phase tasks,' *TPAMI*, vol. 45, no. 11, pp. 12 816–12 831, 2022.
[112] K. Zhu, Y. Cao, W. Zhai, J. Cheng and Z.-J. Zha, 'Self-promoted prototype refinement for few-shot class-incremental learning,' in *CVPR*, 2021, pp. 6801–6810.
[113] A. Cheraghian et al., 'Synthesized feature based few-shot class-incremental learning on a mixture of subspaces,' in *CVPR*, 2021, pp. 8661–8670.

[114] A. Cheraghian, S. Rahman, P. Fang, S. K. Roy, L. Petersson and M. Harandi, 'Semantic-aware knowledge distillation for few-shot class-incremental learning,' in *CVPR*, 2021, pp. 2534–2543.

[115] H. Zhuang, Z. Weng, R. He, Z. Lin and Z. Zeng, 'Gkeal: Gaussian kernel embedded analytic learning for few-shot class incremental task,' in *CVPR*, 2023.

[116] Y. Zou, S. Zhang, Y. Li and R. Li, 'Margin-based few-shot class-incremental learning with class-level overfitting mitigation,' *NIPS*, vol. 35, pp. 27 267–27 279, 2022.

[117] Y. Zou, S. Zhang, H. Zhou, Y. Li and R. Li, 'Compositional few-shot class-incremental learning,' *ICML*, 2024.

[118] Q.-W. Wang, D.-W. Zhou, Y.-K. Zhang, D.-C. Zhan and H.-J. Ye, 'Few-shot class-incremental learning via training-free prototype calibration,' *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[119] Y.-M. Tang, Y.-X. Peng, J. Meng and W.-S. Zheng, 'Rethinking few-shot class-incremental learning: Learning from yourself,' *ECCV*, 2024.

[120] Z. Song, Y. Zhao, Y. Shi, P. Peng, L. Yuan and Y. Tian, 'Learning with fantasy: Semantic-aware virtual contrastive constraint for few-shot class-incremental learning,' in *CVPR*, 2023, pp. 24 183–24 192.

[121] D.-W. Zhou, F.-Y. Wang, H.-J. Ye, L. Ma, S. Pu and D.-C. Zhan, 'Forward compatible few-shot class-incremental learning,' in *CVPR*, 2022, pp. 9046–9056.

[122] C. Peng, K. Zhao, T. Wang, M. Li and B. C. Lovell, 'Few-shot class-incremental learning from an open-set perspective,' in *ECCV*, Springer, 2022, pp. 382–397.

[123] H. Zhao, Y. Fu, M. Kang, Q. Tian, F. Wu and X. Li, 'Mgsvf: Multi-grained slow versus fast framework for few-shot class-incremental learning,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 3, pp. 1576–1588, 2024. DOI: 10.1109/TPAMI.2021.3133897.

[124] L. Zhao et al., 'Few-shot class-incremental learning via class-aware bilateral distillation,' in *CVPR*, 2023.

[125] C. Zhang, N. Song, G. Lin, Y. Zheng, P. Pan and Y. Xu, 'Few-shot incremental learning with continually evolved classifiers,' *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 450–12 459, 2021.

[126] B. Yang et al., 'Learnable expansion-and-compression network for few-shot class-incremental learning,' *arXiv preprint arXiv:2104.02281*, 2021.

[127] B. Yang et al., 'Dynamic support network for few-shot class incremental learning,' *TPAMI*, vol. 45, no. 3, pp. 2945–2951, 2022.

[128] C. Liu et al., 'Few-shot class incremental learning with attention-aware self-adaptive prompt,' *ECCV*, 2024.
[129] K.-H. Park, K. Song and G.-M. Park, 'Pre-trained vision and language transformers are few-shot incremental learners,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 881–23 890.
[130] H. Sun, J. Zhou, X. He, J. Xu and Y. Peng, 'Finefmpl: Fine-grained feature mining prompt learning for few-shot class incremental learning,' *IJCAI*, 2024.
[131] X. Wang, Z. Ji, X. Liu, Y. Pang and J. Han, 'On the approximation risk of few-shot class-incremental learning,' in *ECCV*, Springer, 2025, pp. 162–178.
[132] Y. Guo, B. Liu and D. Zhao, 'Dealing with cross-task class discrimination in online continual learning,' in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 11 878–11 887.
[133] Y. Gu, X. Yang, K. Wei and C. Deng, 'Not just selection, but exploration: Online class-incremental continual learning via dual view consistency,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 7442–7451.
[134] J. Yoo, Y. Liu, F. Wood and G. Pleiss, 'Layerwise proximal replay: A proximal point method for online continual learning,' in *Proceedings of the 41st International Conference on Machine Learning*, ser. ICML'24, Vienna, Austria: JMLR.org, 2024.
[135] M. Wang, N. Michel, J. Mao and T. Yamasaki, 'Dealing with synthetic data contamination in online continual learning,' in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [Online]. Available: https://openreview.net/forum?id=Lc8gemv97Y.
[136] M. Seo, H. Koh and J. Choi, 'Budgeted online continual learning by adaptive layer freezing and frequency-based sampling,' in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=dOAkHmsjRX.
[137] F. Huo, W. Xu, J. Guo, H. Wang and Y. Fan, 'Non-exemplar online class-incremental continual learning via dual-prototype self-augment and refinement,' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 12 698–12 707.
[138] M. Seo et al., 'Learning equi-angular representations for online continual learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 933–23 942.
[139] N. Michel, M. Wang, L. Xiao and T. Yamasaki, 'Rethinking momentum knowledge distillation in online continual learning,' in *Proceedings of the*

41st International Conference on Machine Learning, ser. ICML'24, Vienna, Austria: JMLR.org, 2024.

[140] M. Wang, N. Michel, L. Xiao and T. Yamasaki, 'Improving plasticity in online continual learning via collaborative learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 460–23 469.

[141] M. Wang, N. Michel, L. Xiao and T. Yamasaki, 'Improving plasticity in online continual learning via collaborative learning,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 23 460–23 469.

[142] W. Xinrui, C. Geng, W. Wan, S.-Y. Li and S. Chen, 'Forgetting, ignorance or myopia: Revisiting key challenges in online continual learning,' vol. 37, 2024, pp. 58 341–58 375.

[143] A. Prabhu, S. Sinha, P. Kumaraguru, P. Torr, O. Sener and P. Dokania, 'Randumb: Random representations outperform online continually learned representations,' *Advances in Neural Information Processing Systems*, vol. 37, pp. 37 988–38 006, 2024.

[144] M. D. McDonnell, D. Gong, A. Parvaneh, E. Abbasnejad and A. Van den Hengel, 'Ranpac: Random projections and pre-trained models for continual learning,' *Advances in Neural Information Processing Systems*, vol. 36, pp. 12 022–12 053, 2023.

[145] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y Arcas, 'Communication-efficient learning of deep networks from decentralized data,' in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.

[146] J. Zhang, C. Chen, W. Zhuang and L. Lv, 'Addressing catastrophic forgetting in federated class-continual learning,' *arXiv preprint arXiv:2303.06937*, 2023.

[147] D. Qi, H. Zhao and S. Li, 'Better generative replay for continual federated learning,' *arXiv preprint arXiv:2302.13001*, 2023.

[148] A. Odena, C. Olah and J. Shlens, 'Conditional image synthesis with auxiliary classifier gans,' in *International conference on machine learning*, PMLR, 2017, pp. 2642–2651.

[149] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith, 'Federated optimization in heterogeneous networks,' *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[150] H. Shin, J. K. Lee, J. Kim and J. Kim, 'Continual learning with deep generative replay,' vol. 30, 2017.

[151] A. Usmanova, F. Portet, P. Lalanda and G. Vega, 'A distillation-based approach integrating continual learning and federated learning for pervasive services,' *arXiv preprint arXiv:2109.04197*, 2021.

[152] G. Bagwe, X. Yuan, M. Pan and L. Zhang, 'Fed-cprompt: Contrastive prompt for rehearsal-free federated continual learning,' *arXiv preprint arXiv:2307.04869*, 2023.
[153] J. Liu et al., 'Federated class-incremental learning with prompting,' *arXiv preprint arXiv:2310.08948*, 2023.
[154] H. Guo, F. Zhu, W. Liu, X.-Y. Zhang and C.-L. Liu, 'Pilora: Prototype guided incremental lora for federated class-incremental learning,' in *Proceedings of the European Conference on Computer Vision*, 2024.
[155] T. Hospedales, A. Antoniou, P. Micaelli and A. Storkey, 'Meta-learning in neural networks: A survey,' *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2021.
[156] M. Kaya and H. Ş. Bilge, 'Deep metric learning: A survey,' *Symmetry*, vol. 11, no. 9, p. 1066, 2019.
[157] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li and Y. Gao, 'A survey on federated learning,' *Knowledge-Based Systems*, vol. 216, p. 106 775, 2021.
[158] T. Kumar, R. Brennan, A. Mileo and M. Bendechache, 'Image data augmentation approaches: A comprehensive survey and future directions,' *IEEE Access*, 2024.
[159] W. Zheng, J. Lu and J. Zhou, 'Deep metric learning via adaptive learnable assessment,' *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2957–2966, 2020.
[160] R. Volpi, D. Larlus and G. Rogez, 'Continual adaptation of visual representations via domain randomization and meta-learning,' in *CVPR*, 2021.
[161] G. M. van de Ven and A. Tolias, 'Three scenarios for continual learning,' *ArXiv*, vol. abs/1904.07734, 2019.
[162] F. M. Castro, M. J. Marín-Jiménez, N. G. Mata, C. Schmid and A. Karteek, 'End-to-end incremental learning,' *ArXiv*, vol. abs/1807.09536, 2018.
[163] Q. Pham, C. Liu, D. Sahoo and S. Hoi, 'Contextual transformation networks for online continual learning,' in *ICLR*, 2021.
[164] Y. Wang, J. Lin, J. Zou, Y. Pan, T. Yao and T. Mei, 'Improving self-supervised learning with automated unsupervised outlier arbitration,' in *NeurIPS*, 2021.
[165] A. van der Vaart, 'Asymptotic statistics: Frontmatter,' 1998.
[166] S. Chen, E. Dobriban and J. Lee, 'A group-theoretic framework for data augmentation,' *arXiv: Machine Learning*, 2020.
[167] Z. Chen and B. Liu, 'Lifelong machine learning,' in *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 2016.
[168] J. Zhang et al., 'Class-incremental learning via deep model consolidation,' in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1131–1140.

[169] A. Chaudhry, A. Gordo, P. Dokania, P. Torr and D. Lopez-Paz, 'Using hindsight to anchor past knowledge in continual learning,' in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 6993–7001.
[170] A. Chaudhry et al., 'On tiny episodic memories in continual learning,' *arXiv: Learning*, 2019.
[171] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, 'Gradient-based learning applied to document recognition,' *Proc. IEEE*, vol. 86, pp. 2278–2324, 1998.
[172] A. Krizhevsky, 'Learning multiple layers of features from tiny images,' 2009.
[173] W. Hu, Q. Qin, M. Wang, J. Ma and B. Liu, 'Continual learning by using information of each class holistically,' in *AAAI*, 2021.
[174] G. M. van de Ven, H. T. Siegelmann and A. S. Tolias, 'Brain-inspired replay for continual learning with artificial neural networks,' *Nature Communications*, vol. 11, 2020.
[175] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra et al., 'Matching networks for one shot learning,' vol. 29, 2016.
[176] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov and J. van de Weijer, 'Class-incremental learning: Survey and performance evaluation,' *arXiv preprint arXiv:2010.15277*, 2020.
[177] H. Liu, Y. Zhou, B. Liu, J. Zhao, R. Yao and Z. Shao, 'Incremental learning with neural networks for computer vision: A survey,' *Artificial intelligence review*, vol. 56, no. 5, pp. 4557–4589, 2023.
[178] M. Biesialska, K. Biesialska and M. R. Costa-Jussa, 'Continual lifelong learning in natural language processing: A survey,' *arXiv preprint arXiv:2012.09823*, 2020.
[179] Z. Tian, D. Zhang and H.-N. Dai, 'Continual learning on graphs: A survey,' *arXiv preprint arXiv:2402.06330*, 2024.
[180] K. Shaheen, M. A. Hanif, O. Hasan and M. Shafique, 'Continual learning for real-world autonomous systems: Algorithms, challenges and frameworks,' *Journal of Intelligent & Robotic Systems*, vol. 105, no. 1, p. 9, 2022.
[181] X. Cao, H. Lu, L. Huang, X. Liu and M.-M. Cheng, 'Generative multimodal models are good class incremental learners,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 28 706–28 717.
[182] Y. Shen, X. Wang, P. Gao and M. Lin, 'Auxiliary modality learning with generalized curriculum distillation,' in *International Conference on Machine Learning*, PMLR, 2023, pp. 31 057–31 076.
[183] D. Hendrycks et al., 'The many faces of robustness: A critical analysis of out-of-distribution generalization,' *ICCV*, 2021.
[184] C. Wah, S. Branson, P. Welinder, P. Perona and S. Belongie, 'The caltech-ucsd birds-200-2011 dataset,' 2011.

[185] J. Qiao, X. Tan, C. Chen, Y. Qu, Y. Peng, Y. Xie et al., 'Prompt gradient projection for continual learning,' in *The Twelfth International Conference on Learning Representations*, 2023.

[186] G. Zhang, L. Wang, G. Kang, L. Chen and Y. Wei, 'Slca: Slow learner with classifier alignment for continual learning on a pre-trained model,' in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 148–19 158.

[187] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan and S. Wermter, 'Continual lifelong learning with neural networks: A review,' *Neural networks : the official journal of the International Neural Network Society*, vol. 113, pp. 54–71, 2019.

[188] K. Chen and C.-G. Lee, 'Incremental few-shot learning via vector quantization in deep embedded space,' in *ICLR*, 2021.

[189] P. Sun, Y. Ouyang, W. Zhang and X. Dai, 'Meda: Meta-learning with data augmentation for few-shot text classification,' in *IJCAI*, 2021.

[190] S. Kullback, 'Mathematical statistics. john e. freund. prentice-hall, englewood cliffs, n.j., 1962. 260 pp. illus. $7.50,' *Science*, vol. 136, pp. 1114–1115, 1962.

[191] B. Kang et al., 'Decoupling representation and classifier for long-tailed recognition,' *ArXiv*, vol. abs/1910.09217, 2020.

[192] J. Kalla and S. Biswas, 'S3c: Self-supervised stochastic classifiers for few-shot class-incremental learning,' *ArXiv*, vol. abs/2307.02246, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:253099859.

[193] Z. Pan, X. Yu, M. Zhang and Y. Gao, 'Ssfe-net: Self-supervised feature enhancement for ultra-fine-grained few-shot class incremental learning,' in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, pp. 6275–6284.

[194] S. D. Bhat, B. Banerjee, S. Chaudhuri and A. Bhattacharya, 'Cilea-net: Curriculum-based incremental learning framework for remote sensing image classification,' *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 5879–5890, 2021.

[195] D. Ye, J. Peng, H. Li and L. Bruzzone, 'Better memorization, better recall: A lifelong learning framework for remote sensing image scene classification,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[196] M. Zhai, H. Liu and F. Sun, 'Lifelong learning for scene recognition in remote sensing images,' *IEEE Geoscience and Remote Sensing Letters*, vol. 16, pp. 1472–1476, 2019.

[197] N. Ammour, 'Continual learning using data regeneration for remote sensing scene classification,' *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–5, 2022.

[198] X. Lu, X. Sun, W. Diao, Y. Feng, P. Wang and K. Fu, 'Lil: Lightweight incremental learning approach through feature transfer for remote sensing image scene classification,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–20, 2021.

[199] Y. Zhang, W. Li, M. Zhang, Y. Qu, R. Tao and H. Qi, 'Topological structure and semantic information transfer network for cross-scene hyperspectral image classification,' *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, pp. 2817–2830, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237545097.

[200] Y. Zhang, W. Li, W. Sun, R. Tao and Q. Du, 'Single-source domain expansion network for cross-scene hyperspectral image classification,' *IEEE Transactions on Image Processing*, vol. 32, pp. 1498–1512, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252089530.

[201] Y. Zhang, M. Zhang, W. Li, S. Wang and R. Tao, 'Language-aware domain generalization network for cross-scene hyperspectral image classification,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–12, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252110949.

[202] M. Zhang, W. Li, Y. Zhang, R. Tao and Q. Du, 'Hyperspectral and lidar data classification based on structural optimization transmission,' *IEEE Transactions on Cybernetics*, vol. 53, pp. 3153–3164, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248777741.

[203] Y. Zhang, W. Li, M. Zhang, S. Wang, R. Tao and Q. Du, 'Graph information aggregation cross-domain few-shot learning for hyperspectral image classification.,' *IEEE transactions on neural networks and learning systems*, vol. PP, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250145582.

[204] Z. Huang, J. Chen, J. Zhang and H. Shan, 'Learning representation for clustering via prototype scattering and positive sampling.,' *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2021.

[205] M. A. Ma'sum, M. Pratama, L. Liu, E. D. Lughofer, Habibullah and R. Kowalczyk, 'Few-shot continual learning via flat-to-wide approaches,' *ArXiv*, vol. abs/2306.14369, 2023.

[206] S. Dang, Z. Cui, Z. Cao, Y. Pi and X. Feng, 'Distribution reliability assessment-based incremental learning for automatic target recognition,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–13, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258822540.

[207] Y. Liu, Z. Xiong, Y. Yuan and Q. Wang, 'Distilling knowledge from super-resolution for efficient remote sensing salient object detection,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–16, 2023.

[208] Y. Liu, Z. Xiong, Y. Yuan and Q. Wang, 'Transcending pixels: Boosting saliency detection via scene understanding from aerial imagery,' *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[209] Y. Fu, Z. Liu, C. Wu, F. Wu and M. Liu, 'Class-incremental recognition of objects in remote sensing images with dynamic hybrid exemplar selection,' *IEEE Transactions on Aerospace and Electronic Systems*, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267529798.

[210] L. Wang, X. Yang, H. Tan, X. Bai and F. Zhou, 'Few-shot class-incremental sar target recognition based on hierarchical embedding and incremental evolutionary network,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 61, pp. 1–11, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257147398.

[211] Y. Liu et al., 'Few-shot object detection in remote-sensing images via label-consistent classifier and gradual regression,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–14, 2024. DOI: 10.1109/TGRS.2024.3369666.

[212] X. Lu et al., 'Few-shot incremental object detection in aerial imagery via dual-frequency prompt,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–17, 2024. DOI: 10.1109/TGRS.2024.3400595.

[213] W. Wang et al., 'Gradient guided multi-scale feature collaboration networks for few-shot class-incremental remote sensing scene classification,' *TGRS*, 2024.

[214] Z. Wu, Y. Xiong, S. X. Yu and D. Lin, 'Unsupervised feature learning via non-parametric instance discrimination,' *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

[215] X. Chen and K. He, 'Exploring simple siamese representation learning,' *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15 745–15 753, 2020.

[216] Y. Yang and S. Newsam, 'Bag-of-visual-words and spatial extensions for land-use classification,' in *ACM SIGSPATIAL International Workshop on Advances in Geographic Information Systems*, 2010.

[217] G.-S. Xia et al., 'Aid: A benchmark data set for performance evaluation of aerial scene classification,' *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, pp. 3965–3981, 2016.

[218] H. Li, C. Tao, Z. Wu, J. Chen, J. Gong and M. Deng, 'Rsi-cb: A large-scale remote sensing image classification benchmark using crowdsourced data,' *Sensors (Basel, Switzerland)*, vol. 20, 2017.

[219] G. Cheng, J. Han and X. Lu, 'Remote sensing image scene classification: Benchmark and state of the art,' *Proceedings of the IEEE*, vol. 105, no. 10, pp. 1865–1883, Oct. 2017, ISSN: 1558-2256. DOI: 10.1109/jproc.2017.2675998. [Online]. Available: http://dx.doi.org/10.1109/JPROC.2017.2675998.

[220] P. Helber, B. Bischke, A. Dengel and D. Borth, 'Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification,' *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[221] Y. Fu, Z. Liu and Z. Zhang, 'Progressive learning vision transformer for open set recognition of fine-grained objects in remote sensing images,' *IEEE Transactions on Geoscience and Remote Sensing*, 2023.

[222] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich and A. T. Suresh, 'Scaffold: Stochastic controlled averaging for federated learning,' in *International conference on machine learning*, PMLR, 2020, pp. 5132–5143.

[223] N. Shoham et al., 'Overcoming forgetting in federated learning on non-iid data,' *arXiv preprint arXiv:1910.07796*, 2019.

[224] H. Hwang et al., 'Towards the practical utility of federated learning in the medical domain,' in *Conference on Health, Inference, and Learning*, PMLR, 2023, pp. 163–181.

[225] W. He, H. Yao, T. Mai, F. Wang and M. Guizani, 'Three-stage stackelberg game enabled clustered federated learning in heterogeneous uav swarms,' *IEEE Transactions on Vehicular Technology*, 2023.

[226] M. J. Baucas, P. Spachos and K. N. Plataniotis, 'Federated learning and blockchain-enabled fog-iot platform for wearables in predictive healthcare,' *IEEE Transactions on Computational Social Systems*, 2023.

[227] Y. Tan et al., 'Fedproto: Federated prototype learning across heterogeneous clients,' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 8432–8440.

[228] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang and J. Feng, 'No fear of heterogeneity: Classifier calibration for federated learning with non-iid data,' *Advances in Neural Information Processing Systems*, vol. 34, pp. 5972–5984, 2021.

[229] A. Vaswani et al., 'Attention is all you need,' *Advances in neural information processing systems*, vol. 30, 2017.

[230] X. Li, K. Huang, W. Yang, S. Wang and Z. Zhang, 'On the convergence of fedavg on non-iid data,' *arXiv preprint arXiv:1907.02189*, 2019.

[231] L. Bottou, F. E. Curtis and J. Nocedal, 'Optimization methods for large-scale machine learning,' *SIAM review*, vol. 60, no. 2, pp. 223–311, 2018.

[232] M. Kang, J. Park and B. Han, 'Class-incremental learning by knowledge distillation with adaptive feature consolidation,' in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 16 071–16 080.

[233] A. Douillard, A. Ramé, G. Couairon and M. Cord, 'Dytox: Transformers for continual learning with dynamic token expansion,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9285–9295.

[234] H. Ahn, J. Kwak, S. Lim, H. Bang, H. Kim and T. Moon, 'Ss-il: Separated softmax for incremental learning,' in *Proceedings of the IEEE/CVF International conference on computer vision*, 2021, pp. 844–853.

[235] C. Simon, P. Koniusz and M. Harandi, 'On learning the geodesic path for incremental learning,' in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 1591–1600.

[236] X. Hu, K. Tang, C. Miao, X.-S. Hua and H. Zhang, 'Distilling causal effect of data in class-incremental learning,' in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 3957–3966.

[237] A. Douillard, M. Cord, C. Ollion, T. Robert and E. Valle, 'Podnet: Pooled outputs distillation for small-tasks incremental learning,' in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, Springer, 2020, pp. 86–102.

[238] B. Zhang, X. Li, S. Feng, Y. Ye and R. Ye, 'Metanode: Prototype optimization as a neural ode for few-shot learning,' in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 9014–9021.

[239] J. Kalla and S. Biswas, 'S3c: Self-supervised stochastic classifiers for few-shot class-incremental learning,' pp. 432–448, 2022.

[240] D. Goswami, B. Twardowski and J. Van De Weijer, 'Calibrating higher-order statistics for few-shot class-incremental learning with pre-trained vision transformers,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4075–4084.

[241] R. T. Chen, Y. Rubanova, J. Bettencourt and D. K. Duvenaud, 'Neural ordinary differential equations,' *Advances in neural information processing systems*, vol. 31, 2018.

[242] R. Alexander, 'Solving ordinary differential equations i: Nonstiff problems (e. hairer, sp norsett, and g. wanner),' *Siam Review*, vol. 32, no. 3, p. 485, 1990.

[243] M. A. Ma'sum, M. Pratama, S. Ramasamy, L. Liu, H. Habibullah and R. Kowalczyk, 'Pip: Prototypes-injected prompt for federated class incremental learning,' ser. CIKM '24, Boise, ID, USA: Association for Comput-

ing Machinery, 2024, pp. 1670–1679, ISBN: 9798400704369. DOI: 10.1145/3627673.3679794.

[244] Y. Jiang, Y. Cheng, D. Wang and B. Song, 'Rra-ffscil: Inter-intra classes representation and relationship augmentation federated few-shot incremental learning,' *Neurocomputing*, p. 127 956, 2024.

[245] H. Zhuang et al., 'F-oal: Forward-only online analytic learning with fast training and low memory footprint in class incremental learning,' *Advances in Neural Information Processing Systems*, vol. 37, pp. 41 517–41 538, 2024.

[246] L. Zhao, X. Zhang, K. Yan, S. Ding and W. Huang, 'Safe: Slow and fast parameter-efficient tuning for continual learning with pre-trained models,' *Advances in Neural Information Processing Systems*, vol. 37, pp. 113 772–113 796, 2025.

[247] A. Krizhevsky, G. Hinton et al., 'Learning multiple layers of features from tiny images,' 2009.

[248] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt and D. Song, 'Natural adversarial examples,' *CVPR*, 2021.

[249] M. Riemer et al., 'Learning to learn without forgetting by maximizing transfer and minimizing interference,' *arXiv preprint arXiv:1810.11910*, 2018.

[250] L. Caccia, R. Aljundi, N. Asadi, T. Tuytelaars, J. Pineau and E. Belilovsky, 'New insights on reducing abrupt representation change in online continual learning,' *arXiv preprint arXiv:2104.05025*, 2021.

[251] Y. Guo, B. Liu and D. Zhao, 'Online continual learning through mutual information maximization,' in *International conference on machine learning*, PMLR, 2022, pp. 8109–8126.

[252] G. Zhou, K. Sohn and H. Lee, 'Online incremental feature learning with denoising autoencoders,' in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:5805711.

[253] D. Sahoo, Q. Pham, J. Lu and S. C. H. Hoi, 'Online deep learning: Learning deep neural networks on the fly,' in *Proceedings of International Joint Conference on Artificial Intelligence*, 2018.

[254] M. Das, M. Pratama, S. Savitri and J. Zhang, 'Muse-rnn: A multilayer self-evolving recurrent neural network for data stream classification,' *Proceedings of 2019 IEEE International Conference on Data Mining (ICDM)*, pp. 110–119, 2019.

[255] M. Pratama, C. Za'in, A. Ashfahani, Y. Ong and W. Ding, 'Automatic construction of multi-layer perceptron network from streaming examples,' *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019.

[256] A. Ashfahani and M. Pratama, 'Autonomous deep learning: Continual learning approach for dynamic environments,' in *Proceedings of SIAM International Conference on Data Mining (SDM)*, 2019.

[257] E. D. Lughofer, M. Pratama and I. Skrjanc, 'Online bagging of evolving fuzzy systems,' *Inf. Sci.*, vol. 570, pp. 16–33, 2021.

[258] M. Pratama, W. Pedrycz and E. D. Lughofer, 'Evolving ensemble fuzzy classifier,' *IEEE Transactions on Fuzzy Systems*, vol. 26, pp. 2552–2567, 2018.

[259] M. Pratama, W. Pedrycz and G. I. Webb, 'An incremental construction of deep neuro fuzzy system for continual learning of nonstationary data streams,' *IEEE Transactions on Fuzzy Systems*, vol. 28, pp. 1315–1328, 2020.

[260] V. Cerqueira, L. Torgo, F. Pinto and C. Soares, 'Arbitrated ensemble for time series forecasting,' in *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, 2017.

[261] A. Saadallah, M. Jakobs and K. Morik, 'Explainable online deep neural network selection using adaptive saliency maps for time series forecasting,' in *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, 2021.

[262] E. Eldele et al., 'Time-series representation learning via temporal and contextual contrasting,' *ArXiv*, vol. abs/2106.14112, 2021.

[263] H. Zhou et al., 'Informer: Beyond efficient transformer for long sequence time-series forecasting,' in *Proceedings of AAAI Conference on Artificial Intelligence*, 2021.

[264] H. Wu, J. Xu, J. Wang and M. Long, 'Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting,' in *Proceedings of Neural Information Processing Systems*, 2021.

[265] Q. H. Pham, C. Liu and S. C. H. Hoi, 'Dualnet: Continual learning, fast and slow,' in *Proceedings of Neural Information Processing System*, 2021.

[266] L. Liu, W. Hamilton, G. Long, J. Jiang and H. Larochelle, 'A universal representation transformer layer for few-shot image classification,' *ArXiv*, vol. abs/2006.11702, 2021.

[267] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun and R. Jin, 'Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting,' *ArXiv*, vol. abs/2201.12740, 2022.

[268] A. Zeng, M.-H. Chen, L. Zhang and Q. Xu, 'Are transformers effective for time series forecasting?' *ArXiv*, vol. abs/2205.13504, 2022.

[269] R. R. Chowdhury, X. Zhang, J. Shang, R. K. Gupta and D. Hong, 'Tarnet: Task-aware reconstruction for time-series transformer,' *Proceedings of the*

28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022.

[270] J. Zbontar, L. Jing, I. Misra, Y. LeCun and S. Deny, 'Barlow twins: Self-supervised learning via redundancy reduction,' in *Proceedings of International Conference on Machine Learning*, 2021.

[271] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy and A. Bouchachia, 'A survey on concept drift adaptation,' *ACM Computing Surveys (CSUR)*, vol. 46, pp. 1–37, 2014.

[272] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty and C. Eickhoff, 'A transformer-based framework for multivariate time series representation learning,' *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

[273] E. Perez, F. Strub, H. de Vries, V. Dumoulin and A. C. Courville, 'Film: Visual reasoning with a general conditioning layer,' in *Proceedings of AAAI Conference on Artificial Intelligence*, 2018.

[274] A. Vaswani et al., 'Attention is all you need,' *ArXiv*, vol. abs/1706.03762, 2017.

[275] N. Kitaev, L. Kaiser and A. Levskaya, 'Reformer: The efficient transformer,' *ArXiv*, vol. abs/2001.04451, 2020.

[276] S. LI et al., 'Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting,' *ArXiv*, vol. abs/1907.00235, 2019.

[277] G. Lai, W.-C. Chang, Y. Yang and H. Liu, 'Modeling long- and short-term temporal patterns with deep neural networks,' *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

[278] S. Hochreiter and J. Schmidhuber, 'Long short-term memory,' *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[279] S. Bai, J. Z. Kolter and V. Koltun, 'An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,' *ArXiv*, vol. abs/1803.01271, 2018.

Appendix A: Code Repositories

This following repositories contains material from the included publications.

1. The code of AGLA (chapter 3) is published in https://github.com/anwarmaxsum/AGLA
2. The code of LEAPGen (chapter 4) is published in https://github.com/anwarmaxsum/LEAPGEN
3. The code of FLOWER (chapter 5) is published in https://github.com/anwarmaxsum/FLOWER
4. The code of UNISA (chapter 6) is published in https://github.com/anwarmaxsum/UNISA
5. The code of PIP (chapter 7) is published in https://github.com/anwarmaxsum/PIP
6. The code of FFSCIL (chapter 8) is published in https://github.com/anwarmaxsum/FFSCIL
7. The code of PROL (chapter 9) will be published in https://github.com/anwarmaxsum/PROL
8. The code of MANTRA (chapter 10) is published in https://github.com/anwarmaxsum/MANTRA

Appendix B: Statements of Authorship

This appendix contains the statements of authorship from the included publications.

University of South Australia

Statement of Authorship

Title of Paper	Assessor-guided learning for continual environments
Publication Status	■ Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M Pratama, E Lughofer, W Ding, W Jatmiko. Assessor-guided learning for continual environments. Information Sciences 640, 119088. doi.org/10.1016/j.ins.2023.119088

PRINCIPAL AUTHOR

Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum	
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis, editing the manuscript, and revising the manuscript.	
Overall Percentage (%)	50	
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14 May 2025

CO-AUTHOR CONTRIBUTIONS

By signing the Statement of Authorship, each author certifies that:

I. the candidate's stated contribution to the publication is accurate (as detailed above);

II. permission is granted for the candidate in include the publication in the thesis; and

III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.

Name of Co-Author 1	Mahardhika Pratama

% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 2	Edwin Lughofer (Passed away)	
% Contribution to the Paper	7	
Signature *Signatures will be redacted by the Library at the time of publication*		Date:
Name of Co-Author 3	Weiping Ding	
% Contribution to the Paper	6	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 15/05/2025
Name of Co-Author 4	Wisnu Jatmiko	
% Contribution to the Paper	6	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 5		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	Vision and language synergy for rehearsal free continual learning
Publication Status	■ Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M Pratama, S. Ramasamy, L Liu, H Habibullah, R Kowalczyk. Vision and Language Synergy for Rehearsal Free Continual Learning. The Thirteenth International Conference on Learning Representations (ICLR 2025). openreview.net/forum?id=9aZ2ixiYGd
PRINCIPAL AUTHOR	
Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis, preparing the manuscript, editing the manuscript, and revising the manuscript.
Overall Percentage (%)	50
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted] Date: 14 May 2025
CO-AUTHOR CONTRIBUTIONS	
By signing the Statement of Authorship, each author certifies that: I. the candidate's stated contribution to the publication is accurate (as detailed above); II. permission is granted for the candidate in include the publication in the thesis; and III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.	
Name of Co-Author 1	Mahardhika Pratama

% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 2	Savitha Ramasamy	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14 May 2025
Name of Co-Author 3	Lin Liu	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14 May 2025
Name of Co-Author 4	Habibullah Habibullah	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 15 May 2025
Name of Co-Author 5	Ryszard Kowalczyk	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	**14/5/25** Date
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	Few-shot continual learning via flat-to-wide approaches
Publication Status	■ Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M Pratama, E Lughofer, L Liu, H Habibullah, R Kowalczyk. Few-shot continual learning via flat-to-wide approaches. IEEE Transactions on Neural Networks and Learning Systems. 36 (5), 8966-8978. doi.org/10.1109/TNNLS.2024.3415048
PRINCIPAL AUTHOR	
Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis,, editing the manuscript, and revising the manuscript.
Overall Percentage (%)	50
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted] Date: 14 May 2025
CO-AUTHOR CONTRIBUTIONS	
By signing the Statement of Authorship, each author certifies that: I. the candidate's stated contribution to the publication is accurate (as detailed above); II. permission is granted for the candidate in include the publication in the thesis; and III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.	
Name of Co-Author 1	Mahardhika Pratama

% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*		Date: 14/05/2025
Name of Co-Author 2	Edwin Lughofer (Passed away)	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 3	Lin Liu	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*		Date 14 May 2025
Name of Co-Author 4	Habibullah Habibullah	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*		Date 15 May 2025
Name of Co-Author 5	Ryszard Kowalczyk	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*		Date14/5/25
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	Unsupervised Few-Shot Continual Learning for Remote Sensing Image Scene Classification
Publication Status	■ Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M Pratama, R Savitha, L Liu, H Habibullah, R Kowalczyk. Unsupervised Few-Shot Continual Learning for Remote Sensing Image Scene Classification. IEEE Transactions on Geoscience and Remote Sensing 62, 4707214. doi.org/10.1109/TGRS.2024.3445887

PRINCIPAL AUTHOR

Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum	
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis, editing the manuscript, and revising the manuscript.	
Overall Percentage (%)	50	
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14 May 2025

CO-AUTHOR CONTRIBUTIONS

By signing the Statement of Authorship, each author certifies that:

I. the candidate's stated contribution to the publication is accurate (as detailed above);

II. permission is granted for the candidate in include the publication in the thesis; and

III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.

Name of Co-Author 1	Mahardhika Pratama

% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 2	Savitha Ramasamy	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14/05/2025
Name of Co-Author 3	Lin Liu	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14 May 2025
Name of Co-Author 4	Habibullah Habibullah	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 15 May 2025
Name of Co-Author 5	Ryszard Kowalczyk	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date14/5/25
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	PIP: Prototypes-Injected Prompt for Federated Class Incremental Learning
Publication Status	■ Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M Pratama, S Ramasamy, L Liu, H Habibullah, R Kowalczyk. PIP: Prototypes-Injected Prompt for Federated Class Incremental Learning. Proceedings of the 33rd ACM International Conference on Information and Management (CIKM). 2024: 1670–1679. doi.org/10.1145/3627673.3679794

PRINCIPAL AUTHOR

Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis, preparing the manuscript, editing the manuscript, and revising the manuscript.
Overall Percentage (%)	50
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted] Date: 14 May 2025

CO-AUTHOR CONTRIBUTIONS

By signing the Statement of Authorship, each author certifies that:

I. the candidate's stated contribution to the publication is accurate (as detailed above);

II. permission is granted for the candidate in include the publication in the thesis; and

III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.

Name of Co-Author 1	Mahardhika Pratama

% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 2	Savitha Ramasamy	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14/05/2025
Name of Co-Author 3	Lin Liu	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14 May 2025
Name of Co-Author 4	Habibullah Habibullah	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 15 May 2025
Name of Co-Author 5	Ryszard Kowalczyk	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date14/5/25
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	Federated Few-Shot Class-Incremental Learning.
Publication Status	■ Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M Pratama, L Liu, H Habibullah, R Kowalczyk. Federated Few-Shot Class-Incremental Learning. The Thirteenth International Conference on Learning Representations (ICLR 2025). openreview.net/forum?id=ZiPoAlKf9Y
PRINCIPAL AUTHOR	
Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis, preparing the manuscript, editing the manuscript, and revising the manuscript.
Overall Percentage (%)	50
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted] Date: 14 May 2025
CO-AUTHOR CONTRIBUTIONS	
By signing the Statement of Authorship, each author certifies that: I. the candidate's stated contribution to the publication is accurate (as detailed above); II. permission is granted for the candidate in include the publication in the thesis; and III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.	
Name of Co-Author 1	Mahardhika Pratama

% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 2	Lin Liu	
% Contribution to the Paper	7	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14 May 2025
Name of Co-Author 3	Habibullah Habibullah	
% Contribution to the Paper	6	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 15 May 2025
Name of Co-Author 4	Ryszard Kowalczyk	
% Contribution to the Paper	6	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14/5/25
Name of Co-Author 5		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	PROL : Rehearsal Free Continual Learning in Streaming Data via Prompt Online Learning
Publication Status	□ Published □ Accepted for Publication ■ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M. Pratama, S. Ramasamy, L. Liu, H. Habibullah, and R Kowalczyk. "PROL : Rehearsal Free Continual Learning in Streaming Data via Prompt Online Learning". Submitted to International Conference on Computer Vision 2025. https://openreview.net/forum?id=mKvOmPHnhO
PRINCIPAL AUTHOR	
Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis, preparing the manuscript, editing the manuscript, and revising the manuscript.
Overall Percentage (%)	50
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted] Date: 14 May 2025
CO-AUTHOR CONTRIBUTIONS	
By signing the Statement of Authorship, each author certifies that: I. the candidate's stated contribution to the publication is accurate (as detailed above); II. permission is granted for the candidate in include the publication in the thesis; and III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.	
Name of Co-Author 1	Mahardhika Pratama

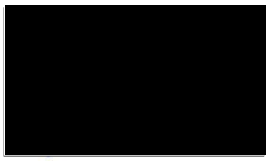
% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 2	Savitha Ramasamy	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14.05.2025
Name of Co-Author 3	Lin Liu	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14 May 2025
Name of Co-Author 4	Habibullah Habibullah	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 15 May 2025
Name of Co-Author 5	Ryszard Kowalczyk	
% Contribution to the Paper	5	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14/5/25
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	Dynamic Long-Term Time-Series Forecasting via Meta Transformer Networks
Publication Status	■ Published □ Accepted for Publication □ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, MD Sarkar, M Pratama, S Ramasamy, S Anavatti, L Liu, R Kowlczyk. Dynamic Long-Term Time-Series Forecasting via Meta Transformer Networks. IEEE Transactions on Artificial Intelligence 5 (8), 4258-4268. doi.org/10.1109/TAI.2024.336577
PRINCIPAL AUTHOR	
Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum
Contribution to the Paper Brief description of your work in this publication	Developing the proposed method, conducting experiments and analysis, editing the manuscript, and revising the manuscript.
Overall Percentage (%)	30
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted] Date: 14 May 2025
CO-AUTHOR CONTRIBUTIONS	
By signing the Statement of Authorship, each author certifies that: I. the candidate's stated contribution to the publication is accurate (as detailed above); II. permission is granted for the candidate in include the publication in the thesis; and III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.	
Name of Co-Author 1	MD Rasel Sarkar

% Contribution to the Paper	30	
Signature *Signatures will be redacted by the Library at the time of publication*		Date: 15/05/2025
Name of Co-Author 2	Mahardhika Pratama	
% Contribution to the Paper	20	
Signature *Signatures will be redacted by the Library at the time of publication*		Date: 14/05/2025
Name of Co-Author 3	Savitha Ramasamy	
% Contribution to the Paper	4	
Signature *Signatures will be redacted by the Library at the time of publication*		Date: 14/05/2025
Name of Co-Author 4	Sreenatha Anavatti	
% Contribution to the Paper	4	
Signature *Signatures will be redacted by the Library at the time of publication*		Date: 14/05/2025
Name of Co-Author 5	Lin Liu	
% Contribution to the Paper	4	
Signature *Signatures will be redacted by the Library at the time of publication*		Date: 14 May 2025
Name of Co-Author 6	Habibullah Habibullah	
% Contribution to the Paper	4	
Signature *Signatures will be redacted by the Library at the time of publication*		Date: 15/05/2025
Name of Co-Author 7	Ryszard Kowalczyk	
% Contribution to the Paper	4	

Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date 14/5/25
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date

University of South Australia

Statement of Authorship

Title of Paper	Latest Advancements Towards Catastrophic Forgetting under Data Scarcity: A Comprehensive Survey on Few-Shot Class Incremental Learning.
Publication Status	□ Published □ Accepted for Publication ■ Submitted for Publication □ Unpublished and Unsubmitted work written in manuscript style
Publication Details Relevant reference details	MA Ma'sum, M Pratama, I Skrjanc. Latest Advancements Towards Catastrophic Forgetting under Data Scarcity: A Comprehensive Survey on Few-Shot Class Incremental Learning. ArXiv:2502.08181. doi.org/10.48550/arXiv.2502.08181
PRINCIPAL AUTHOR	
Name of Principal Author (Candidate)	Muhammad Anwar Ma'sum
Contribution to the Paper Brief description of your work in this publication	Designing the survey, preparing the manuscript, editing the manuscript, and revising the manuscript.
Overall Percentage (%)	50
Certification	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted] Date: 14 May 2025
CO-AUTHOR CONTRIBUTIONS	

By signing the Statement of Authorship, each author certifies that:

I. the candidate's stated contribution to the publication is accurate (as detailed above);

II. permission is granted for the candidate in include the publication in the thesis; and

III. the sum of all co-author contribution is equal to 100% less the candidate's stated contribution.

Name of Co-Author 1	Mahardhika Pratama

% Contribution to the Paper	40	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 14/05/2025
Name of Co-Author 2	Igor Skrjanc	
% Contribution to the Paper	10	
Signature *Signatures will be redacted by the Library at the time of publication*	[redacted]	Date: 15/05/2025
Name of Co-Author 3		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 4		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 5		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 6		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 7		
% Contribution to the Paper		

Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 8		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 9		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date
Name of Co-Author 10		
% Contribution to the Paper		
Signature *Signatures will be redacted by the Library at the time of publication*		Date