

THE UNIVERSITY of ADELAIDE

The Role of Rank in Efficient Learning

Jianqiao Zheng
Australian Institute for Machine Learning (AIML)
School of Computer and Mathematical Sciences

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY
The University of Adelaide

November 3, 2025

Contents

Abstract xi

Declaration of Authorship xiii

Acknowledgements xv

Publications xvii

1 Introduction 1
- 1.1 Rank in MLPs . . . 1
- 1.2 Rank in Transfomers . . . 3

2 Rethinking Positional Encoding 7
- 2.1 Introduction . . . 7
- 2.2 Related work . . . 9
- 2.3 Positional encoding: a theoretical walk-through . . . 10
  - 2.3.1 Rank of the embedded representation . . . 10
  - 2.3.2 Bandlimited embedders . . . 11
  - 2.3.3 Distance preservation . . . 11
  - 2.3.4 Analysis of possible embedders . . . 12
  - 2.3.5 Connection to the Random Fourier Features . . . 14
- 2.4 Simplicity vs complexity in positional encoding . . . 15
  - 2.4.1 2D simple encoding . . . 15
  - 2.4.2 2D complex encoding . . . 15
  - 2.4.3 High dimensional encoding . . . 17
- 2.5 Experiments . . . 18
  - 2.5.1 Method Notations . . . 18
  - 2.5.2 1D: Rank of input & depth of Network . . . 19
  - 2.5.3 2D: image reconstruction . . . 19
  - 2.5.4 Visual results for 2D images . . . 20
  - 2.5.5 3D: video reconstruction . . . 21
- 2.6 Theoretical results . . . 21
  - 2.6.1 1D postional encoding . . . 21
  - 2.6.2 2D complex encoding . . . 26
    - 2.6.2.1 Closed form solution for separable coordinates . . . 26
    - 2.6.2.2 Blending matrix for non-separable coordinates . . . 27
    - 2.6.2.3 HD complexity . . . 28
- 2.7 Conclusion . . . 30

3 Robust Point Cloud Processing 37
- 3.1 Introduction . . . 37
- 3.2 Related work . . . 39
- 3.3 Method . . . 40

3.3.1 Band-limited per-point encoder . . . 41
3.3.2 Smoothness of per-point embedding . . . 42
3.3.3 Trade-off – bandwidth and smoothness . . . 43
3.4 Experiments . . . 44
3.4.1 Experimental settings . . . 44
3.4.2 Classification on clean data . . . 45
3.4.3 Robustness to noisy data . . . 45
3.4.4 Ablation study of different pooling functions . . . 48
3.4.5 Ablation study of PCT and PE-AT . . . 48
3.4.6 Point cloud registration . . . 49
3.5 Theoretical results . . . 50
3.5.1 RFF Derivation . . . 50
3.5.1.1 RFF in 2D . . . 50
3.5.1.2 RFF in 3D . . . 55
3.5.1.3 Frequency distribution derivation . . . 56
3.5.2 Pooling functions . . . 57
3.5.2.1 Proof of Lemma 1 . . . 58
3.6 Limitations . . . 60
3.7 Conclusion . . . 61

4 All You Need is the First Layer 67
4.1 Introduction . . . 67
4.2 Preliminaries . . . 68
4.2.1 MLP Network . . . 68
4.2.2 Neural Tangent Kernel . . . 69
4.2.3 Appoximation of rank . . . 69
4.3 From NTK to Jacobian . . . 70
4.3.1 Definition of Jacobian . . . 70
4.3.2 Relation to Performance through NTK . . . 70
4.3.3 Decomposition of Jacobian into Layers . . . 71
4.3.4 Decomposition of Jacobian into Weights and Embeddings . . . 72
4.4 Why the Rank of Embeddings in the First Layer . . . 73
4.5 Our Initialization Strategy . . . 77
4.5.1 1-D Case . . . 77
4.5.2 2-D Case . . . 77
4.6 Experiments . . . 78
4.6.1 Rank of Input Embedding . . . 78
4.6.2 2-D Image Representation . . . 79
4.6.3 3-D Occupancy Representation . . . 79
4.7 Conclusion . . . 80

5 Structured Initialization for Vision Transformers 83
5.1 Background . . . 83
5.1.1 Similarity between ConvMixer and ViT . . . 83
5.1.2 Rank of Spatial Mixing Functions . . . 84
5.2 Introduction . . . 85
5.3 Related Work . . . 86
5.4 Why Random Impulse Filters Work? . . . 87
5.5 Structured Initialization for Attention Map . . . 89
5.5.1 Expected Initialized Attention Map Structure . . . 89
5.5.2 Solving $\mathbf{Q}_{\text{init}}$ and $\mathbf{K}_{\text{init}}$ . . . 90

5.6 Experiments and Analysis . . . 91
5.6.1 Small and Medium-Scale Datasets . . . 92
5.6.2 Large-Scale Datasets . . . 92
5.6.3 ViT Variants . . . 94
5.6.4 Attention Maps . . . 97
5.6.5 Pre-trained Model . . . 97
5.7 Additional Results . . . 98
5.7.1 Theorectical Results . . . 98
5.7.1.1 Proof for Proposition 1 . . . 98
5.7.1.2 Convolutional Representation Matrix . . . 98
5.7.2 Experimental Results . . . 100
5.7.2.1 Training Details and Training Curves for ViT-Base . . . 100
5.7.2.2 Dataset Scale . . . 101
5.7.2.3 Additional Statistical Results for Small and Medium-Scale Datasets . . . 102
5.7.2.4 Experimental Validation of the Proposition and Corollaries . . . 102
5.7.2.5 Attention Maps . . . 103
5.8 Limitations . . . 103
5.9 Conclusion . . . 105

6 Conclusion **107**
6.1 Limitations and Future Work . . . 108

List of Figures

2.1 Overview of the proposed non-Fourier positional encoding scheme by shifted basis functions. . . . 8
2.2 Stable rank of the Gaussian embedder vs the standard deviation for different numbers of samples. . . . 13
2.3 Quantitative comparison of the stable rank and the distance preservation of different embedders and RFFs. . . . 14
2.4 The stable rank and distance preservation of Gaussian embedder and RFF across different standard deviations. . . . 15
2.5 Complex encoding uses a linear interpolation of grid points positional encodings to represent positional encodings of non-grid points. . . . 16
2.6 Illustration of different methods to extend 1D encoding of length K to N-dimensional case. . . . 17
2.7 The performance of the reconstruction depends on the depth of the network and the rank of the input embedding. . . . 19
2.8 The normalized singular values of different 1D embeddings. . . . 21
2.9 Reconstruction results of an archway using separable coordinates (regular-grid sampled training points) with different combinations of simple or complex encodings and network depths. . . . 31
2.10 Reconstruction results of a heap of walnuts using separable coordinates (regular-grid sampled training points) with different combinations of simple or complex encodings and network depths. . . . 32
2.11 Reconstruction results of a lion using non-separable coordinates (randomly sampled training points) with different combinations of simple or complex encodings and network depths. . . . 33
2.12 Reconstruction results of a seaside residential area using non-separable coordinates (randomly sampled training points) with different combinations of simple or complex encodings and network depths. . . . 34

3.1 The robustness of positional embedding (PE)-based methods (PE-AT and PE) in comparison to end-to-end trained methods (PointNet and PCT). . . . 38
3.2 Illustration of different point cloud processing architectures. . . . 39
3.3 Illustration of a 1D point cloud and its embeddings. . . . 41
3.4 Encoding distance *vs*. point distance for (A) one ReLU layer and (B) RFF-based PE. . . . 43
3.5 Error rate of our PE method with different pooling functions under different data corruptions. . . . 47
3.6 Point cloud registration accuracy on ModelNet40 20 categories with Gaussian noise. . . . 49
3.7 A visualization of point cloud registration using (a) PointNetLK Revisited and (b) PE on ModelNet40 dataset. . . . 52
3.8 Illustration of how 1D and 2D band-limited embeddings sample the frequencies. . . . 53

3.9 Frequency distributions in D-dimensional space where D=2, 3, 4. . . . 54
3.10 Classification accuracy of different methods on our data corruption dataset of various severity. . . . 62
3.11 Classification accuracy of different methods on the original ModelNet40-C dataset. . . . 63
3.12 Continued figure of classification accuracy of different methods on the original ModelNet40-C dataset. . . . 64
3.13 Visualization of noise and outliers. . . . 65

4.1 Reconstruction results of the ReLU-activated MLP using the default initialization compared to our proposed initialization. . . . 68
4.2 Random weights are naturally well-conditioned. . . . 74
4.3 Illustration of a thought experiment on the capacity of a coordinate MLP. 75
4.4 Image representation results of a 3-layer ReLU-MLP with no batch normalization and batch normalization at layer 1, 2, or 3 individually. 76
4.5 Illustration of initialization in 1-D case. . . . 77
4.6 Illustration of our initialization in 2-D. . . . 78
4.7 Comparison of SVD of input embeddings and NTK kernel with various methods. . . . 79
4.8 Visualization of meshes generated with occupancy volumes with various methods. . . . 80

5.1 The architectures of ConvMixer Trockman and Kolter [2022] and Simple ViT Beyer et al. [2022]. . . . 84
5.2 Illustration of conventional generative initialization and structured initialization (ours) strategies for the weights **Q** and **K** of the attention map in transformers. . . . 86
5.3 Illustration of why random spatial convolution filters are effective. . . . 88
5.4 Visualization of attention maps in ViT-T using ours, mimetic Trockman and Kolter [2023], and default Xu et al. [2024] initializations. . . . 94
5.5 Training curves of ViT-Base using default, mimetic, and impulse initialization under three different training configurations. . . . 101
5.6 Classification accuracy of ViT-Tiny with different initialization methods on different datasets (mean of 5 runs). . . . 102
5.7 Visualization of attention maps in ViT-T using our impulse initialization method, mimetic [Trockman and Kolter, 2023], and default [Wightman, 2019] initializations. . . . 104

List of Tables

2.1 Performance of natural 2D image reconstruction without random sampling (separable coordinates). 20
2.2 Performance of 2D image reconstruction with randomly sampled inputs (non-separable coordinates). 20
2.3 Performance of video reconstruction without random sampling (separable coordinates). 22
2.4 Performance of video reconstruction with randomly sampled inputs (non-separable coordinates). 22

3.1 Classification accuracy(%) on ModelNet40 of RFF-based PE with different scales of RFF. 44
3.2 Classification results of different models on modified ModelNet40-C (generated by larger corruption severities). 46
3.3 Ablation study on different architectures of PCT and PE-AT. 48
3.4 Performance on other data corruptions as provided in ModelNet-C . . 50
3.5 Classification results of different models on ModelNet40-C with 3 different data corruptions. 51

4.1 2-D image representation PSRN of a collection of 16 images from Div2k. 79

5.1 Classification accuracy of ViT-Tiny with different initialization methods on different datasets. 93
5.2 Classification accuracy of ViT-Tiny, ViT-Small and ViT-Base on ImageNet-1K dataset with different initialization methods. 95
5.3 Classification accuracy of different Swin Transformers and MLP-Mixer on different datasets with different initialization methods. 96
5.4 Classification accuracy of ViT-Tiny on small-scale datasets with the weight selection method. 99
5.5 Ablation study on training settings of ViT-Base on ImageNet-1K dataset. 100
5.6 Detailed scale of all the datasets used in the Chapter. 101
5.7 Classification accuracy(%) of ConvMixer (depth 8) with different filter sizes, embedding dimensions on CIFAR-10. 103

The University of Adelaide

Abstract

The Role of Rank in Efficient Learning

by Jianqiao Zheng

Despite the remarkable success of deep learning in recent years, the underlying mechanisms that enable neural networks to perform effectively remain poorly understood. In this thesis, we investigate the behavior of deep learning through the lens of *rank*. We find that the rank of embeddings within neural networks plays a critical role in enabling efficient learning, in both shallow MLPs and deep models.

Shallow MLPs have regained popularity in implicit neural representations (INRs) due to their architectural simplicity. However, MLPs alone are insufficient for effectively capturing complex signals. To address this limitation, various architectural enhancements have been proposed for INRs, including alternative input encodings, unconventional activation functions, and batch normalization. These modifications yield models that better capture high-frequency information.

We begin our study with one of the most widely used techniques: positional encoding (PE), and identify *rank* as a key factor underlying its effectiveness. We demonstrate that the performance of PE is governed by a trade-off between the *stable rank* and the distance preservation between embedded coordinates. Moreover, we broaden the conventional Fourier analysis perspective by showing that alternative, non-Fourier embedding functions can also serve as effective positional encodings.

Next, we illustrate this in the context of 3D point cloud processing. By modulating the rank-distance trade-off, we can control the effective bandwidth of features extracted from point clouds, thereby enhancing robustness to out-of-distribution noise.

Furthermore, we reveal that beyond PE, the primary effect of these techniques is to increase the rank of the embeddings, particularly in the first layer. Based on a direct connection we establish between the Neural Tangent Kernel (NTK) and the Jacobians of each network layer, this improved conditioning of the embeddings subsequently enhances the NTK spectrum, ultimately enabling MLPs to learn more efficiently.

Analyzing the rank of embeddings in deep models such as convolutional neural networks (CNNs) and vision Transformers (ViTs) is challenging due to the large number of parameters and the diverse formats of their embeddings. We begin our investigation by focusing on attention maps, a key architectural component that differentiates ViTs from CNNs. While randomly initialized attention maps are typically low-rank, we propose a novel initialization strategy that incorporates both a convolutional structure and high rank. This initialization enables ViTs to learn more efficiently from small-scale datasets.

Declaration of Authorship

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Jianqiao Zheng

June 2025

Acknowledgements

I still remember the first time I met Prof. Simon Lucey, shortly after completing my master's degree at the University of Michigan, Ann Arbor. I had just arrived in Pittsburgh to work with him as a research assistant at Carnegie Mellon University. He was a young and energetic professor who opened the door to research for me. His constant enthusiasm and encouragement always lifted me up, especially during challenging times. He has not only been my PhD advisor but also a lifelong mentor who has guided and inspired me both academically and personally.

I would also like to sincerely thank Dr. Sameera Ramasinghe and Dr. Hemanth Saratchandran for their invaluable guidance during the first and final years of my PhD. Their support and insights played a crucial role in shaping my research and helping me grow as a scholar.

Special thanks to Xueqian Li. Although we were in the same year, your experience and guidance in research have helped me tremendously throughout my PhD journey. Beyond academics, I also cherish the days we spent exploring Adelaide together when we first arrived — even on foot!

To all my friends in Adelaide, whether still around or gone, I will always carry you — and the life we shared in this city — in my heart.

Finally, I am deeply grateful to my parents for allowing me to pursue what I love.

"This mind is luminous; what more is there to say?"
— Wang Yangming

Publications

This thesis contains the following works that have been published or prepared for publication:

- *Trading Positional Complexity vs Deepness in Coordinate Networks*
  Jianqiao Zheng, Sameera Ramasinghe, Xueqian Li, Simon Lucey
  The European Conference on Computer Vision (ECCV), 2022

- *Robust Point Cloud Processing through Positional Embedding*
  Jianqiao Zheng, Xueqian Li, Sameera Ramasinghe, Simon Lucey
  International Conference on 3D Vision (3DV), 2024

- *All You Need is the First Layer: Rethinking Positional Encoding in MLPs*
  Jianqiao Zheng, Hemanth Saratchandran, Simon Lucey
  Manuscript, 2025

- *Structured Initialization for Vision Transformers*
  Jianqiao Zheng, Xueqian Li, Simon Lucey
  Conference on Neural Information Processing Systems (NeurIPS), 2025

In addition, I have the following papers not included in this thesis:

- *Fast Neural Scene Flow*
  Xueqian Li, **Jianqiao Zheng**, Francesco Ferroni, Jhony Kaesemodel Pontes, Simon Lucey
  The IEEE / CVF International Conference on Computer Vision (ICCV), 2023

- *Rethinking Softmax: Self-Attention with Polynomial Activations*
  Hemanth Saratchandran, **Jianqiao Zheng**, Yiping Ji, Wenbo Zhang, Simon Lucey
  ArXiv preprint, 2024

Chapter 1

Introduction

Rank is a foundational concept that determines (a) whether a system has a solution, and (b) whether that solution can be reached via gradient-based optimization methods Lucey [2025]. Consider a simple linear system:

$$\mathbf{Y} = \phi(\mathbf{X})\mathbf{W}. \tag{1.1}$$

It is well known that to guarantee the existence of a solution for the weights $\mathbf{W}$ that satisfies any target $\mathbf{Y}$, the embedding matrix $\phi(\mathbf{X})$ must be full rank. Meanwhile, in gradient descent-based training, the weight update is governed by $\phi(\mathbf{X})^\top$, and the reconstruction error evolves according to $\mathbf{I} - \phi(\mathbf{X})\phi(\mathbf{X})^\top$. This implies that for the reconstruction error to approach zero within a limited number of steps, the singular values of $\phi(\mathbf{X})$ should be well-conditioned—that is, they should not differ excessively in magnitude. This characterization highlights the role of rank in efficient learning, and a detailed theoretical walkthrough is provided in Chapter 4.

To some extent, these two aspects—existence of a solution and efficient convergence—are closely intertwined. A solution that exists in theory but cannot be reached within a reasonable number of iterations due to ill-conditioning is effectively meaningless in the context of deep learning, where non-linearity and training efficiency are critical. Nonetheless, in certain cases, an analytical solution can be integrated into a complex system or used to guide the design of improved initialization schemes, thereby making the theoretical solution practically accessible. Although the above discussion pertains to a linear system, the underlying principles extend naturally to more complex, nonlinear deep learning models.

Following this motivation, the first part of this thesis investigates the role of rank in multilayer perceptrons (MLPs), which serve as a relatively simple yet informative model in deep learning, in Chapters 2 to 4. Beyond simple MLPs, rank continues to play a critical role in more complex and deeper architectures such as Transformers in Chapter 5.

1.1 Rank in MLPs

Implicit Neural Representations (INRs)—typically implemented as multilayer perceptrons (MLPs) that map low-dimensional (often coordinate-like) inputs to continuous signal outputs such as 2D images, 3D shapes, or 5D radiance fields—have attracted significant attention in computer vision and machine learning. However, a major limitation of training such coordinate-based MLPs on raw input coordinates is their suboptimal ability to model high-frequency content. To address this, recent works have empirically demonstrated that projecting input coordinates into a higher-dimensional space using sinusoidal functions of varying frequencies (i.e., Fourier feature mappings) enables these networks to learn high-frequency information more effectively Mildenhall

et al. [2020]; Zhong et al. [2019]. Tancik *et al*. Tancik et al. [2020] further provided a theoretical perspective, showing that this projection alters the spectrum of the neural tangent kernel (NTK), thereby facilitating the learning of fine-grained signal structures.

Despite their empirical success, Fourier-based positional encodings exhibit several limitations. First, their performance is highly sensitive to the choice of frequency bands. Most methods rely on stochastic frequency sampling, which introduces instability—especially when the number of sampled frequencies is limited. Second, restricting the interpretation of positional encoding to a purely Fourier-based framework obscures some of the core principles behind its effectiveness. These concerns motivate the need for a more general and principled understanding of positional encoding mechanisms.

In Chapter 2, we address these limitations by introducing a broader theoretical framework for positional encoding, one that generalizes beyond Fourier features. A key contribution of our work is demonstrating that non-Fourier embedding functions—specifically, shifted continuous basis functions parameterized by spatial coordinates—can serve as effective alternatives. Unlike the heuristic or random sampling typically used in Fourier mappings, our formulation yields a clear and interpretable relationship between sampling density and the behavior of the embedding.

We identify two principal factors that govern the performance of such embeddings: (i) the approximate rank of the embedding matrix across coordinate positions, which influences the model's memorization capacity; and (ii) distance preservation, which measures how well inner products between embedded vectors reflect Euclidean distances between coordinates—thereby impacting generalization. Strikingly, we find that any continuous basis function can be used for positional encoding, provided that an appropriate trade-off between rank and distance preservation is maintained. Within this framework, we can also reinterpret the strengths and weaknesses of Fourier feature mappings, offering new insights into their behavior.

To further expand the expressivity and efficiency of INRs, we introduce a complex positional encoding scheme that enables the network to approximate high-dimensional signal mappings with a single linear layer. A central idea here is coordinate separability. For instance, in 1D signals, embeddings occur along a single direction. In higher-dimensional cases—such as 2D natural images or 3D video sequences—the coordinate axes remain separable, allowing us to construct full embeddings via Kronecker products across each axis. In contrast, for signals with non-separable coordinates, we introduce a blending matrix that linearly interpolates the separable components, yielding a unified embedding that captures cross-dimensional interactions.

End-to-end trained per-point embeddings are an essential ingredient of any state-of-the-art 3D point cloud processing, such as detection or alignment. Methods like PointNet Qi et al. [2017a], or the more recent point cloud transformer Guo et al. [2021]—and its variants—all employ learned per-point embeddings. Despite impressive performance, such approaches are sensitive to out-of-distribution (OOD) noise and outliers. In Chapter 3, we explore the hypothesis of whether a more robust performance can be afforded by frameworks such as PointNet or PCT if we replace the learned PPEs with analytical embeddings. This hypothesis has two motivations. First, we hypothesize that the earliest modules of modern point cloud networks—specifically the PPE—such as those found in PointNet and PCT, should be most generic in task and domain. As a result, learning such PPEs in an end-to-end fashion solely from data may make the entire pipeline overly sensitive to changes in the point cloud statistics Sun et al. [2022]. Second, the success of positional embedding (PE) in machine learning frameworks such as transformer networks Vaswani et al. [2017] and implicit

neural functions Mildenhall et al. [2021] offers an alternate surrogate for end-to-end PPEs. PE Zheng et al. [2022] shares similar characteristics to learned PPEs in that they attempt to transform a scalar or even vector position into a multi-dimensional embedding. They differ fundamentally, however, in that the entire embedding is analytical requiring only a single tunable hyper-parameter—bandwidth Zheng et al. [2022].

Beyond positional encoding (PE), a range of architectural modifications have been proposed for implicit neural representations (INRs), including alternative input encodings Xie et al. [2023], unconventional activation functions Ramasinghe and Lucey [2022], and batch normalization Cai et al. [2024], all aimed at enhancing the network's capacity to model high-frequency content. Previous studies Sitzmann et al. [2020]; Tancik et al. [2020] have leveraged Neural Tangent Kernel (NTK) theory Jacot et al. [2018] to explain the effectiveness of these approaches by linking spectral bias to the singular value distribution of a network's Jacobian. However, NTK provides only a global perspective and lacks the resolution to identify which architectural components most significantly impact this bias.

In Chapter 4, we address this limitation by introducing a more granular and interpretable framework. We decompose the NTK into contributions from the Jacobians of individual layers and further relate these to the rank of the hidden-layer embeddings. This rank-based view complements the classical NTK theory and reveals a unifying principle underlying seemingly diverse architectural interventions: all these techniques serve to increase the rank of the input or intermediate embeddings, thereby improving the network's spectral properties and its capacity to model high-frequency signals.

Crucially, we observe that parameters associated with low-rank embeddings are poorly optimized during training, as they reside in flat regions of the loss landscape and receive weak gradients. Thus, increasing the embedding rank early in the network—particularly in the first layer—is essential to ensure a favorable NTK spectrum and effective training. We empirically demonstrate that applying these architectural interventions (e.g., PE, batch normalization, or alternative activations) only at the first layer yields comparable performance to applying them throughout the network.

Motivated by this observation, we propose a simple yet effective initialization strategy for the first layer that raises the rank of the initial embeddings. This initialization alone enables MLPs operating on raw coordinate inputs to perform competitively, without requiring any architectural modifications.

1.2 Rank in Transfomers

Beyond simple MLPs, rank continues to play a crucial role in deeper and more complex architectures, such as Transformers. Vision Transformers (ViTs), in particular, have demonstrated remarkable performance when trained on large-scale datasets. However, their performance drops significantly when applied to small-scale datasets. To bridge this gap, several strategies have been explored, including self-supervised pretraining on large datasets [Dosovitskiy et al., 2021; Touvron et al., 2021], advanced data augmentation [Cubuk et al., 2020; Yun et al., 2019], and the integration of convolutional layers [Dai et al., 2021; Li et al., 2023; Liu et al., 2021a; Wu et al., 2021; Yuan et al., 2021].

More recently,Zhang et al. [2022] investigated initializing ViTs with pretrained weights, inspiring follow-up work that aims to improve ViT performance on small-scale datasets without modifying the core architecture. For example,Trockman and Kolter [2023] proposed a "mimetic" initialization method that mimics the distribution

of pretrained ViT weights. Similarly, Xu et al. [2024] directly sample weights from pretrained large models. However, both strategies rely on access to pretrained large-scale models, which are often infeasible in domain-specific or resource-constrained applications. Moreover, these approaches primarily focus on matching the distribution of attention weights, rather than explicitly structuring the attention maps themselves.

In Chapter 5, we introduce structured initialization for ViTs—an approach that avoids pretraining and instead focuses on imposing a meaningful structure on attention maps from the outset. This strategy is inspired by theoretical findings in Cazenavette et al. [2025], which show that structured bias in randomly initialized depth-wise convolutions enables comparable performance to learned counterparts in ConvMixer [Trockman and Kolter, 2022] and ResNet [He et al., 2016]. Crucially, unlike approaches that incorporate convolutions directly into the architecture [Dai et al., 2021; Li et al., 2023; Yuan et al., 2021], our method preserves the architectural purity and flexibility of ViTs, making it attractive for plug-and-play use.

An additional and intriguing observation is that many of these initialization strategies—whether implicit or explicit—tend to produce high-rank attention maps and embeddings. This suggests that the commonly used initialization schemes may not be well-suited for softmax-based attention, which can suffer from early training degeneracies when initialized improperly. While a precise rank-based analysis of attention mechanisms remains challenging due to the architectural complexity of Transformers, our findings offer an empirical foundation and promising direction for future research. We hope that this perspective sparks new insights into principled initialization strategies for attention-based models.

Statement of Authorship

Title of Paper	Trading positional complexity vs deepness in coordinate networks
Publication Status	☒ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Zheng, J., Ramasinghe, S., Li, X., & Lucey, S. (2022, October). Trading positional complexity vs deepness in coordinate networks. In European Conference on Computer Vision (pp. 144-160). Cham: Springer Nature Switzerland.

Principal Author

Name of Principal Author (Candidate)	Jianqiao Zheng		
Contribution to the Paper	Proposed ideas, conducted experiments, and wrote the manuscript.		
Overall percentage (%)			
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		Date	June 3, 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 contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Sameera Ramasinghe		
Contribution to the Paper	Supervised developement of work; Helped in discussion and writing revision.		
Signature		Date	June 11, 2025

Name of Co-Author	Xueqian Li		
Contribution to the Paper	Helped in discussion and writing revision.		
Signature		Date	June 7, 2025

Please cut and paste additional co-author panels here as required.

Name of Co-Author	Simon Lucey			
Contribution to the Paper	Supervised developement of work; Helped in discussion and writing revision.			
Signature			Date	23/6/25

Chapter 2

Rethinking Positional Encoding

Preface

This chapter is based on the following published paper:

> Zheng, Jianqiao, Sameera Ramasinghe, Xueqian Li, and Simon Lucey. *"Trading Positional Complexity vs Deepness in Coordinate Networks."* In: *European Conference on Computer Vision (ECCV)*. Cham: Springer Nature Switzerland, 2022, pp. 144–160.

This paper has been adapted for inclusion in this thesis. Minor edits have been made for consistency in notation and formatting. The work in this chapter contributes to the thesis by demonstrating that it is the *rank*, rather than the commonly assumed frequencies, that underlies the effectiveness of positional encoding.

2.1 Introduction

Positional encoding is an umbrella term used for representing the coordinates of a structured object as a finite-dimensional embedding. Such embeddings are fast becoming critical instruments in modern language models Bao et al. [2020]; Devlin et al. [2018]; Gehring et al. [2017]; Lewis et al. [2019]; Vaswani et al. [2017]; Yang et al. [2019] and vision tasks that involve encoding a signal (*e.g.*, 2D image, 3D object, etc.) as weights of a neural network Barron et al. [2021]; Gafni et al. [2020]; Li et al. [2020]; Martin-Brualla et al. [2020]; Mildenhall et al. [2020]; Ost et al. [2020]; Park et al. [2020]; Zhong et al. [2019]. Of specific interest in this chapter is the use of positional encodings when being used to enhance the performance of *coordinate-MLPs*. Coordinate-MLPs are fully connected networks, trained to learn the structure of an object as a continuous function, with coordinates as inputs. However, the major drawback of training coordinate-MLPs with raw input coordinates is their sub-optimal performance in learning high-frequency content Rahaman et al. [2019].

As a remedy, recent studies empirically confirmed that projecting the coordinates to a higher-dimensional space using sine and cosine functions of different frequencies (*i.e.*, Fourier frequency mapping) allows coordinate-MLPs to learn high-frequency information more effectively Mildenhall et al. [2020]; Zhong et al. [2019]. This observation was recently characterized theoretically by Tancik *et al*. Tancik et al. [2020], showing that the above projection permits tuning the spectrum of the neural tangent kernel (NTK) of the corresponding MLP, thereby enabling the network to learn high-frequency information. Despite impressive empirical results, encoding position through Fourier frequency mapping entails some unenviable attributes. First, prior research substantiates the belief that the performance of the Fourier feature mapping

is sensitive to the choice of frequencies. Leading methods for frequency selection, however, employ a stochastic strategy (*i.e.*, random sampling) which can become volatile as one attempts to keep to a minimum the number of sampled frequencies. Second, viewing positional encoding solely through a Fourier lens obfuscates some of the fundamental principles behind its effectiveness. These concerns have heightened the need for an extended analysis of positional encoding.

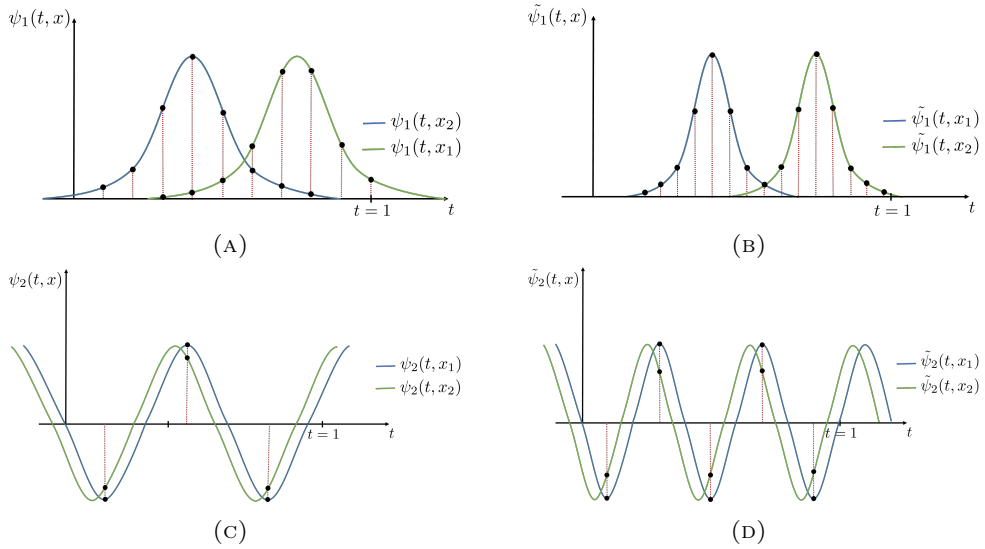


FIGURE 2.1. Overview of the proposed positional encoding scheme. Positions are encoded as equidistant samples from shifted basis functions (embedders). The shifts are determined by the corresponding coordinate positions we want to embed. In (a) and (b), x_1 and x_2 are encoded as samples from shifted Gaussians with a higher and a lower standard deviation, respectively. Note that we need a higher number of samples for (b) due to higher bandwidth (see Sec. 2.3). In (c) and (d), x_1 and x_2 are encoded with sinusoidal signals with different frequencies. Note that although different sampling rates are employed for (c) and (d), the same two values are repeated across the samples. Hence, sampling more than twice is redundant.

This chapter aims to overcome the aforesaid limitations by developing an alternative and more comprehensive understanding of positional encoding. The foremost benefit of our work is allowing non-Fourier embedding functions to be used in the positional encoding. Specifically, we show that positional encoding can be accomplished via systematic sampling of shifted continuous basis functions, where the shifts are determined by the coordinate positions, as illustrated in Fig. 2.1. In comparison to the ambiguous frequency sampling in Fourier feature mapping, we derive a more interpretable relationship between the sampling density and the behavior of the embedding scheme. In particular, we discover that the effectiveness of the proposed embedding scheme primarily relies on two factors: (i) the approximate matrix rank of the embedded representation across positions, and (ii) the distance preservation between the embedded coordinates. Distance preservation measures the extent to which the inner product between the shifted functions correlates with the Euclidean distance between the corresponding coordinates. Intuitively, a higher approximate matrix rank causes better memorization of the training data, while the distance preservation correlates with generalization. Remarkably, we establish that any given continuous function can be used for positional encoding—as performance is simply determined by the trade-off between the aforementioned two factors. Further, we assert that the effectiveness and

shortcomings of Fourier feature mapping can also be analyzed in the context of this newly developed framework. We also propose a complex positional encoding to relax the expressibility of the coordinate network into a single linear layer, which largely speeds the instance-based optimization. An essential idea here is the separation of the coordinates. For a simple 1D signal, the input is only embedded in one direction. As for 2D natural images and 3D video sequences, the coordinates are still separable, which enables us to use the Kronecker product to gather input embedding in every single direction. With signals that have non-separable coordinates, we add a blending matrix to linearly interpolate to get the final embedding. In summary, the contribution of this chapter is fourfold:

- We expand the current understanding of positional encoding and show that it can be formulated as a systematic sampling scheme of shifted continuous basis functions. Compared to the popular Fourier frequency mapping, our formulation is more interpretative in nature and less restrictive.
- We develop theoretical formulae to show that the performance of the encoding is governed by the approximate rank of the embedding matrix (sampled at different positions) and the distance preservation between the embedded coordinates. We further solidify this new insight using empirical evaluations.
- As a practical example, we employ a Gaussian signal as the embedding function and show that it can deliver on-par performance with the Fourier frequency mapping. Most importantly, we demonstrate that the Gaussian embedding is more efficient in terms of the embedding dimension while being less volatile.
- We show that trading a complex positional encoding for a deep network allows us to encode high-frequency features with a substantial speedup by circumventing the heavy computation for a simple positional encoding combined with a deep neural network. Promising empirical reconstruction performance is obtained on 1D, 2D, and 3D signals using our proposed embedding function in conjunction with coordinate networks.

2.2 Related work

Positional encoding became a popular topic among the machine learning community after the seminal work on Transformers by Vaswani *et al*. Vaswani et al. [2017]. Since the attention mechanism used in the Transformers is position-insensitive, they employed a sinusoidal signal to encode the positions before feeding them to the higher blocks. A contemporary work by Gehring *et al*. Gehring et al. [2017] also proposed a convolutional seq2seq model, adapting a positional encoding mechanism. Since then, using positional encoding in language models became a common trend Dai et al. [1901]; He et al. [2020]; Kitaev and Klein [2018]; Raffel et al. [2019]; Shaw et al. [2018]. Notably, Wang *et al*. Wang et al. [2019a] extended the embedding space from real numbers to complex values. Another critical aspect of their work is replacing the pre-defined encoding mechanism with a learnable one. There have also been other exciting attempts to improve positional encoding, such as extending the sequential positional encoding to tree-based positional encoding Shiv and Quirk [2019], untying the correlations between words and positions while embedding coordinates Ke et al. [2020], and modeling positional encoding using dynamical systems Liu et al. [2020].

In parallel, positional encoding is also gaining attention in computer vision, specifically with coordinate-MLPs. Coordinate-MLPs provide an efficient method to encode

objects such as images Nguyen et al. [2015]; Stanley [2007] and 3D scenes Niemeyer et al. [2020]; Saito et al. [2019]; Sitzmann et al. [2019] as their weights. Remarkably, Mildenhall *et al*. Mildenhall et al. [2020] and Zhong *et al*. Zhong et al. [2019] found that encoding coordinates with sinusoidal signals allow coordinate-MLPs to learn high-frequency content better. One of the earliest roots of this approach can perhaps be traced to the work by Rahimi and Recht Rahimi and Recht [2007], where they used random Fourier features to approximate an arbitrary stationary kernel function by applying Bochner's theorem. More recently, Tancik *et al*. Tancik et al. [2020], leveraging the NTK theory Arora et al. [2019]; Bietti and Mairal [2019]; Du et al. [2018]; Jacot et al. [2018]; Lee et al. [2019], recently added theoretical rigor to this particular practice by showing that such embeddings enable tuning the spectrum of the NTK of the corresponding MLP. In contrast, the goal of this chapter is to show that one does not have to be limited to the Fourier embedding for positional encoding. We demonstrate that alternative functions can be used for positional encoding while gaining similar or better performance compared to Fourier embedding.

2.3 Positional encoding: a theoretical walk-through

This section contains an exposition of the machinery and fundamentals necessary to understand the proposed framework. We begin our analysis by considering a simple linear learner since a rigorous characterization of a linear learner is convenient compared to a nonlinear model. Therefore, we study a linear learner and empirically show that the gathered insights are extendable to the non-linear models.

First, we show that the capacity to memorize a given set of training data entirely depends on the (approximate) rank of the embedding matrix. Next, we establish that for generalization, the rank should be upper-bounded against the number of coordinates, *i.e*., the embedding function should be bandlimited [1]. We incur a crucial insight here that positional encoding essentially portrays a trade-off between memorization and generalization. Afterward, we discuss the importance of distance preservation between embedded coordinates and their relationship to bandlimited embedding functions. Finally, we consider several possible embedder functions and analyze their behavior using the developed tools.

2.3.1 Rank of the embedded representation

Let $\mathbf{x}{=}[x_1, x_2, \cdots, x_N]^T$ be a vector of 1D coordinates, in which $x_i{\in}[0, C]$. And let $\mathbf{y}{=}[y_1, y_2, \cdots, y_n]^T$ be the corresponding outputs of a function $f{:}\mathbb{R}{\rightarrow}\mathbb{R}$. Our goal is to find a d-dimensional embedding $\Psi{:}\mathbb{R}{\rightarrow}\mathbb{R}^d$ for these positions, so that a linear model can be employed to learn the mapping f as,

$$\mathbf{w}^T \Psi(\mathbf{x}) + b \approx f(\cdot)\,, \tag{2.1}$$

where $\mathbf{w}{\in}\mathbb{R}^d$ and $b{\in}\mathbb{R}$ are the learnable weights and the bias, respectively. Then, it is straightforward to show that for the perfect reconstruction of *any possible* given $\mathbf{y}$ using Eq. (2.1), the following condition should be satisfied:

$$\text{Rank}\left\{[\Psi(x_1)\, \Psi(x_2)\, \ldots\, \Psi(x_N)]\right\} = N\,. \tag{2.2}$$

Thus, we establish the following Proposition:

[1] We assume that in regression, the smoothness of a model is implicitly related to generalization.

Proposition 1. *Consider a set of coordinates* $\mathbf{x}=[x_1, x_2, \cdots, x_N]^T$*, corresponding outputs* $\mathbf{y}=[y_1, y_2, \cdots, y_N]^T$*, and a* d *dimensional embedding* $\Psi:\mathbb{R}\rightarrow\mathbb{R}^d$*. Under perfect convergence, the sufficient condition for a linear model to perfectly memorize the mapping between* $\mathbf{x}$ *and* $\mathbf{y}$ *is for* $\mathbf{X}=[\Psi(x_1), \Psi(x_2), \ldots, \Psi(x_N)]^T$ *to have full rank.*

2.3.2 Bandlimited embedders

One possible way of enforcing the condition in Eq. (2.2) is to define an embedding scheme where the rank of the embedded matrix strictly monotonically increases with N (for a sufficiently large d). As depicted in Sec. 2.3.1, this would ensure that the model can memorize the training data and therefore perfectly reconstruct $\mathbf{y}$. However, memorization alone does not yield a good model. On the contrary, we also need our model to be generalizable to unseen coordinates.

To this end, let us define elements of $\Psi(\cdot)$ as sampled values from a function $\psi:\mathbb{R}^2\rightarrow\mathbb{R}$ such that for a given x,

$$\Psi(x) = [\psi(0, x), \psi(s, x), \ldots, \psi((d-1)s, x)]^T \,, \tag{2.3}$$

where $s=Cd^{-1}$ is the sampling interval. We shall refer to $\psi(\cdot)$ as the *embedder*. As discussed above, for better generalization, we need,

$$\psi(t, x) \approx \sum_{b=0}^{B} \alpha_b \beta_b(t) \,, \tag{2.4}$$

where α_b and $\beta_b(t)$ are weights and shifted basis functions, respectively, that can approximately estimate $\psi(t, x)$ at any arbitrary position x. We refer to such embedders as *bandlimited embedders* with a bandwidth B. This is equivalent to saying that the embedding matrix has a bounded rank, *i.e.*, the rank cannot increase arbitrarily with N. The intuition here is that if B is too small, the model will demonstrate poor memorization and overly smooth generalization. On the other hand, if B is extremely high, the model is capable of perfect memorization but poor generalization. Therefore, we conclude that for ideal performance, the embedder should be chosen carefully, such that it is both bandlimited and has a sufficient rank. As we shall discuss, the bandwidth B can also act as a guide for the minimal value of d.

2.3.3 Distance preservation

Intuitively, the embedded coordinates should preserve the distance between the original coordinates, irrespective of the absolute position. The embedded distance (or similarity) $\mathrm{D}(\cdot, \cdot)$ between two coordinates (x_1, x_2) can be measured via the inner product $\mathrm{D}(x_1, x_2)=\int_0^1 \psi(t, x_1)\psi(t, x_2)dt$. For ideal distance preservation, we need,

$$\|x_1 - x_2\| \propto \mathrm{D}(x_1, x_2) \,. \tag{2.5}$$

Interestingly, this property is also implicitly related to the limited bandwidth requirement. Note that in practice, we employ sampled embedders to construct Ψ as shown in Eq. (2.3). Hence, the dot product between the sampled $\psi(t, x_1)$ and $\psi(t, x_2)$ should be able to approximate D as,

$$\mathrm{D}(x_1, x_2) = \int_0^C \psi(t, x_1)\psi(t, x_2)dt \approx \sum_{d=0}^{d-1} \psi(s \cdot d, x_1)\psi(s \cdot d, x_2) \,, \tag{2.6}$$

which is possible, if and only if, ψ is bandlimited. In that case, $d{=}B$ is sufficient where B is the bandwidth of ψ (by Nyquist sampling theory). In practice, we choose $C{=}1$.

Remark 1. *The embedder should be bandlimited for better generalization (equivalently, the rank of the embedded matrix should be upper-bounded). Further, the ideal embedder should essentially face a trade-off between memorization and generalization. Here, memorization correlates with the rank of the embedded matrix, while generalization relates to the distance preservation between the embedded coordinates.*

2.3.4 Analysis of possible embedders

Although our derivations in Sec. 2.3 are generic, it is imperative to carefully choose a specific form of $\psi(\cdot,\cdot)$, such that properties of candidate embedders can be conveniently analyzed. Hence, we define embedders in terms of shifted basis functions, *i.e.*, $\psi(t,x){=}\phi(t{-}x)$. Such a definition permits us to examine embedders in a unified manner, as we shall see below.

Moreover, the rank of a matrix can be extremely noisy in practice. Typically, we need to heuristically set an appropriate threshold to the singular values, leading to unstable calculations. Therefore, we use the stable rank Rudelson and Vershynin [2007] instead of the rank in all our experiments. In particular, the stable rank is a more stable surrogate for the rank and is defined as $\frac{\|\mathbf{A}\|_F^2}{\|\mathbf{A}\|_2^2}$, where $\mathbf{A}$ is the matrix, $\|\cdot\|_F$ is the Frobenius norm and $\|\cdot\|_2$ is the matrix norm. From here onwards, we will use the terms rank, approximate rank, and stable rank interchangeably.

Impulse embedder. One simple way to satisfy the condition of Eq. (2.2) for an arbitrary large N is to define $\psi(t,x){=}\delta(t{-}x)$, where $\delta(\cdot)$ is the impulse function. Note that using an impulse embedder essentially converts the embedding matrix to a set of one-hot encodings. With the impulse embedder, we can perfectly memorize a given set of data points, since the embedded matrix has full rank. However, the obvious drawback is that the bandwidth of the impulse embedder is infinite *i.e.*, assuming a continuous domain d needs to reach infinity to learn outputs for all possible positions. Hence, the distance preservation is hampered and, consequently, the learned model lacks generalization.

Rectangle embedder. As an approximation of the impulse function (unit pulse), rectangular function $rect(x){=}1$ when $|x|{<}\frac{1}{2}$ and $rect(x){=}0$ when $|x|{>}\frac{1}{2}$. We can define $\psi(x){=}rect\left(\frac{x-t}{d}\right)$, where d is the width of the impulse. Immediately we know that the stable rank of the rectangular embedder is $\min\left(N,\frac{1}{d}\right)$, where N is the number of coordinates sampled, the distance function $D(x_1,x_2){=}tri(x_1{-}x_2)$, where $tri(\cdot)$ is the triangular function. From an intuitive perspective, the rectangle embedder can be viewed as performing nearest neighbor regression.

Triangle embedder. A better choice to approximate the impulse function may be the triangular function, which is defined as $tri(x){=}\max(1{-}|x|,0)$. Thus, the embedder is defined as $\psi(x){=}tri\left(\frac{x-t}{0.5d}\right)$. Here, the factor 0.5 determines the width of the triangle d. The stable rank of the triangular embedder is $\min\left(N,\frac{4/3}{d}\right)$. When d is the same, the triangle embedder has a higher stable rank than the rectangle embedder. The distance function of the triangular embedder is $D(x_1,x_2){=}\frac{1}{4}\max\left(d{-}|x_1{-}x_2|,0\right)^2$. This distance function is very close to a Gaussian function, as illustrated in Fig. 2.3. From an intuitive perspective, the triangular embedder can be viewed as performing linear interpolation between neighboring points.

Sine embedder. Consider $\psi(t,x)=\sin(f(t-x))$ for an arbitrary fixed function f. Since $\sin(f(t-x))=\sin(ft)\cos(fx)-\cos(ft)\sin(fx)$, elements of any row of the embedding matrix can be written as a linear combination of the corresponding $\sin(ft)$ and $\cos(ft)$. Thus, the rank of the embedding matrix is upper-bounded at 2. Consequently, the expressiveness of the encoding is limited, leading to poor memorization and overly smooth generalization (interpolation) at unseen coordinates.

Square embedder. Let us denote a square wave with unit amplitude and period 2π as $\text{sgn}(\sin(t))$, where sgn is the sign function. Then, define $\psi(t,x)=\text{sgn}(\sin(t-x))$. It is easy to deduce that the embedded distance $D(x_1,x_2)=1-2\|x_1-x_2\|, \forall |x|\leq 1$ which implies perfect distance preservation. The drawback, however, is that the square wave is not bandlimited. Thus, it cannot approximate the inner product $\int \psi(t,x)\psi(t,x')$ using a finite set of samples as in Eq. (2.6). However, an interesting attribute of the square wave is that it can be decomposed into a series of sine waves with odd-integer harmonic frequencies as $\text{sgn}(\sin(t))=\frac{4}{\pi}\left[\sin(t)+\frac{1}{3}\sin(3t)+\frac{1}{5}\sin(5t)+\frac{1}{7}\sin(7t)+\dots\right]$. In other words, its highest energy (from a signal processing perspective) is contained in a sinusoidal with the same frequency. Thus, the square wave can be *almost* approximated by a sinusoidal signal. In fact, the square wave and the sinusoidal demonstrate similar properties in terms of the stable rank and the distance preservation (see Fig. 2.3).

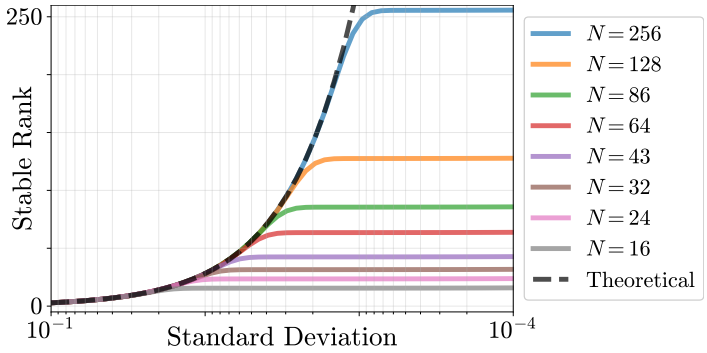
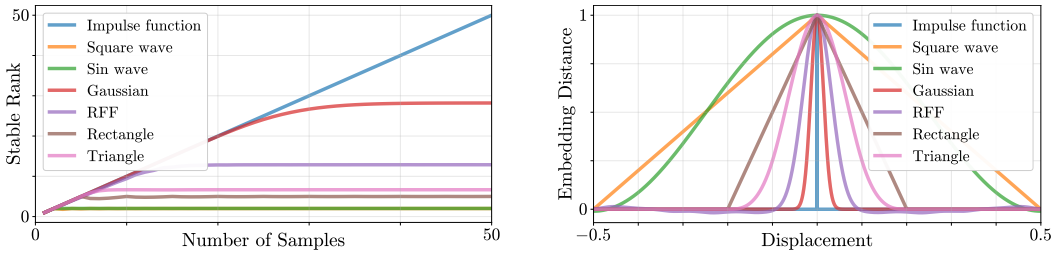


FIGURE 2.2. Stable rank of the Gaussian embedder vs the standard deviation for different numbers of samples. The dash line is the theoretical stable rank $\frac{1}{2\sqrt{\pi}\sigma}$

Gaussian embedder. We define the Gaussian embedder as $\psi(t,x)=\exp\left(-\frac{\|t-x\|^2}{2\sigma^2}\right)$ where σ is the standard deviation. The Gaussian embedder is also approximately bandlimited like the square embedder. However, the Gaussian embedder has a higher upper bound for the stable rank that can be controlled by σ. More precisely, when the embedding dimension is large enough, the stable rank of the Gaussian embedding matrix and the embedded distance between coordinates can be obtained analytically as shown below. (see Fig. 2.2 for an experimental illustration). It is clear from proposition 2 that as the number of sampled positions goes up, the stable rank of the Gaussian embedding matrix will linearly increase until it reaches its upper bound. Finally, Fig. 2.3 empirically validates the theoretically discussed properties of different embedders.

Proposition 2. *Let the Gaussian embedder be denoted as* $\psi(t,x)=\exp\left(-\frac{\|t-x\|^2}{2\sigma^2}\right)$. *With a sufficient embedding dimension, the stable rank of the embedding matrix obtained using the Gaussian embedder is* $\min\left(N,\frac{1}{2\sqrt{\pi}\sigma}\right)$ *where* N *is the number of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}(x_1,x_2)=\exp\left(-\frac{\|x_1-x_2\|^2}{4\sigma^2}\right)$.



(A) (B)

FIGURE 2.3. Quantitative comparison of (A) the stable rank and (B) the distance preservation of different embedders and RFFs. As expected, the stable rank of the impulse embedder strictly increases with the number of sampled points, causing poor distance preservation. The stable rank of the sine embedder is upper-bounded at 2. The stable ranks of the square embedder and the sine embedder almost overlap. However, if the sample numbers are extremely high (not shown in the figure), their stable ranks begin to deviate. Similarly, the square embedder demonstrates perfect distance preservation, and the sine embedder is a close competitor. In contrast, the Gaussian embedder and the RFF showcase mid-range upper bounds for the stable rank and adequate distance preservation, exhibiting a much better trade-off between memorization and generalization.

2.3.5 Connection to the Random Fourier Features

The prominent way of employing Fourier frequency mapping is via Random Fourier Features (RFF) mapping Tancik et al. [2020], where the frequencies are randomly sampled from a Gaussian distribution with a certain standard deviation σ. In this Section, we show that RFF mapping can be analyzed through the lens of our theoretical framework discussed thus far. To this end, we first establish the following proposition:

Proposition 3. *Let the RFF embedding be denoted as* $\gamma(x)=[\cos(2\pi\mathbf{b}x), \sin(2\pi\mathbf{b}x)]$, *where* $\mathbf{b}$ *are sampled from a Gaussian distribution. When the embedding dimension is large enough, the stable rank of RFF will be* $\min\left(N, \sqrt{2\pi}\sigma\right)$, *where* N *is the number of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}(x_1, x_2)=\sum_j \cos 2\pi b_j(x_1-x_2)$.

As shown in Fig. 2.4, the stable rank of RFF increases linearly with the number of samples until it gets saturated at $\sqrt{2\pi}\sigma$. This indicates a relationship between RFF and the Gaussian embedder. Let σ_g and σ_f be the standard deviations of the Gaussian embedder and RFF. When their stable ranks are equal, $\frac{1}{2\sqrt{\pi}\sigma_g}=\sqrt{2\pi}\sigma_f$ (from proposition 2, proposition 3). This implies that when $\sigma_g\sigma_f=\frac{1}{2\sqrt{2\pi}}$, these two embedders are equivalent in terms of the stable rank and distance preservation (see Fig. 2.4 when σ_g=0.01 and σ_f=0.1).

A common observation with RFFs is that when σ_f is too low, the reconstruction is overly smooth, and if σ_f is too high, it gives noisy interpolation Tancik et al. [2020]. This observation directly correlates to our theory. In Fig. 2.4, as the standard deviation increases, the stable rank increases and distance preservation decreases. Similarly, when the standard deviation is too low, the stable rank decreases while distance preservation increases.

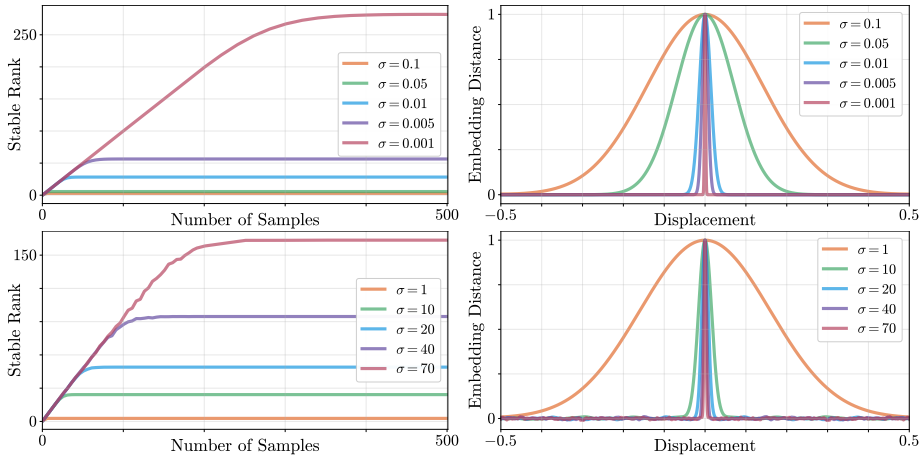


FIGURE 2.4. The stable rank (left column) and distance preservation (right column) of Gaussian embedder (top row) and RFF (bottom row) across different standard deviations.

2.4 Simplicity vs complexity in positional encoding

Thus far, we have established that the positional encoding can be achieved by sampling shifted basis functions, and the well-known RFF-embedding can also be analyzed through this lens. However, the analysis so far focused only on 1D coordinates. In this section, we shall investigate how to extend these positional embedding schemes to high-dimensional signals, *e.g.*, images and 3D signals.

2.4.1 2D simple encoding

Suppose $\psi(\cdot)$ is an arbitrary positional encoding function. We define simple positional encoding as the concatenation of the 1D encoding in each dimension: $\Psi(x, y) = [\psi(x), \psi(y)]$. Then, with a linear model, we have,

$$I(x, y) \approx \mathbf{w}^T \Psi(x, y) = \mathbf{w}_x^T \psi(x) + \mathbf{w}_y^T \psi(y) \,. \tag{2.7}$$

The above formula can be written in the matrix form as,

$$\begin{bmatrix} I(1,1) & \dots & I(N,1) \\ \vdots & \ddots & \vdots \\ I(1,N) & \dots & I(N,N) \end{bmatrix} \approx \underbrace{\begin{bmatrix} \mathbf{w}_x^T\psi(x_1) & \dots & \mathbf{w}_x^T\psi(x_N) \\ \vdots & \ddots & \vdots \\ \mathbf{w}_x^T\psi(x_1) & \dots & \mathbf{w}_x^T\psi(x_N) \end{bmatrix}}_{\mathbf{A}} + \underbrace{\begin{bmatrix} \mathbf{w}_y^T\psi(y_1) & \dots & \mathbf{w}_y^T\psi(y_1) \\ \vdots & \ddots & \vdots \\ \mathbf{w}_y^T\psi(y_N) & \dots & \mathbf{w}_y^T\psi(y_N) \end{bmatrix}}_{\mathbf{B}} . \tag{2.8}$$

Clearly, $\mathbf{A}$ and $\mathbf{B}$ are rank 1 matrices. Therefore, a linear network can only reconstruct a 2D image signal with at most rank 2. This drawback can be addressed in most practical cases using deeper non-linear MLPs, since the rank of the representations can be increased with multiple layers.

2.4.2 2D complex encoding

As opposed to simple encoding, we propose an alternative method for positional encoding in higher dimensions using the Kronecker product. With this approach, we can obtain a higher rank for the positional embedding matrix. For example, consider

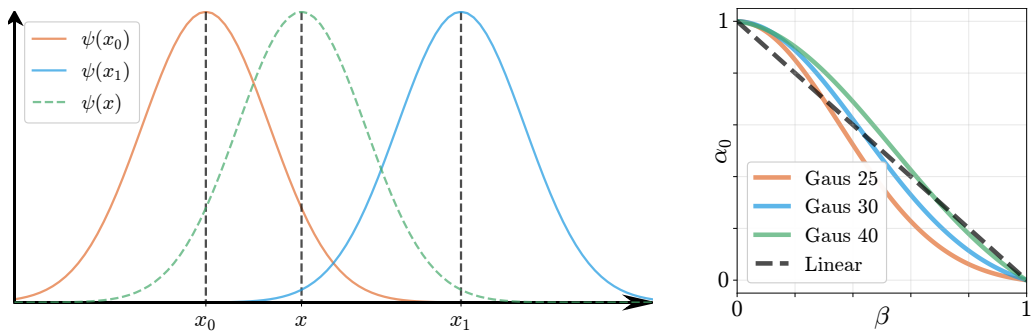


FIGURE 2.5. **Relationship between α_0 and β.** (a) is an example positional encoding at x_0, x_1 and x. The x-axis is along the feature dimension, and the y-axis is the encoding value. Here, we want to express $\psi(x)$ as a linear combination of $\psi(x_0)$ and $\psi(x_1)$. (b) is the relationship between the interpolation weight α_0 and the position ratio β. Instead of simple linear interpolation (dashed line), the actual relation depends on the encoding function.

2D inputs. Then, we can obtain the complex encoding as $\Psi(\mathbf{x},\mathbf{y})=\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y})$, where $\Psi(\mathbf{x})$ and $\Psi(\mathbf{y})$ are 1D encodings along each dimension. Also, the following relationship holds:

$$\text{Rank}(\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y})) = \text{Rank}(\Psi(\mathbf{x}))\text{Rank}(\Psi(\mathbf{y}))\,. \tag{2.9}$$

However, the drawback is also obvious. The embedding dimension is squared, which takes significantly more memory and computational cost. However, we propose an elegant workaround for this problem, given that the points are sampled on a regular grid, *i.e.*, when the coordinates are separable, using the following property of the Kronecker product,

$$\text{vec}(\text{S})^T \approx \text{vec}(\mathbf{W})^T(\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y})) = \text{vec}(\Psi(\mathbf{y})^T\mathbf{W}\Psi(\mathbf{x}))^T\,, \tag{2.10}$$

where $\text{S}_{i,j}=I(x_i, y_j)$. For instance, suppose we have N^2 number of 2D separable points where the feature length is K for each dimension. The naive Kronecker product leads to $O(N^2K^2)$ computational complexity and $O(N^2K^2)$ memory complexity. Using Eq. (2.10), we reduce it dramatically to $O(NK(N+K))$ computational complexity and $O(2NK+K^2)$ memory complexity. The key advantage of the complex encoding mechanism is that although it leads to a larger encoding matrix, the ability to achieve full rank allows us to use a single linear layer instead of an MLP, which reduces the computational cost of the neural network substantially. In addition, this enables us to obtain a closed-form solution instead of using stochastic gradient descent, leading to dramatically faster optimization. More precisely, we need to solve,

$$\arg\min_{\mathbf{W}} \|\text{vec}(\text{S}) - (\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y}))^T\text{vec}(\mathbf{W})\|_2^2\,, \tag{2.11}$$

where the solution can be obtained analytically as,

$$\mathbf{W} = (\Psi(\mathbf{y})\Psi(\mathbf{y})^T)^{-1}\Psi(\mathbf{y})\text{S}\Psi(\mathbf{x})^T(\Psi(\mathbf{x})\Psi(\mathbf{x})^T)^{-1}\,. \tag{2.12}$$

When the coordinates are not separable, we can still take advantage of Eq. (2.10) by adding a sparse blending matrix B as,

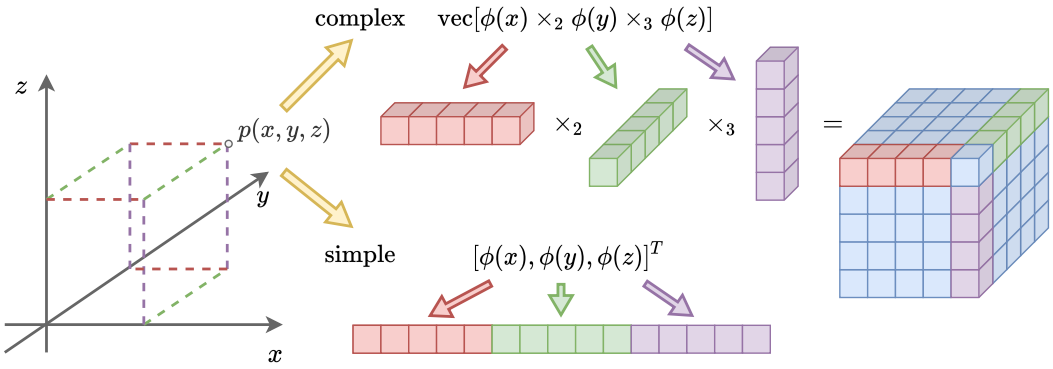


FIGURE 2.6. Illustration of different methods to extend 1D encoding of length K to N-dimensional case. Simple encoding is the widely used method currently, which is only the concatenation of embeddings in each dimension to get an NK size encoding. We propose an alternative embedding method: conduct mode-n product to get the N-dimensional cube, which we define as the complex embedding.

$$\text{vec}(I(\mathbf{x}, \mathbf{y}))^T \approx \text{vec}(\mathbf{W})^T(\Psi(\mathbf{x}) \otimes \Psi(\mathbf{y}))B = \text{vec}(\Psi(\mathbf{y})^T \mathbf{W} \Psi(\mathbf{x}))^T B \,. \tag{2.13}$$

This procedure is equivalent to evaluating virtual points (sampled on a regular grid) and interpolating to arbitrary coordinates using interpolation weights of B. The nature of the interpolation depends on the basis function used for positional encoding. Suppose x_0 and x_1 are two grid points and $x_0 \leq x \leq x_1$. We want $\psi(x) \approx \alpha_0 \psi(x_0) + \alpha_1 \psi(x_1)$. It can be solved by

$$\arg\min_{\alpha} \|\psi(x) - \begin{bmatrix}\psi(x_0) & \psi(x_1)\end{bmatrix} \alpha\|_2^2 \,, \tag{2.14}$$

where $\alpha = \begin{bmatrix}\alpha_0 & \alpha_1\end{bmatrix}^T$. With the definition $D(x)$ in Sec. 2.3.1, consider $d = x_1 - x_0$, which is the grid interval. Then, $x = x_0 + \beta d$, where $0 \leq \beta \leq 1$, and we have

$$\alpha = \frac{1}{D^2(0) - D^2(d)} \begin{bmatrix} D(0) & -D(d) \\ -D(d) & D(0) \end{bmatrix} \begin{bmatrix} D(\beta d) \\ D((1-\beta)d) \end{bmatrix} \,. \tag{2.15}$$

Please refer to Sec. 2.6.2 for more details.

2.4.3 High dimensional encoding

Both simple and complex encoding methods discussed in the previous sections can be extended easily to arbitrarily high dimensions. Suppose $\mathbf{x} = [x_1, x_2, \cdots, x_D]^T$ are the coordinates of a D dimensional point, $\psi : \mathbb{R} \to \mathbb{R}^K$ is the 1D encoder and $\Psi^{(D)}(\cdot)$ is the D dimensional encoding function. Then, the simple encoding is

$$\Psi^{(D)}(\mathbf{x}) = [\psi(x_1), \psi(x_2), \cdots, \psi(x_D)]^T \,. \tag{2.16}$$

Similarly, the complex encoding can be obtained via the Kronecker product between each encoding as,

$$\Psi^{(D)}(\mathbf{x}) = \text{vec}([\psi(x_1) \times_2 \psi(x_2) \times_3 \cdots \psi(x_3) \cdots \times_D \psi(x_D)]) \,. \tag{2.17}$$

Then, we can again extend the workaround we used in Eq. Eq. (2.10) to multiple dimensions as,

$$\text{vec}(\mathbf{W})^T \Psi^{(D)}(\mathbf{x}) = \mathbf{W} \times_1 \psi(x_1) \times_2 \psi(x_2) \cdots \times_D \psi(x_D) \,. \tag{2.18}$$

Fig. Fig. 2.6 graphically illustrates the simple and complex positional encoding schemes.

2.5 Experiments

In this Section, we empirically confirm the advantages of using the proposed embedding procedure and verify that the theoretically predicted properties in the previous sections hold in practice. To this end, five encoding methods are compared: linearly sampled frequency (LinF), log-linearly sampled frequency (LogF), RFF, shifted Gaussian (Gau), and shifted triangle encoder (Tri).

2.5.1 Method Notations

For 1D encoding experiments, we used Fourier-feature-based encodings with linearly, log-linearly, or randomly sampled frequencies, and shifted encodings whose bases are Gaussian or triangular function. We give a brief introduction to these methods below.
LinF (Fourier feature-based encoding using linearly sampled frequency).

$$\phi(x) = \left[\cdots, \cos\left(2\pi \cdot \left(\tfrac{K-i}{K} 2^0 + \tfrac{i}{K} 2^\sigma\right) x\right), \sin\left(2\pi \cdot \left(\tfrac{K-i}{K} 2^0 + \tfrac{i}{K} 2^\sigma\right) x\right), \cdots\right]^T \,, \tag{2.19}$$

where $i{=}0, \ldots, K{-}1$ and σ is the hyperparameter for the frequency range that sampled linearly from base frequency (2^0) to max frequency (2^σ).
LogF (Fourier feature-based encoding using log-linearly sampled frequency).

$$\phi(x) = \left[\cdots, \cos\left(2\pi \cdot 2^{\sigma i/K} x\right), \sin\left(2\pi \cdot 2^{\sigma i/K} x\right), \cdots\right]^T \,, \tag{2.20}$$

where $i{=}0, \ldots, K{-}1$ and σ is the hyperparameter for frequency range. The frequency are sampled log-linearly from base frequency (2^0) to max frequency (2^σ).
RFF (Fourier feature-based encoding using randomly sampled frequency) Tancik et al. [2020].

$$\phi(x) = \left[\cos\left(2\pi \mathbf{b} x\right)^T, \sin\left(2\pi \mathbf{b} x\right)^T\right]^T \,, \tag{2.21}$$

where $\mathbf{b} \in \mathbb{R}^{K \times 1}$ is random frequencies sampled from $\mathcal{N}(0, \sigma^2)$, where σ is the hyperparameter for frequency range.
Tri (shifted triangle encoding).

$$\phi(x) = \left[\cdots, \max\left(1 - \left|\tfrac{x - i/K}{d}\right|, 0\right), \cdots\right]^T \,, \tag{2.22}$$

where $i{=}0, \ldots, K{-}1$ and d is the hyperparameter for the width of triangle wave.
Gau (shifted Gaussian encoding).

$$\phi(x) = \left[\cdots, e^{-\frac{x - i/K}{2d^2}}, \cdots\right]^T \,, \tag{2.23}$$

where $i{=}0, \ldots, K{-}1$ and d is the hyperparameter for the width of Gaussian wave.

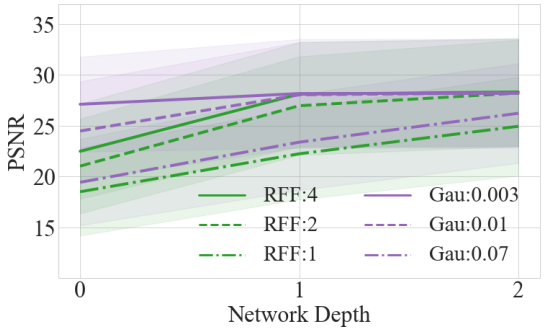


FIGURE 2.7. The performance of the reconstruction depends on the depth of the network and the rank of the input embedding.

2.5.2 1D: Rank of input & depth of Network

In this experiment, we randomly sample 16 columns and 16 rows from 512×512 natural images from the image dataset in Tancik et al. [2020]. And we used 256 equally spaced points for training and the rest 256 points for testing. The encoding length is set to be 256, the same as the number of training points. Parameters were chosen carefully for each encoder to show the relationship of the encoding rank and the depth of the network. The depth of the network changes from 0 (linear layer) to 2 with a fixed width of 256. From Fig. 2.4, we already know that the rank of the encoding matrix drops when σ of Gau increases or σ of RFF decreases. The result in Fig. 2.7 shows that when the rank is high, a linear network (0 depth) also works well. When the rank drops (*e.g.*, when the σ of Gau changes from 0.003 to 0.01 to 0.07), the performance of a single linear layer also drops. Adding one layer to the linear network makes up for the performance drop, while adding more layers does not help a lot. In conclusion, as the rank drops, a deeper network is needed for better performance.

2.5.3 2D: image reconstruction

For this experiment, we used the image dataset in Tancik et al. [2020], which contains 32 natural images of size 512×512. For simple encoding, we used a 4-layer MLP with hidden units of 256 widths, and for complex encoding, we only used a single linear layer. The results discussed below are the average metrics for 16 images where the networks are trained for 2000 epochs.

Separable coordinates. 256×256 grid points were evenly sampled for training, and the rest were used for testing. As shown in Tab. 2.1, complex encoding is around 20 times faster compared to simple encoding. In fact, although both the encodings were trained for 2000 epochs, we observed that complex encodings achieved good performance in a significantly lower number of epochs. Complex encoding can also be solved in closed form without training, which is orders of magnitude faster than simple encoding and maintains a good performance. Frequency encodings (LinF, LogF, RFF) did not perform well with complex encoding since they were rank-deficient. Although complex frequency encodings did not perform as well as complex shifted encoding when combined with a single linear layer, they still outperformed simple encodings followed by a single linear layer.

Non-separable coordinates. For non-separable coordinates, the training points were randomly sampled 25% of the 512×512 natural images. Complex encoding contains a blending matrix representing virtual separable coordinates, which can be precomputed. As illustrated in Tab. 2.2, the performance of complex encodings was comparable and converged faster than the simple encodings.

TABLE 2.1. Performance of natural 2D image reconstruction without random sampling (separable coordinates). ● are simple positional encodings. ● are complex positional encodings with gradient descent. ● are complex positional encodings with closed-form solution. We did not find a result for the complex encoding LogF, since it is ill-conditioned (rank-deficient).

	PSNR	No. of params (memory)	Time (s)
● LinF	26.12 ± 3.99	$329,475$ $(1.32M)$	22.59
● LogF	26.11 ± 4.04	$329,475$ $(1.32M)$	22.67
● RFF Tancik et al. [2020]	$\mathbf{26.58 \pm 4.18}$	$329,475$ $(1.32M)$	22.62
● Tri	25.96 ± 3.88	$329,475$ $(1.32M)$	22.98
● Gau	26.17 ± 4.17	$329,475$ $(1.32M)$	23.12
● LinF	19.73 ± 2.49	$196,608$ $(0.79M)$	1.93
● LogF	23.55 ± 2.99	$196,608$ $(0.79M)$	1.85
● RFF Tancik et al. [2020]	24.34 ± 3.24	$196,608$ $(0.79M)$	1.55
● Tri	26.65 ± 3.57	$196,608$ $(0.79M)$	1.48
● Gau	$\mathbf{26.69 \pm 3.74}$	$196,608$ $(0.79M)$	2.01
● Tri	26.36 ± 3.39	$196,608$ $(0.79M)$	0.03
● Gau	$\mathbf{26.69 \pm 3.76}$	$196,608$ $(0.79M)$	0.18

TABLE 2.2. Performance of 2D image reconstruction with randomly sampled inputs (non-separable coordinates). ● are simple positional encodings. ● are complex positional encodings with stochastic gradient descent using smart indexing.

	PSNR	No. of params (memory)	Time (s)
● LinF	24.58 ± 3.74	$329,475$ $(1.32M)$	21.97
● LogF	24.76 ± 3.82	$329,475$ $(1.32M)$	22.13
● RFF Tancik et al. [2020]	$\mathbf{24.90 \pm 3.78}$	$329,475$ $(1.32M)$	22.28
● Tri	24.24 ± 3.56	$329,475$ $(1.32M)$	22.50
● Gau	24.63 ± 3.70	$329,475$ $(1.32M)$	22.86
● LinF	19.64 ± 2.04	$196,608$ $(0.79M)$	11.51
● LogF	22.35 ± 2.82	$196,608$ $(0.79M)$	11.52
● RFF Tancik et al. [2020]	20.54 ± 2.61	$196,608$ $(0.79M)$	3.08
● Tri	23.21 ± 3.58	$196,608$ $(0.79M)$	1.90
● Gau	$\mathbf{23.97 \pm 3.44}$	$196,608$ $(0.79M)$	2.51

2.5.4 Visual results for 2D images

Here we show 2D image visual results for separable coordinates in Figs. 2.9 and 2.10, and non-separable coordinates in Figs. 2.11 and 2.12. For simple encoding, the five aforementioned encoders were tested with a 256-width MLP of 0 and 4 hidden ReLU layers (0 means only a linear layer). For complex encoding, the same five encoders were tested with 0 and 1 hidden ReLU MLPs.

As shown in column 1 of these figures, when we used simple encodings and the network only had a single linear layer (0 hidden layers), the reconstructed images are of low quality, showing low-resolution color grids (LinF, LogF), cross-strip colors (Tri, Gau), or random color blobs (RFF). The results clearly support our claim that a linear network can only reconstruct a 2D image signal with at most rank 2. When we introduced non-linear layers and increased the hidden layer depth (depth 4, column 2), the reconstruction quality improved, leading to a better PSNR.

On the contrary, even with a single linear layer (depth 0, column 3), our complex encoding methods can achieve comparable results with methods that used a simple encoding combined with deeper non-linear networks. Note that Fourier feature-based (frequency-based) complex encodings (LinF, LogF, RFF) performed worse than shifted-based complex encodings (Tri, Gau) when there was only one linear layer due

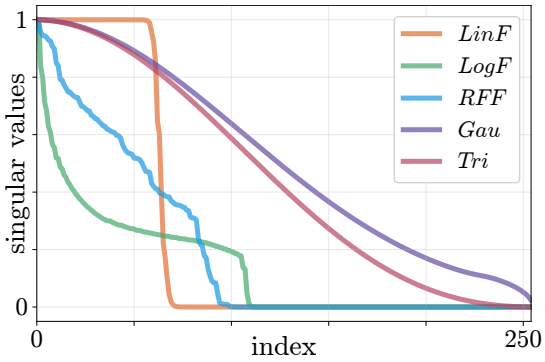


FIGURE 2.8. The normalized singular values of different 1D embeddings $\Psi(\mathbf{x}) \in \mathbb{R}^{N\times K}$. Here $N{=}K{=}256$ and $\mathbf{x}$ is sampled equally spaced from 0 to 1. Fourier feature-based encodings (LinF, LogF, RFF) tend to have much fewer non-zero singular values, which results in a low rank. While shifted encodings (Tri, Gau) usually have sufficient non-zero singular values, which leads to a high rank. When $\mathbf{x}$ is randomly sampled, the rank deficiency in Fourier feature-based encodings becomes worse.

to the deficiency of the embedding rank (shown in Fig. 2.8). Adding an extra non-linear layer (depth 1, column 4) did not substantially improve the performance of shifted-based complex encodings, while adding more details for frequency-based complex encodings.

2.5.5 3D: video reconstruction

As dimensionality of data increases, the faster convergence of the complex encoding becomes more notable. We use the YouTube video Real et al. [2017] for our experiments. In our experiments, we extracted 256 frames from 5 different videos and rescaled each frame to 256×256. Then, a central 128×128×128 cube was cropped to create our dataset, since some videos contain borders. For simple encoding, a 5-layer MLP with 512-width was used, while for complex encoding, only a single linear layer was used. We trained the networks for 500 epochs.

Separable coordinates. The training points were regularly sampled from a 64×64×64 grid, and the rest of the points were used for testing. The quantitative results are shown in Tab. 2.3. Qualitative results are illustrated in Fig. 2.9.

Non-separable coordinates. Training points were *randomly* sampled (12.5% from the total number of points) of a 64×64×64 grid, and the rest of the points were used for testing. The results are shown in Tab. 2.4. Similar to our observations in the main chapter, complex encodings combined with a single linear layer have comparable performance to simple encodings combined with deep (4-layer MLPs) networks while being 10x faster. Complex frequency-based encodings (LinF, LogF, RFF) have inferior results to complex shifted-based encodings (Tri, Gau) due to deficient rank.

2.6 Theoretical results

2.6.1 1D postional encoding

Proposition 4. *Consider a set of coordinates* $\mathbf{x}{=}[x_1, x_2, \cdots, x_N]^T$*, corresponding outputs* $\mathbf{y}{=}[y_1, y_2, \cdots, y_N]^T$*, and a d dimensional embedding* $\Psi{:}\mathbb{R}{\to}\mathbb{R}^d$*. Assuming perfect*

TABLE 2.3. Performance of video reconstruction without random sampling (separable coordinates). ● are simple positional encodings. ● are complex positional encodings with gradient descent; ● are complex positional encodings with closed-form solution.

	PSNR	No. of params (memory)	Time (s)
● LinF	21.49 ± 3.29	$1,445,891$ $(5.78M)$	77.06
● LogF	21.46 ± 3.09	$1,445,891$ $(5.78M)$	77.86
● RFF Tancik et al. [2020]	$\mathbf{21.06 \pm 3.26}$	$1,445,891$ $(5.78M)$	77.86
● Tri	20.86 ± 3.09	$1,445,891$ $(5.78M)$	78.91
● Gau	21.21 ± 3.26	$1,445,891$ $(5.78M)$	79.09
● LinF	9.76 ± 3.95	$786,432$ $(3.15M)$	0.69
● LogF	9.98 ± 3.84	$786,432$ $(3.15M)$	0.68
● RFF Tancik et al. [2020]	20.33 ± 3.06	$786,432$ $(3.15M)$	0.69
● Tri	22.24 ± 2.89	$786,432$ $(3.15M)$	0.68
● Gau	$\mathbf{22.11 \pm 3.14}$	$786,432$ $(3.15M)$	0.66
● RFF Tancik et al. [2020]	18.05 ± 2.36	$786,432$ $(3.15M)$	0.01
● Tri	22.04 ± 2.95	$786,432$ $(3.15M)$	0.009
● Gau	$\mathbf{21.83 \pm 3.33}$	$786,432$ $(3.15M)$	0.008

TABLE 2.4. Performance of video reconstruction with randomly sampled inputs (non-separable coordinates). ● are simple positional encodings. ● are complex positional encodings with stochastic gradient descent using smart indexing. Complex encodings with a single linear network are 10x faster than simple encodings with deep networks.

	PSNR	No. of params (memory)	Time (s)
● LinF	26.15 ± 3.33	$1,445,891$ $(5.78M)$	349.49
● LogF	25.63 ± 2.93	$1,445,891$ $(5.78M)$	350.62
● RFF Tancik et al. [2020]	25.46 ± 2.73	$1,445,891$ $(5.78M)$	350.92
● Tri	$\mathbf{26.44 \pm 4.24}$	$1,445,891$ $(5.78M)$	352.66
● Gau	24.58 ± 2.76	$1,445,891$ $(5.78M)$	351.45
● LinF	12.87 ± 2.38	$786,432$ $(3.15M)$	3.37
● LogF	16.76 ± 2.55	$786,432$ $(3.15M)$	3.41
● RFF Tancik et al. [2020]	21.40 ± 2.88	$786,432$ $(3.15M)$	3.62
● Tri	23.91 ± 2.81	$786,432$ $(3.15M)$	3.39
● Gau	$\mathbf{24.44 \pm 2.79}$	$786,432$ $(3.15M)$	3.46

convergence, the necessary and sufficient condition for a linear model to perfectly memorize the mapping between $\mathbf{x}$ *and* $\mathbf{y}$ *is for* $\mathbf{X}=[\Psi(x_1), \Psi(x_2), \ldots, \Psi(x_N)]$ *to have full rank.*

Proof: Let us refer to the row vectors of $\mathbf{X}$ as $[\mathbf{p}_1, \ldots, \mathbf{p}_d]^T$. In order to perfectly reconstruct $\mathbf{y}$ using a linear learner with weights $\mathbf{w}=[w_1, w_2, \ldots, w_d]$ as

$$\mathbf{y} = \sum_{i=1}^{d} w_i \mathbf{p}_i + b \,, \tag{2.24}$$

one needs $\mathbf{X}$ to be of rank N (since $\mathbf{y}$ needs to completely span $\{\mathbf{p}_i\}_{i=1}^{d}$). If $d > N$ then there is no unique solution to $\{\mathbf{w}, b\}$ without some regularization. In the unlikely scenario that the row vectors of $\mathbf{X}$ have zero mean, then $\mathbf{X}$ needs to be of rank $N-1$ since the bias term b can account for that missing linear basis. □

Proposition 5. *Let the Gaussian embedder be denoted as* $\psi(t, x) = \exp\left(-\frac{\|t-x\|^2}{2\sigma^2}\right)$. *With a sufficient embedding dimension, the stable rank of the embedding matrix obtained using the Gaussian embedder is* $\min\left(N, \frac{1}{2\sqrt{\pi}\sigma}\right)$ *where* N *is the number of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}(x_1, x_2) = \exp\left(-\frac{\|x_1-x_2\|^2}{4\sigma^2}\right)$.

Proof: Let us define the Gaussian embedder as $\psi(t,x)=\exp\left(-\frac{\|t-x\|^2}{2\sigma^2}\right)$, where σ is the standard deviation. Given d samples points $[t_1,\ldots,t_d]$ and N input coordinates $[x_1,\ldots,x_N]$, the elements of the embedding matrix are

$$\Psi_{i,j} = \psi(t_i, x_j)\,. \tag{2.25}$$

To make sure the stable rank is saturated, we assume that d and N is large enough. Then, Ψ is approximately a circulant matrix. We know that the singular value decomposition of a circulant matrix C, whose first row is c, can be written as

$$C = \frac{1}{n} F_n^{-1} diag\left(F_n c\right) F_n\,, \tag{2.26}$$

where F_n is the Fourier transform matrix. This means the singular values of a circulant matrix are the Fourier transform of the first row. When N is large enough, we can approximate the first row of Ψ as a continuous signal, which is $\psi(x,t{=}0){=}\exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$, so the singular values are

$$s(\xi) = \mathcal{F}\left(\psi(x;t=0)\right) = \sqrt{2\pi}\sigma \exp\left(-2\sigma^2\|\pi\xi\|^2\right)\,. \tag{2.27}$$

Therefore, we can calculate stable rank directly from the definition,

$$\text{Stable Rank}(\Psi){=}\sum_{i=1}^{N}\frac{s_i^2}{s_1^2}{=}\int_{-\infty}^{+\infty}\frac{s^2(\xi)}{s^2(0)}d\xi = \int_{-\infty}^{+\infty}\exp\left(-4\sigma^2\|\pi\xi\|^2\right)d\xi{=}\frac{1}{2\sqrt{\pi}\sigma}\,. \tag{2.28}$$

Considering the general case, where N might not be large enough, the stable rank will be $\min\left(N, \frac{1}{2\sqrt{\pi}\sigma}\right)$.

The distance (or similarity) between two embedded coordinates can be obtained via the inner product:

$$\begin{aligned}
\mathrm{D}(x_1,x_2) &= \int_{-\infty}^{+\infty}\psi(t,x_1)\psi(t,x_2)dt \\
&= \int_{-\infty}^{+\infty} e^{-\frac{(t-x_1)^2}{2\sigma^2}} e^{-\frac{(t-x_2)^2}{2\sigma^2}} dt \\
&= \int_{-\infty}^{+\infty} e^{-\frac{(t-x_1)^2+(t-x_2)^2}{2\sigma^2}} dt \\
&= \int_{-\infty}^{+\infty} e^{-\frac{t^2-2x_1t+x_1^2+t^2-2x_2t+x_2^2}{2\sigma^2}} dt \\
&= \int_{-\infty}^{+\infty} e^{-\frac{2t^2-2(x_1+x_2)t+\frac{(x_1+x_2)^2}{2}+\frac{(x_1-x_2)^2}{2}}{2\sigma^2}} dt \\
&= \int_{-\infty}^{+\infty} e^{-\frac{(t-\frac{x_1+x_2}{2})^2}{\sigma^2}} e^{-\frac{(x_1-x_2)^2}{4\sigma^2}} dt \\
&= e^{-\frac{(x_1-x_2)^2}{4\sigma^2}} \int_{-\infty}^{+\infty} e^{-\frac{(t-\frac{x_1+x_2}{2})^2}{\sigma^2}} dt \\
&= \sqrt{\pi}\sigma e^{-\frac{(x_1-x_2)^2}{4\sigma^2}}\,.
\end{aligned} \tag{2.29}$$

which is also a Gaussian with a standard deviation $\sqrt{2}\sigma$. We can empirically define that the distance between two embedded coordinates x_1 and x_2 is preserved if $D(x_1, x_2) \geq 10^{-k}$, for an interval $x_1 - x_2 \leq l$, where k is a threshold. In the Gaussian embedder, we can analytically obtain a σ for an arbitrary l using the relationship $\sigma = \frac{l}{2\sqrt{k \ln 10}}$. □

Proposition 6. *Let the RFF embedding be denoted as* $\gamma(x) = [\cos 2\pi \mathbf{b}x, \sin 2\pi \mathbf{b}x]$, *where* $\mathbf{b}$ *are sampled from a Gaussian distribution. When the embedding dimension is large enough, the stable rank of RFF will be* $\min\left(N, \sqrt{2\pi}\sigma\right)$, *where* N *is the number of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}(x_1, x_2) = \sum_j \cos 2\pi b_j (x_1 - x_2)$.

Proof: Given $\frac{d}{2}$ samples for $\mathbf{b}$ as $[b_1, \ldots, b_{\frac{d}{2}}]$ from a Gaussian distribution with a standard deviation σ and N input coordinates $[x_1, \ldots, x_N]$, RFF embedding is defined as $\gamma(x) = [\cos 2\pi \mathbf{b}x_i, \sin 2\pi \mathbf{b}x_i]$.

To make sure the stable rank is saturated, we assume that d and N are large enough. Although the RFF embedding matrix is not circulant, it is naturally frequency-based, so we already know its spectrum, which is its singular value distribution

$$s(\xi) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\xi^2}{2\sigma^2}\right) . \tag{2.30}$$

Similarly,

$$\text{Stable Rank}(\gamma) = \sum_{i=1}^{N} \frac{s_i^2}{s_1^2} = \int_{-\infty}^{+\infty} \frac{s^2(\xi)}{s^2(0)} d\xi = \int_{-\infty}^{+\infty} \exp\left(-\frac{\xi^2}{2\sigma^2}\right) d\xi = \sqrt{2\pi}\sigma , \tag{2.31}$$

Considering the general case, the stable rank is $\min\left(N, \sqrt{2\pi}\sigma\right)$.

From the basic trigonometry, it can be easily deduced that the distance function that $\mathrm{D}(x_1, x_2) = \sum_j \cos 2\pi b_j (x_1 - x_2)$. When d is extremely large it can be considered as $f(\xi) = \cos 2\pi\xi(x_1 - x_2)$ where ξ is a Gaussian random variable with standard deviation σ. Then the above sum can be replaced with the integral,

$$\begin{aligned}
\mathrm{D}(x_1, x_2) &= \int_{-\infty}^{+\infty} e^{-\frac{\xi^2}{2\sigma^2}} \cos 2\pi\xi(x_1 - x_2) d\xi \\
&= 2\int_{0}^{+\infty} e^{-\frac{\xi^2}{2\sigma^2}} \cos 2\pi\xi(x_1 - x_2) d\xi \\
&= 2\int_{0}^{+\infty} e^{-\frac{\xi^2}{2\sigma^2}} \frac{1}{2}(e^{i2\pi(x_1 - x_2)\xi} + e^{-i2\pi(x_1 - x_2)\xi}) d\xi \\
&= \int_{0}^{+\infty} e^{-\frac{\xi^2}{2\sigma^2} + i2\pi(x_1 - x_2)\xi} + e^{-\frac{\xi^2}{2\sigma^2} - i2\pi(x_1 - x_2)\xi} d\xi .
\end{aligned} \tag{2.32}$$

Further,

$$\int_0^{+\infty} e^{-ax^2 + bx} dx = e^{-\frac{b^2}{4a}} \int_0^{+\infty} e^{-a(x - i\frac{b}{2a})^2} dx = \frac{1}{2}\left(1 + \text{erfi}\left(\frac{b}{2\sqrt{a}}\right)\right)\sqrt{\frac{\pi}{a}} e^{-\frac{b^2}{4a}} . \tag{2.33}$$

Let $a = \frac{1}{2\sigma^2}$ and $b = \pm 2\pi(x_1 - x_2)$. Then, we have

$$\mathrm{D}(x_1, x_2) = \sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x_1 - x_2)^2} . \tag{2.34}$$

□

Proposition 7. *Let the Rectangular embedder be denoted as* $\psi(t,x)$=*rect* $\left(\frac{x-t}{\sigma}\right)$ =$(1-\frac{|x-t|}{0.5\sigma})$>0. *With a sufficient embedding dimension, the stable rank of the embedding matrix obtained using the Rectangular embedder is* $\min\left(N,\frac{1}{\sigma}\right)$ *where* N *is the number of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}\left(x_1,x_2\right)$ =$\sigma tri\left(\frac{|x_1-x_2|}{\sigma}\right)$ =$\sigma\max(1-\frac{|x_1-x_2|}{\sigma},0)$.

Proof: Let us define the Rectabgular embedder as $\psi(t,x)$=rect $\left(\frac{x-t}{\sigma}\right)=\left(1-\frac{|x-t|}{0.5\sigma}\right)>0$, where σ is the width of the rectangle impulse. Given d samples points $[t_1,\ldots,t_d]$ and N input coordinates $[x_1,\ldots,x_N]$, the elements of the embedding matrix are

$$\Psi_{i,j}=\psi(t_i,x_j)\,. \tag{2.35}$$

To make sure the stable rank is saturated, we assume that d and N are large enough. Then, Ψ is approximately a circulant matrix. We know that the singular value decomposition of a circulant matrix C, whose first row is c, can be written as

$$C=\frac{1}{n}F_n^{-1}diag\left(F_nc\right)F_n\,, \tag{2.36}$$

where F_n is the Fourier transform matrix. This means the singular values of a circulant matrix are the Fourier transform of the first row. When N is large enough, we can approximate the first row of Ψ as a continuous signal, which is $\psi(x,t{=}0)$=rect$(\frac{x}{\sigma})$, so the singular values are

$$s(\xi)=\mathcal{F}\left(\psi(x;t=0)\right)=\sigma\mathrm{sinc}(\sigma\xi)\,, \tag{2.37}$$

where $\mathrm{sinc}(\xi)$=$\frac{\sin(\pi x)}{\pi x}$. Therefore, we can compute the stable rank directly from the definition,

$$\text{Stable Rank}(\Psi)=\sum_{i=1}^{N}\frac{s_i^2}{s_1^2}=\int_{-\infty}^{+\infty}\frac{s(\xi)^2}{s(0)^2}d\xi=\int_{-\infty}^{+\infty}\mathrm{sinc}^2(\sigma\xi)d\xi=\frac{1}{\sigma}\,. \tag{2.38}$$

Considering the general case, where N might not be large enough, the stable rank will be $\min\left(N,\frac{1}{\sigma}\right)$.

The distance (or similarity) between two embedded coordinates can be obtained via the inner product:

$$\begin{aligned}\mathrm{D}(x_1,x_2)&=\int_{-\infty}^{+\infty}\psi(t,x_1)\psi(t,x_2)dt\\&=\int_{-\infty}^{+\infty}\mathrm{rect}\left(\frac{x_1-t}{\sigma}\right)\mathrm{rect}\left(\frac{x_2-t}{\sigma}\right)dt\\&=\sigma\mathrm{tri}\left(\frac{x_1-x_2}{\sigma}\right)\,.\end{aligned} \tag{2.39}$$

□

Proposition 8. *Let the Triangular embedder be* $\psi(t,x)$=$tri\left(\frac{x-t}{0.5\sigma}\right)=\max(1-\frac{|x-t|}{0.5\sigma},0)$. *With a sufficient embedding dimension, the stable rank of the embedding matrix obtained using the Triangular embedder is* $\min(N,\frac{4}{3\sigma})$ *where* N *is the number of embedded coordinates. Under the same conditions, the embedded distance between two coordinates* x_1 *and* x_2 *is* $\mathrm{D}(x_1,x_2)=\frac{1}{4}\sigma^2tri^2(\frac{|x_1-x_2|}{\sigma})=\frac{1}{4}\sigma^2\max(1-\frac{|x_1-x_2|}{\sigma},0)^2$.

Proof: Let us define the Triangle embedder as $\psi(t,x)$=tri $\left(\frac{x-t}{0.5\sigma}\right) = \max\left(1-\frac{|x-t|}{0.5\sigma}, 0\right)$, where σ is the width of the Triangular impulse. Given d samples points $[t_1,\ldots,t_d]$ and N input coordinates $[x_1,\ldots,x_N]$, the elements of the embedding matrix are

$$\Psi_{i,j} = \psi(t_i, x_j)\,. \tag{2.40}$$

To make sure the stable rank is saturated, we assume that d and N are large enough. Then, Ψ is approximately a circulant matrix. We know that the singular value decomposition of a circulant matrix C, whose first row is c, can be written as

$$C = \frac{1}{n} F_n^{-1} diag\left(F_n c\right) F_n\,, \tag{2.41}$$

where F_n is the Fourier transform matrix. This means the singular values of a circulant matrix are the Fourier transform of the first row. When N is large enough, we can approximate the first row of Ψ as a continuous signal, which is $\psi(x,t{=}0)$=tri $\left(\frac{x}{\sigma}\right)$, so the singular values are

$$s(\xi) = \mathcal{F}\left(\psi(x; t=0)\right) = \frac{\sigma}{2}\text{sinc}^2\left(\frac{\sigma}{2}\xi\right)\,, \tag{2.42}$$

where sinc(ξ)=$\frac{\sin(\pi x)}{\pi x}$. Therefore, we can compute stable rank directly from the definition as,

$$\text{Stable Rank}(\Psi) = \sum_{i=1}^{N}\frac{s_i^2}{s_1^2} = \int_{-\infty}^{+\infty}\frac{s(\xi)}{s(0)}^2 d\xi = \int_{-\infty}^{+\infty}\text{sinc}^4\left(\frac{\sigma}{2}\xi\right)d\xi = \frac{4}{3\sigma}\,. \tag{2.43}$$

Considering the general case, where N might not be large enough, the stable rank will be min $\left(N, \frac{1}{\sigma}\right)$.

The distance (or similarity) between two embedded coordinates can be obtained via the inner product:

$$\begin{aligned}\text{D}(x_1,x_2) &= \int_{-\infty}^{+\infty}\psi(t,x_1)\psi(t,x_2)dt \\ &= \int_{-\infty}^{+\infty}\text{tri}\left(\frac{x-t}{0.5\sigma}\right)\text{tri}\left(\frac{x-t}{0.5\sigma}\right)dt \\ &= \frac{1}{4}\sigma^2\max\left(1-\frac{|x_1-x_2|}{\sigma}, 0\right)^2\,.\end{aligned} \tag{2.44}$$

□

2.6.2 2D complex encoding

2.6.2.1 Closed form solution for separable coordinates

If pixels are sampled on a regular grid formed by samples $\mathbf{x}$=$[x_1, x_2, \cdots, x_N]^T$ and samples $\mathbf{y}$=$[y_1, y_2, \cdots, y_M]^T$, then the coordinates of these pixels are separable. Let $\mathbf{S}\in\mathbb{R}^{M\times N}$ be the signal defined as $\mathbf{S}_{i,j}$=$I(x_i, y_j)$, where i=$1,2,\cdots,N$, j=$1,2,\cdots,M$, and Ψ:$\mathbb{R}\to\mathbb{R}^K$ be the 1D encoder. We want to find the weights $\mathbf{W}\in\mathbb{R}^{K\times K}$ of the linear layer by optimizing the following equation,

$$\arg\min_{\mathbf{W}} \|\text{vec}(\mathbf{S}) - \left(\Psi(\mathbf{y})\otimes\Psi(\mathbf{x})\right)\text{vec}\left(\mathbf{W}\right)\|_2^2\,, \tag{2.45}$$

where $\Psi(\mathbf{x}) \in \mathbb{R}^{N \times K}$ is the encoding for $\mathbf{x}$, $\Psi(\mathbf{y}) \in \mathbb{R}^{M \times K}$ is the encoding for $\mathbf{y}$. This is a linear least squares problem. Based on the properties of the Kronecker product, we find the optimal solution $\mathbf{W}^*$ as,

$$\begin{aligned}
\operatorname{vec}(\mathbf{W}^*) &= \arg\min_{\mathbf{W}} \|\operatorname{vec}(\mathbf{S}) - (\Psi(\mathbf{y}) \otimes \Psi(\mathbf{x}))\operatorname{vec}(\mathbf{W})\|_2^2 \\
&= \left((\Psi(\mathbf{y}) \otimes \Psi(\mathbf{x}))^T (\Psi(\mathbf{y}) \otimes \Psi(\mathbf{x}))\right)^{-1} (\Psi(\mathbf{y}) \otimes \Psi(\mathbf{x}))^T \operatorname{vec}(\mathbf{S}) \\
&= \left(\left(\left(\Psi(\mathbf{y})^T\Psi(\mathbf{y})\right)^{-1}\Psi(\mathbf{y})\right) \otimes \left(\left(\Psi(\mathbf{x})^T\Psi(\mathbf{x})\right)^{-1}\Psi(\mathbf{x})\right)\right) \operatorname{vec}(\mathbf{S}) \\
&= \operatorname{vec}\left(\left(\left(\Psi(\mathbf{x})^T\Psi(\mathbf{x})\right)^{-1}\Psi(\mathbf{x})\right)\mathbf{S}\left(\left(\Psi(\mathbf{y})^T\Psi(\mathbf{y})\right)^{-1}\Psi(\mathbf{y})\right)^T\right) \\
&= \operatorname{vec}\left(\left(\Psi(\mathbf{x})^T\Psi(\mathbf{x})\right)^{-1}\Psi(\mathbf{x})\mathbf{S}\Psi(\mathbf{y})^T\left(\Psi(\mathbf{y})^T\Psi(\mathbf{y})\right)^{-1}\right),
\end{aligned} \tag{2.46}$$

which means,

$$\mathbf{W}^* = \left(\Psi(\mathbf{x})^T\Psi(\mathbf{x})\right)^{-1}\Psi(\mathbf{x})\mathbf{S}\Psi(\mathbf{y})^T\left(\Psi(\mathbf{y})^T\Psi(\mathbf{y})\right)^{-1}. \tag{2.47}$$

2.6.2.2 Blending matrix for non-separable coordinates

First, we focus on 1D encoders. Given a 1D encoder $\Psi{:}\mathbb{R}\to\mathbb{R}^K$ and two points x_0, x_1, we want to express $\Psi(x)\approx\alpha_0\Psi(x_0)+\alpha_1\Psi(x_1)$ for $x_0\le x\le x_1$. This problem can be solved by

$$\arg\min_{\alpha} \left\|\Psi(x) - \begin{bmatrix}\Psi(x_0) & \Psi(x_1)\end{bmatrix}\alpha\right\|_2^2, \tag{2.48}$$

where $\alpha=\begin{bmatrix}\alpha_0 & \alpha_1\end{bmatrix}^T$. Note here that $\Psi(x)$, $\Psi(x_0)$, and $\Psi(x_1)$ are $K\times 1$ vectors. This is equivalent to a least-squares problem; thus, the optimal solution α^* can be solved by,

$$\begin{aligned}
\alpha^* &= \arg\min_{\alpha} \left\|\Psi(x) - \begin{bmatrix}\Psi(x_0) & \Psi(x_1)\end{bmatrix}\alpha\right\|_2^2 \\
&= \left(\begin{bmatrix}\Psi(x_0) & \Psi(x_1)\end{bmatrix}^T\begin{bmatrix}\Psi(x_0) & \Psi(x_1)\end{bmatrix}\right)^{-1}\begin{bmatrix}\Psi(x_0) & \Psi(x_1)\end{bmatrix}^T\Psi(x) \\
&= \left(\begin{bmatrix}\Psi(x_0)^T \\ \Psi(x_1)^T\end{bmatrix}\begin{bmatrix}\Psi(x_0) & \Psi(x_1)\end{bmatrix}\right)^{-1}\begin{bmatrix}\Psi(x_0)^T \\ \Psi(x_1)^T\end{bmatrix}\Psi(x) \\
&= \begin{bmatrix}\Psi(x_0)^T\Psi(x_0) & \Psi(x_0)^T\Psi(x_1) \\ \Psi(x_1)^T\Psi(x_0) & \Psi(x_1)^T\Psi(x_1)\end{bmatrix}^{-1}\begin{bmatrix}\Psi(x_0)^T\Psi(x) \\ \Psi(x_1)^T\Psi(x)\end{bmatrix}.
\end{aligned} \tag{2.49}$$

With the definition $D(x_1, x_2)$ in Sec. 2.6, this can be written as,

$$\alpha^* = \begin{bmatrix}D(x_0,x_0) & D(x_0,x_1) \\ D(x_1,x_0) & D(x_1,x_1)\end{bmatrix}^{-1}\begin{bmatrix}D(x_0,x) \\ D(x_1,x)\end{bmatrix}. \tag{2.50}$$

Typically, this distance function only depends on the difference of the inputs, as examples shown in Sec. 2.6. Therefore, we can have a closed-form solution for $D{:}\mathbb{R}\to\mathbb{R}$.

Let $d{=}x_1{-}x_0$, and $x{=}x_0{+}\beta d$, where $0{\leq}\beta{\leq}1$. Then, the solution becomes,

$$\begin{aligned}\alpha^* &= \begin{bmatrix} D(x_0,x_0) & D(x_0,x_1) \\ D(x_1,x_0) & D(x_1,x_1) \end{bmatrix}^{-1} \begin{bmatrix} D(x_0,x) \\ D(x_1,x) \end{bmatrix} \\ &= \begin{bmatrix} D(0) & D(d) \\ D(d) & D(0) \end{bmatrix}^{-1} \begin{bmatrix} D(\beta d) \\ D\left((1-\beta)\,d\right) \end{bmatrix} \\ &= \frac{1}{D^2(0)-D^2(d)} \begin{bmatrix} D(0) & -D(d) \\ -D(d) & D(0) \end{bmatrix} \begin{bmatrix} D(\beta d) \\ D\left((1-\beta)\,d\right) \end{bmatrix} .\end{aligned} \tag{2.51}$$

Based on the 1D analysis, encoding 2D non-separable points can also be expressed as a non-linear interpolation of 2D separable coordinates. Suppose that the settings are the same as in Sec. 2.6.2.1. The virtual pixels are sampled on a regular grid formed by samples $\mathbf{x}{=}[x_1, x_2, \cdots, x_N]^T$ and samples $\mathbf{y}{=}[y_1, y_2, \cdots, y_M]^T$. The query points are randomly sampled in the space as $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_P]^T$, where P is the number of points and each $\mathbf{q}_i{\in}\mathbb{R}^{2\times1}$ is a random 2D coordinate. Let $\mathbf{s}{\in}\mathbb{R}^{P\times1}$ be the signal, and $\Psi{:}\mathbb{R}{\rightarrow}\mathbb{R}^K$ be the 1D encoder. We want to find the weights $\mathbf{W}{\in}\mathbb{R}^{K\times K}$ of the linear layer by optimizing the following equation,

$$\arg\min_{\mathbf{W}} \|\mathbf{s}{-}B(\mathbf{Q})\left(\Psi(\mathbf{y}) \otimes \Psi(\mathbf{x})\right)\mathrm{vec}\left(\mathbf{W}\right)\|_2^2 , \tag{2.52}$$

where $B{:}\mathbb{R}^2{\rightarrow}\mathbb{R}^{MN}$ is the non-linear interpolation coefficients function, *i.e.*, $B(\mathbf{Q}) \in \mathbb{R}^{P\times MN}$ is the blending matrix. Note that although B is large, it is extremely sparse and only has 4 non-zero values on each row of MN elements. Consider a certain point $\mathbf{q}_p$ is in the grid whose corner points are (x_i, y_j), (x_{i+1}, y_j), (x_i, y_{j+1}), and (x_{i+1}, y_{j+1}), which means $\mathbf{x}_i{\leq}\mathbf{q}_{p0}{\leq}\mathbf{x}_{i+1}$ and $\mathbf{y}_j{\leq}\mathbf{q}_{p1}{\leq}\mathbf{y}_{j+1}$. Then we can obtain the encoding for $\mathbf{q}_{p0}$ and $\mathbf{q}_{p1}$ as follows,

$$\begin{aligned}\Psi(\mathbf{q}_{p0}) &\approx \alpha_0\Psi(x_i) + \alpha_1\Psi(x_{i+1}), \\ \Psi(\mathbf{q}_{p1}) &\approx \beta_0\Psi(y_j) + \beta_1\Psi(y_{j+1}) .\end{aligned} \tag{2.53}$$

Then, the 2D encoding for $\mathbf{q}_p$ is,

$$\begin{aligned}\Psi(\mathbf{q}) =& \Psi(\mathbf{q}_{p0}, \mathbf{q}_{p1}) \\ =& \Psi(\mathbf{q}_{p1}) \otimes \Psi(\mathbf{q}_{p0}) \\ \approx& \left(\beta_0\Psi\left(y_j\right) + \beta_1\Psi\left(y_{j+1}\right)\right) \otimes \left(\alpha_0\Psi\left(x_i\right) + \alpha_1\Psi\left(x_{i+1}\right)\right) \\ =& \alpha_0\beta_0\Psi\left(y_j\right) \otimes \Psi\left(x_i\right) + \alpha_0\beta_1\Psi\left(y_{j+1}\right) \otimes \Psi\left(x_i\right) \\ &+ \alpha_1\beta_0\Psi\left(y_j\right) \otimes \Psi\left(x_{i+1}\right) + \alpha_1\beta_1\Psi\left(y_{i+1}\right) \otimes \Psi\left(x_{i+1}\right) \\ =& \alpha_0\beta_0\Psi\left(x_i, y_j\right) + \alpha_0\beta_1\Psi\left(x_i, y_{j+1}\right) \\ &+ \alpha_1\beta_0\Psi\left(x_{i+1}, y_j\right) + \alpha_1\beta_1\Psi\left(x_{i+1}, y_{i+1}\right) ,\end{aligned} \tag{2.54}$$

which means $B(\mathbf{q}_p){\in}\mathbb{R}^{1\times MN}$ are all zeros except $\alpha_0\beta_0$ at index $jN{+}i$, $\alpha_0\beta_1$ at index $(j{+}1)N{+}i$, $\alpha_0\beta_0$ at index $jN{+}i{+}1$ and $\alpha_0\beta_0$ at index $(j{+}1)N{+}i{+}1$.

2.6.2.3 HD complexity

Let $\mathbf{X}{\in}\mathbb{R}^{N^D\times D}$ be N^D points in D dimensional space, $\Psi{:}\mathbb{R}{\rightarrow}\mathbb{R}^K$ be the 1D encoder, and we want to know the memory and computational complexity when the encoding is multiplied a linear layer $\mathbf{W}$.

Simple encoding. The embedding $\Psi(\mathbf{X}){\in}\mathbb{R}^{N^D\times DK}$ and the weights $\mathbf{W}{\in}\mathbb{R}^{DK\times1}$, so the memory complexity is $O(DKN^D)$ and the computational complexity is $O(DKN^D)$.

Complex encoding (naive implementation). The embedding $\Psi(\mathbf{X}) \in \mathbb{R}^{N^D \times K^D}$ and the weights $\mathbf{W} \in \mathbb{R}^{K^D \times 1}$, so the memory complexity is $O(K^D N^D)$ and the computational complexity is $O(K^D N^D)$.
Complex encoding (separable coordinates). The embedding $\Psi(\mathbf{X}) \in \mathbb{R}^{N \times K}$ and the weights $\mathbf{W} \in \mathbb{R}^{K^D}$, so the memory complexity is $O(K^D + NK)$ and the computational complexity is $\sum_{i=1}^{D} N^i K^{D+1-i} = O(NK \frac{N^D - K^D}{N-K})$. A special case of $N=K$ will be discussed later.
Complex encoding (non-separable coordinates). The embedding $\Psi(\mathbf{X}) \in \mathbb{R}^{N \times K}$, the weights $\mathbf{W} \in \mathbb{R}^{K^D}$ and the Blending matrix $B(\mathbf{X}) \in \mathbb{R}^{N^D \times N^D}$ (sparse matrix with only $N^D \times 2^D$ non-zeros values), so the memory complexity is $O(K^D + NK + 2^D N^D)$, the computational complexity is $2^D N^D + \sum_{i=1}^{D} N^i K^{D+1-i} = O(2^D N^D + NK \frac{N^D - K^D}{N-K})$.
Special case $N=K$. Both simple encoding and separable complex encoding have $O(DN^{D+1})$ computational complexity. Memory complexity is $O(DN^{D+1})$ for simple encoding while it is $O(N^D + 2N)$ for separable encoding. However, the rank of the latter one is the power of D to the first one.

2.7 Conclusion

In this chapter, we show that the performance of a positional encoding scheme is mainly governed by the stable rank of the embedding matrix and the distance preservation between the embedded coordinates. In light of this discovery, we propose a novel framework that can incorporate arbitrary continuous signals as potential embedders, under certain constraints. We also propose a positional encoding scheme that enables dramatically faster convergence, allowing a single linear network to encode signals with high fidelity.

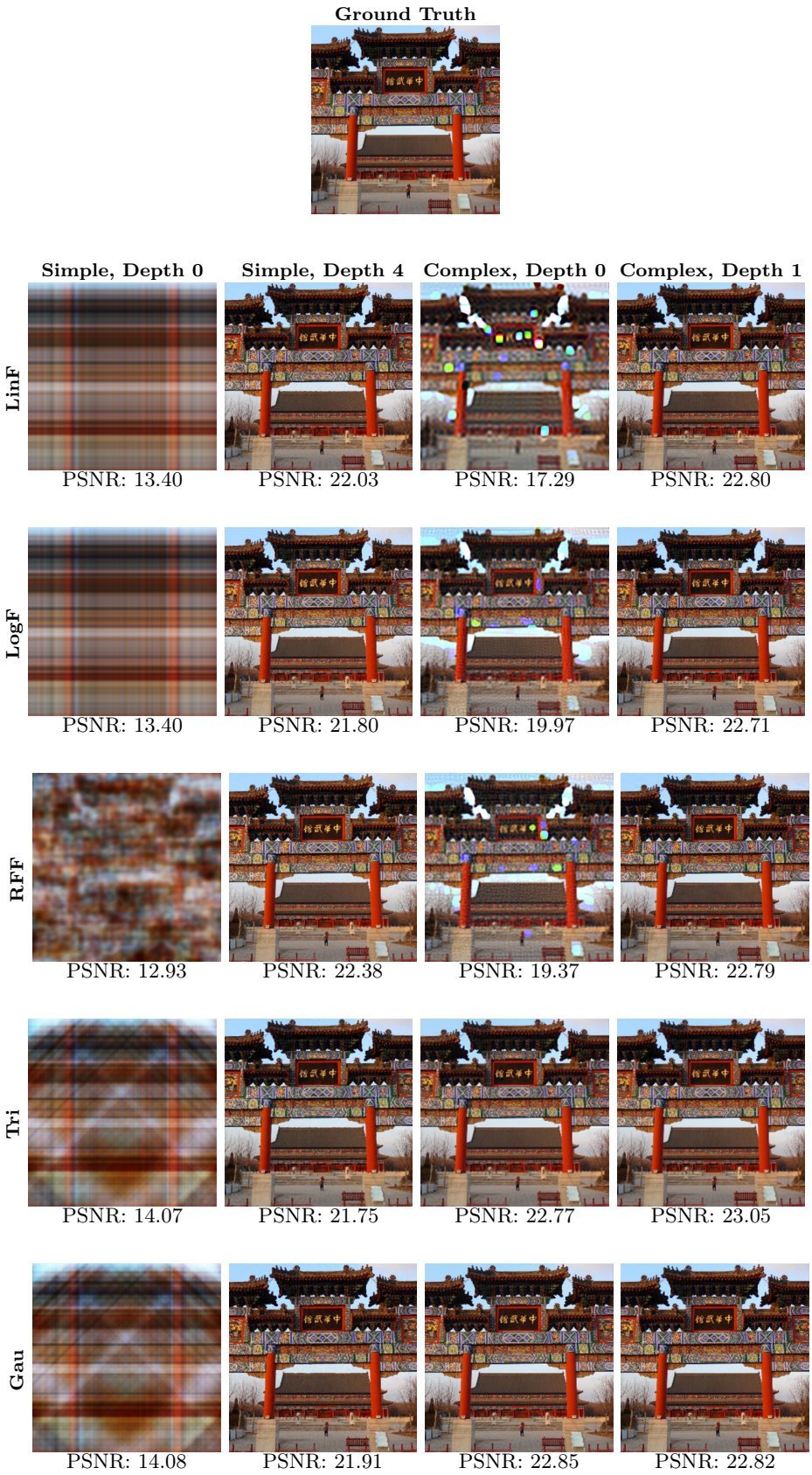


FIGURE 2.9. Reconstruction results of an archway using separable coordinates (regular-grid sampled training points) with different combinations of simple or complex encodings and network depths.

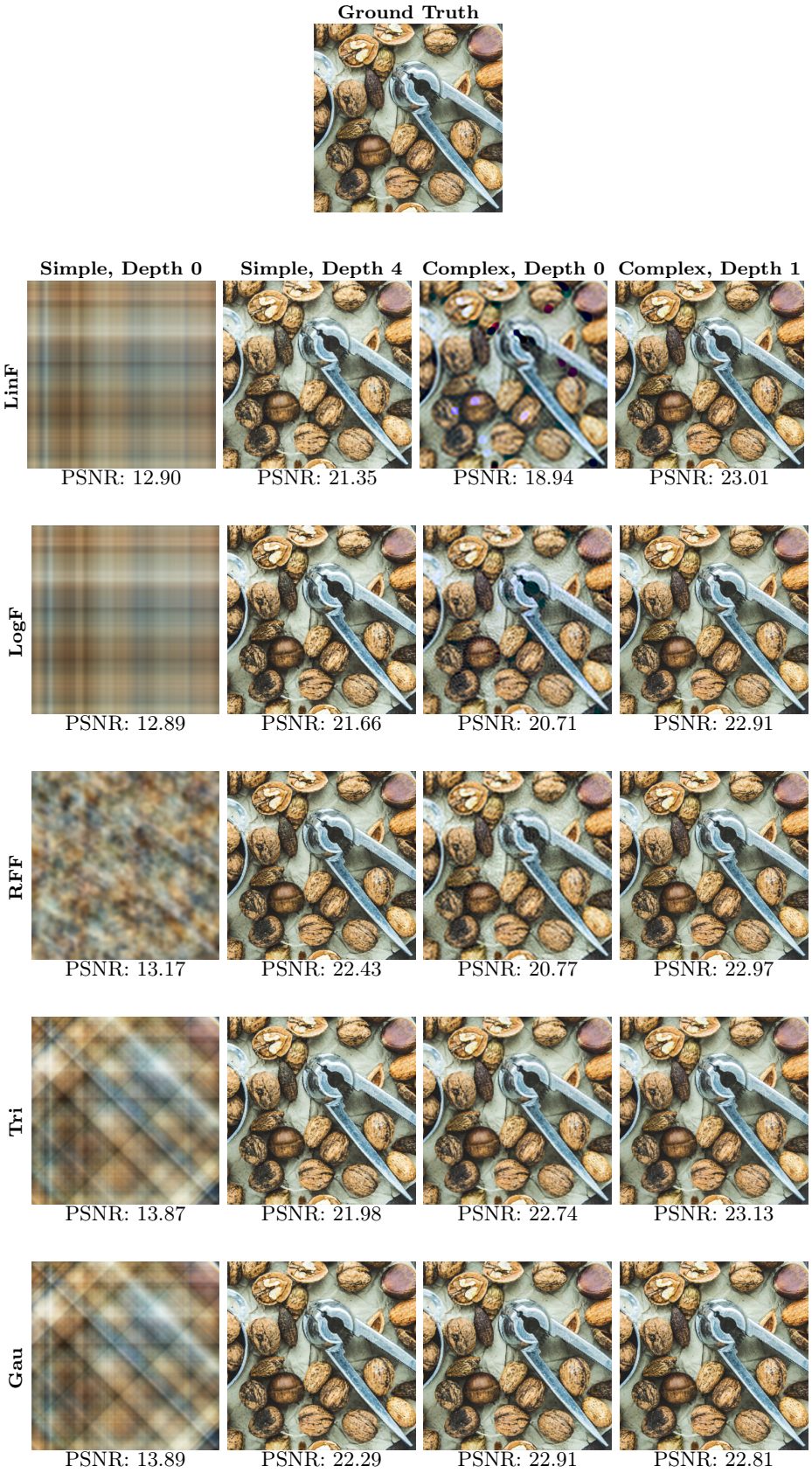


FIGURE 2.10. Reconstruction results of a heap of walnuts using separable coordinates (regular-grid sampled training points) with different combinations of simple or complex encodings and network depths.

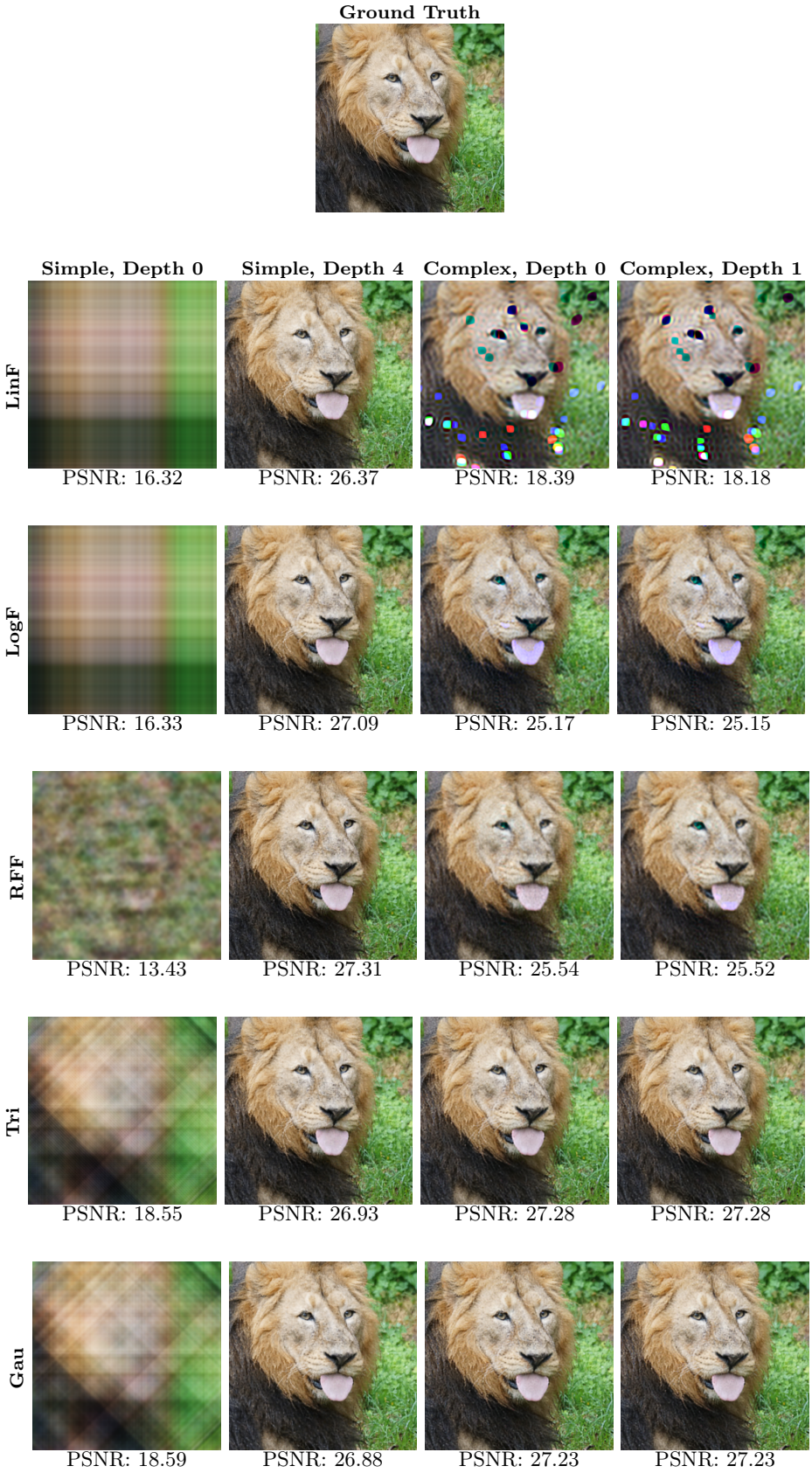


FIGURE 2.11. Reconstruction results of a lion using non-separable coordinates (randomly sampled training points) with different combinations of simple or complex encodings and network depths.

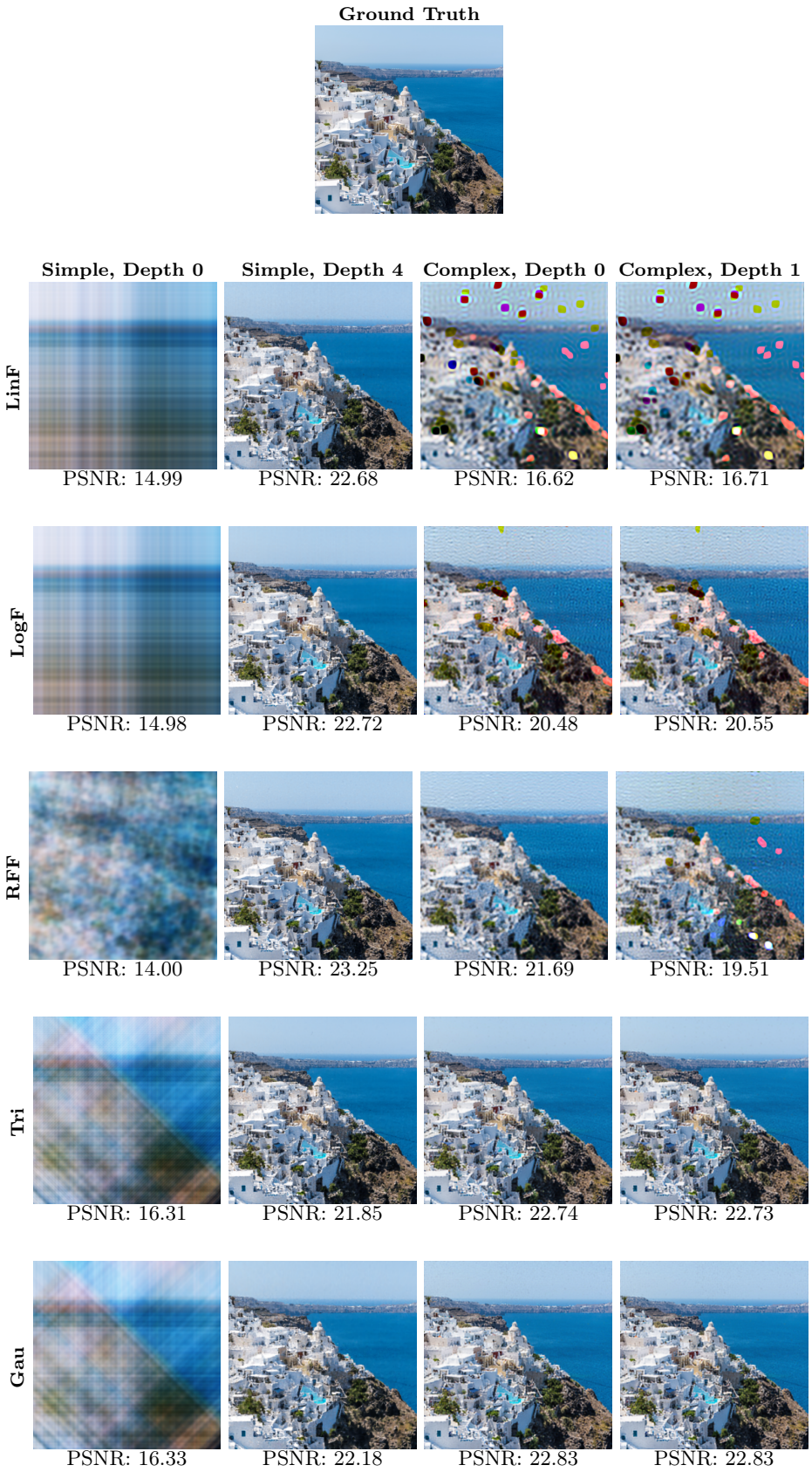


FIGURE 2.12. Reconstruction results of a seaside residential area using non-separable coordinates (randomly sampled training points) with different combinations of simple or complex encodings and network depths.

Statement of Authorship

Title of Paper	Robust point cloud processing through positional embedding
Publication Status	☒ Published ☐ Accepted for Publication ☐ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Zheng, J., Li, X., Ramasinghe, S., & Lucey, S. (2024, March). Robust point cloud processing through positional embedding. In 2024 International Conference on 3D Vision (3DV) (pp. 1403-1412). IEEE.

Principal Author

Name of Principal Author (Candidate)	Jianqiao Zheng		
Contribution to the Paper	Proposed ideas, conducted experiments, and wrote the manuscript.		
Overall percentage (%)			
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		Date	June 3, 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 contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Xueqian Li		
Contribution to the Paper			
Signature		Date	June 7, 2025

Name of Co-Author	Sameera Ramasinghe		
Contribution to the Paper	Helped in discussion and writing revision.		
Signature		Date	June 11, 2025

Please cut and paste additional co-author panels here as required.

Name of Co-Author	Simon Lucey			
Contribution to the Paper	Supervised developement of work; Helped in discussion and writing revision.			
Signature			Date	23/6/25

Chapter 3

Robust Point Cloud Processing through Positional Embedding

Preface

This chapter is based on the following published paper:

> Zheng, Jianqiao, Xueqian Li, Sameera Ramasinghe, and Simon Lucey. *"Robust Point Cloud Processing through Positional Embedding."* In: *Proceedings of the 2024 International Conference on 3D Vision (3DV)*. IEEE, 2024, pp. 1403–1412.

This paper has been adapted for inclusion in this thesis. Minor edits have been made for consistency in notation and formatting. The work in this chapter contributes to the thesis by demonstrating that *rank* also plays an important role in multilayer perceptrons (MLPs) for 3D point cloud processing.

3.1 Introduction

PointNet Qi et al. [2017a] has been a critical innovation in the area of deep geometry, providing much of the backbone to innovations in object classification, segmentation, detection Qi et al. [2018], and alignment Aoki et al. [2019]; Li et al. [2021] on 3D point clouds. Many variants Guo et al. [2021]; Qi et al. [2017b]; Thomas et al. [2019]; Wang et al. [2019b]; Wu et al. [2019]; Zhao et al. [2021] have followed, most notably, point cloud transformer (PCT) Guo et al. [2021] [1] that combines the strengths of PointNet and transformer networks into a state-of-the-art framework for 3D point cloud processing. All these variants have a common element—a learned—typically deep ReLU network—per-point embedding (PPE).

In this chapter, we explore the hypothesis of whether a more robust performance can be afforded by frameworks such as PointNet or PCT if we replace the learned PPEs with analytical embeddings. This hypothesis has two motivations. First, we hypothesize that the earliest modules of modern point cloud networks—specifically the PPE—such as those found in PointNet and PCT, should be most generic in task and domain. As a result, learning such PPEs in an end-to-end fashion solely from data may make the entire pipeline overly sensitive to changes in the point cloud statistics Sun et al. [2022]. Second, the success of positional embedding (PE) in machine learning frameworks such as transformer networks Vaswani et al. [2017] and implicit neural functions Mildenhall et al. [2021] offers an alternate surrogate for end-to-end

[1] We acknowledge that there is concurrent similar work such as point transformer Zhao et al. [2021]. In this chapter, we will focus on PCT Guo et al. [2021].

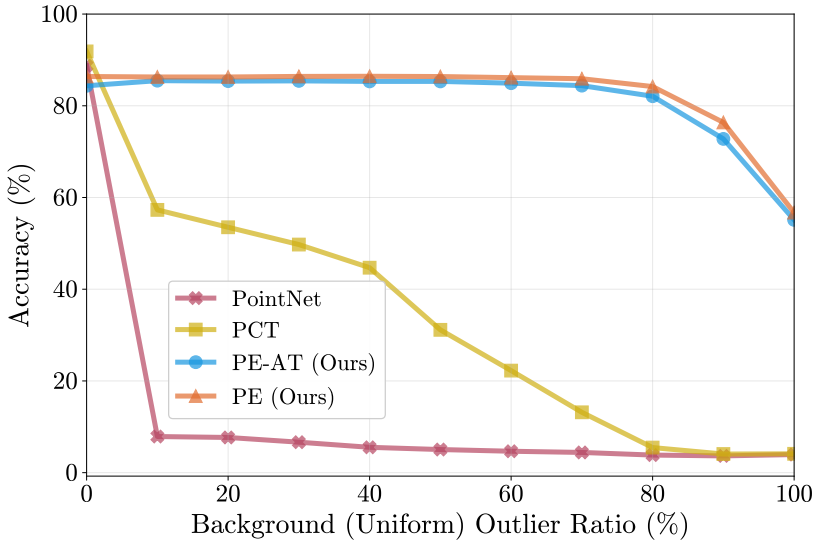


FIGURE 3.1. The robustness of positional embedding (PE)-based methods (PE-AT and PE) in comparison to end-to-end trained methods (PointNet and PCT). We show the classification accuracy on ModelNet40-C Sun et al. [2022] versus the number of outliers (ratio to the number of point clouds) uniformly sampled from $[-1, 1]^3$. We argue in this chapter that PE, especially random Fourier features (RFF), embodies many of the properties one needs in a robust embedding without the need for learning, making it more robust than end-to-end trained per-point embeddings (PPEs).

PPEs. PE Zheng et al. [2022] shares similar characteristics to learned PPEs in that they attempt to transform a scalar or even vector position into a multi-dimensional embedding. They differ fundamentally, however, in that the entire embedding is analytical, requiring only a single tunable hyper-parameter—bandwidth Zheng et al. [2022].

The success of PointNet/PCT is fundamentally wedded to the data it is trained upon. When the distribution of the test environment differs substantially from the training dataset, catastrophic performance can occur (see Fig. 3.1 red and yellow curve). PointNet/PCT involves two components: (i) a per-point high-dimensional embedding (PPE), which refers to all components before the pooling function, and (ii) a pooling operation that aggregates a global feature vector. The theoretical role of (ii) is well understood as a mechanism for providing invariance to permutation Soelch et al. [2019]. In contrast, the role of (i) is largely shrouded in mystery since most of the community treats it as an opaque black box that must be learned. In this chapter, we attempt to theoretically characterize the idealized properties of (i). In particular, we argue that such PPEs should be band-limited and smooth. Further, we argue that learned deep ReLU embeddings—commonly used in most implementations of PointNet/PCT—are inherently flawed in this regard. To make up for this shortcoming they are unnecessarily reliant on aggressive aggregation operators like max-pool to ensure good performance.

The PPEs used in techniques such as PointNet and PCT typically take the form of ReLU-MLPs. The weights of these ReLU-MLPs are learned end-to-end for the task at hand (*e.g.*, alignment, detection, *etc.*). Although the data-dependent training implicitly achieves appropriate bandwidth, it inherently produces in-distribution bias. Inspired by positional embeddings (PE)—recently made popular in transformer networks Vaswani et al. [2017] and implicit neural functions such as NeRF Mildenhall

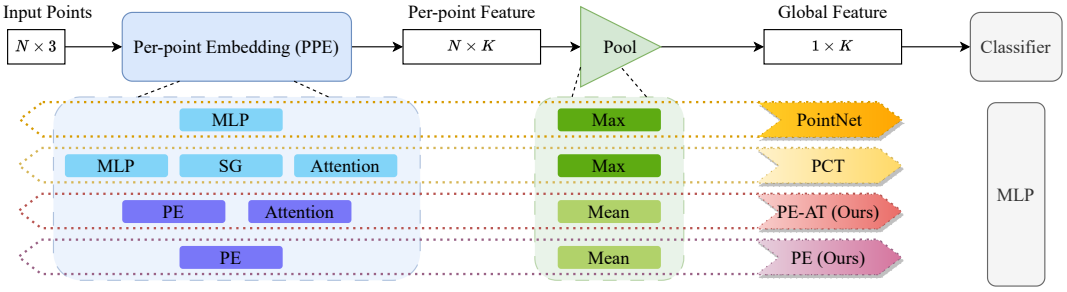


FIGURE 3.2. Illustration of different point cloud processing architectures. SG: sampling and grouping. PE: positional encoding. In per-point embedding (PPE), light blue PPEs are trained end-to-end while purple ones are randomly initialized. Both PointNet-based and PCT-based methods process points with a per-point encoder first and then use a pooling (usually max-pool) to get a feature vector for the point cloud. We found that all the pre-processing stages in PCT could be replaced by a simple PE (our PE-AT model) while maintaining good robustness to OOD noise. Furthermore, we can use PE as a non-learned PPE (our model) and achieve comparable performance. A pooling operation is used to get the global feature vector. And all these models can be used in various downstream tasks, such as classification.

et al. [2021]—we explore the use of sinusoid-based PEs as a surrogate for learned PPEs in PointNet/PCT. Unlike deep ReLU, the per-point high-dimensional embedding provided by PEs does *not* need to be trained on any data—simply tuning the variance of the randomly initialized weights yields effective performance. Our claims are further supported by strong empirical evidence by showcasing the performance of PE-based methods to the ReLU-MLPs on the object classification task (Fig. 3.1). Comparable performance is achieved when testing these methods in the in-distribution data domain (*i.e.*, background outlier ratio is 0%). However, the real advantage of PEs becomes apparent when applied to more practical out-of-distribution (OOD) circumstances. As the mismatch between the train and test distributions increases, PE-based embeddings significantly outperform current learning methods by a large margin.

We conduct substantial experiments across popular benchmarks—ModelNet40 Wu et al. [2015] and ModelNet40-C Sun et al. [2022]—and we show that one can achieve remarkably robust classification performance in the presence of various unseen data corruptions, such as noise and outliers. Our chapter makes the following contributions:

1. We theoretically show why RFF-based PEs are excellent candidates for use as PPEs within PointNet and PCT. In particular, we connect the variance of the weights to the bandwidth of the embeddings and spatial locality—speculating upon their utility in the presence of noise and outliers.

2. Strong empirical evidence that demonstrates that PE-based embeddings significantly outperform current end-to-end trained models when evaluated in a variety of OOD data corruption conditions on both classification and alignment tasks.

3.2 Related work

Learned PPE. Approaches such as PointNet Qi et al. [2017a] and DeepSets Zaheer et al. [2017] opened a new direction of directly processing unsorted 3D point cloud sets.

Usually, this architecture uses MLPs to process the points, followed by a permutation-invariant aggregation function. PointNet++ Qi et al. [2017b] enhances its ability to capture local information and get impressive results. Recently, point cloud transformer (PCT) Guo et al. [2021] combined the attention mechanism into per-point encoder. However, all these models need to be trained end-to-end. In a recent work Sanghi and Jayaraman [2020], it was demonstrated that good performance could be achieved with a randomly initialized deep ReLU PPE within PointNet.

Positional Encoding. Transformers Vaswani et al. [2017] heavily utilize PE so as to bake positional information into the attention mechanism. The popular Neural radiance fields (NeRF) Mildenhall et al. [2021] framework also uses PE for its coordinate encodings. In fact, 2D coordinate networks already widely use positional encodings in tasks such as image reconstruction Nguyen et al. [2015]; Stanley [2007]. Of particular note is the work of Mildenhall *et al.* Mildenhall et al. [2020] and Zhong *et al.* Zhong et al. [2019] who found empirically that using sinusoidal positional encodings allows networks to converge faster, and learn high-frequency signal content better. The most popular PE method in literature is random Fourier features (RFF) Tancik et al. [2020] – the authors motivate their mechanism of positional encoding through NTK theory Arora et al. [2019]; Bietti and Mairal [2019]; Du et al. [2018]; Jacot et al. [2018]; Lee et al. [2019]. Subsequent works have offered expanded theoretical motivations for PE Zheng et al. [2021] and Zheng et al. [2022], which use rank and smoothness to evaluate the utility of a positional encoding scheme.

Pooling Function. In the PointNet architecture, the pooling operation is critical so as to ensure permutation invariance. In the original PointNet Qi et al. [2017a] paper, max and mean poolings are discussed. Bueno *et al.* Bueno and Hylton [2021] explained the mechanism of pooling functions on sets and add sum pooling as an aggregation function. Pl-Net3D Mukhaimar et al. [2019] used deep declarative networks Gould et al. [2019], used M-estimators Rousseeuw and Leroy [2005] to provide additional robustness in pooling. However, due to its high computational cost and complexity, it is not useful in most practical scenarios. Mukhaimar et al. [2021] and Soelch et al. [2019] explored more robust options for pooling functions like median, histogram (mode), and RANSAC (mode). In this paper, we propose to use PEs to replace the learned PPEs in PointNet and PCT to enable the use of robust pooling functions.

3.3 Method

In this section, we provide a brief overview of point cloud processing architecture (Fig. 3.2), which was proposed by PointNet/PCT. Suppose we have a 3D point cloud $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ as input, where $\mathbf{x}_i \in \mathbb{R}^3$ and N is the number of points. The entire point cloud encoder computes a permutation invariant feature $\mathbf{y} \in \mathbb{R}^K$ induced by a mapping function $f : \mathbb{R}^{3\times N} \to \mathbb{R}^K$, as described in Eq. (3.1).

$$f(\mathbf{X}) = f(\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}) \ . \tag{3.1}$$

The mapping function f can be divided into two parts: i) a per-point encoder $h : \mathbb{R}^3 \to \mathbb{R}^K$ and ii) a symmetric (permutation invariant) aggregation function, usually max pooling. Thus, the complete point cloud encoder can be represented as:

$$\mathbf{y} = \max(h(\mathbf{x}_1), h(\mathbf{x}_2), \cdots, h(\mathbf{x}_N)) \ , \tag{3.2}$$

For example, in PointNet Qi et al. [2017a], h is learned mainly via MLPs; in naive PCT Guo et al. [2021], h is the attention layer; in addition, a complete PCT Guo

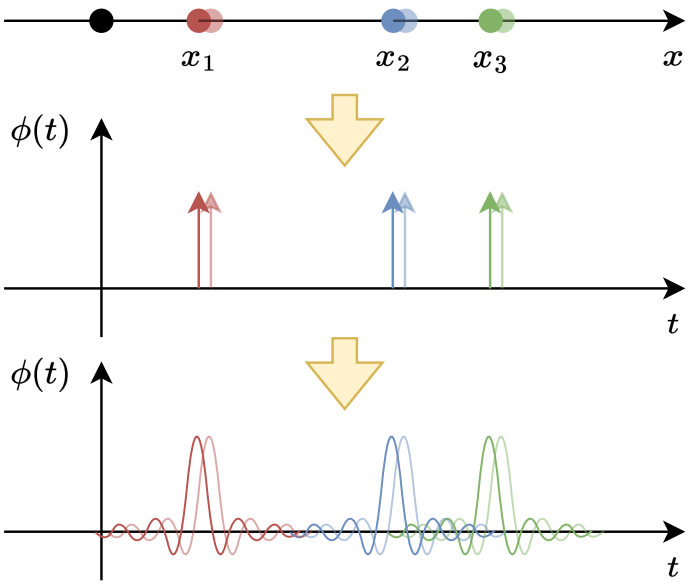


FIGURE 3.3. Illustration of a 1D point cloud and its embeddings. The upper row shows a simple 1D point cloud. The middle row is the encoding with $\delta(\cdot)$. The bottom row is the encoding with $\text{sinc}(\cdot)$. The faded ones indicate points with noise.

et al. [2021] introduces MLP-based input pre-processing and sampling grouping layers to improve the performance of the naive PCT. These three different types of h directly operate on each point and can be viewed as per-point encoders. After h, a max pooling operation is added to aggregate the final feature vector for the entire point cloud. The entire network is trained end-to-end.

However, a recent study Sanghi and Jayaraman [2020] demonstrates that the deep ReLU PPE in PointNet does not necessarily need to be trained; instead, randomly initialized deep ReLU PPE performs well in point cloud classification tasks. This finding motivates us to explore other properties of the PPE beyond its learning capabilities. While randomness makes the encoder data-independent, which potentially helps its performance on data with OOD noise, properties of the PPEs and pooling options are also crucial to the OOD performance.

3.3.1 Band-limited per-point encoder

An obvious strategy to obtain a permutation-invariant lossless feature is to encode points with the impulse function:

$$\phi(t) = \sum_{i=1}^{N} h(x_i, t) = \sum_{i=1}^{N} \delta(t - x_i)\,, \tag{3.3}$$

where $\delta(\cdot)$ is the impulse function and $t \in (0, 1)$. A drawback is that $\phi(t)$ becomes an infinite bandwidth signal, which means we need infinite feature dimensions in discrete space to represent it (observe that this is equivalent to doing one-hot encoding in a continuous space). Another drawback is that when there is noise in the points, the embedding obtained by the impulse function will have a large difference from the embedding of the clean point cloud. Additionally, the error between the clean point embedding and the noisy point embedding is not proportional to the noise intensity.

Nonetheless, a common approximation to the impulse function is $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$, which is band-limited and easy to discretize. The feature can be represented as:

$$\phi(t) = \sum_{i=1}^{N} h(x_i, t) = \sum_{i=1}^{N} \text{sinc}(K(t - x_i)) \,, \tag{3.4}$$

where K is a hyperparameter that controls the bandwidth. Fig. 3.3 provides a visual example. Note that shifting a signal does not affect its spectrum, and the Fourier transform is a linear operation. Therefore, the bandwidth of $\phi(t)$ is K, and it can be represented as K equally spaced, weighted sinc functions as,

$$\phi(t) = \sum_{k=1}^{K} \gamma_k \text{sinc}\left(K\left(t - u_k\right)\right) \,, \tag{3.5}$$

where u_k are equally spaced (Nyquist Shannon sampling theorem Shannon [1949]). This can equivalently be represented in the Fourier domain as,

$$\phi(t) = \sum_{b=0}^{B-1} \gamma_b \cos\left(2\pi b t\right) + \gamma_{b+B} \sin\left(2\pi b t\right) \,, \tag{3.6}$$

where $B = \frac{K}{2}$, and γs are coefficients.

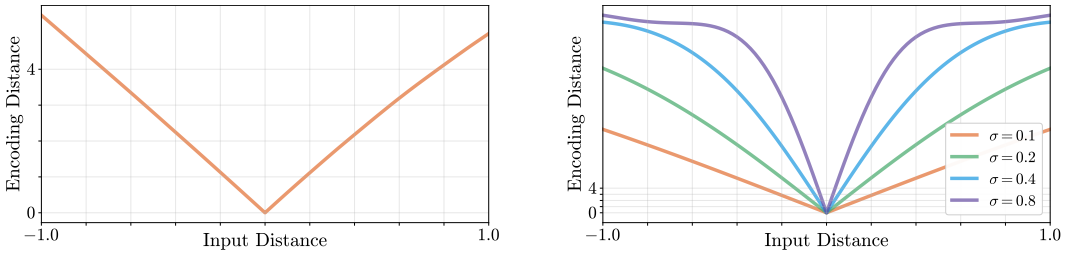
For high-dimensional cases, such as 3D point clouds, all axes must be sampled and encoded, and the resulting embeddings are multiplied to obtain a high-dimensional representation. The main drawback of sinc encoding is the large number of required basis functions, as the encoding dimension grows exponentially with the input dimension. However, a key advantage of sinusoidal functions is that, although sinc and sinusoidal bases appear equivalent in one dimension, in higher-dimensional settings, sinusoids can achieve comparable or better encoding quality with significantly fewer feature dimensions due to trigonometric identities.

In theory, the number of terms in sinc or sinusoidal bases is the same in high dimensions. However, the distribution of these sinusoidal components becomes non-uniform and tends toward a Gaussian whose variance is proportional to the bandwidth B. When subsampling from this large set of basis functions, such a non-uniform distribution provides a principled approach for frequency selection. This insight helps explain why random Fourier features (RFF) Tancik et al. [2020] are particularly effective embeddings for high-dimensional data. We refer readers to Zheng et al. [2022] for a more detailed treatment of this topic.

3.3.2 Smoothness of per-point embedding

According to Zheng et al. [2021], a good embedding should also be smooth (*i.e.*, distance preserving), which means closer point clouds in the input space have more similar features. The feature is also important for the robustness of the model. When the input point cloud has noise or outliers, a smooth per-point encoder can generate a similar embedding to the clean one. In the ideal case, feature distance should be proportional to the distance between input point clouds, which is

$$\begin{aligned} \mathrm{D}_f(\mathbf{y}_1, \mathbf{y}_2) &= \|\phi(\{x_{1,i}\}_{i=1}^{N}, t) - \phi(\{x_{2,i}\}_{i=1}^{N}, t)\|_2^2 \\ &\propto \sum_{i=1}^{N} \|x_{1,i} - x_{2,i}\|_2^2 \,, \end{aligned} \tag{3.7}$$



(A) (B)

FIGURE 3.4. Encoding distance *vs*. point distance for (A) one ReLU layer and (B) RFF-based PE.

where D_f is distance between features, $\{x_{1,i}\}_{i=1}^N$, $\{x_{2,i}\}_{i=1}^N$ are two point clouds and ϕ_1, ϕ_2 are their features.

When the PPE is not band-limited, such as the impulse function, this will not hold since no matter how far the point moves, the distance changes in the feature are the same. When the PPE is band-limited as we discussed in Sec. 3.3.1, based on the theory in Zheng et al. [2021], for each point, the encoding distance is proportional to the input distance, which means

$$\mathrm{D}_e(\mathbf{e}_1, \mathbf{e}_2) = \|\phi(x_1, t) - \phi_2(x_2, t)\|_2^2 \propto \|x_1 - x_2\|_2^2 \,, \tag{3.8}$$

where D_e means distance between encodings, x_1, x_2 are two point clouds and $\mathbf{e}_1$, $\mathbf{e}_2$ are their features. Although it is not necessarily equivalent, it strongly implies Eq. (3.7) will hold since features are just a summation of encodings. In addition, a critical advantage of using our band-limited encoders is that one can easily tune the bandwidth of the representation by varying the variance of the frequency distribution. Zheng et al. [2021] shows that by varying the bandwidth, the distance preservation (or the feature similarity) between nearby point encodings can be controlled. This is a vital property that can be exploited when dealing with OOD data.

3.3.3 Trade-off – bandwidth and smoothness

In the original PointNet architecture, the encoder is an MLP, and the entire network is trained end-to-end. The deep ReLU point embedding for PointNet can be defined as

$$\mathcal{H}_\ell(\mathbf{x}) = \mathrm{ReLU}[\mathbf{W}_\ell \circ \mathcal{H}_{\ell-1}(\mathbf{x}) + \mathbf{b}_{\ell-1}] \,, \tag{3.9}$$

where $\mathcal{H}_0(\mathbf{x}) = \mathbf{x}$. The PPE in PCT can be defined as

$$\begin{aligned} \mathbf{F}_1 &= \mathrm{AT}^1(\mathbf{F}_e) \,, \\ \mathbf{F}_i &= \mathrm{AT}^i(\mathbf{F}_{i-1}) \,, \; i = 2, 3, 4 \\ \mathbf{F}_o &= \mathrm{concat}(\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3, \mathbf{F}_4) \,, \end{aligned}$$

where AT^i is the i-th attention layer, $\mathbf{F}_e$ is the input embedding through some MLPs and sampling and grouping layer, and $\mathbf{F}_o$ is the final embedding. The deep ReLU network and attention layers map low-dimensional encodings to a high-dimensional space. This mapping is band-limited and preserves distance information when randomly initialized. However, after training end-to-end with data, its ability to preserve distance entirely depends on the training data. There is no clear geometry information in the learned features; therefore, when OOD noise is added to the data, it is

TABLE 3.1. Classification accuracy(%) on ModelNet40 of RFF-based PE with different scales of RFF.

Scale	0.05	0.1	0.5	1	5	10	15	20
Train	80.9	82.8	95.6	99.2	100	100	100	100
Test	66.2	79.2	86.1	86.1	76.1	33.8	23.0	19.6

unpredictable how the features will change in the feature space and whether they will pass the decision boundary of the learned classifier or not.

Moreover, even though random initialized MLPs can be viewed as a non-trained per-point encoder, it is unclear how the initialization affects the behavior of the embeddings. On the contrary, PE has a well-established theory—by tuning the hyperparameters (usually called "scale" in PE), we can theoretically know the bandwidth and smoothness changes and predict the performance of the embedding. The sinusoids encoder we discussed in Sec. 3.3.1 turns into PE—RFF, which is defined as

$$\mathcal{G}(\mathbf{x}) = \begin{bmatrix} \cos(\mathbf{W}\mathbf{x}) \\ \sin(\mathbf{W}\mathbf{x}) \end{bmatrix} . \tag{3.10}$$

The distance preservation property of these encoders is shown in Fig. 3.4. For the ReLU network, the distance is kept linear in the point space, while for RFF, this property can be tuned by tuning the scale (standard deviation of the random weight matrix). Even if we change the standard deviation of the ReLU network, *i.e.*, multiply the weights with a specific scale, we will get an output that is also multiplied by that specific scale, instead of changing the shape of the distance preservation curve. Note that for RFF, when the scale is large, such as σ=8 (shown in Fig. 3.4), only local distance is preserved, which means only local information is considered. This property makes the RFF encoder robust to noise and outliers.

3.4 Experiments

In this section, we first show that PE can replace PPE with classification results on the clean ModelNet40 Wu et al. [2015] dataset and have the ability to control bandwidth by tuning the scale of RFF-based PE. Then we use our modified ModelNet40-C Sun et al. [2022] dataset that contains various corruptions to show that our PE and PE-AT methods are much more robust than current end-to-end trained methods when dealing with noisy data. We show the performance of using different pooling functions to validate our theory of how to get a robust model. Finally, we showcase point cloud registration results on ModelNet40 in OOD scenarios.

3.4.1 Experimental settings

As mentioned before, the architecture of PointNet and PCT can be divided into three parts, *i.e.*, a PPE (ReLU-MLPs), a pooling function (max pooling), and a downstream network (fully connected networks, FCNs). Here, we introduce our settings for these three components separately.

Per-point embeddings. We used two different types of PPEs. We denote the attention layer as AT, the linear layer as FC (fully connected), the ReLU activation function as ReLU, and the random Fourier features as RFF. The dimension of the input and output feature to the encoder is denoted as (X, Y). **PE-AT (RFF-based**

PE + attention) is a variation of PCT. We use PE as the input embedding and directly feed the embeddings into 4 attention layers: RFF (3, 256) → AT. The scale of RFF is 0.9. **PE (RFF-based PE)** uses PE as the surrogate PPE: RFF (3, 1024). The scale of RFF for max pooling is 0.09, and for all the other pooling functions (mean/median), we used 0.9.

Pooling functions. We used three types of pooling operations: max pooling, mean pooling, and median pooling. Max pooling is widely used in current models, which results in poor performance in OOD cases. Mean pooling is used in our models to maintain robustness to noise. Median pooling is used in the noisy data experiments to further demonstrate that PE enables a more robust choice of pooling functions that better deal with OOD noise.

Downstream networks. For the classifier task, we used the exact network in the original PointNet Qi et al. [2017a]: FC (1024, 512) → BN → ReLU → FC (512, 256)→ Dropout (0.4) → BN → ReLU → FC (256, #class). Here, Dropout (0.4) is the dropout layer with 40% of the probability of an element being zero, and BN refers to the batch normalization layer. The number of object categories in classification (#class) is 40. Note that the classifier in the original PCT differs; for a fair comparison, we replaced it with the PointNet classifier. This modification does not affect the overall performance of the method.

3.4.2 Classification on clean data

We followed the standard training protocol in Sun et al. [2022] and trained all models on clean ModelNet40 Wu et al. [2015] data with 1024 points and 300 epochs. We did not use any data augmentation strategy except random scaling and translation as proposed in DGCNN Wang et al. [2019b]—which is usually considered as "unaugmented". We show the classification results on clean data in the first columns of Tab. 3.2. We observed that end-to-end trained methods achieve good performance and PointNet++ has the lowest error rate. Both randomly initialized PE-AT and PE achieve comparable but slightly worse results compared to learned PointNet, indicating the ability of PE-based methods in replacing learned PPEs.

Ablation study on different scales of PE

As mentioned in Sec. 3.3.2, different scales (standard deviations) decide the bandwidth and distance preservation of RFF-based PE, therefore affecting the performance. We demonstrate this behavior in Tab. 3.1, where increasing the standard deviation in RFF leads to higher training accuracy but lower testing accuracy beyond a certain threshold. This property also applies to all other PEs, such as the Gaussian encoder proposed in Zheng et al. [2021].

3.4.3 Robustness to noisy data

To compare the robustness of different PPEs and pooling functions, we trained the model (the entire model or the classifier alone if PPEs do not need to be trained) on clean data and tested these trained models on test data under six different types of noise and outlier corruptions. We selected five noise/outlier types from the ModelNet40-C Sun et al. [2022] dataset: uniform noise, Gaussian noise, impulse noise, upsampling outliers, and background outliers. However, the corruption generated by ModelNet40-C is too small, which makes the dataset less challenging. Therefore, we enhanced the severity of the corruption by either enlarging the variance of the noise or increasing

TABLE 3.2. Classification results of different models on modified ModelNet40-C (generated by larger corruption severities). All results shown here are classification errors. The upper tabular between **blue bars** are methods with end-to-end training. The lower tabular between **orange bars** are methods with randomly initialized PPEs and trained classifiers. All the training is on ModelNet40 clean data.

Method	Clean	Corruption (avg.)	Noise Corruption			Outliers Corruption				
			Uniform	Gaussian	Impulse	Upsampling	Background	Ball(10%)	Ball(50%)	Ball(100%)
PointNet Qi et al. [2017a]	11.3	55.8	18.3	41.4	28.5	13.4	94.7	82.5	83.7	83.9
PointNet++ Qi et al. [2017b]	**7.5**	43.8	37.3	70.1	47.2	12.4	49.1	27.1	48.3	58.8
DGCNN Wang et al. [2019b]	8.1	52.7	35.2	67.0	35.7	21.9	66.5	50.9	44.6	—
RSCNN Liu et al. [2019]	9.3	52.7	30.4	65.5	49.9	**10.5**	60.6	54.6	75.3	74.4
PCT Guo et al. [2021]	8.1	51.9	22.2	57.8	37.1	12.4	71.5	61.2	75.0	77.9
PE-AT (Ours)	15.6	23.3	15.1	**17.1**	16.0	14.7	19.4	16.0	33.4	55.1
PE (Ours)	13.6	**22.1**	**14.1**	**17.1**	**15.7**	13.7	**17.9**	**14.6**	**31.8**	**52.3**

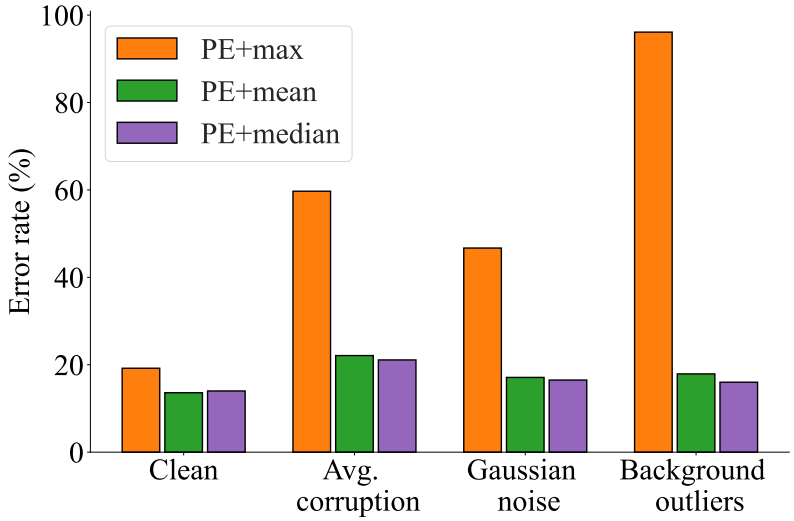


FIGURE 3.5. Error rate of our PE method with different pooling functions under different data corruptions.

the number of outliers, adding to the clean point cloud. We also included a ball outlier to better demonstrate the effectiveness and robustness of PE-based methods. The detailed settings are listed below.

Noise corruption is a type of data corruption where the total number of points in a point cloud is unchanged, but a subset of points is corrupted by noise. For example, a part of or all points are shifted from their original positions by noise. Usually, this noise can be represented by a random distribution, such as uniform, Gaussian, or impulse. The severity range of the noise we used in the experiment is as follows. **1) Uniform noise:** 10 different severity levels ranging from 0.01 to 0.1 were used for all points. **2) Gaussian noise:** a mean 0 and standard deviation ranging from 0.01 to 0.1 Gaussian noise was added to all points. **3) Impulse noise:** an impulse with 0.1 magnitude, $[1, 1, 1]$ or $[-1, -1, -1]$ direction was added to 2% to 100% of points.

Outlier corruption is a type of data corruption where a small portion of additional points was added to the clean point cloud, yielding an increase in the total number of points. We used three different types of outliers, *i.e.*, upsampling, background, and ball outliers. We list the severity parameters of the outliers we added as follows. **4) Upsampling outliers:** 10% to 100% of points were randomly copied and modified with uniform noise in $[-0.05, 0.05]$. Then these points were added to the original point cloud as outliers. **5) Background outliers:** 2% to 100% of points sampled uniformly from $[-1, 1]$ were added as additional points. **6) Ball outliers:** We designed a ball outlier that was uniformly sampled on a ball surface with a radius ranging from 0.3 to 3.0. We added 10%, 50%, and 100% of points as additional points.

We show the performance of our methods compared to other learned PPEs under different data corruptions in Tab. 3.2. Our methods achieve compelling results across all data corruption types, indicating the great advantage of using untrained PE as a surrogate for learned PPEs under OOD scenarios. Specifically, our methods maintain great accuracy, yielding a substantial decrease in error rate when common corruptions like Gaussian noise and background outliers are presented in the data. The superior performance of our methods further validates the robustness of PE-based embeddings in dealing with OOD cases such as noise and outliers. Moreover, the bandwidth of PE could be easily tuned through the scale of the embedding to control the ability to handle high-level noise corruption.

TABLE 3.3. Ablation study on different architectures of PCT and PE-AT. The upper tabular methods between **blue bars** use trained end-to-end PPEs, and the lower tabular methods between **orange bars** use randomly initialized PPEs.

Model	Pooling	Clean ER	Corruption ER
PCT Guo et al. [2021]	max	**8.1**	51.9
PCT Guo et al. [2021]	mean	8.8	52.7
PE-AT (Ours)	max	10.1	47.7
PE-AT (Ours)	mean	11.2	37.7
PCT Guo et al. [2021]	max	14.3	55.2
PCT Guo et al. [2021]	mean	13.0	49.4
PE-AT (Ours)	max	17.3	39.3
PE-AT (Ours)	mean	15.6	**23.3**

Our methods achieve state-of-the-art results on almost all corruption types. However, in the upsampling outlier corruption case, all methods (except DGCNN) achieve similarly good performance. The reason behind this is that the point outliers were generated from upsampled points with a very small uniform noise ranging in $[-0.05, 0.05]$ that does not account for a strong OOD case. For other outlier corruption experiments with uniformly sampled background outliers and outliers sampled on a ball surface, PointNet Qi et al. [2017a] fails catastrophically due to its PPE processing each point independently and only contains global features, making the point embedding easily affected by these outliers. Other learned PPEs Guo et al. [2021]; Liu et al. [2019]; Qi et al. [2017b]; Wang et al. [2019b] combine local information in neighboring points that help the resistance to small outliers. Nevertheless, these local patterns are still biased to the training data and cannot achieve robust performance when handling large noise/outlier corruptions. We have also noticed that the randomly initialized attention (PE-AT) will not harm the performance of the untrained PE (PE-AT) and achieve comparable results to PE. We speculate that the relation map between points in a point cloud is not biased on the training data.

3.4.4 Ablation study of different pooling functions

We show the performance of our PE embedding on different corrupted data using different pooling functions, such as max, mean, and median in Fig. 3.5. On clean data, all pooling functions achieve similarly good performance, and mean/median poolings have slightly better accuracy. However, when the data is corrupted, for example, Gaussian noise or background outliers are added, max pooling yields catastrophic errors, while mean and median poolings maintain robust and good accuracy.

3.4.5 Ablation study of PCT and PE-AT

In Tab. 3.3, we show the results under different PPEs (end-to-end trained or randomly initialized) and pooling (max or mean) settings of PCT and PE-AT using both clean and corrupted data. On the clean data, all methods using mean pooling show slightly worse performance than using max pooling when doing end-to-end training, since max pooling allows for more powerful feature learning. On the contrary, when using randomly initialized PPEs, all methods using mean pooling result in slightly better

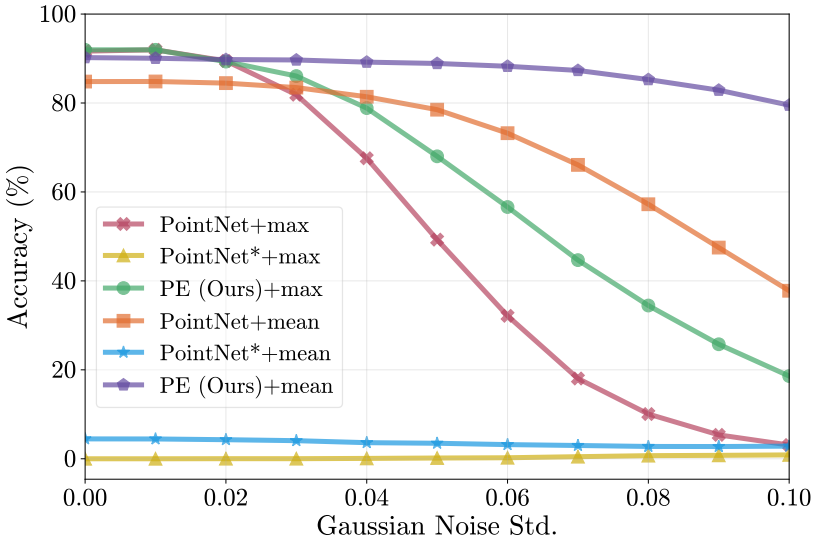


FIGURE 3.6. Point cloud registration accuracy on ModelNet40 20 categories with Gaussian noise. Our method achieves robust registration results on noisy data, especially with mean pooling. All methods use the Lucas-Kanade algorithm for alignment. Note that PointNet* refers to PointNet with randomly initialized PPE.

accuracy due to the data-unbiased nature of these random PPEs. For corruption cases, if we only change one condition from the original model (PCT+Max, 51.9%), changing max pooling to mean pooling (PCT+Mean, 52.7%) or just using a randomly initialized per-point encoder (PCT(R)+Max, 55.2%) will not work. However, only changing the input embedding to PE (PE-AT+Max, 47.7%) works slightly better. When two conditions are changed, the one without PE (PCT(R)+Mean, 49.4%) is the worst. The ones with PE (PE-AT+Mean, 37.7% & PE-AT(R)+Max, 39.3%) already show good robustness on OOD noise and outliers. Finally, if we change all three conditions (PE-AT(R)+Mean, 23.3%), the model works best on corruption cases.

3.4.6 Point cloud registration

We test the robustness of different pooling functions and different PPEs (PointNet or PE) on the point cloud registration task using the ModelNet40 dataset.

We chose the point cloud registration method Li et al. [2021] that uses a simple PointNet structure—3-layer MLPs—as a feature encoder as our baseline. In PointNetLK Revisited Li et al. [2021], a point-based Lucas-Kanade algorithm is used for registration, which is essentially a Gauss-Newton method. Note that the learning-based method was only trained on a clean ModelNet dataset. We then replaced the PPE in PointNet with PE and kept the Gauss-Newton registration algorithm as our method.

Similar to Sec. 3.4.3, we added Gaussian noise with a zero mean and a standard deviation ranging from 0.01 to 0.1 with a step size of 0.01. As shown in Fig. 3.6, PointNetLK Revisited Li et al. [2021] achieves good performance when there is only small noise presented. However, when large noise is added, the performance drops dramatically, especially for untrained PPE. In contrast, our method (PE) maintains good performance even when large amounts of noise are present. Note that mean pooling yields the most robust performance, indicating the effectiveness of our method. The visual comparison is shown in Fig. 3.7.

TABLE 3.4. Performance on other data corruptions as provided in ModelNet-C Sun et al. [2022]. We showcased our performance using increased density, cutout density, rotations, and shear transformations.

Model	Density Inc.	Cutout	Rotation	Shear
PointNet	**11.7**	13.2	34.8	25.8
PointNet++	17.5	**11.6**	27.6	15.0
PCT	11.8	13.3	**16.4**	**10.8**
PE-AT (Ours)	20.1	19.0	36.3	31.5
PE (Ours)	18.3	17.3	27.1	22.3

3.5 Theoretical results

3.5.1 RFF Derivation

In this section, we extend the derivation of Random Fourier Features (RFF) from band-limited embeddings discussed in the main text Sec. 3.1. In particular,

1. We derive a per-point embedding (PPE) not trained on data that is better for out-of-domain (OOD) generalization.
2. Empirically, sinusoidal embeddings are better than ReLU embeddings under randomly initialized conditions. This finding can be explained by the tradeoff between the controllable bandwidth and the local smoothness.
3. We show why RFF is a band-limited PE and derive RFF from sinusoidal embeddings. Since RFF is a derivative of sinusoidal embeddings, it explains why it is a more robust PPE than other band-limited PEs.

3.5.1.1 RFF in 2D

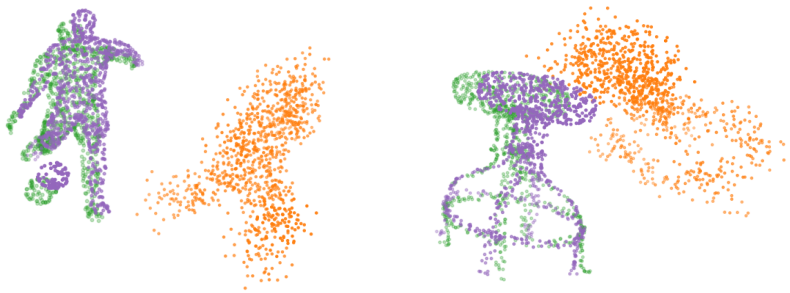
We show that for the 2D case, instead of using a sinc function, we can use sinusoidal embeddings. Both of these embeddings form orthonormal bases in a K-dimensional space.

We want to emphasize that different encodings are equivalent to coordinates in different coordinate systems. For example, in the 2D case, the K band-limited embedding by sinc can be viewed as

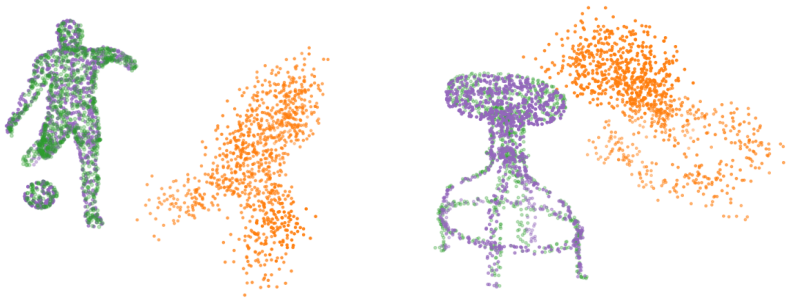
$$\phi(u,v) = \sum_{i=1}^{K}\sum_{j=1}^{K} \gamma_{i,j}\text{sinc}\left(K\left(u-x_i\right)\right)\text{sinc}\left(K\left(v-y_i\right)\right) \ , \tag{3.11}$$

TABLE 3.5. Classification results of different models on ModelNet40-C Sun et al. [2022] with 3 different data corruptions. All results shown here are classification errors. The upper tabular between **blue bars** are methods with end-to-end training. The lower tabular between **orange bars** are methods with randomly initialized PPEs and trained classifiers. All the training is on ModelNet40 clean data.

Method	Corruption	Density Corruption					Noise Corruption					Transformation Corruption				
	(avg.)	Occlusion	LiDAR	Density Inc.	Density Dec.	Cutout	Uniform	Gaussian	Impulse	Upsampling	Background	Rotation	Shear	FFD	RBF	Inv. RBF
PointNet Qi et al. [2017a]	28.9	54.4	56.9	11.7	12.7	13.2	14.1	15.2	28.7	14.9	93.2	34.8	25.8	21.1	18.9	18.2
PointNet++ Qi et al. [2017b]	26.1	56.3	71.7	17.5	11.2	11.6	24.0	19.9	43.2	21.8	22.4	27.6	15.0	15.6	17.2	16.5
DGCNN Wang et al. [2019b]	26.4	60.6	82.4	14.4	18.1	15.7	16.5	18.4	25.7	16.6	51.0	20.0	12.3	13.4	15.2	14.2
RSCNN Liu et al. [2019]	27.8	53.4	75.4	18.3	15.0	15.4	24.4	14.4	54.3	17.8	30.0	26.9	16.3	17.7	19.1	17.8
PCT Guo et al. [2021]	24.0	58.2	73.4	11.8	13.8	13.3	12.0	13.7	32.1	15.5	53.7	16.4	10.8	11.6	12.2	11.6
PE-AT (Ours)	26.8	57.8	55.6	20.1	24.3	19.0	19.8	14.9	18.1	19.3	14.9	36.3	31.5	25.3	22.9	22.5
PE (Ours)	24.1	55.2	51.4	18.3	22.2	17.3	16.5	13.5	15.0	16.0	13.6	34.5	27.1	22.3	19.8	19.3



(a) PointNetLK Revisited



(b) PE (Ours)

FIGURE 3.7. A visualization of point cloud registration using (a) PointNetLK Revisited Li et al. [2021] and (b) PE on ModelNet40 dataset. Gaussian noise with a standard deviation of 0.05 is presented. Green: clean target points, orange: transformed source points with Gaussian noise, and purple: source points transformed by the estimated pose. Note that we only show purple points in clean data to better visualize the shape of the object. In the experiments, we used noisy data as shown in orange. The learning-based method with the PointNet encoder cannot properly handle noise, leaving a large discrepancy between the deformed source point cloud and the target point cloud. In contrast, our PE method successfully aligns two point clouds even when large noise is added.

which requires K^2 samples for each combination of x_i and y_i. Therefore, the sinusoidal embeddings can be represented as

$$\begin{aligned}\phi(u,v) = &\sum_{i=0}^{B-1}\sum_{j=0}^{B-1} A_{i,j}\cos(2\pi iu)\cos(2\pi jv) \\ &+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1} A_{i+B,j}\sin(2\pi iu)\cos(2\pi jv) \\ &+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1} A_{i,j+B}\cos(2\pi iu)\sin(2\pi jv) \\ &+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1} A_{i+B,j+B}\sin(2\pi iu)\sin(2\pi jv)\ , \end{aligned} \tag{3.12}$$

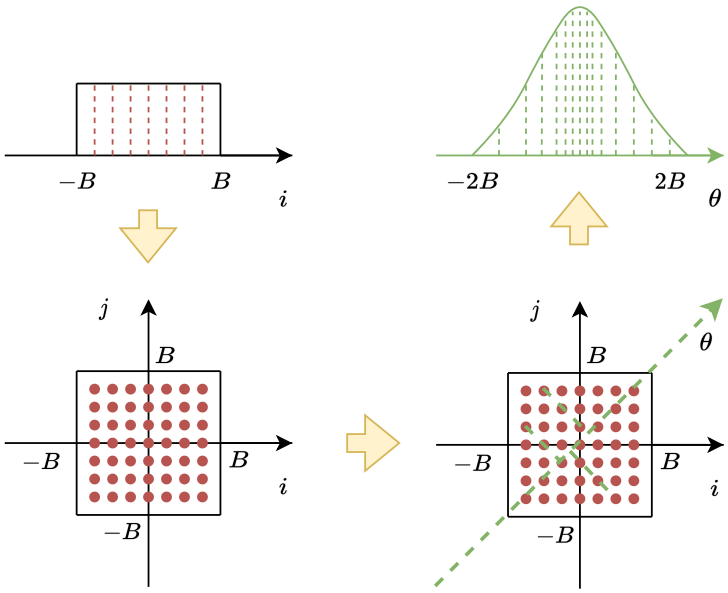
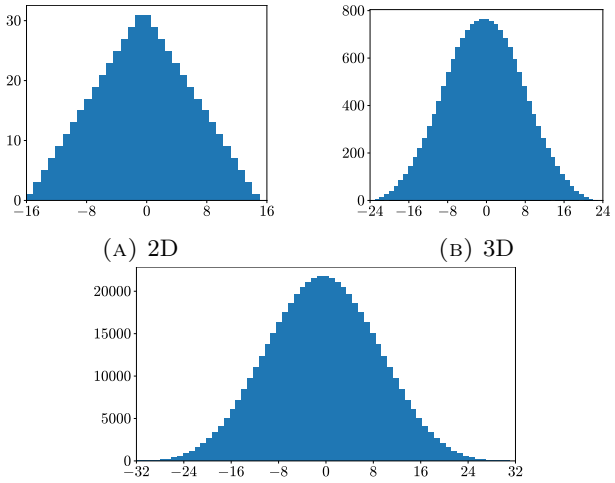


FIGURE 3.8. Illustration of how 1D and 2D band-limited embeddings sample the frequencies. The upper left corner depicts a histogram of coefficients for sinusoidal embeddings in 1D, where frequencies are equally spaced between $-B$ to B, denoting the bandwidth. For the 2D case (see Eq. (3.12)), a histogram of these coefficients can be seen in the bottom left corner, where the frequencies are equally spaced in 2D. The trigonometric identities used in Eq. (3.13) are equivalent to the rotation of a 2D grid onto a new axis (see bottom right corner). In the top right corner, one can see how the position of these frequencies on this rotated axis now tends towards a Gaussian—no longer a uniform distribution. Note that there are still K^2 green dashed lines on the θ axis, which means there are still K^2 different frequencies in 2D. However, the frequency density shape gives us an idea of how to sub-sample the frequencies.

which needs K^2 terms for each combination of i and j. Following trigonometric identities

$$\begin{aligned}
\sin\alpha\cos\beta &= \frac{\sin(\alpha+\beta)+\sin(\alpha-\beta)}{2} \\
\cos\alpha\cos\beta &= \frac{\cos(\alpha+\beta)+\cos(\alpha-\beta)}{2} \\
\sin\alpha\sin\beta &= \frac{\cos(\alpha-\beta)-\cos(\alpha+\beta)}{2},
\end{aligned} \tag{3.13}$$



(A) 2D (B) 3D

(C) 4D

FIGURE 3.9. Frequency distributions in D-dimensional space where D=2, 3, 4. The x-axis is the frequency, and the y-axis denotes the frequency distribution. Here, we set $B = 8$ and ignore the 2π term in the frequency.

we have,

$$\begin{aligned}\phi(u,v) = &\sum_{i=0}^{B-1}\sum_{j=0}^{B-1} C_{i,j}\cos\left(2\pi(iu+jv)\right) \\ &+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1} C_{i+B,j}\cos\left(2\pi(iu-jv)\right) \\ &+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1} C_{i,j+B}\sin\left(2\pi(iu+jv)\right) \\ &+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1} C_{i+B,j+B}\sin\left(2\pi(iu-jv)\right)\ .\end{aligned} \tag{3.14}$$

Note that Eq. (3.14) and Eq. (3.12) are mathematically equivalent to each other. The coefficients C and A are coordinates under different coordinate systems. Although there are still K^2 terms, the transformation is not orthonormal. The distributions of these sinusoids are no longer uniform but in the triangle shape of width $4B$—which is the sum of two uniform distributions on $[-B, B]$. If one attempts to sub-sample from these K^2 frequencies, this non-uniform distribution offers a principled strategy for frequency selection. Please see Fig. 3.8 (a) for an example.

3.5.1.2 RFF in 3D

In the 3D space, we have similar derivations. The sinusoidal embeddings can be represented as

$$
\begin{aligned}
&\phi(u,v,w) \\
&= \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i,j,k}\cos(iu)\cos(jv)\cos(kw) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i,j,k+B}\cos(iu)\cos(jv)\sin(kw) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i,j+B,k}\cos(u)\sin(jv)\cos(kw) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i,j+B,k+B}\cos(iu)\sin(jv)\sin(kw) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i+B,j,k}\sin(iu)\cos(jv)\cos(kw) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i+B,j,k+B}\sin(iu)\cos(jv)\sin(kw) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i+B,j+B,k}\sin(u)\sin(jv)\cos(kw) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} A_{i+B,j+B,k+B}\sin(iu)\sin(jv)\sin(kw)\ ,
\end{aligned}
\tag{3.15}
$$

which needs K^3 terms for each combination of i, j and k. Here we omit 2π of each frequency for simplicity. Following the same trigonometric identities, Eq. (3.15)

becomes,

$$
\begin{aligned}
&\phi(u,v,w) \\
&= \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i,j,k}\cos\left(2\pi(iu+jv+kw)\right) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i,j,k+B}\cos\left(2\pi(iu+jv-kw)\right) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i,j+B,k}\cos\left(2\pi(iu-jv+kw)\right) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i,j+B,k+B}\cos\left(2\pi(iu-jv-kw)\right) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i+B,j,k}\sin\left(2\pi(iu+jv+kw)\right) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i+B,j,k+B}\sin\left(2\pi(iu+jv-kw)\right) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i+B,j+B,k}\sin\left(2\pi(iu-jv+kw)\right) \\
&+ \sum_{i=0}^{B-1}\sum_{j=0}^{B-1}\sum_{k=0}^{B-1} C_{i+B,j+B,k+B}\sin\left(2\pi(iu-jv-kw)\right) \ . \qquad (3.16)
\end{aligned}
$$

Similarly to the 2D case, these K^3 samples of frequencies can also be sub-sampled according to the shape of frequency distributions as in Fig. 3.9 (b). The sinusoidal embeddings are expected to work more efficiently in higher dimensions and this shape becomes more and more similar to a Gaussian distribution (Fig. 3.9 (c)), which leads to the definition of RFF embedding.

3.5.1.3 Frequency distribution derivation

As the dimension D of the points increases, the efficiency of these band-limited embeddings becomes more prominent. The shape of the frequency distribution becomes the sum of D uniform distributions on $[-B, B]$, which yields the Irwin-Hall distribution. According to the Central Limit Theorem, we know that the sum of n uniform distributions on $[0, 1]$ (Irwin-Hall distribution) approximate Gaussian distribution with $\mu = \frac{n}{2}$ and $\sigma^2 = \frac{n}{12}$. Therefore, in D-dimensional space, the distribution of projection

frequencies $\mathbf{W}$ is

$$\begin{aligned}\mathbf{W} &= \sum_{i=1}^{D} \mathcal{U}(-B, B) \\ &= \sum_{i=1}^{D} (2B\mathcal{U}(0,1) - B) \\ &= 2B \sum_{i=1}^{D} \mathcal{U}(0,1) - DB \\ &\approx 2B\mathcal{N}(\frac{D}{2}, \frac{D}{12}) - DB \\ &= \mathcal{N}(0, \frac{DB^2}{3}),\end{aligned} \tag{3.17}$$

where $\mathcal{U}(a,b)$ is uniform distribution on $[a,b]$ and $\mathcal{N}(\mu, \sigma^2)$ is Gaussian distribution with mean μ and standard deviation σ, D is the dimension and B is the band-width. We therefore know that the projection frequencies $\mathbf{W}$ (as well as the weight matrix in RFF) should follow the Gaussian distribution $\mathcal{N}(0, \frac{DB^2}{3})$. This gives some intuitions on why RFF is a more attractive embedding for higher-dimensional data. To achieve comparable performance with sinc or a similar approximated function (*e.g.*, Gaussian), we need K^D samples. In contrast, for RFF, only a limited number of sub-samples is needed. Additionally, when tuning the standard deviation of the scale of RFF (standard deviation of the Gaussian distribution), one can easily control the bandwidth of this encoder.

3.5.2 Pooling functions

Thus far, the encoding of the point cloud discussed in the main text and aforementioned section all used sum pooling to get the final point representation. However, max pooling is the most common pooling mechanism used in the context of learned PPEs such as PointNet Qi et al. [2017a]. Mean pooling is also used as a comparison oftentimes. Here, we first show that all these pooling methods can be formulated as a form of sum pooling. Then we dive deeper into the difference between max pooling and mean pooling.

Mean pooling is straightforwardly connected to sum pooling. The only difference between these two pooling functions is a scale parameter:

$$\begin{aligned}\mathbf{y} &= \text{mean}\left(\{h(\mathbf{p}_1), h(\mathbf{p}_2), \cdots, h(\mathbf{p}_N)\}\right) \\ &= \frac{1}{N} \sum_{i=1}^{N} h(\mathbf{p}_i) \\ &= \sum_{i=1}^{N} \frac{1}{N} h(\mathbf{p}_i) \\ &= \sum_{i=1}^{N} h'(\mathbf{p}_i)\,,\end{aligned} \tag{3.18}$$

where $h'(\cdot) = \frac{1}{N} h(\cdot)$. In Eq. (3.18), we take the scale parameter into the PPE h and convert mean pooling to sum pooling. Max pooling can be converted similarly since max pooling can be seen as zero masking all the elements in one column, except for the maximum value in that column, and summing over columns. If we take the zero

masking process into $h'(\cdot)$, then we have

$$h'(\mathbf{p}_i)[k] = \begin{cases} h(\mathbf{p}_i)[k], & \text{if } \forall j \neq i, h(\mathbf{p}_i)[k] \geq h(\mathbf{p}_j)[k] \,, \\ 0, & \text{otherwise} \,. \end{cases} \tag{3.19}$$

Note that we ignore the case when there are multiple maximum values in one column, which rarely happens in real-world data. Following this analogy, any pooling function can be converted to sum pooling. The magic step is to take all the operations before the last sum operation into the PPE function. Therefore, the bandwidth and smoothness theory we discussed in the main text and aforementioned sections holds for all types of point encoders.

In PointNet Qi et al. [2017a] and its variants Liu et al. [2019]; Qi et al. [2017b]; Wang et al. [2019b]; Wu et al. [2019], max pooling has been the most popular pooling choice due to its good performance over mean pooling. However, the reason why max pooling has better performance is unclear. In order to have a better understanding of the difference between max pooling and mean pooling, we show an extreme case where the PPE only contains one linear layer, *i.e.*, $h(\mathbf{x}) = \mathbf{A}\mathbf{x}+\mathbf{b}$, without ReLU or any other nonlinear activation functions. In this case, mean pooling fails while max pooling works surprisingly well. With mean pooling, the entire network can be represented as

$$\mathbf{y} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{A}\mathbf{x}_i + \mathbf{b}) = \mathbf{A}\bar{\mathbf{x}} + \mathbf{b} \,, \tag{3.20}$$

where $\bar{\mathbf{x}} = \frac{1}{N}\sum_i^N \mathbf{x}_i$ is the center (mean) of the point cloud. In this case, mean pooling fails since the feature is only a linear transformation of the center point. This can also be explained by the bandwidth and smoothness properties. Even though the distance information is perfectly preserved, the bandwidth of the PPE is always 3 (for 3D point cloud) when using mean pooling, which does not increase at all. However, max pooling acts as a nonlinear operator. Eq. (3.19) shows that the max pooling operator is similar to an extreme ReLU function. Following this observation, we introduce the following lemma.

Lemma 1. *For an input $\mathbf{X} \in \mathbb{R}^{N\times K}$, max pooling by column is equivalent to a sequence of max-normalization (parameterized by $\mathbf{b}$ and c, which are dependent on $\mathbf{X}$, similar to a weaker version of Instance Normorlizaiton), ReLU and a sum pooling by column, which can be written as*

$$\max(\mathbf{X}) = \text{mean}(\text{ReLU}(\frac{\mathbf{X} - \mathbf{b}}{c})) \tag{3.21}$$

As discussed in Zheng et al. [2022], ReLU can inherently increase bandwidth and keep some distance at the same time, so max pooling can work well even with only a linear layer.

3.5.2.1 Proof of Lemma 1

Now let us consider how we do max pooling given a matrix $\mathbf{X} \in \mathbb{R}^{N\times K}$. We want to apply max pooling to its columns to get a feature vector $\hat{\mathbf{x}} \in \mathbb{R}^{1\times K}$, which is equivalent to first zero masking all the elements except for the maximum values in each column

as $\ddot{\mathbf{X}}$), and then do a sum pooling

$$\begin{aligned}\hat{\mathbf{x}} &= \max(\mathbf{X}) \\ &= \begin{bmatrix}\max(\mathbf{X}_{:,1}) & \max(\mathbf{X}_{:,2}) & \cdots & \max(\mathbf{X}_{:,K})\end{bmatrix} \\ &= \begin{bmatrix}\text{sum}(\ddot{\mathbf{X}}_{:,1}) & \text{sum}(\ddot{\mathbf{X}}_{:,2}) & \cdots & \text{sum}(\ddot{\mathbf{X}}_{:,K})\end{bmatrix} \\ &= \text{sum}(\ddot{\mathbf{X}}) \,.\end{aligned} \tag{3.22}$$

The maximum value masking operation is similar to a ReLU function. If we represent $\ddot{\mathbf{X}}$ using a ReLU function, we can obtain the masked matrix by setting all elements except the maximum value in each column to be negative. Specifically, we can achieve this by subtracting a value $\tilde{x}_i$ from each element in the column, where $\tilde{x}_i$ is between the largest value m_i and the second-largest value n_i in column i, for $i = 1, \ldots, K$. After the ReLU operation, since all the elements except the largest value are zeros in each column, we can multiply each column of the matrix by $\tilde{c}_i = \frac{m_i}{m_i - \tilde{x}_i}$ to get the maximum value back:

$$\hat{\ddot{\mathbf{X}}} = \tilde{\mathbf{c}} \circ \text{ReLU}(\mathbf{X} - \tilde{\mathbf{x}}) \,, \tag{3.23}$$

where $\tilde{\mathbf{c}} = [\frac{m_1}{m_1 - \tilde{x}_1} \frac{m_2}{m_2 - \tilde{x}_2}, \cdots, \frac{m_K}{m_K - \tilde{x}_K}]$ and $\tilde{\mathbf{x}} = [\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_K]$ are the multiple and deducted value in each column. It is worth noting that the choice of $\tilde{x}_i$ is flexible and can be any value between m_i and n_i, which makes $\tilde{c}_i = \frac{m_i}{m_i - \tilde{x}_i}$ vary. However, we can choose $\tilde{x}_i$ such that all the $\tilde{c}_i$ take the same value, say $\frac{1}{\tilde{c}}$, and feed this common value to the ReLU function to get

$$\hat{\ddot{\mathbf{X}}} = \text{ReLU}\left(\frac{\mathbf{X} - \tilde{\mathbf{x}}}{\tilde{c}}\right) \,. \tag{3.24}$$

The part inside the ReLU function can be viewed as a form of batch normalization. In total, the operation by column is equivalent to a sequence of max-normalization, which is parameterized by $\tilde{\mathbf{x}}$ and $\tilde{c}$, both of which depend on the input matrix $\mathbf{X}$. The max pooling, therefore, can be written as a ReLU function and a column-wise mean pooling function:

$$\hat{\mathbf{x}} = \max(\mathbf{X}) = \text{mean}\left(\text{ReLU}\left(\frac{\mathbf{X} - \tilde{\mathbf{x}}}{\tilde{c}}\right)\right) \,. \tag{3.25}$$

Compared to mean pooling, max pooling contains a weak normalization. As a result, if we add a max pooling to a linear PPE, then the entire network can be written as

$$\mathbf{y} = \max(\mathbf{A}\mathbf{x}_i + \mathbf{b}) = \text{mean}\left(\text{ReLU}\left(\frac{\mathbf{A}\mathbf{x}_i + \mathbf{b} - \tilde{\mathbf{x}}}{\tilde{c}}\right)\right) \,, \tag{3.26}$$

where $\tilde{\mathbf{x}}$ and $\tilde{c}$ are parameters that depend on $\mathbf{x}_i$. Therefore, max pooling works well even with a single-layer linear PPE. Specifically, it is equivalent to mean pooling with a linear normalization and ReLU function. After carefully investigating this simple example, we prove that max pooling has inherent nonlinearity that leads to better performance than mean pooling in MLPs. Even when we use multiple layers with ReLU functions, this nonlinearity helps increase the bandwidth of PPE; the performance is still slightly better than using mean pooling. However, max pooling is only applicable for an MLP PPE, specifically in PointNet. If a Positional Encoder is used, the PE function itself already contains nonlinearity and the bandwidth is already high enough. As a result, the advantage of max pooling is meaningless in PE-based PPEs.

3.6 Limitations

Our proposed PE-based PPEs have shown compelling robustness on noise and outlier corruptions. However, our method does not show superior advantages over learned PPEs with corruptions like density changes and transformations (see Tab. 3.4). We acknowledge that this is due to the limitations of the RFF-based PEs. For point density changes, a more specialized PE function is expected. For transformation corruptions, pre-processed data normalization is needed. The full experiments and analysis are provided in Figs. 3.10 and 3.11.Nevertheless, the focus of the chapter is to demonstrate that untrained PE is a robust point cloud embedding for OOD noise presented in data.

The quantitative classification results of different models tested under data corruptions other than in the main text are shown in Tab. 3.5.

We provide the full results of different models tested under different severities of different data corruptions on our modified dataset in Fig. 3.10. The full results of the original ModelNet-C Sun et al. [2022] dataset are shown in Fig. 3.11 and Fig. 3.12.

The visualization of our modified ModelNet40-C Sun et al. [2022] dataset with noise and outlier corruptions is demonstrated in Fig. 3.13.

3.7 Conclusion

In this chapter, we theoretically and empirically show that an untrained positional embedding can be a robust surrogate for learned per-point embedding in point cloud processing. Moreover, we show that the per-point embeddings should be both band-limited and smooth to maintain good robustness to OOD corruptions, where noise and outliers are presented in the data. One of the advantages of using these untrained PPEs is that the bandwidth and the smoothness of the embedding can be manually tuned to control the resistance to the OOD noise. We further demonstrate the downstream tasks, such as object classification and point cloud registration, through thorough experiments.

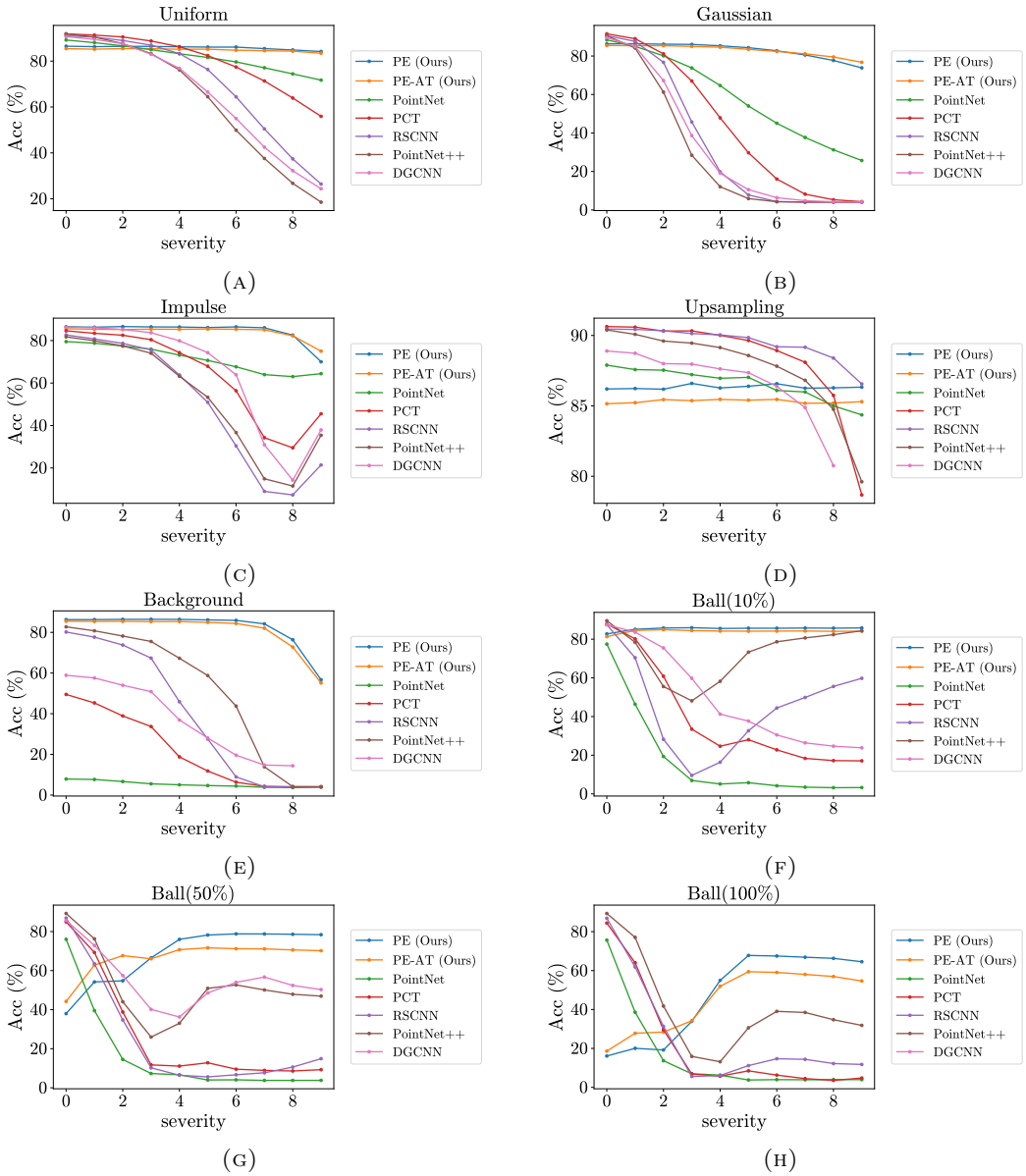


FIGURE 3.10. Classification accuracy of different methods on our data corruption dataset of various severity. As severity increases, corruption becomes stronger. However, the meaning of severity varies between corruption types, such as the variance or range of the noise and the number of outliers added. DGCNN fails to get reasonable performance when the number of input data is too large. Our proposed PE-based PPE methods maintain robust and good performance across almost all categories of data corruptions.

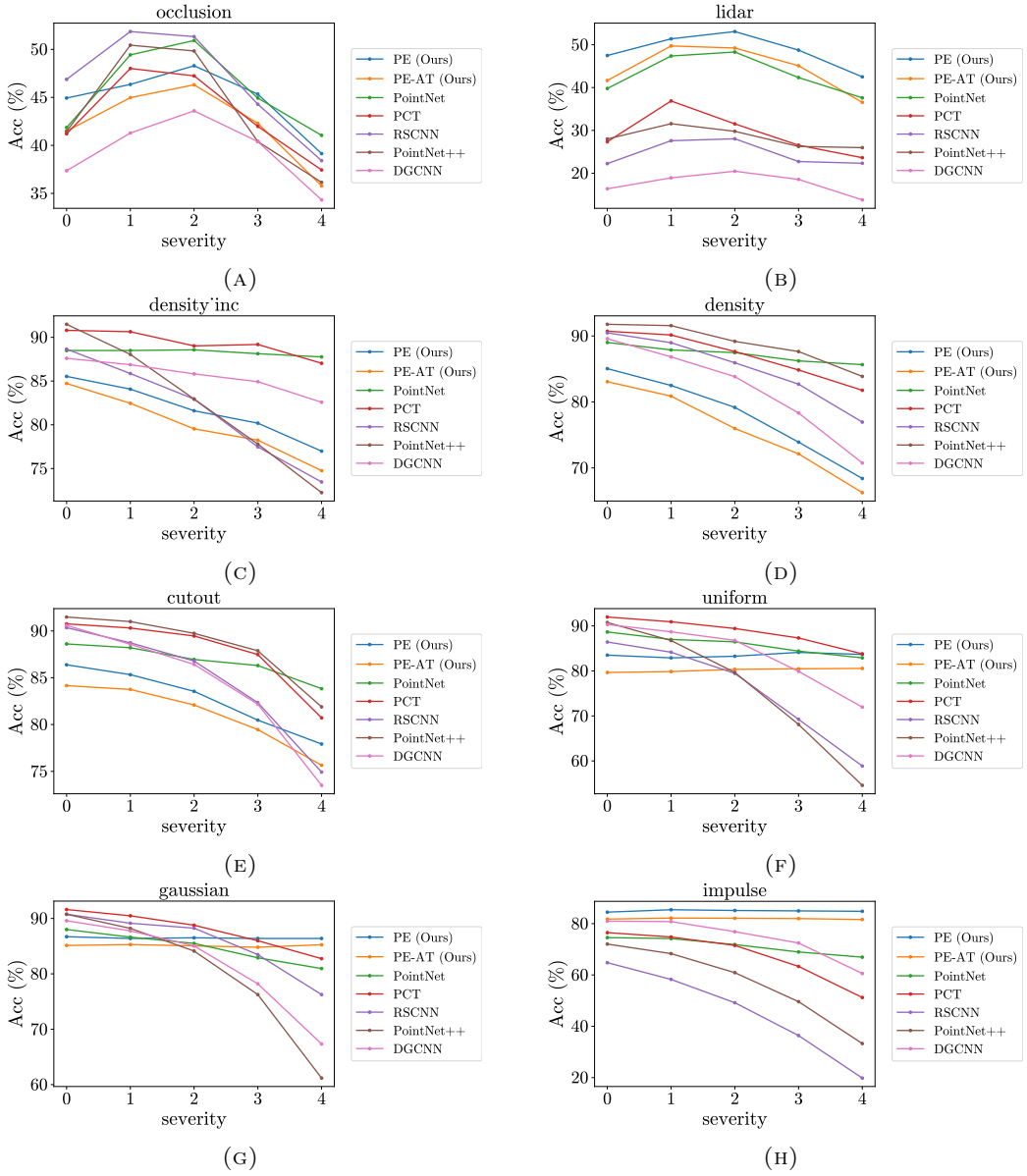


FIGURE 3.11. Classification accuracy on the original ModelNet40-C Sun et al. [2022] dataset. Continued in Fig. 3.12

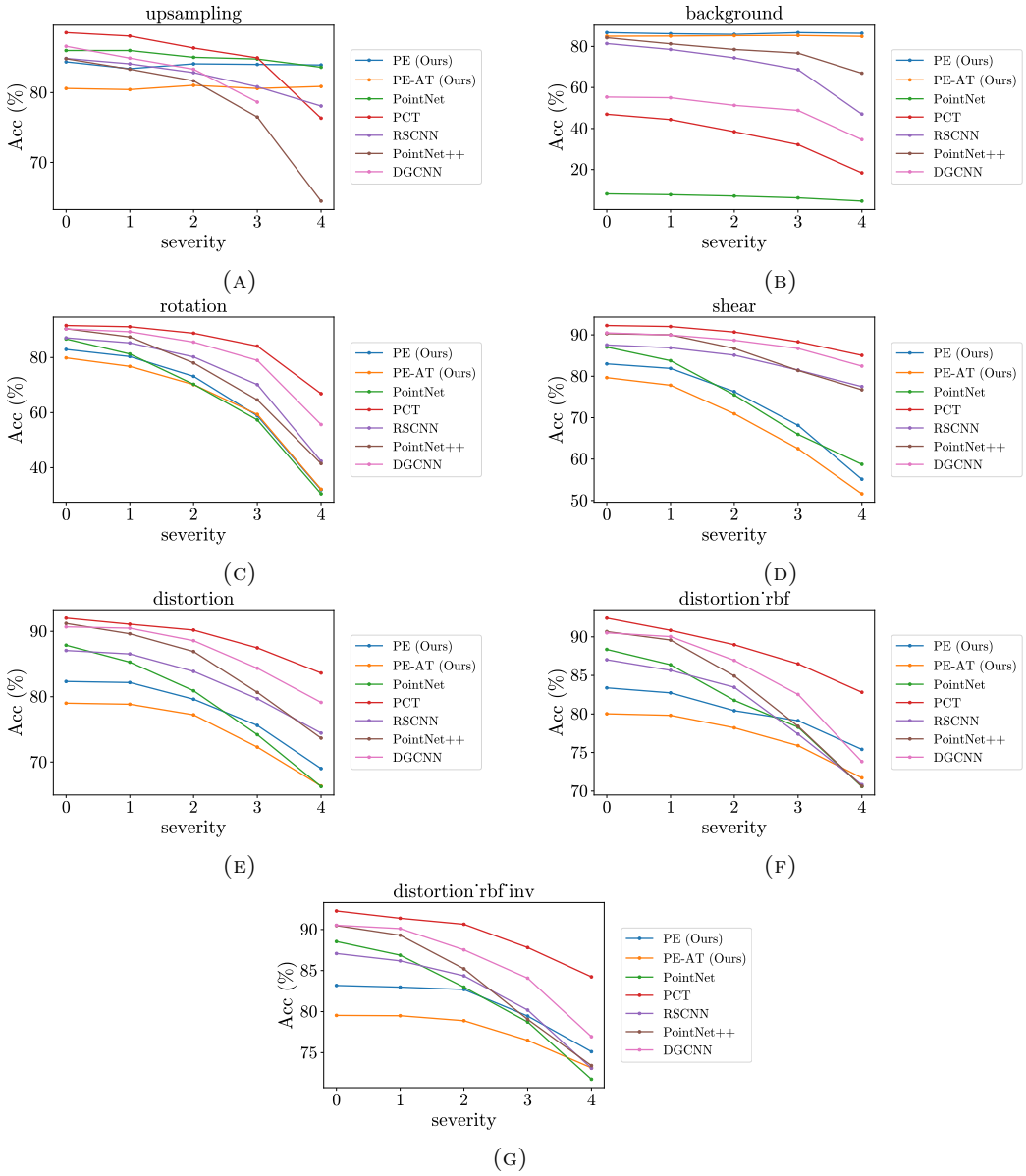
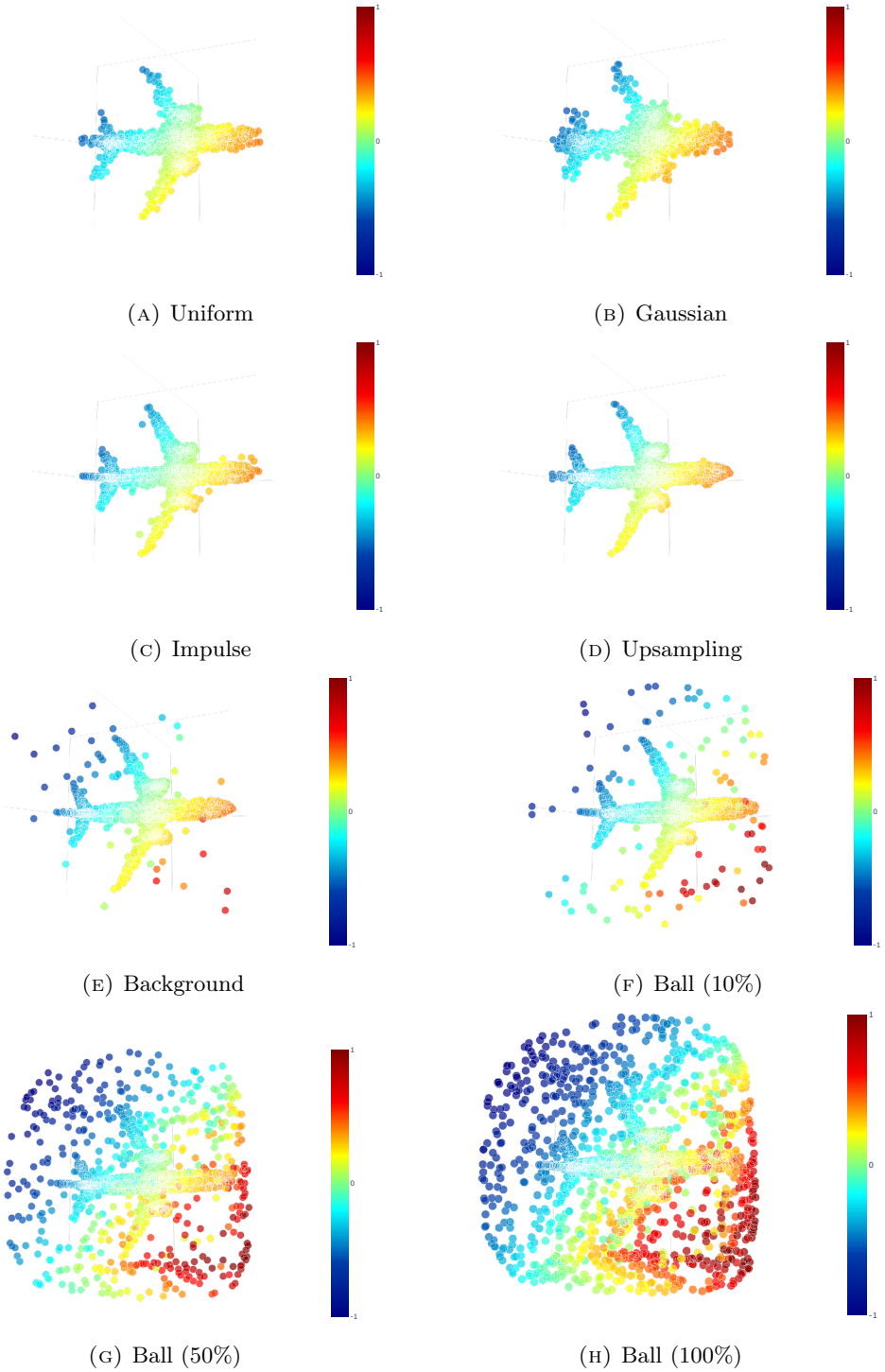


FIGURE 3.12. Continued figure of classification accuracy on the original ModelNet40-C Sun et al. [2022] dataset. As severity increases, corruption becomes stronger. Note that the severity is smaller than our modified dataset, which is less challenging. In this case, the clean part of the points (*e.g.*, the original points) may dominate the entire point cloud, yielding good performance for learned PPEs.



(A) Uniform (B) Gaussian

(C) Impulse (D) Upsampling

(E) Background (F) Ball (10%)

(G) Ball (50%) (H) Ball (100%)

FIGURE 3.13. Visualization of noise and outliers. Colors are fake depth for better 3D view.

Statement of Authorship

Title of Paper	All You Need is the First Layer: Rethinking Positional Encoding in MLPs
Publication Status	☐ Published ☐ Accepted for Publication ☐ Submitted for Publication ☒ Unpublished and Unsubmitted w ork w ritten in manuscript style
Publication Details	Almost finished. Will be on arxiv.

Principal Author

Name of Principal Author (Candidate)	Jianqiao Zheng			
Contribution to the Paper	Proposed ideas, conducted experiments, and wrote the manuscript.			
Overall percentage (%)	90%			
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			Date	June 3, 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 contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Hemanth Saratchandran			
Contribution to the Paper	Helped in discussion and writing revision.			
Signature			Date	09/07/2025

Name of Co-Author	Simon Lucey			
Contribution to the Paper	Supervised developement of work; Helped in discussion and writing revision.			
Signature			Date	23/6/25

Please cut and paste additional co-author panels here as required.

Chapter 4

All You Need is the First Layer: Rethinking Positional Encoding in MLPs

Preface

This chapter is based on a draft that has been adapted for inclusion in this thesis. Minor edits have been made for consistency in notation and formatting. The work in this chapter contributes to the thesis by establishing the relationship between the spectra of the Neural Tangent Kernel (NTK) and the Jacobian matrix, highlighting the importance of rank, and proposing a new initialization strategy derived from these insights.

4.1 Introduction

The spectral bias of ReLU-MLPs in implicit neural representations (INRs) has garnered significant attention in computer vision and machine learning. This bias often hampers the ability of these networks to accurately reconstruct low-dimensional signals, such as 2D images, 3D shapes, and 5D radiance fields. In response, researchers have proposed various architectural modifications, including positional encoding, non-traditional activation functions, and batch normalization, to mitigate these limitations.

Previous studies Sitzmann et al. [2020]; Tancik et al. [2020] have leveraged Neural Tangent Kernel (NTK) theory Jacot et al. [2018] to establish a theoretical connection between spectral bias and the singular value distribution of a network's Jacobian. While NTK analysis provides valuable insights, it offers only a global perspective and lacks the granularity needed to pinpoint the specific network components most responsible for this bias. In this paper, we address this limitation by introducing a more targeted framework that enables interventions beyond architectural modifications.

Specifically, we establish a theoretical connection between the spectral bias of INRs and the rank of hidden-layer embeddings. Our framework builds upon the NTK perspective to show that common interventions—such as positional encoding, non-traditional activation functions, and batch normalization—fundamentally serve the same purpose. We further demonstrate that these architectural modifications are unnecessary and that a similar effect can be achieved through a carefully chosen initialization, particularly in the first layer.

Our proposed initialization is inspired by prior work on the memorization of 1D signals using shallow ReLU networks. Unlike conventional approaches such as Xavier



(A) ReLU (default init) (B) ReLU (our init)

FIGURE 4.1. Reconstruction results of the ReLU-activated MLP using the default initialization compared to our proposed initialization. The PSNR of a ReLU-activated MLP can be improved significantly—from 24.14 (left) to 27.69 (right)—by modifying only the initialization of the first layer from the default scheme to our proposed initialization.

initialization, we show that in low-dimensional settings, the correct selection of bias is crucial for performance in ReLU-MLPs.

Applying NTK analysis to real-world networks is computationally expensive and memory-intensive. A key advantage of our approach is that it simplifies this analysis by shifting the focus from the full network Jacobian to the rank of hidden-layer embeddings. This refinement significantly reduces computational overhead and opens new avenues for applying NTK analysis to larger and more complex networks.

4.2 Preliminaries

4.2.1 MLP Network

An MLP network can be conceptually divided into three components Chen et al. [2024]; Sitzmann et al. [2020]; Yüce et al. [2022]: the input embedding, hidden layers, and output embedding. Let the input be denoted as $\mathbf{X} \in \mathbb{R}^{N \times P}$. The input embedding function $\phi : \mathbb{R}^{N \times P} \to \mathbb{R}^{N \times D}$ projects the input from the original dimension P (typically low) to a higher-dimensional hidden space D, as given by

$$\mathbf{Z}_0 = \phi(\mathbf{X}), \tag{4.1}$$

where ϕ may be instantiated as a linear layer followed by a ReLU activation, a sine-activated linear layer, a linear layer with ReLU and batch normalization (BN), or a positional encoding method such as Random Fourier Features.

The network then comprises L isotropic hidden layers with identical activation functions, defined as

$$\mathbf{Z}_i = \rho\left(\mathbf{Z}_{i-1}\mathbf{W}_i + \mathbf{b}_i\right), \quad i = 1, 2, \ldots, L, \tag{4.2}$$

where $\mathbf{W}_i \in \mathbb{R}^{D \times D}$ and $\mathbf{b}_i \in \mathbb{R}^{1 \times D}$ are trainable parameters, and ρ denotes a non-linear activation function, commonly chosen as ReLU, or more recently, sinusoidal or Gaussian functions.

Finally, the output embedding is a linear layer given by

$$\mathbf{Y} = \mathbf{Z}_L \mathbf{W}_o + \mathbf{b}_o, \tag{4.3}$$

where $\mathbf{W}_o \in \mathbb{R}^{D \times M}$ and $\mathbf{b}_o \in \mathbb{R}^{1 \times M}$ define a linear transformation that maps the final hidden representation $\mathbf{Z}_L$ from the high-dimensional space D to the output dimension M (typically lower). The resulting output $\mathbf{Y} \in \mathbb{R}^{N \times M}$ matches the dimensionality of the target signal.

4.2.2 Neural Tangent Kernel

The Neural Tangent Kernel (NTK) Jacot et al. [2018] provides a theoretical framework for understanding the training dynamics of neural networks and has gained significant attention, particularly for simple MLPs used in coordinate-based neural representations. The NTK is defined in terms of the network's Jacobian as follows:

$$\mathbf{K} = (\nabla_\theta f(\mathbf{X}; \theta))^\top \nabla_\theta f(\mathbf{X}; \theta), \tag{4.4}$$

where $f(\mathbf{X}; \theta)$ is the network output with parameters θ, and $\nabla_\theta f$ denotes the Jacobian with respect to the parameters.

When the network width tends to infinity and training is conducted using gradient descent with a sufficiently small learning rate η and an L_2 loss, the training dynamics can be approximated by a linear kernel regression model. In the discrete case, the convergence after n training steps is approximated as Lee et al. [2019]:

$$\mathbf{Y}^{(n)} - \hat{\mathbf{Y}} \approx (\mathbf{I} - \eta \mathbf{K})^n (\mathbf{Y}^{(0)} - \hat{\mathbf{Y}}), \tag{4.5}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{Y}^{(0)}$ is the initial output, and $\hat{\mathbf{Y}}$ is the target output. Since the NTK matrix admits an eigen-decomposition $\mathbf{K} = \mathbf{Q} \Lambda \mathbf{Q}^\top$, the approximation of the error becomes:

$$\mathbf{Y}^{(n)} - \hat{\mathbf{Y}} \approx \mathbf{Q}(\mathbf{I} - \eta \Lambda)^n \mathbf{Q}^\top (\mathbf{Y}^{(0)} - \hat{\mathbf{Y}}). \tag{4.6}$$

This formulation highlights that the initial error is projected into the eigenspace of $\mathbf{K}$ and subsequently decays along each eigendirection at a rate governed by the corresponding eigenvalues.

4.2.3 Appoximation of rank

The rank of a matrix plays a crucial role in characterizing the expressiveness of its column or row space. Suppose a matrix $A \in \mathbb{R}^{m \times n}$ has singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$. If all singular values are strictly greater than zero, the matrix is said to be of full rank. However, the mathematical definition of rank can be highly sensitive to numerical noise in practice. To address this, a threshold-based definition known as *numerical rank* Feng et al. [2008] is often employed:

$$\operatorname{nrank}_\epsilon(A) = \#\left\{i : \sigma_i > \epsilon\right\}. \tag{4.7}$$

This approach depends heavily on the choice of the threshold ϵ, which can lead to unstable results. To overcome these limitations, more robust measures such as the *stable rank* Rudelson and Vershynin [2007] and *effective rank* Roy and Vetterli [2007]

have been introduced:

$$\operatorname{srank}(A) = \frac{\|A\|_F^2}{\|A\|_2^2} = \frac{\sum_{i=1}^{\min(m,n)} \sigma_i^2}{\sigma_1^2}, \tag{4.8}$$

and

$$p_i = \frac{\sigma_i}{\sum_{j=1}^{\min(m,n)} \sigma_j}, \tag{4.9}$$

$$\operatorname{erank}(A) = \exp\left(- \sum_{i=1}^{\min(m,n)} p_i \log p_i \right). \tag{4.10}$$

4.3 From NTK to Jacobian

4.3.1 Definition of Jacobian

Based on the NTK theory discussed in Sec. 4.2.2, we understand that the eigenvalues of the Neural Tangent Kernel (NTK) are critical for the performance of MLPs. However, directly utilizing the NTK is computationally expensive in terms of both memory and time complexity.

To mitigate this, we shift our focus from analyzing the NTK kernel itself to examining the properties of the Jacobian matrix. This is feasible because the NTK kernel is essentially the Gram matrix of the Jacobian and thus shares the same eigenvectors and squared singular values.

Using the notation established in Sec. 4.2.1, we define the Jacobian matrix as:

$$\nabla_\theta f(\mathbf{X}; \theta) = \mathbf{J}_{\text{all}}^L := \frac{\partial \operatorname{vec}(\mathbf{Z}_L)}{\partial \operatorname{vec}(\mathbf{W}_{\text{all}})}, \tag{4.11}$$

where $\mathbf{W}_{\text{all}} := [\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_L]$, and the superscript and subscript of $\mathbf{J}_{\text{all}}^L$ indicate that it is the Jacobian of the output from the final layer L with respect to the weights of all layers.

Let the singular value decomposition (SVD) of $\mathbf{J}_{\text{all}}^L$ be given by $\mathbf{J}_{\text{all}}^L = \mathbf{U}\mathbf{S}\mathbf{V}^T$. Since the NTK is defined as $\mathbf{K} = \mathbf{J}^T\mathbf{J}$, we obtain:

$$\mathbf{Q}\Lambda\mathbf{Q}^T = \mathbf{K} = {\mathbf{J}_{\text{all}}^L}^T \mathbf{J}_{\text{all}}^L = \mathbf{V}\mathbf{S}^2\mathbf{V}^T. \tag{4.12}$$

This shows that the column space onto which the initialization error is projected coincides with the column space of $\mathbf{V}$, and the eigenvalues of the NTK (left) are simply the squared singular values of the Jacobian matrix (right). For simplicity, we normalize all singular values in this paper such that the largest singular value is 1, unless stated otherwise.

4.3.2 Relation to Performance through NTK

According to NTK theory and Eq. (4.6), the reconstruction error after n training epochs is governed by the eigenvalues of the NTK kernel. The learning rate η is typically chosen based on the largest eigenvalue to ensure convergence stability. Meanwhile, convergence of the remaining components is proportional to their relative eigenvalue magnitude.

Theoretically, if the NTK kernel is full rank (i.e., all eigenvalues are strictly positive), then the model can perfectly fit any training signal, given sufficient training time. In practice, however, the number of training epochs is limited, so only those eigenvalues exceeding a relatively high threshold (e.g., 10^{-2}, which is picked through the convergence under 2000 epochs) contribute meaningfully to convergence within a reasonable timeframe, as discussed in Lucey [2025].

A common way to characterize this is via the "numerical rank" of a matrix, which counts the number of singular values greater than a given threshold. In this paper, unlike the conventional use of a small threshold (e.g., 10^{-12}), we use a larger threshold (e.g., 10^{-2}) to reflect practical training dynamics. This motivates our use of quotation marks around "numerical rank" above, though for simplicity we will drop the quotes in the remainder of this paper. Thus we have the following proposition 9

Proposition 9. *The number of singular values of the Jacobian matrix exceeding a given threshold (e.g., 10^{-2}), referred to as the "numerical rank", determines the number of eigenvalues of the NTK kernel exceeding the square of that threshold (e.g., 10^{-4}), and thus reflects the expressivity of the network.*

In other words, the parameters contributing to this rank can be considered *effective parameters*, as they drive learning dynamics. The remaining parameters are effectively redundant and do not enhance the expressivity of the network.

4.3.3 Decomposition of Jacobian into Layers

The analysis of the Jacobian matrix can be decomposed into multiple steps to facilitate understanding and computation. First, the Jacobian of the network output with respect to all weights, denoted as $\mathbf{J}_{\text{all}}^L$, can be expressed as the concatenation of the Jacobians with respect to the weights of each individual layer:

$$\mathbf{J}_{\text{all}}^L = \begin{bmatrix} \mathbf{J}_L^L & \mathbf{J}_{L-1}^L & \cdots & \mathbf{J}_1^L \end{bmatrix}, \tag{4.13}$$

where the superscript and subscript of each $\mathbf{J}_k^L$ denote that it is the Jacobian of the output from layer L with respect to the weights in the k-th layer, i.e.,

$$\mathbf{J}_k^L := \frac{\partial \operatorname{vec}(\mathbf{Z}_L)}{\partial \operatorname{vec}(\mathbf{W}_k)}. \tag{4.14}$$

Each $\mathbf{J}_k^L$ has the same number of rows as $\operatorname{vec}(\mathbf{Z}_L)$, and its number of columns equals the number of parameters in $\mathbf{W}_k$. Thus, $\mathbf{J}_{\text{all}}^L$ is a horizontal concatenation of these matrices.

Proposition 10. *The (numerical) rank of the full Jacobian matrix $\mathbf{J}_{all}^L$ is bounded as follows:*

$$\max_{k=1,\ldots,L} \operatorname{rank}(\mathbf{J}_k^L) \;\le\; \operatorname{rank}(\mathbf{J}_{all}^L) \;\le\; \sum_{k=1}^{L} \operatorname{rank}(\mathbf{J}_k^L), \tag{4.15}$$

and similarly, for the numerical rank with a threshold ϵ,

$$\min_{k=1,\ldots,L} \operatorname{nrank}_\epsilon(\mathbf{J}_k^L) \;\le\; \operatorname{nrank}_\epsilon(\mathbf{J}_{all}^L) \;\le\; \sum_{k=1}^{L} \operatorname{nrank}_\epsilon(\mathbf{J}_k^L). \tag{4.16}$$

Discussion.

The lower bound for the (exact) rank arises from the fact that the concatenated matrix must at least preserve the number of independent columns from the single most expressive layer. The upper bound comes from the assumption that no singular directions overlap across layers, resulting in additive rank contributions.

For the numerical rank, the situation is subtler. Although one might expect the maximum of the component ranks to serve as a lower bound, this is not always true. If the components $\mathbf{J}_k^L$ align heavily along the same dominant singular directions, the second or third largest singular values might fall below the threshold, reducing the overall numerical rank. Hence, we use the minimum as a conservative lower bound for numerical rank.

This decomposition and bounding technique allows us to analyze the structure and expressivity of the full Jacobian matrix $\mathbf{J}_{\text{all}}^L$ by studying smaller and more manageable submatrices $\mathbf{J}_k^L$. This insight will be useful in subsequent sections where we consider layer-wise initialization and activation strategies.

4.3.4 Decomposition of Jacobian into Weights and Embeddings

Proposition 11. *Matrix multiplication in each layer can be reformulated (ignoring bias) as:*

$$\text{vec}(\mathbf{Z}_i) = (\mathbf{I}_D \otimes \mathbf{Z}_{i-1})\text{vec}(\mathbf{W}_i) = (\mathbf{W}_i^\top \otimes \mathbf{I}_N)\text{vec}(\mathbf{Z}_{i-1}), \tag{4.17}$$

so the Jacobians for each linear layer are:

$$\begin{aligned} \frac{\partial \mathbf{Z}_i}{\partial \mathbf{Z}_{i-1}} &= \mathbf{W}_i^\top \otimes \mathbf{I}_N, \\ \frac{\partial \mathbf{Z}_i}{\partial \mathbf{W}_{i-1}} &= \mathbf{I}_D \otimes \mathbf{Z}_{i-1}. \end{aligned} \tag{4.18}$$

This formulation allows us to compute the Jacobian of the output with respect to the weights of each layer via the chain rule.

Corollary 1. *For a **linear** network without any nonlinear activation (i.e., ρ is the identity function), the Jacobian is:*

$$\begin{aligned} \mathbf{J}_k^L = \frac{\partial \mathbf{Z}_L}{\partial \mathbf{W}_k} &= \left(\prod_{i=L}^{k+1} \frac{\partial \mathbf{Z}_i}{\partial \mathbf{Z}_{i-1}} \right) \frac{\partial \mathbf{Z}_k}{\partial \mathbf{W}_k} \\ &= \left(\prod_{i=L}^{k+1} (\mathbf{W}_i \otimes \mathbf{I}_N) \right) (\mathbf{I}_D \otimes \mathbf{Z}_{k-1}) \\ &= (\mathbf{W}_L \otimes \mathbf{I}_N) \cdots (\mathbf{W}_{k+1} \otimes \mathbf{I}_N)(\mathbf{I}_D \otimes \mathbf{Z}_{k-1}) \\ &= \left(\mathbf{I}_D \prod_{i=L}^{k+1} \mathbf{W}_i \right) \otimes \mathbf{Z}_{k-1}. \end{aligned} \tag{4.19}$$

Lemma 2. *Suppose $C = A \otimes B$, and let the singular value decompositions be given by: $A = U_A S_A V_A^\top$, $B = U_B S_B V_B^\top$, and $C = U_C S_C V_C^\top$. Then, by properties of the Kronecker product:*

$$\begin{aligned} U_C &= U_A \otimes U_B, \\ S_C &= S_A \otimes S_B, \\ V_C &= V_A \otimes V_B, \\ \text{rank}(C) &= \text{rank}(A) \cdot \text{rank}(B). \end{aligned} \tag{4.20}$$

From lemma 2 and corollary 1, we derive:

$$\begin{aligned} U_{\mathbf{J}_k^L} &= U_{\mathbf{P}_k} \otimes U_{\mathbf{Z}_{k-1}}, \\ S_{\mathbf{J}_k^L} &= S_{\mathbf{P}_k} \otimes S_{\mathbf{Z}_{k-1}}, \\ V_{\mathbf{J}_k^L} &= V_{\mathbf{P}_k} \otimes V_{\mathbf{Z}_{k-1}}, \\ \operatorname{rank}(\mathbf{J}_k^L) &= \operatorname{rank}(\mathbf{P}_k) \cdot \operatorname{rank}(\mathbf{Z}_{k-1}), \\ \operatorname{nrank}(\mathbf{J}_k^L) &\leq \operatorname{nrank}(\mathbf{P}_k) \cdot \operatorname{nrank}(\mathbf{Z}_{k-1}), \\ \operatorname{nrank}(\mathbf{J}_k^L) &\geq \min(\operatorname{nrank}(\mathbf{P}_k), \operatorname{nrank}(\mathbf{Z}_{k-1})), \end{aligned} \tag{4.21}$$

where $\mathbf{P}_k := \mathbf{I}_D \prod_{i=L}^{k+1} \mathbf{W}_i$. While the numerical rank differs from the strict Kronecker rank property, it still offers a meaningful characterization of the Jacobian's structure in relation to the embeddings.

Corollary 2. *For a network with element-wise nonlinear activations (e.g., ReLU, sinusoidal, or Gaussian functions), the Jacobian becomes:*

$$\begin{aligned} \mathbf{J}_k^L = \frac{\partial \mathbf{Z}_L}{\partial \mathbf{W}_k} &= \left(\prod_{i=L}^{k+1} \frac{\partial \mathbf{Z}_i}{\partial \mathbf{Z}_{i-1}} \right) \frac{\partial \mathbf{Z}_k}{\partial \mathbf{W}_k} \\ &= \left(\prod_{i=L}^{k+1} \mathbf{D}_i^A (\mathbf{W}_i \otimes \mathbf{I}_N) \right) \left(\mathbf{D}_k^A (\mathbf{I}_D \otimes \mathbf{Z}_{k-1}) \right), \end{aligned} \tag{4.22}$$

where $\mathbf{D}_i^A := \operatorname{diag}(\operatorname{vec}(\mathbf{Z}_i > 0))$ *is a diagonal matrix capturing the gradient of the activation function.*

In corollary 2, the superscript A denotes "activation". Since the activation function is applied element-wise, $\mathbf{D}_i^A$ cannot be expressed as a Kronecker product, though its binary structure (e.g., 0 or 1 for ReLU) remains interpretable.

Corollary 3. *For a network using both ReLU activation and Batch Normalization, the Jacobian becomes:*

$$\begin{aligned} \mathbf{J}_k^L = \frac{\partial \mathbf{Z}_L}{\partial \mathbf{W}_k} &= \left(\prod_{i=L}^{k+1} \frac{\partial \mathbf{Z}_i}{\partial \mathbf{Z}_{i-1}} \right) \frac{\partial \mathbf{Z}_k}{\partial \mathbf{W}_k} \\ &= \left(\prod_{i=L}^{k+1} (\mathbf{D}_i^B \otimes \mathbf{I}_N) \mathbf{D}_i^A (\mathbf{W}_i \otimes \mathbf{I}_N) \right) \left((\mathbf{D}_k^B \otimes \mathbf{I}_N) \mathbf{D}_k^A (\mathbf{I}_D \otimes \mathbf{Z}_{k-1}) \right), \end{aligned} \tag{4.23}$$

.

where $\mathbf{D}_i^B := \operatorname{diag}(\gamma / \text{running_var})$ *encodes the gradient of the Batch Normalization operation.*

In corollary 3, the superscript B denotes "batch normalization". Since batch normalization operates uniformly across spatial locations, the associated diagonal matrix $\mathbf{D}_i^B$ can also be expressed using Kronecker structure in some cases.

4.4 Why the Rank of Embeddings in the First Layer

Random weights are naturally well-conditioned.

Recall from Eq. (4.15) that the numerical rank of the overall Jacobian $\mathbf{J}^L$ directly influences the expressivity and trainability of the network. This rank is bounded by

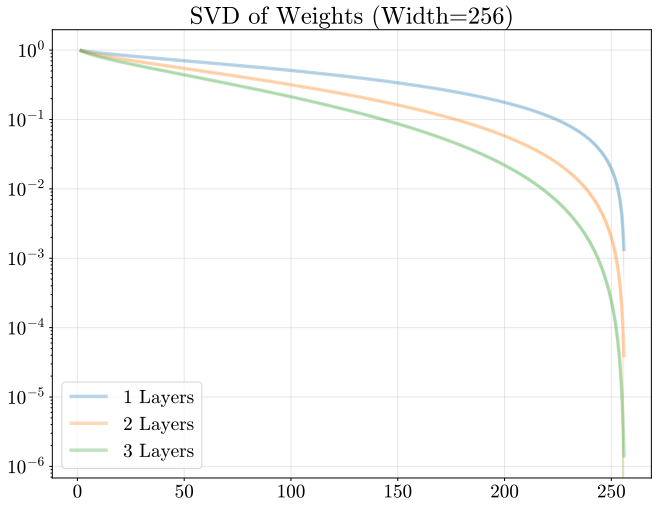


FIGURE 4.2. Singular values of weights $\mathbf{P}_k = \mathbf{I}_D \prod_{i=L}^{k+1} \mathbf{W}_i$ for $k = 1, 2, 3$. Most singular values are well above a small threshold (e.g., 10^{-2}), indicating that these matrices are well-conditioned.

the numerical ranks of the intermediate Jacobians $\mathbf{J}_k^L$. Due to the Kronecker product structure of these Jacobians from Eq. (4.21), the rank of each $\mathbf{J}_k^L$ is further bounded by the product of the ranks of the associated weight matrices and input embeddings:

$$\begin{aligned} \operatorname{nrank}(\mathbf{J}_{all}^L) &\geq \min_k \operatorname{nrank}(\mathbf{J}_k^L) \geq \min_k \left(\min\left(\operatorname{nrank}(\mathbf{P}_k), \operatorname{nrank}(\mathbf{Z}_{k-1})\right)\right), \\ \operatorname{nrank}(\mathbf{J}_{all}^L) &\leq \sum_k \operatorname{nrank}(\mathbf{J}_k^L) \leq \sum_k \left(\operatorname{nrank}(\mathbf{P}_k) \cdot \operatorname{nrank}(\mathbf{Z}_{k-1})\right). \end{aligned} \tag{4.24}$$

Here, $\mathbf{P}_k = \mathbf{I}_D \prod_{i=L}^{k+1} \mathbf{W}_i$ denotes the composition of weights from layer $k+1$ to L. As shown in Fig. 4.2, these matrices, when initialized with Gaussian random values, typically exhibit high numerical rank and good conditioning. In contrast, the embedding matrices $\mathbf{Z}_k$, particularly $\mathbf{Z}_0$ formed from raw input coordinates, are often rank-deficient because the input space (e.g., 2D or 3D coordinates) is intrinsically low-dimensional. As a result, we can immediately get $\operatorname{nrank}(\mathbf{P}_k) \approx D$ and $\min\left(\operatorname{nrank}(\mathbf{P}_k), \operatorname{nrank}(\mathbf{Z}_{k-1})\right) = \operatorname{nrank}(\mathbf{Z}_{k-1})$, so we have,

$$\min_k \left(\operatorname{nrank}(\mathbf{Z}_{k-1})\right) \leq \operatorname{nrank}(\mathbf{J}_{all}^L) \leq D \sum_k \left(\operatorname{nrank}(\mathbf{Z}_{k-1})\right). \tag{4.25}$$

This implies that the primary bottleneck in the overall Jacobian—and thus in the neural tangent kernel (NTK)—often lies in the embedding rank, not the weight matrices. As a result, improving or analyzing the rank of embeddings, especially in early layers, provides a more tractable and insightful approach than analyzing the full NTK. Unlike NTK computation, which is expensive (quadratic or worse in data size), evaluating embedding rank is computationally efficient—roughly square root in scale—and strongly indicative of learning capacity.

Thought experiment on capacity and parameter efficiency.

Given that the NTK condition and overall trainability of a coordinate MLP are strongly tied to the rank of its embeddings, we consider a thought experiment to illustrate how embedding rank governs effective model capacity.

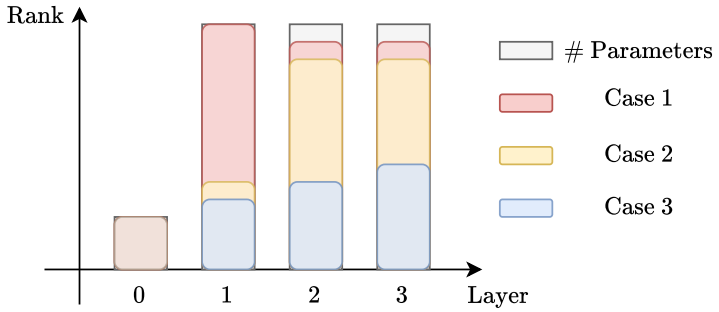


FIGURE 4.3. Thought experiment on the capacity of a coordinate MLP. Layer 0 represents the raw input coordinates. The bars for Layer 1 to Layer 3 depict the numerical rank of the embeddings at each layer. All cases share the same number of parameters (gray background), yet their effective capacities differ. In Case 1, the embeddings have nearly full rank throughout, so the capacity is close to the parameter count. In Cases 2 and 3, rank deficiency in early embeddings limits the network's effective capacity, wasting later parameters.

Imagine multiple MLPs with identical architectures and parameter counts, but differing in the rank of their internal embeddings, as shown in Fig. 4.3. Although traditional notions of capacity focus on the number of parameters, this view neglects the structural bottlenecks caused by low-rank embeddings.

For instance, in Case 3, a ReLU MLP with raw coordinate inputs exhibits a significant mismatch between embedding dimensionality and rank. While the embedding layer increases dimensionality, the actual rank remains low—constrained by the low dimensionality of the input coordinates—thus leading to poor expressivity and underutilized parameters.

In contrast, Case 1 demonstrates a scenario where embeddings maintain near full rank throughout the network. Here, the capacity closely matches the parameter count, and the model can fully exploit its expressiveness. In fact, many commonly used techniques—such as positional encoding, batch normalization, and alternative activation functions—aim to address this rank deficiency and effectively fall into Case 1. However, they differ in how and where they influence the network. Batch normalization and alternative activations are global methods; they modify the behavior of multiple layers, including the isotropic ones. In contrast, positional encoding specifically targets the input layer, enhancing the rank of the initial embedding only. The subsequent layers remain standard ReLU layers, which neither increase nor decrease the rank significantly.

This observation suggests that while ReLU activations are not effective at increasing rank, they also preserve it—making them neutral in terms of rank propagation. Thus, if one manages to increase the embedding rank at any intermediate layer (as illustrated in Case 2 of Fig. 4.3), then all layers after that point can be fully effective, while the layers before remain underutilized.

This leads us to a key insight about MLP capacity, summarized in remark 2.

Remark 2. *The effective capacity of a coordinate MLP is governed not by its parameter count alone, but by the numerical rank of its embeddings, which bounds the rank of the Jacobian and the NTK kernel.*

Experimental evidence.

To validate Case 2 of the thought experiment, we conduct an experiment using four 3-layer ReLU MLPs to represent a grayscale image. One model is a plain ReLU MLP with no batch normalization, while the other three each include a single batch normalization (BN) layer inserted at Layer 1, 2, or 3, respectively.

Batch normalization is introduced here as a tool to increase the embedding rank at a specific layer. The reconstruction results, NTK spectra, and SVDs of the embeddings from each layer are shown in Fig. 4.4.

As expected from NTK theory, the overall NTK spectrum correlates well with the PSNR of the reconstruction. Moreover, our decomposition reveals that the NTK spectrum is largely determined by the embedding ranks at each layer. The plain ReLU MLP—with poor NTK spectrum and low-rank embeddings across all layers—fails to represent the image accurately, even though it is overparameterized. This clearly demonstrates that the true capacity of a coordinate MLP is not governed by the number of parameters, but rather by the conditioning of its NTK, which in turn is shaped by the embedding ranks.

When batch normalization is applied at Layer 3, only the embedding rank of that layer increases—resulting in a modest improvement in both NTK spectrum and reconstruction performance. Moving the BN layer to Layer 2 improves the embedding rank of both Layers 2 and 3, and further boosts performance. Finally, placing batch normalization at Layer 1 increases the embedding rank across all layers, leading to the best NTK spectrum and reconstruction PSNR—despite no normalization being applied in later layers.

This experiment empirically validates the thought experiment: the effective capacity of a coordinate MLP stems from the NTK condition, not simply the parameter count. Furthermore, it reinforces that ReLU layers tend to preserve rank rather than increase or degrade it. This implies that global methods such as alternative activation functions or batch normalization may work effectively even when applied only at the first layer, as further supported by results in Tab. 4.1.

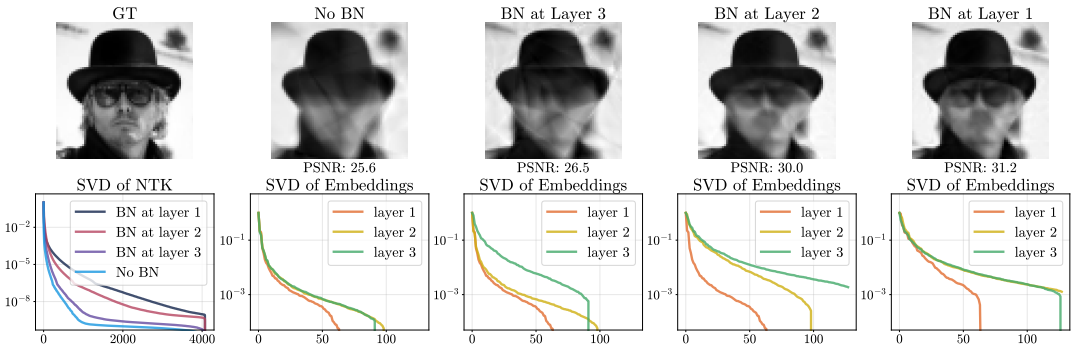


FIGURE 4.4. Image representation results of a 3-layer ReLU-MLP with either no batch normalization or batch normalization applied at Layer 1, 2, or 3 individually. The NTK spectrum strongly correlates with reconstruction quality (measured by PSNR), and is governed by the embedding rank at each layer. Batch normalization boosts rank at the inserted layer and improves performance accordingly.

Remark 3. *The primary reason for the effectiveness of coordinate MLP techniques—such as positional encoding, batch normalization, and alternative activation functions—is*

their ability to increase the rank of the input embedding after the initial dimension expansion. This rank enhancement in the input embedding layer is critical, while similar interventions in later (isotropic) layers tend to have a less significant effect.

Remark 4. *If the embedding input to a given layer is ill-conditioned (i.e., its singular values decay rapidly), the weights in that layer will be difficult to optimize and are likely to remain close to their initialization during training.*

4.5 Our Initialization Strategy

Following the insight in remark 3, the most problematic part of a ReLU MLP is often the input embedding layer. The input dimension increases dramatically, the resulting embedding is typically low-rank, and this layer contains significantly fewer parameters than subsequent layers. These factors make standard ReLU MLPs fail in coordinate-based tasks.

Instead of modifying the architecture (e.g., through positional encoding, batch normalization, or alternative activations), we propose to increase the embedding rank through a carefully designed initialization of the weights and biases in the input ReLU layer. Specifically, we define:

$$\mathbf{Z}_0 = \phi(\mathbf{X}) = \mathrm{ReLU}(\mathbf{X}\mathbf{W}^* + \mathbf{b}^*), \tag{4.26}$$

where the input $\mathbf{X} \in \mathbb{R}^{N\times P}$ (P is usually as small as 1,2, or 6 as coordinates), and the problem becomes how to find the ideal initialization $\mathbf{W}^* \in \mathbb{R}^{P\times D}$ and $\mathbf{b}^* \in \mathbb{R}^{1\times D}$.

4.5.1 1-D Case

In the 1-D case, the number of input coordinates increases linearly with resolution, so the embedding dimension can be similar to the number of inputs. Previous works have explored how to construct weights and biases so that $\mathbf{Z}_0$ has full rank. We adopt a simple approach from Zhang et al. [2017]:

Proposition 12. *Let a ReLU linear layer take 1-D coordinate inputs* $\mathbf{x}=[x_1, x_2, \ldots, x_N]$, *where* $x_1 < x_2 < \cdots < x_N$. *Define the weight vector* $\mathbf{W}^* = \mathbf{1}$ *and the bias vector* $\mathbf{b}^* = [-x_1 + \epsilon, \ldots, -x_N + \epsilon]$, *where* ϵ *is a small positive constant. Then the output* $\mathbf{Z}_0 = ReLU(\mathbf{X}\mathbf{W}^* + \mathbf{b}^*)$ *has full rank, as illustrated in Fig. 4.5.*



FIGURE 4.5. Illustration of initialization in the 1-D case. The key idea is to use biases that separate the activation thresholds across different input coordinates, ensuring a full-rank ReLU activation matrix.

4.5.2 2-D Case

Inspired by proposition 12, we extend the idea to the 2-D case, where the number of input coordinates grows quadratically with resolution and the embedding dimension is usually much smaller than the number of input points.

Proposition 13. *Let a ReLU linear layer take normalized 2-D coordinates* $\mathbf{X} \in [-1, 1]^2$*, and let the embedding dimension be* $D = p^2$*. Define a set of grid points:*

$$G_{p(i-1)+j} = \left[\frac{2i}{p+1} - 1, \ \frac{2j}{p+1} - 1 \right], \tag{4.27}$$

for $i, j = 1, \ldots, p$*. Then set:*

$$\mathbf{W}^*_{p(i-1)+j} = \frac{G_{p(i-1)+j}}{\|G_{p(i-1)+j}\|}, \quad \mathbf{b}^*_{p(i-1)+j} = -\|G_{p(i-1)+j}\| + \epsilon, \tag{4.28}$$

where ϵ *is a small positive constant. This initialization is a sufficient condition to ensure that the embedding* $\mathbf{Z}_0$ *is full rank (see Fig. 4.6).*

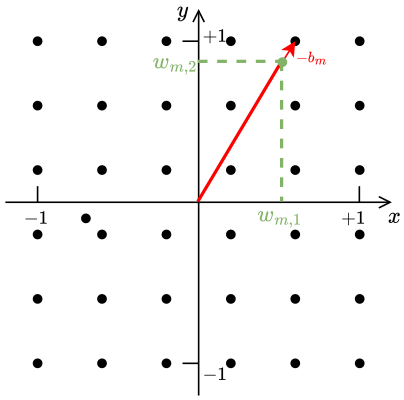


FIGURE 4.6. Illustration of our initialization in the 2-D case. Each black point represents a grid location G, which defines a weight vector (via normalization) and a corresponding bias (the negative norm plus ϵ). This initialization spreads ReLU activation thresholds across space, promoting a high-rank embedding.

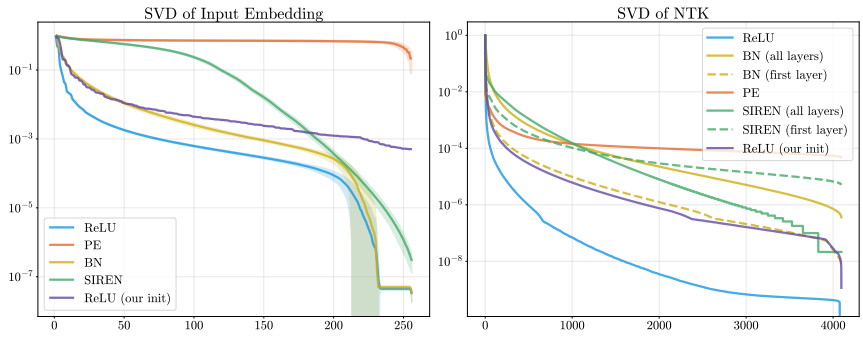
4.6 Experiments

4.6.1 Rank of Input Embedding

In Sec. 4.5, we argue that our initialization strategy can increase the rank of the input embedding without requiring any architectural modifications. To validate this, we conduct experiments using 256×256 coordinate grids as input, with hidden layer dimensions also set to 256. We compare several configurations: a ReLU layer with default initialization, our proposed initialization, a ReLU layer with batch normalization (using default initialization), Random Fourier Features (as a positional encoding method), and a SIREN layer. The results are presented in Fig. 4.7.

In the left plot, our initialization (purple line) significantly increases the rank of the input embedding compared to the default ReLU initialization (blue line). In the right plot, a ReLU-MLP using our initialization also exhibits a much-improved overall NTK spectrum compared to the default setting.

These results not only demonstrate the effectiveness of our initialization in boosting the embedding rank but also highlight a strong correlation between the rank of the input embedding and the overall NTK spectrum.



(A) SVD of embeddings (B) SVD of NTK

FIGURE 4.7. (A) Comparison of the rank of input embeddings with various methods, which proves that our initialization method can increase the rank of the input embedding. Shadow areas are the standard deviation over 100 runs. (B) Comparison of SVD of NTK kernel with various methods. By increasing the rank of the first layer output, our method can also improve the spectrum of the NTK kernel.

TABLE 4.1. 2-D image representation PSRN of a collection of 16 images from DIV2K Agustsson and Timofte [2017]. "BN" stands for batch normalization. "first" means the method is only used in the first layer, and the rest of the layers are normal ReLU-MLPs.

ReLU	PE	SIREN	SIREN (first)	BN (first)	BN	ReLU (our init)
23.57	27.15	27.29	26.15	25.82	25.04	25.61

4.6.2 2-D Image Representation

We evaluate our initialization strategy on a 2-D image representation task, which is a common benchmark for implicit neural representations (INRs). In Tab. 4.1, we report the average reconstruction PSNR over a collection of 16 images from the DIV2K dataset Agustsson and Timofte [2017], comparing various techniques.

The results show that for global methods such as SIREN (with sine activation) and batch normalization, applying the technique solely to the first layer yields comparable performance to applying it across all layers. Notably, our method—without introducing any architectural changes and simply modifying the initialization of the first layer—achieves a substantial improvement of approximately 2 dB in PSNR.

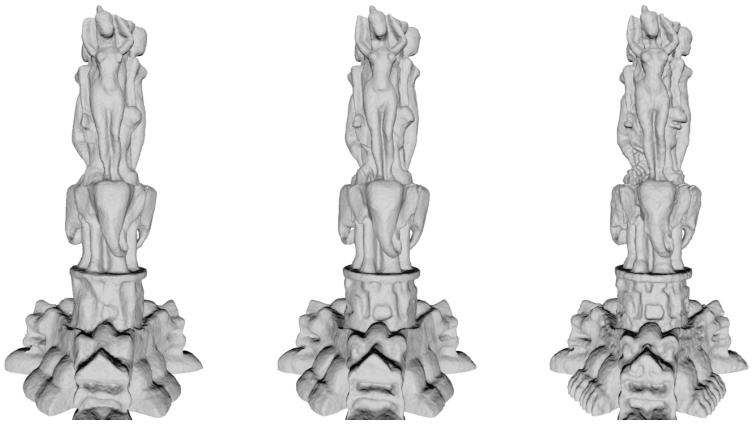
4.6.3 3-D Occupancy Representation

We further evaluate our initialization strategy on a 3-D occupancy representation task. As shown in Fig. 4.8, a ReLU-MLP with default initialization fails to reconstruct a high-quality mesh of the shape. In contrast, applying our initialization only to the first layer enables the ReLU-MLP to achieve both visually and quantitatively competitive performance, comparable to state-of-the-art techniques.

Consistent with our findings in the 2-D image task, global methods such as SIREN (with sine activation) or batch normalization exhibit similar performance whether applied solely to the first layer or throughout the entire network.



(A) ReLU (default): 90.98 (B) PE: 97.50 (C) SIREN: 96.36 (D) SIREN (first layer): 97.01



(E) BN: 95.17 (F) BN (first layer): 96.15 (G) ReLU (our init): 95.75

FIGURE 4.8. Visualization of meshes generated with occupancy volumes with various methods.

4.7 Conclusion

In this chapter, we establish a theoretical connection between the Neural Tangent Kernel (NTK) and the rank of intermediate embeddings across layers in ReLU-based MLPs. Our analysis reveals that the key bottleneck in standard coordinate-based MLPs lies in the rank deficiency of embeddings produced by the first layer. We further demonstrate that the success of various existing techniques—including positional encoding, alternative activation functions, and batch normalization—can be attributed primarily to their ability to increase the rank of these initial embeddings.

Motivated by this insight, we propose a simple yet effective initialization strategy specifically for the first layer of MLPs. This strategy enhances the embedding rank without altering the network architecture, thereby enabling vanilla ReLU-MLPs to achieve significantly improved performance. Our findings not only shed light on the fundamental limitations of coordinate MLPs but also point to a promising new direction for improving their performance in both 2D and 3D applications. Moreover, this work highlights the critical role of initialization in determining the expressive capacity of MLPs.

Statement of Authorship

Title of Paper	Structured Initialization for Vision Transformers
Publication Status	☐ Published ☐ Accepted for Publication ☒ Submitted for Publication ☐ Unpublished and Unsubmitted work written in manuscript style
Publication Details	Submitted to NeurIPS 2025

Principal Author

Name of Principal Author (Candidate)	Jianqiao Zheng		
Contribution to the Paper	Proposed ideas, conducted experiments, and wrote the manuscript.		
Overall percentage (%)	80%		
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		Date	June 3, 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 contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Xueqian Li		
Contribution to the Paper	Helped in discussion and writing revision.		
Signature		Date	June 7, 2025

Name of Co-Author	Hemanth Saratchandran		
Contribution to the Paper	Helped in discussion and writing revision.		
Signature		Date	09/06/2025

Please cut and paste additional co-author panels here as required.

Name of Co-Author	Simon Lucey			
Contribution to the Paper	Supervised developement of work; Helped in discussion and writing revision.			
Signature			Date	23/6/25

Chapter 5

Structured Initialization for Vision Transformers

Preface

This chapter is based on the following paper, which has been accepted at NeurIPS 2025:

> Zheng, Jianqiao, Xueqian Li, Hemanth Saratchandran, and Simon Lucey. *"Structured Initialization for Vision Transformers."* In: *Advances in Neural Information Processing Systems (NeurIPS 2025)*, to appear.

This paper has been adapted for inclusion in this thesis. Minor edits have been made for consistency in notation and formatting. The work in this chapter contributes to the thesis by demonstrating that *rank* also plays an important role in Transformers, particularly in the structure of attention maps.

5.1 Background

Studying rank in deep models such as convolutional neural networks (CNNs) and Transformers is challenging due to their large number of parameters, which makes it difficult to isolate the effectiveness of individual components. However, we can initiate this investigation by comparing the structural differences between CNNs and ViTs to gain insight into the role of rank in their behavioral differences.

5.1.1 Similarity between ConvMixer and ViT

Recent advances in CNNs, most notably ConvMixers Trockman and Kolter [2022], have resulted in architectures that increasingly resemble Vision Transformers (ViTs). Specifically, ConvMixers decompose multi-channel convolution into separate depth-wise and channel-wise convolutions. Unlike earlier CNN variants, ConvMixers maintain a constant spatial resolution across all layers, avoiding any form of downsampling. Similarly, transformer architectures apply channel-wise operations, but replace depth-wise convolution with multi-head self-attention mechanisms (see Fig. 5.1).

In this sense, the only fundamental difference between ConvMixer and ViT lies in the spatial mixing mechanism: ConvMixer employs spatial convolution, whereas ViT uses self-attention. Aside from this, the remaining architectural components are largely comparable. This distinction in spatial mixing results in different behaviors across datasets of varying scale, a phenomenon traditionally attributed to the trade-off between long-range dependencies and local inductive biases. While this explanation is reasonable, we propose an alternative perspective grounded in the rank properties of spatial mixing functions.

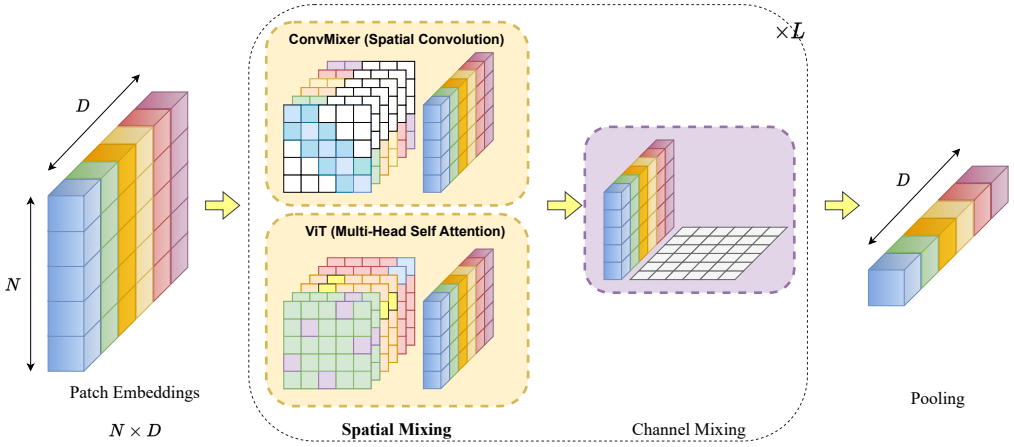


FIGURE 5.1. The architectures of ConvMixer Trockman and Kolter [2022] and Simple ViT Beyer et al. [2022] are quite similar. Both are composed of input patch embedding, several layers of spatial mixing and channel mixing blocks, and then pooling for future fully connected classifiers. Skip connections, BatchNorm or LayerNorm, and ReLU or GeLU are not shown in this figure for simplification. The only difference is the structure of the spatial mixing matrix. In ConvMixer it is in convolution form (the upper one) and each channel has a different filter, while in Simple ViT (the lower one) the channels are divided into heads, and in each head, the spatial mixing matrix is the same which is computed by SoftMax of two low-rank matrices $\mathbf{Q}$ and $\mathbf{K}$.

5.1.2 Rank of Spatial Mixing Functions

Let us define a 2D convolution filter as $\mathbf{h} \in \mathbb{R}^{f\times f}$. In general, this kernel can be represented as a circulant matrix $\mathbf{H} \in \mathbb{R}^{N\times N}$ such that $\mathbf{h}*\mathbf{x} = \mathbf{H}\mathbf{x}$, where $*$ denotes the convolution operator. With input $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D] \in \mathbb{R}^{N\times D}$, the spatial mixing layer in ConvMixer, $\mathcal{T}_S^{\text{Conv}} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$, and in ViT, $\mathcal{T}_S^{\text{ViT}} : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$, can be formulated as:

$$\mathcal{T}_S^{\text{Conv}}(\mathbf{X};\mathbf{H}) = [\mathbf{H}_1\mathbf{x}_1, \mathbf{H}_2\mathbf{x}_2, \ldots, \mathbf{H}_D\mathbf{x}_D], \tag{5.1}$$

$$\mathcal{T}_S^{\text{ViT}}(\mathbf{X};\mathbf{M}) = [\mathbf{M}_1\mathbf{x}_1, \ldots, \mathbf{M}_1\mathbf{x}_d, \mathbf{M}_2\mathbf{x}_{d+1}, \ldots, \mathbf{M}_2\mathbf{x}_{2d}, \ldots, \mathbf{M}_h\mathbf{x}_{(h-1)d+1}, \ldots, \mathbf{M}_h\mathbf{x}_{hd}], \tag{5.2}$$

where h denotes the number of attention heads in ViT and d is the dimensionality of each head, satisfying $D{=}hd$.

From one perspective, $\mathbf{H}$ incorporates a locality inductive bias: each row of $\mathbf{H}_i$ contains non-zero values only near the main diagonal. In contrast, $\mathbf{M}$ reflects the similarity between all possible position pairs, with all entries being non-negative values between 0 and 1 due to the `SoftMax` operation. Beyond this structural distinction, a more fundamental difference lies in their rank properties. Convolutional matrices $\mathbf{H}$ are naturally high-rank due to their structured design, whereas attention maps $\mathbf{M}$ at initialization are typically low-rank, mainly because of random initialization and the `SoftMax` function, which tends to produce concentrated distributions. Trockman and Kolter [2023] observed that pre-trained ViT weights tend to produce attention maps with strong main diagonals. Based on this, they proposed mimicking such a structure during initialization, achieving improved performance when training ViTs on small-scale datasets. From a rank perspective, low-rank attention maps lead to inefficient weight updates in ViTs, requiring more data and longer training to converge.

In contrast, high-rank attention maps result in better-conditioned embeddings and Neural Tangent Kernels (NTKs), as discussed in Chapter 4, thereby enabling more efficient learning even on small-scale datasets.

5.2 Introduction

Despite their success on large-scale training datasets, Vision Transformers (ViTs) often suffer a notable drop in performance when trained on small-scale datasets. This limitation is primarily attributed to their lack of architectural inductive biases, which are crucial for generalization with insufficient data. In contrast, Convolutional Neural Networks (CNNs) possess strong inductive biases that allow them to perform well even with limited training data.

To bridge this performance gap, several strategies have been proposed. These include self-supervised pretraining on large-scale datasets Dosovitskiy et al. [2021]; Touvron et al. [2021], advanced data augmentation techniques Cubuk et al. [2020]; Yun et al. [2019], and hybrid architectures that incorporate convolutional layers into Vision Transformers Dai et al. [2021]; Li et al. [2023]; Liu et al. [2021a]; Wu et al. [2021]; Yuan et al. [2021]. More recently, Zhang *et al*. Zhang et al. [2022] explored the use of pretrained weights to initialize ViTs. Building on this idea, subsequent works have shown that carefully designed network initialization strategies can enhance ViT performance on small-scale datasets without modifying the model architecture. In particular, Trockman and Kolter Trockman and Kolter [2023] introduced mimetic initialization that replicates the weight distribution of pretrained ViTs. Similarly, Xu *et al*. Xu et al. [2024] proposed directly sampling weights from large pretrained models.

While effective, these methods come with three notable limitations: (1) they focus on replicating the distribution of pretrained attention weights rather than structuring attention maps; (2) they rely on access to pretrained models trained on large-scale data, which is often impractical in domain-specific scenarios; and (3) their effectiveness is often tied to specific model architectures.

To overcome the limitations of existing approaches, we propose a novel "structured initialization" strategy for ViTs that imposes convolutional structures on attention maps without requiring any pretrained models. Our approach is grounded in our theoretical insights, aligned with the findings of recent work Cazenavette et al. [2025], that randomly initialized depthwise convolution filters can match the performance of their trained counterparts in models such as ConvMixer Trockman and Kolter [2022] and ResNet He et al. [2016]. Inspired by this, we develop an initialization strategy for ViTs based on random impulse convolution kernels, which impart locality and structure directly into the attention maps, as shown in Fig. 5.2. This structured initialization yields inductive biases characteristic of CNNs, enabling ViTs to generalize effectively on small-scale datasets, while preserving their adaptability for large-scale applications. Unlike prior methods that modify ViT architectures by integrating convolutional components, our method preserves the original transformer structure, making it broadly applicable across a variety of ViT variants.

To conclude, this chapter makes the following contributions:

- We establish a conceptual link between the structural inductive bias of CNNs and initialization in ViTs, and provide a theoretical justification for using random convolution filters to initialize attention maps.
- To the best of our knowledge, we are the first to introduce the initialization strategy that explicitly structures attention maps in ViTs, embedding convolutional

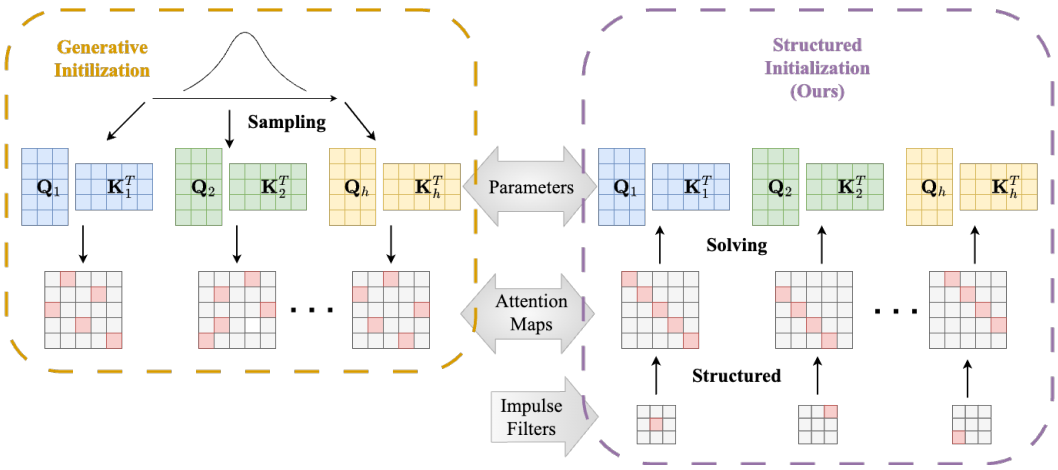


FIGURE 5.2. Illustration of conventional generative initialization and structured initialization (ours) strategies for the weights **Q** and **K** of the attention map in transformers. Conventional generative initialization involves sampling **Q** and **K** from certain distributions, such as Gaussian or Uniform, resulting in unstructured attention maps. In contrast, our structured initialization imposes constraints on the structure of the initial attention maps, specifically requiring them to be random impulse filters. The initialization of **Q** and **K** is computed based on this requirement. Note that in both attention maps and random impulse filters, the pink cells indicate ones, while the gray cells represent zeros.

inductive biases without modifying model architecture, enabling compatibility across diverse ViT variants.

- We demonstrate state-of-the-art performance on small-scale and medium-scale datasets, including Food-101, CIFAR-10, CIFAR-100, STL-10, Flowers, and Pets, while maintaining competitive results on large-scale datasets such as ImageNet-1K, and achieving improved performance across various ViT architectures, such as Swin Transformers and MLP-Mixer.

5.3 Related Work

Introducing inductive bias of CNN to ViT through architecture.

Many efforts have aimed to incorporate a convolutional inductive bias into ViTs through architectural modifications. Dai et al. [2021] proposed to combine convolution and self-attention by mixing the convolutional self-attention layers. Li et al. [2023]; Pan et al. [2021]introduced hybrid models wherein the output of each layer is a summation of convolution and self-attention. Wu et al. [2021] explored using convolution for token projections within self-attention, while Yuan et al. [2021] showed promising results by inserting a depthwise convolution before the self-attention map. d'Ascoli et al. [2021] introduced gated positional self-attention to imply a soft convolution inductive bias. Although these techniques have been proven effective, they aim to introduce the inductive bias of convolution through architectural choices. Our approach, on the other hand, stands out by not requiring any modifications to the architecture, retaining the generalizability to be seamlessly applied to different settings.

Initializating ViTs from pretrained weights.

The exploration of applying inductive bias through initialization within a transformer is limited to date. Zhang et al. [2022] posited that the benefit of pretrained models in ViTs can be interpreted as a more effective strategy for initialization. Trockman and Kolter [2023]; Trockman et al. [2022] recently investigated the empirical distributions of self-attention weights, learned from large-scale datasets, and proposed a mimetic initialization strategy. While this approach lies between structured and generative initialization, it relies on the pretraining results of large models. Li et al. [2024]; Samragh et al. [2023, 2024]; Xu et al. [2024] directly sampled weights from pretrained large-scale models as initialization for smaller models. While effective, the sampled weights must follow the distribution of these pretrained weights. A key difference in our approach is that our method does not require offline knowledge of pretrained models. Instead, our initialized structure is derived from a theoretical analysis of convolution layers.

Convolution as attention.

Since their introduction Dosovitskiy et al. [2021]; Vaswani et al. [2017], the relationship between transformers and CNNs has been a topic of immense interest. Andreoli [2019] studied the structural similarities between attention and convolution, bridging them into a unified framework. Building on this, Cordonnier et al. [2020] demonstrated that self-attention layers can express any convolutional layers through a careful theoretical construction. While these studies highlighted the functional equivalence between self-attention in ViTs and convolutional spatial mixing in CNNs, they did not delve into how the inductive bias of ViTs could be adapted through this theoretical connection. In contrast, our work offers a theoretical insight: a random convolutional impulse filter can be effectively approximated by softmax self-attention.

5.4 Why Random Impulse Filters Work?

It is well established that both ConvMixer and ViTs use alternating blocks of spatial and channel mixing, where ViTs replace the spatial convolutions in ConvNets with attention mechanisms. A fundamental difference, except for the receptive field, lies in their parameterization: spatial convolutions with hundreds of channels typically use distinct kernels for each channel, whereas self-attention relies on a shared mechanism with only a limited number (~ 10) of attention heads. By introducing a key observation that input embeddings are often rank-deficient, we demonstrate that as long as the spatial kernels or attention patterns sufficiently span the kernel space, the uniqueness or repetition of individual kernels becomes less critical. This insight, supported by a recent empirical observation Cazenavette et al. [2025], offers a new perspective on the relationship between ViTs and CNNs, motivating the use of impulse-based structures to embed convolutional inductive biases into the attention map initialization.

In recent work Cazenavette et al. [2025], Cazenavette *et al*. demonstrated a remarkable performance of randomly initialized convolution filters in ConvMixer and ResNet when solely learning the channel mixing parameters. However, they failed to offer any insights into the underlying reasons. In this section, we provide a theoretical analysis of how solely learning channel mixing can be sufficient for achieving reasonably good performance. Our theoretical findings are significant as they establish a conceptual link between the architecture of ConvMixer and the initialization of

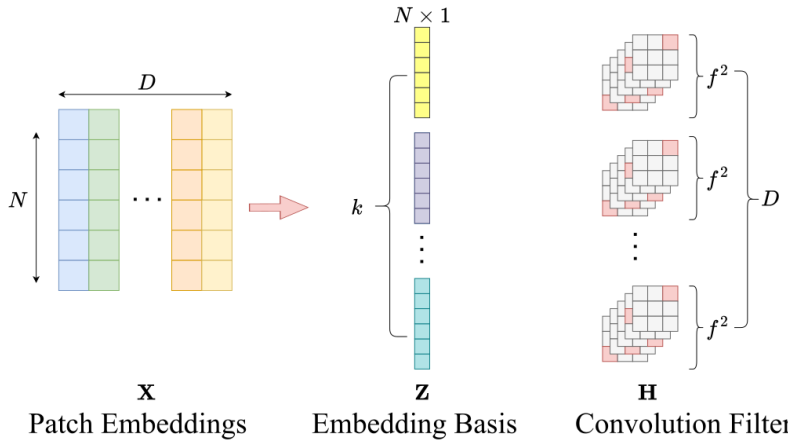


FIGURE 5.3. Illustration of why random spatial convolution filters are effective. Patch embeddings $\mathbf{X} \in \mathbb{R}^{N \times D}$ are typically rank-deficient and can be approximately decomposed to k bases. Meanwhile, a linear combination of f^2 linearly independent filters $\mathbf{h}$ can express any arbitrary filter in the filter space $\mathbb{R}^{f \times f}$. Based on these two observations, we derive the inequality $D \geq kf^2$ from Proposition 14.

ViT, offering a deeper understanding of desired properties for spatial mixing matrices. Without losing generality, we have omitted activations (*e.g.*, GeLU, ReLU, *etc.*), bias, batch normalization, and skip connections in our equations for clarity.

Remark 5. *Let us define the patch embeddings or intermediate layer outputs in ConvMixer as* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D]$*, where* D *is the number of channels and* N *is the number of pixels in the vectorized patch* $\mathbf{x} \in \mathbb{R}^N$*. An interesting observation is the rank (stable rank, defined as* $\sum \sigma^2 / \sigma^2_{max}$*) of* $\mathbf{X}$ *is consistently much smaller than the minimum dimension* $min(N, D)$ *of* $\mathbf{X}$*, indicating a significant amount of redundancy in patch embeddings or intermediate layer outputs in deep networks. This rank deficiency is common in various deep neural networks Feng et al. [2022], especially in ViTs Ji et al. [2025]; Naderi et al. [2024]; Noci et al. [2022].*

Let us define a 2D convolution filter as $\mathbf{h} \in \mathbb{R}^{f \times f}$. In general, this kernel can be represented as a circulant matrix $\mathbf{H} \in \mathbb{R}^{N \times N}$, such that $\mathbf{h} * \mathbf{x} = \mathbf{H}\mathbf{x}$, where $*$ denotes convolution operator. The relation between the convolutional matrix and convolution filters is explained in detail in Sec. 5.7.1.2. A ConvMixer block $\mathbf{T}^{\text{Conv}} : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times D}$ is composed of a spatial mixing layer $\mathbf{T}^{\text{Conv}}_S : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times D}$ and a channel mixing layer $\mathbf{T}^{\text{Conv}}_C : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times D}$, where $\mathbf{T}^{\text{Conv}}_S$ is defined by a sequence of convolution filters $\mathbf{H} = [\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_D] \in \mathbb{R}^{D \times N \times N}$, $\mathbf{H}_i \in \mathbb{R}^{N \times N}$, and $\mathbf{T}^{\text{Conv}}_C$ is defined by a weight matrix $\mathbf{W} \in \mathbb{R}^{D \times D}$. With input $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D] \in \mathbb{R}^{N \times D}$, $\mathbf{T}^{\text{Conv}}$ is

$$\mathbf{T}^{\text{Conv}}_S(\mathbf{X}; \mathbf{H}) = [\mathbf{H}_1\mathbf{x}_1, \mathbf{H}_2\mathbf{x}_2, \ldots, \mathbf{H}_D\mathbf{x}_D], \tag{5.3}$$

$$\mathbf{T}^{\text{Conv}}_C(\mathbf{X}; \mathbf{W}) = \mathbf{X}\mathbf{W}, \tag{5.4}$$

$$\mathbf{T}^{\text{Conv}}(\mathbf{X}) = \mathbf{T}^{\text{Conv}}_C(\mathbf{T}^{\text{Conv}}_S(\mathbf{X}; \mathbf{H}); \mathbf{W}) = [\mathbf{H}_1\mathbf{x}_1, \mathbf{H}_2\mathbf{x}_2, \ldots, \mathbf{H}_D\mathbf{x}_D]\mathbf{W}. \tag{5.5}$$

Definition 1. ***(M–k Spanned Set):*** *Let* $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N\} \subset \mathbb{R}^d$ *be a finite set of vectors. We say that* $\mathcal{V}$ *is* M*–*k spanned *(on* W*) if there exists a partition of* $\mathcal{V}$ *into at least* k *non-overlapping subsets:*

$$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \ldots \cup \mathcal{V}_k, \tag{5.6}$$

such that there exist a M–dimensional subspace $W \subset \mathbb{R}^d$ in the span of each subset $\mathcal{V}_i$

$$W \subseteq Span(\mathcal{V}_i) \quad \forall i = 1, 2, \ldots, k. \tag{5.7}$$

Proposition 14. *A ConvMixer block $\mathbf{T}$ consists of a spatial mixing layer $\mathbf{T}_S(\,\cdot\,;\mathbf{H})$ with convolution filters $\mathbf{H}$ and a channel mixing layer $\mathbf{T}_C$. $\mathbf{T}'$ is another ConvMixer block composed of $\mathbf{T}'_S(\,\cdot\,;\mathbf{H}')$ and $\mathbf{T}'_C$. Let k be the rank of the input $\mathbf{X}$. If $\mathbf{H}$ is M–k spanned on W, then for any set of filters $\mathbf{H}' \subset W$ and any $\mathbf{T}'_C$, there always exists a $\mathbf{T}_C$ such that $\mathbf{T}(\mathbf{X}) = \mathbf{T}'(\mathbf{X})$.*

For simplicity, we include the full proof in Sec. 5.7.1.1. Note that since H are convolution matrices, their span lies in $\mathbb{R}^{f\times f}$ instead of $\mathbb{R}^{N\times N}$, where f is the kernel size. In practice, the fact that $D \geq kf^2$ indicates that randomly initialized spatial convolution kernels are f^2–k spanned and satisfy Proposition 14, as illustrated in Fig. 5.3. Consequently, any trained results of $\mathbf{T}'_C$ and $\mathbf{T}'_S$ can be achieved by solely training the $\mathbf{T}_C$, while keeping a fixed spatial mixing layer $\mathbf{T}_S$. Hence, the following corollaries can be obtained, and Corollary 4 explains the phenomenon found in Cazenavette et al. [2025], mentioned at the beginning of the section. Corollary 5 inspires our proposed structure initialization for ViTs. The related experimental evidence of these corollaries and our proposition is given in the Sec. 5.7.2.4.

Corollary 4. *Randomly initialized spatial convolution filters perform as well as trained spatial convolution filters since the f^2–k spanned condition in Proposition 14 is satisfied.*

Corollary 5. *Random impulse spatial convolution filters perform as well as trained spatial convolution filters since the f^2–k spanned condition in Proposition 14 is satisfied.*

Corollary 6. *Spatial convolution filters with all ones (referred to as "box" filters) will not perform well since they are 1–k spanned and can produce only averaging values.*

5.5 Structured Initialization for Attention Map

5.5.1 Expected Initialized Attention Map Structure

ConvMixer and ViT share most of the components in their architectures. The gap in their performance on small-scale datasets stems from their architectural choices regarding the spatial mixing matrix. Although depthwise convolution (ConvMixer) and multi-head self-attention (ViT) may appear distinct at first glance, their underlying goal remains the same: to identify spatial patterns indicated by the spatial mixing matrix. As defined in Sec. 5.4, similar to the spatial mixing step in ConvMixer defined in Eq. (5.3), the spatial mixing step of multi-head attention can be expressed as

$$\begin{aligned}\mathbf{T}_S^{\mathrm{ViT}}(\mathbf{X};\mathbf{M}) = [&\mathbf{M}_1\mathbf{x}_1, \ldots, \mathbf{M}_1\mathbf{x}_d, \mathbf{M}_2\mathbf{x}_{d+1}, \ldots, \mathbf{M}_2\mathbf{x}_{2d},\\ &\ldots, \mathbf{M}_h\mathbf{x}_{(h-1)d+1}, \ldots, \mathbf{M}_h\mathbf{x}_{h*d}],\end{aligned} \tag{5.8}$$

where d represents the feature dimension in each head, typically set to D/h, with h being the number of heads, and the matrices $\mathbf{M}_i$ for multi-head self-attention can be expressed as follows:

$$\mathbf{M}_i = \mathrm{softmax}(\mathbf{X}\mathbf{Q}_i\,\mathbf{K}_i^T\mathbf{X}^T), \tag{5.9}$$

where $\mathbf{Q}_i$, $\mathbf{K}_i$ $\in \mathbb{R}^{D\times d}$ are attention weight matrices.

It is worth noting that in Eq. (5.3), the spatial matrices $\mathbf{H}$ are in convolutional structure, resulting in a span of $\mathbb{R}^{f\times f}$ instead of $\mathbb{R}^{N\times N}$, despite each $\mathbf{H}_i$ having a size of $N \times N$. This structural constraint ensures that CNNs focus on local features but struggle to capture long-range dependencies. In contrast, the span of spatial matrices $\mathbf{M}$ in Eq. (5.8) is $\mathbb{R}^{N\times N}$, allowing for greater learning capacity without these limitations. However, a randomly initialized $\mathbf{Q}$ and $\mathbf{K}$ contain no structural information, resulting in random matrices as depicted in the bottom left of Fig. 5.2.

Leveraging this insight, we propose to initialize the attention map for each head in ViT to a convolutional structure as denoted in the bottom right of Fig. 5.2. Our initialization strategy preserves both the advantage of locality and the capacity to learn long-range information. For clarity and brevity, the following discussions will focus only on one head of multi-head self-attention. Therefore, from Eq. (5.8) and Eq. (5.3), our structured initialization strategy can be represented as

$$\mathbf{T}_S^{\text{ViT}}(\mathbf{X};\mathbf{M}) \xleftarrow{\text{init}} \mathbf{T}_S^{\text{Conv}}(\mathbf{X};\mathbf{M}) \;\Rightarrow\; \mathbf{M}_{\text{init}} = \text{softmax}(\mathbf{X}\mathbf{Q}_{\text{init}}\mathbf{K}_{\text{init}}^T\mathbf{X}^T) \approx \mathbf{H}. \tag{5.10}$$

Why using impulse filters? Any random convolution filters that satisfy proposition 14 could be a choice of initialization of attention maps to introduce inductive bias. However, random convolution filters $\mathbf{H}$ usually contain both positive and negative values, while the output of the softmax function is always positive, making Eq. (5.10) unreachable. One straightforward option is to use random positive convolution filters with a normalized sum of one, following the property of softmax. However, this approach often proves inefficient as the patterns are too complicated for a softmax function to handle with $\mathbf{QK}$ being of low rank. In Tarzanagh et al. [2023], the authors found that the softmax attention map serves as a feature selection function, which typically tends to select a single related feature. In convolution filters, this selection can be parameterized as impulse filters. According to Corollary 5, random impulse filters are also f^2–k spanned. In conclusion, when initializing a softmax attention map, the most straightforward and suitable choice is random impulse convolution filters.

Pseudo input. The advantage of self-attention is that its spatial mixing map is learned from data. The real input to an attention layer is $\mathbf{P}+\mathbf{X}$ for the first layer and $\mathbf{X}$ (the intermediate output from the previous layer) for the following layers. However, during initialization, there is no prior information about the input. To address this problem, we simply use the initialization of positional encoding $\mathbf{P}$ as the pseudo input in the initialization of $\mathbf{Q}$ and $\mathbf{K}$, replacing the actual input data $\mathbf{P}+\mathbf{X}$ or intermediate outputs. Remember that this only happens when we solve the initialization to avoid data-dependent initialization, while in the training stage, we make no change to the ViT architecture.

With the use of impulse filters and the pseudo input, Eq. (5.10) becomes

$$\mathbf{M}_{\text{init}} = \text{softmax}(\mathbf{P}\mathbf{Q}_{\text{init}}\mathbf{K}_{\text{init}}^T\mathbf{P}^T) \approx \mathbf{H}_{\text{impulse}}. \tag{5.11}$$

5.5.2 Solving $\mathbf{Q}_{\text{init}}$ and $\mathbf{K}_{\text{init}}$

There exist numerous approaches to solve Eq. (5.11) for $\mathbf{Q}_{\text{init}}$ and $\mathbf{K}_{\text{init}}$ with known $\mathbf{H}_{\text{impulse}}$ and $\mathbf{P}$. Here, we apply an SVD-based method. First, we change Eq. (5.11) to exclude the Softmax function as

$$\mathbf{P}\mathbf{Q}_{\text{init}}\mathbf{K}_{\text{init}}^T\mathbf{P}^T = \alpha\mathbf{H}_{\text{impulse}} + \beta\mathbf{Z}\,, \tag{5.12}$$

where $\mathbf{Z} \sim \mathcal{N}(0, \frac{1}{D}\mathbf{I})$. Then to solve for $\mathbf{Q}_{\text{init}}$ and $\mathbf{K}_{\text{init}}$, we put $\mathbf{P}$ to the right-hand side of the equation using the pseudo inverse $\mathbf{P}_{\text{inv}} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$—since $\mathbf{P}$ is randomly initialized, it will be of full rank. Therefore, we get

$$\mathbf{Q}_{\text{init}}\mathbf{K}_{\text{init}}^T = \mathbf{P}_{\text{inv}}(\alpha\mathbf{H}_{\text{impulse}} + \beta\mathbf{Z})\mathbf{P}_{\text{inv}}^T . \tag{5.13}$$

With Eq. (5.13), we do SVD on the right-hand side and get a low-rank approximation of $\mathbf{Q}_{\text{init}}$ and $\mathbf{K}_{\text{init}}$:

$$\mathbf{U}, \mathbf{s}, \mathbf{V}^T = \text{SVD}\left(\mathbf{P}_{\text{inv}}\left(\alpha\mathbf{H}_{\text{impulse}} + \beta\mathbf{Z}\right)\mathbf{P}_{\text{inv}}^T\right), \tag{5.14}$$

$$\widetilde{\mathbf{Q}}_{\text{init}} = \mathbf{U}\sqrt{\mathbf{s}}\,[:, :d], \quad \widetilde{\mathbf{K}}_{\text{init}} = \mathbf{V}\sqrt{\mathbf{s}}\,[:, :d] . \tag{5.15}$$

Finally, we do a normalization to get the final $\mathbf{Q}_{\text{init}}$ and $\mathbf{K}_{\text{init}}$ as

$$\mathbf{Q}_{\text{init}} = \frac{\gamma}{\|\widetilde{\mathbf{Q}}_{\text{init}}\|_F)}\widetilde{\mathbf{Q}}_{\text{init}}, \quad \mathbf{K}_{\text{init}} = \frac{\gamma}{\|\widetilde{\mathbf{K}}_{\text{init}}\|_F}\widetilde{\mathbf{K}}_{\text{init}} . \tag{5.16}$$

In ViT models with $d = 64$, we use a 3×3 convolution filters by setting $f = 3$. We find that the random values in $\mathbf{Z}$ are not decisive for our initialization strategy. To ensure a clear impulse structure, the initialization is primarily governed by selecting an appropriate ratio of α to β. Empirically, we choose a large ratio such that $\alpha{:}\beta = 40{:}1$ with $\gamma = 2$. Notably, the exact values of α and β are not strictly constrained due to the normalization step, which scales these parameters. The pseudo code for our initialization strategy can be found in Algorithm 1.

Algorithm 1 Convolutional Structured Impulse Initialization for ViT

Input: $\mathbf{P}$ ▷ Input positional encoding
Input: $d, f, \alpha, \beta, \gamma$ ▷ Hyperparameters
Output: $\mathbf{Q}_{\text{init}}$, $\mathbf{K}_{\text{init}}$ ▷ Initialized attention parameters

$N, D \leftarrow \text{shape}(\mathbf{P})$
$\mathbf{H}_{\text{impulse}} \leftarrow \text{ImpulseConvMatrix}(N, f)$ ▷ Build 2D impulse convolution matrix
$\widetilde{\mathbf{M}} \leftarrow \alpha\mathbf{H}_{\text{impulse}} + \beta\mathbf{Z}$ ▷ Get ideal map
$\widetilde{\mathbf{X}} \leftarrow \text{LayerNorm}(\mathbf{P})$ ▷ Get pseudo input
$\mathbf{P}_{\text{inv}} \leftarrow (\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}$ ▷ Get pseudo inverse of $\mathbf{P}$
$\hat{\mathbf{M}} \leftarrow \mathbf{P}_{\text{inv}}\widetilde{\mathbf{M}}\mathbf{P}_{\text{inv}}^\top$ ▷ Get patterns $\mathbf{QK}^T$
$\mathbf{U}, \mathbf{s}, \mathbf{V}^\top \leftarrow \text{SVD}(\hat{\mathbf{M}})$ ▷ SVD of $\hat{\mathbf{M}}$
$\widetilde{\mathbf{Q}}_{\text{init}} \leftarrow \mathbf{U}\sqrt{\mathbf{s}}, \quad \widetilde{\mathbf{K}}_{\text{init}} \leftarrow \mathbf{V}\sqrt{\mathbf{s}}$
$\widetilde{\mathbf{Q}}_{\text{init}} \leftarrow \widetilde{\mathbf{Q}}_{\text{init}}[:, : d], \quad \widetilde{\mathbf{K}}_{\text{init}} \leftarrow \widetilde{\mathbf{K}}_{\text{init}}[:, : d]$
$\widetilde{\mathbf{Q}}_{\text{init}} \leftarrow \widetilde{\mathbf{Q}}_{\text{init}}/\text{norm}(\mathbf{Q}), \quad \widetilde{\mathbf{K}}_{\text{init}} \leftarrow \widetilde{\mathbf{K}}_{\text{init}}/\text{norm}(\mathbf{K})$
$\mathbf{Q}_{\text{init}} \leftarrow \gamma\widetilde{\mathbf{Q}}_{\text{init}}, \quad \mathbf{K}_{\text{init}} \leftarrow \gamma\widetilde{\mathbf{K}}_{\text{init}}$
return $\mathbf{Q}_{\text{init}}$, $\mathbf{K}_{\text{init}}$

5.6 Experiments and Analysis

In this section, we show that our impulse initialization can improve the performance of ViT on small-scale datasets in Sec. 5.6.1, while it does not limit the learning flexibility of ViT models on large-scale datasets in Sec. 5.6.2. In Sec. 5.6.5, we show that even for a pretrained method like weight selection Xu et al. [2024], the pretrained model initialized with our impulse structure can provide better pretrained weights with a relatively smaller-scale dataset. Finally, in Sec. 5.6.3, we show that besides ViTs, the

concept of introducing architectural bias into initializations can also be generalized to other models like Swin Transformers and MLP-Mixer. Note that all experiments were conducted on a single node with 8 Tesla V100 SXM3 GPUs, each with 32GB of memory. Specifically, all the experiments on the small-scale datasets took about three hours to train each model, while the experiments on the ImageNet-1K took about two days. Please note that all results were reported based on our experiments with retrained models. As a result, minor discrepancies may exist between our reported results and those in published papers. However, the key focus of this analysis remains on the improvements achieved through different initialization strategies.

5.6.1 Small and Medium-Scale Datasets

In this section, we compare the performance of ViT-Tiny under three initialization strategies: the default Wightman [2019], mimetic Trockman and Kolter [2023], and our impulse initialization. Experiments are conducted on medium-scale datasets (∼50K training images), including Food-101 Bossard et al. [2014], CIFAR-10, and CIFAR-100 Krizhevsky et al. [2009], as well as small-scale datasets (∼5K training images), such as STL-10 Coates et al. [2011], Flowers Nilsback and Zisserman [2008], and Pets Parkhi et al. [2012]. We follow the training recipe from Xu et al. [2024], which is proven to be useful in training ViT-Tiny on small and medium-scale datasets. Their codes are based on the timm library Wightman [2019]. By default, all the weights are initialized with a truncated normal distribution. For a fair comparison, all the experiments were run with identical codes except for the choice of initialization methods.

The experimental results are presented in Tab. 5.1. Our method consistently yields substantial improvements over the default initialization across all evaluated datasets. On medium-scale datasets, it achieves performance gains of 2%∼5%, while on small-scale datasets, improvements can reach approximately 8%, and in some cases, up to 20%. While mimetic initialization also improves the performance, our impulse initialization shows superior efficacy, attributed to the convolutional structure integrated in the attention initialization. Notably, as the dataset size decreases, the performance gap between our method and the default (or mimetic) initialization gets larger. This observation validates that the convolutional inductive bias introduced by our initialization becomes increasingly important when there is less data for the model to learn the spatial dependencies.

5.6.2 Large-Scale Datasets

In this section, we compare the performance of ViT-Tiny, ViT-Small, and ViT-base with default, mimetic, and our impulse initialization on a large-scale dataset—ImageNet-1K (over 1M training images). We follow the training recipe from DeiT Touvron et al. [2021]—a classic and efficient training recipe for training ViT models on ImageNet-1K. We directly use the ViT structure and training codes in the timm library, except for adding our implementation of initialization. All the experiments were trained with the same hyperparameters.

We show the comparison results on the ImageNet-1K dataset in Tab. 5.2. Detailed training hyperparameters and training curves can be found in **??**. Our method maintains comparable performance with default initialization, demonstrating that the convolutional inductive bias introduced during initialization does not hinder the model's

TABLE 5.1. Classification accuracy of ViT-Tiny with different initialization methods on different datasets. **Green** number indicates an increase in accuracy. Note that we compare the performance to the default initialization method (shaded in gray). ● represents small-scale datasets, and ▲ represents medium-scale datasets. The datasets are ranked based on their training scales.

Method \ Data↓	▲ Food-101		▲ CIFAR-10		▲ CIFAR-100		● STL-10		● Flowers		● Pets	
Default Wightman [2019]	77.95		92.29		71.67		61.86		64.60		26.58	
Mimetic Trockman and Kolter [2023]	81.78	3.83↑	93.50	1.21↑	75.16	3.49↑	68.54	6.68↑	71.62	7.02↑	47.63	21.05↑
Ours (impulse)	**81.85**	**3.90↑**	**94.67**	**2.38↑**	**77.02**	**5.35↑**	**70.21**	**8.35↑**	**73.18**	**8.58↑**	**50.84**	**24.26↑**

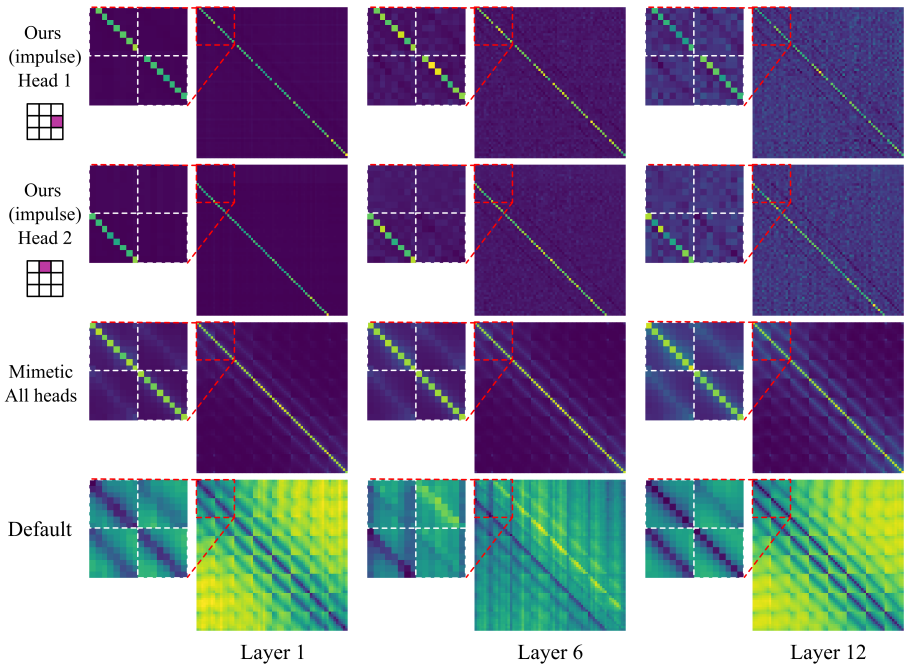


FIGURE 5.4. Visualization of attention maps in ViT-T using ours, mimetic Trockman and Kolter [2023], and default Xu et al. [2024] initializations. Red boxes highlight zoomed-in details of the 16×16 upper left corner in attention maps. White boxes indicate the 8×8 sub-blocks of the zoomed-in attention maps. Our structured initialization method offers distinct attention peaks aligned with the impulse structures across different heads. Head 1 offers a peak at $+1$ offset from the main diagonal. Head 2 offers a peek at -8 offset (equivalent to $-$image_size) from the main diagonal. Both mimetic and random initialization methods initialize all the attention heads identically. Specifically, mimetic initialization primarily strengthens the main diagonal of the attention map for each head, while random initialization shows minimal structural patterns with flatter peak values.

flexibility in learning data-driven dependencies. This indicates that while the transformer architecture begins training with structurally imposed spatial priors, the attention mechanism retains full capacity to learn optimal feature representations when sufficient training data is available. Furthermore, the convolutional structure we introduced in the attention map initialization not only accelerates early convergence but also improves the robustness to variations in training hyperparameters.

In summary, experiments on both small-scale and large-scale datasets show that our initialization method effectively balancing prior architectural knowledge integration with data adaptability—incorporating a convolutional inductive bias during initialization provides structural guidance to capture dependencies when training data is limited, while still allowing the attention mechanism to retain full learning flexibility in scenarios with abundant data.

5.6.3 ViT Variants

Beyond the original ViT architectures, we extend a simplified version of our initialization strategy to the Swin Transformer. Notably, Swin Transformers incorporate relative positional encoding within each attention block—a design choice that aligns with our core objective of instilling convolutional structure into attention maps. For

TABLE 5.2. Classification accuracy of ViT-Tiny, ViT-Small and ViT-Base on ImageNet-1K dataset with different initialization methods. ■ indicates large-scale datasets.

Method \ Model	■ ViT-Tiny		■ ViT-Small		■ ViT-Base	
Default Wightman [2019]	72.71		79.68		76.25[1]	
Mimetic Trockman and Kolter [2023]	72.90	0.19↑	80.26	0.58↑	79.12	2.87↑
Ours (impulse)	**72.76**	**0.05↑**	**80.40**	**0.72↑**	**80.48**	**4.23↑**

[1] Although we strictly follow the training recipe reported in DeiT, we acknowledge that our baseline performance appears slightly lower. This discrepancy may stem from subtle differences in training code or hardware configurations. However, the primary purpose of this table is to demonstrate that our initialization does not compromise the performance of large models.

TABLE 5.3. Classification accuracy of different Swin Transformers and MLP-Mixer on different datasets with different initialization methods. **Red**/**green** number indicates accuracy decrease/increase. We compare the performance to the default initialization method (shaded in gray).

Method \ Model&Data	Swin-B on ■ ImageNet-1K		Swin-T on ▲ CIFAR-10		MLP-Mixer on ▲ CIFAR-10	
Default Wightman [2019]	83.14		89.85		87.00	
Mimetic Trockman and Kolter [2023]	83.14	—	89.24	0.61↓	—	
Ours (impulse)	**83.55**	**0.41↑**	**91.19**	**1.34↑**	**88.78**	**1.78↑**

these models, we achieve this by directly initializing the relative positional embeddings with our impulse pattern.

We evaluated our approach with the Swin Transformer-Base (Swin-B) architecture on the ImageNet-1K following the training recipe in the original Swin Transformer paper Liu et al. [2021b]. To further demonstrate the flexibility of our approach, we applied our impulse initialization to the MLP-Mixer. For experiments with Swin Transformer-Tiny (Swin-T) and MLP-Mixer, we follow training settings in Yoshioka [2024], which is specifically designed for training different models on CIFAR-10.

The results are summarized in Tab. 5.3. For models with strong learning capacity but limited inherent inductive biases at initialization, introducing a structured initialization consistently enhances performance without compromising the model's capacity to learn complex data dependencies. In particular, while Swin Transformer incorporates convolutional inductive bias through windowed self-attention, applying our structured initialization further improves the performance by 1.34% on smaller-scale datasets such as CIFAR-10, with no degradation on large-scale datasets like ImageNet-1K. MLP-Mixer, which replaces the spatial convolutions in ConvMixer with MLP layers, typically struggles to train effectively on small-scale datasets. However, initializing the spatial MLPs with a convolutional structure leads to a 1.78% performance gain on CIFAR-10. In contrast, mimetic initialization—designed to replicate empirical weight distributions from pretrained ViTs—shows negligible benefits or degrades performance, highlighting its limited generalizability outside the specific pretrained ViT structures.

5.6.4 Attention Maps

The initialized attention maps produced by default, mimetic, and our impulse initializations are shown in Fig. 5.4. For better visualization, we use 32×32 CIFAR-10 images as input with a 4×4 patch size, yielding the token size as 8×8=64. We observe that the default initialization generates near-identical attention values with little distinguishable spatial structure, while the mimetic initialization introduces diagonally dominant values by adding an identity matrix during initialization. Notably, these two initialization methods initialize all the attention heads in the same way. In contrast, our initialization assigns different impulse patterns to each attention head, producing spatially diverse activations with off-diagonal peaks on specific impulse positions.

5.6.5 Pre-trained Model

In this section, we demonstrate that our initialization method remains compatible with existing pretraining pipelines. Building on the work of Xu *et al*. Xu et al. [2024], who showed that weight selection from ImageNet-21K-pretrained ViT-Small models effectively initializes ViT-Tiny architectures, we extend this approach to ImageNet-1K pertaining. We pretrained three ViT-Small models on ImageNet-1K using three initialization strategies: default, mimetic, and our proposed impulse initialization. From these, we derived three ViT-Tiny initialization variants—termed "1K-default", "1K-mimetic", and "1K-impulse"—using the same weight selection methodology on smaller-scale datasets. When training these initialized ViT-Tiny models on small and medium-scale datasets (results shown in Tab. 5.4), we observed that: (1) Switching from ImageNet-21K to ImageNet-1K—less training data—pretraining with default initialization typically incurs a performance drop of ∼1%. (2) Mimetic initialization fails to mitigate this data degradation, (3) The ImageNet-1K pretraining model with

our impulse-based initialization achieves performance parity with ImageNet-21K pretrained baselines. This highlights the robustness of our method even under reduced pretraining data regimes.

5.7 Additional Results

5.7.1 Theorectical Results

5.7.1.1 Proof for Proposition 1

Let $\mathbf{w} = [w_1, w_2, \ldots, w_D]^T \in \mathbb{R}^{D\times 1}$ be the channel mixing weights for one output channel and $\mathbf{H}_1, \mathbf{H}_2, \ldots, \mathbf{H}_D$ are the corresponding spatial convolution filters for each channel. Therefore, the result $\mathbf{y} \in \mathbb{R}^N$ after spatial and channel mixing can be represented as,

$$\mathbf{y} = \sum_{i=1}^{D} w_i \mathbf{H}_i \mathbf{x}_i, \tag{5.17}$$

With remark 5, we can suppose the rank of $\mathbf{X} \approx \mathbf{ZA}$ is k, where $\mathbf{Z} = [\mathbf{z}_1, \ldots, z_k]$ and $k \ll D$, as illustrated in Fig. 5.3. We then obtain

$$\mathbf{y} \approx \sum_{i=1}^{D}\sum_{j=1}^{k} w_i a_{ji} \mathbf{H}_i \mathbf{z}_j = \sum_{j=1}^{k} \tilde{\mathbf{H}}_j \mathbf{z}_j, \tag{5.18}$$

where a_{ji} refers to the row j, column i element of $\mathbf{A}$, and $\tilde{\mathbf{H}}_j = \sum_{i=1}^{D} w_i\, a_{ji}\, \mathbf{H}_i$.

Remember that a linear combination of f^2 linearly independent filters $\mathbf{h}$ can express any arbitrary filter in filter space $\mathbb{R}^{f\times f}$, where $\mathbf{h}$ serves as the basis. Consequently, any desired $\tilde{\mathbf{H}}_1, \tilde{\mathbf{H}}_2, \ldots, \tilde{\mathbf{H}}_D$ can be achieved by only learning the channel mixing weights $\mathbf{w}$. Therefore, we obtain the following proposition.

5.7.1.2 Convolutional Representation Matrix

In Sec. 5.4, we interchangeably use the terms convolution filter $\mathbf{h}$ and convolution matrix $\mathbf{H}$. Additionally, we represent the impulse filter as a convolutional matrix. Here, we offer a detailed explanation of the relationship between the convolutional filters and the convolutional matrices.

Let us define a 2D convolution filter as $\mathbf{h} \in \mathbb{R}^{f\times f}$ with elements

$$\mathbf{h} = \begin{pmatrix} h_{11} & \cdots & h_{1f} \\ \vdots & \ddots & \vdots \\ h_{f1} & \cdots & h_{ff} \end{pmatrix}. \tag{5.19}$$

When $\mathbf{h}$ is convolved with an image $\mathbf{x} \in \mathbb{R}^{H\times W}$, this convolution operation is equivalent to a matrix multiplication

$$\mathrm{vec}(\mathbf{h} * \mathbf{x}) = \mathbf{H}\,\mathrm{vec}(\mathbf{x}), \tag{5.20}$$

where $\mathbf{H}$ is composed from the elements in $\mathbf{h}$ and zeros in the following format:

$$\mathbf{H} = \begin{pmatrix} \mathbf{F_1} & \mathbf{F_2} & \cdots & \mathbf{F_f} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{F_1} & \mathbf{F_2} & \cdots & \mathbf{F_f} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{F_1} & \mathbf{F_2} & \cdots & \mathbf{F_f} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{F_1} & \mathbf{F_2} & \cdots & \mathbf{F_f} \end{pmatrix}, \tag{5.21}$$

TABLE 5.4. Classification accuracy of ViT-Tiny on small-scale datasets with the weight selection method. Here, the weights selected for the experiments were pretrained on the ImageNet-1K (shortened as 1K, shown above the **purple** dashed line) and ImageNet-21K (shortened as 21K, shown below the **purple** dashed line) datasets with different initialization methods. Please note that pretraining on ImageNet-21K with default initialization is the original weight selection method Xu et al. [2024]. We compare the performance to the default initialization method pretrained on ImageNet-1K (shaded in **gray**).

Method+Model \ Data↓	▲ Food-101		▲ CIFAR-10		▲ CIFAR-100		● STL-10	
1K+Default Xu et al. [2024]	85.42		96.61		79.64		82.58	
1K+Mimetic Trockman and Kolter [2023]	86.32	0.90↑	96.37	0.24↓	79.86	0.22↑	82.39	0.19↓
1K+Ours (impulse)	**87.43**	**2.01**↑	**97.19**	**0.58**↑	80.92	1.28↑	**83.89**	**1.31**↑
21K+Default Xu et al. [2024]	87.14	1.72↑	97.07	0.46↑	**81.07**	**1.43**↑	83.23	0.65↑

TABLE 5.5. Ablation study on training settings of ViT-Base on ImageNet-1K dataset. Note that we compare the performance to the default initialization method (shaded in gray). The performance we reported in the main experiments section is on the first row (shaded in cyan).

Scaled LR	Repeated Augment	Default	Mimetic		Ours (impulse)	
7e-4	1.0	76.25	79.12	2.87↑	**80.48**	**4.23↑**
1e-3	1.0	80.09	80.17	0.08↑	**81.55**	**1.07↑**
1e-3	3.0	81.24	80.56	0.68↓	**81.83**	**0.59↑**

where

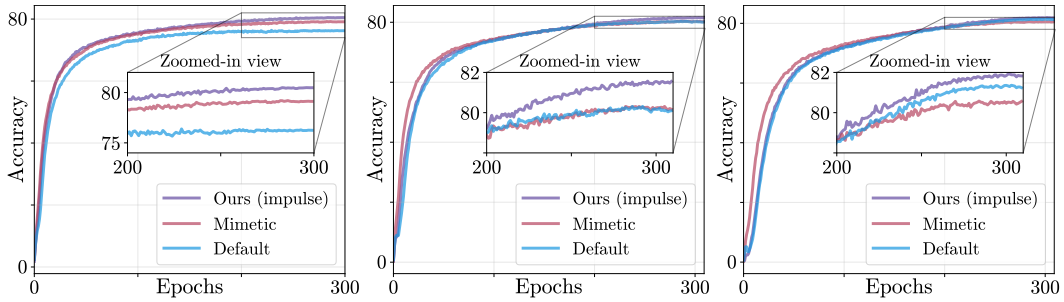
$$\mathbf{F_i} = \begin{pmatrix} h_{i1} & h_{i2} & \cdots & h_{if} & 0 & 0 & \cdots & 0 \\ 0 & h_{i1} & h_{i2} & \cdots & h_{if} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & h_{i1} & h_{i2} & \cdots & h_{if} & 0 \\ 0 & \cdots & 0 & 0 & h_{i1} & h_{i2} & \cdots & h_{if} \end{pmatrix}, \tag{5.22}$$

for $i = 1, 2, \ldots, f$. $\mathbf{F_i}$s are circulant matrices and $\mathbf{H}$ is a block circulant matrix with circulant block (BCCB). Note that convolutions may employ various padding strategies, but the circulant structure remains consistent. Here, we show the convolution matrix without any padding as an example.

5.7.2 Experimental Results

5.7.2.1 Training Details and Training Curves for ViT-Base

In the main experiments section, we found it challenging to reproduce the ViT-Base performance reported in DeiT Touvron et al. [2021], even when strictly following their specified training settings. Upon a careful examination of code differences across multiple versions of the `timm` library Wightman [2019], an open-source repository hosted on GitHub, we identified two critical discrepancies that likely contributed to the performance gap: **(a) Learning rate scaling strategy:** In DeiT, the learning rate is linearly scaled with respect to the batch size: $\text{lr}_{\text{scaled}} = \text{lr}_{\text{base}} \times \frac{\text{batch size}}{512}$. In contrast, the current version of timm uses a square root scaling rule as the default for the AdamW optimizer: $\text{lr}_{\text{scaled}} = \text{lr}_{\text{base}} \times \sqrt{\frac{\text{batch size}}{512}}$. This discrepancy in the default setting leads to different effective learning rates, even when all the other hyperparameters are identical, and can substantially affect performance. **(b) Repeated data augmentation setting:** DeiT emphasizes the importance of the repeated data augmentation strategy, stating a drop in top-1 accuracy from 81.8% to 76.5% when it is disabled. However, they did not specify the exact augmentation weighting value in their original paper. After inspecting their specific version of code, we discovered that the default value for the repeated data augmentation was set to `3`, whereas we used `1` in our main experiments, which may partially explain the performance discrepancy. We also aligned several other minor settings, such as the minimum learning rate during warm-up and at the end of training, as well as an additional 10 epochs for cool-down. However, we believe that these factors have only a marginal effect on the final performance, while the two aforementioned reasons remain the primary contributors to the observed discrepancy.



(a) Scaled LR: 7e-4 Repeated Augementation: 1.0 (b) Scaled LR: 1e-3 Repeated Augmentation: 1.0 (c) Scaled LR: 1e-3 Repeated Augmentation: 3.0

FIGURE 5.5. Training curves of ViT-Base using default, mimetic, and impulse initialization under three different training configurations. The zoomed-in box shows the training curve in the final training stage from epoch 200 to epoch 300.

TABLE 5.6. Detailed scale of all the datasets used in this Chapter. ■ represents large-scale datasets, ▲ represents medium-scale datasets, and ● represents small-scale datasets. The datasets are ranked based on their scales following Xu et al. [2024]. A '+' means we use both train and validation dataset in training. Note for Flowers and Pets datasets, we use both training and validation data for training.

Dataset ↓	#Classes	#Train images in each class	#Train images in total	#Images in total
■ ImageNet-1K	1K	∼1K	1.3M	1.4M
▲ Food-101	101	750	75K	101K
▲ CIFAR-10	10	5K	50K	60K
▲ CIFAR-100	100	500	50K	60K
● STL-10	10	500	5K	13K
● Flowers	102	10+10	2K	8K
● Pets	37	50+50	3.7K	7.3K

We present an ablation study on these settings in Tab. 5.5. With the two settings properly adjusted, we achieved a top-1 accuracy of 81.24%, which is comparable to the 81.8% reported in DeiT Touvron et al. [2021]. Under the same settings, our impulse initialization method still achieves an accuracy of 81.83% with a 0.59% improvement, showcasing the robustness to different training hyperparameters, and still outperforming the mimetic initialization. Furthermore, our structured initialization method demonstrates robustness across different training configurations, consistently yielding over 80% accuracy and outperforming the mimetic initialization in all cases.

In Fig. 5.5, we also present the training accuracy curves across epochs for different initialization methods with three different training configurations. Our impulse initialization consistently outperforms the default initialization throughout the entire training process, with a particularly significant advantage in the final training stage. While the mimetic initialization shows relatively faster initial convergence, it ultimately degrades performance in the final training stage across different training configurations.

5.7.2.2 Dataset Scale

For completeness, we provide the detailed dataset scales used in the main experiments section in Tab. 5.6. The ordering of the datasets follows that in Xu et al. [2024].

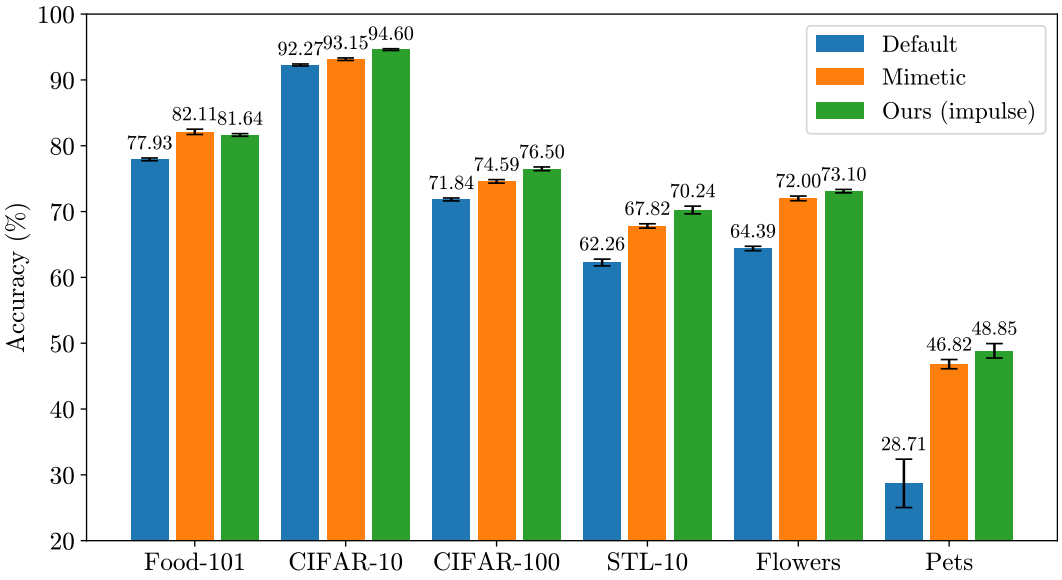


FIGURE 5.6. Classification accuracy of ViT-Tiny with different initialization methods on different datasets (mean of 5 runs). Our method consistently outperforms the default and the mimetic initializations. Except for the smallest-scale dataset Pets, all the methods show robust performance across different datasets with small error bars. Note that from left to right, the dataset scale decreases.

5.7.2.3 Additional Statistical Results for Small and Medium-Scale Datasets

Here we provided the mean and standard deviation of 5 runs for ViT-Tiny with different initialization methods on small and medium-scale datasets in Fig. 5.6, which serves as additional statistical results of Tab. 5.1 in the main experiments section. Notably, our initialization method still outperforms other methods across small and medium-scale datasets. Especially for the smaller-scale datasets, our methods show larger performance improvements, aligning with our findings in the main experiments section.

5.7.2.4 Experimental Validation of the Proposition and Corollaries

In this section, we conduct a series of preliminary experiments on CIFAR-10 to validate the propositions and corollaries in Sec. 5.4, with particular focus on demonstrating that in ConvMixer, fixed random impulse spatial filters can achieve comparable performance to learned filters. All experiments adhere to the training protocol proposed in Yoshioka [2024], which is specifically tailored for evaluating diverse architectures on small-scale datasets such as CIFAR-10.

To support our theoretical findings in Sec. 5.4 concerning the effectiveness of random filters, we train ConvMixer Trockman and Kolter [2022] models with an embedding dimension of 256, a depth of 8, and a patch size of 2 on the CIFAR-10 dataset, using spatial filter sizes of 3, 5, and 8. We also include a variant with an embedding dimension of 512 to examine the impact of feature width. We evaluate the end-to-end trained ConvMixer alongside three initialization strategies: random (Corollary 4), impulse (Corollary 5), and box (Corollary 6). Note that in all three cases, only the spatial convolution filters are initialized—the models are evaluated without any training. The box filters use all-one values, effectively performing average pooling. The results are summarized in Tab. 5.7.

TABLE 5.7. Classification accuracy(%) of ConvMixer (depth 8) with different filter sizes, embedding dimensions on CIFAR-10.

Kernel Size	Embedding Dimension = 256				Embedding Dimension = 512			
	Trained	Random	Impulse	Box	Trained	Random	Impulse	Box
3	91.76	90.72	90.68	81.70	92.82	92.15	92.20	81.90
5	92.69	90.87	90.41	80.57	93.90	92.72	91.91	81.19
8	92.34	88.12	87.82	78.95	92.96	90.09	89.61	80.10

In conclusion, these experimental results reveal several key insights. First, comparing across columns (*i.e.*, initialization methods), both random and impulse initializations achieve performance comparable ($\geq$90%) to that of fully trained models, whereas the box initialization leads to significantly worse performance ($\sim$80%). This discrepancy can be attributed to the deficient rank of the box filters, which fail to span the full f^2-dimensional filter space, unlike random and impulse filters that are capable of forming a complete basis.

Second, when comparing across rows (*i.e.*, kernel sizes) with an embedding dimension of 256, the performance gap between trained and untrained (random or impulse) filters grows with kernel size, from 1% (size 3) to 2% (size 5) and 5% (size 8). This occurs because larger kernels require more distinct filters to effectively span the filter space. However, the fixed embedding dimension constrains the number of such filters, reducing their ability to match the input rank as shown in Proposition 14. Notably, when the embedding dimension is doubled to 512, this performance gap narrows. In particular, for a kernel size of 3, the random and impulse initializations nearly match the performance of the trained filters, suggesting that sufficient embedding width compensates for the limitations of fixed filters.

5.7.2.5 Attention Maps

Here we provide additional visualization of the attention maps for all 12 layers in Fig. 5.7. In particular, our structured initialization method offers different attention peaks on various heads, showing alignment with the impulse structures, while the mimetic initialization only presents main-diagonal peaks, and the default initialization shows little to no patterns. As stated in the main experiments section, both mimetic and default initialization methods use identical initialization for all attention heads.

5.8 Limitations

There exist several limitations in our initialization method: **(1) Positional encoding.** Although positional encoding is a natural and effective choice for pseudo-inputs—due to its simplicity and data-independent nature—even simpler alternatives may exist for initializing the $\mathbf{Q}$ and $\mathbf{K}$ matrices. In this work, we focus solely on the initialization of $\mathbf{Q}$ and $\mathbf{K}$, but a more comprehensive initialization strategy that also considers patch embeddings and positional encodings could offer greater control over the structure of attention maps. Notably, since positional encoding is only added at the input layer, the influence on attention maps may diminish with increasing network depth, as illustrated in Fig. 5.4. **(2) Hard constraints.** Our Corollary 5 of Proposition 14 is based on the presumption that the filters are f^2–k spanned, which is usually a characteristic inherent in CNNs. However, in ViTs, the limited number of heads may be inadequate to span the filter space of a small kernel. Finding better adaptations in this scenario remains a challenge. **(3) Value initialization.** Our

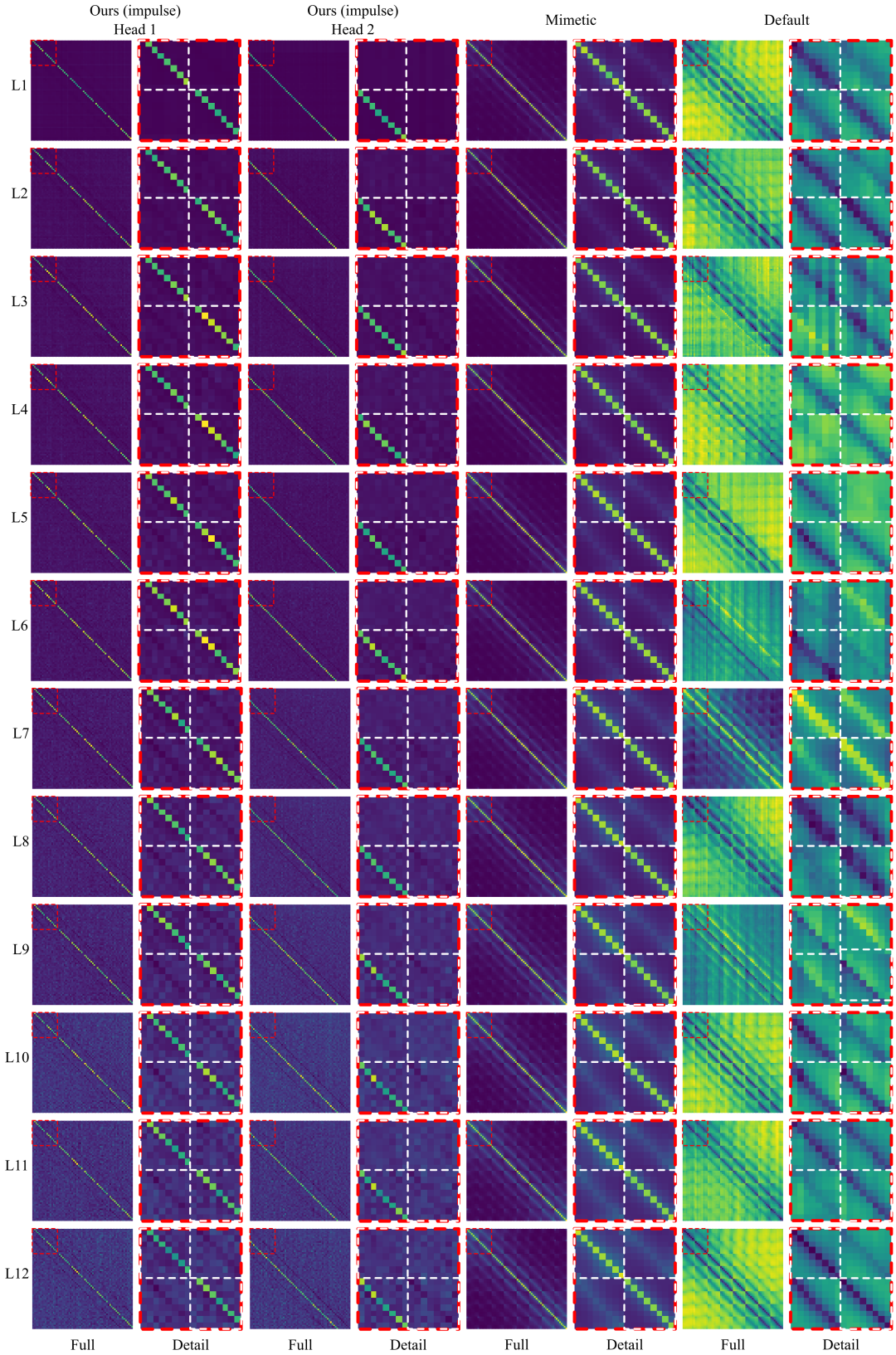


FIGURE 5.7. Visualization of attention maps in ViT-T using our impulse initialization method, mimetic [Trockman and Kolter, 2023], and default [Wightman, 2019] initializations.

method does not consider the initialization for the value weights $\mathbf{V}$ and the projection matrix.

5.9 Conclusion

In this chapter, we propose a structured initialization method with convolutional impulse filters for attention maps in ViTs. Our method preserves both the advantage of locality within CNNs and the capacity to learn long-range dependencies inherited from ViTs. We also provide a thorough theoretical explanation of the spatial and channel mixing in ConvMixer and ViT, building connections between the structural bias in CNNs and the initialization of ViTs. Our results on small-scale datasets validate the effectiveness of the convolutional structural bias, while on-par performance on large-scale datasets indicates the preservation of architectural flexibility. Our initialization also accelerates early-stage convergence but also enhances the model's robustness to variations in training hyperparameters. Furthermore, we demonstrate that our method consistently provides benefits across a range of architectures and even under pre-training strategies.

Chapter 6

Conclusion

In conclusion, understanding why deep neural networks perform well remains a fundamental question, and the concept of rank emerges as a key component in this endeavor. As discussed throughout this thesis, rank plays a critical role in (a) parameter efficiency and (b) accelerated convergence under limited training data. At least in gradient descent-based optimization, the rank is closely related to how training parameters are updated.

In multilayer perceptrons (MLPs), especially those used in implicit neural representations (INRs) where inputs are low-dimensional coordinates, the most problematic component is often the first layer. Although the feature dimension increases significantly, the rank of the embedding remains low. To address this issue, various methods have been proposed, including alternative input embeddings (e.g., sinusoidal positional encodings, hash-based embeddings) and alternative activation functions (e.g., sinusoidal activations, batch normalization).

This thesis demonstrates that the common mechanism behind these diverse innovations is the increase in the rank of the input embeddings, thereby resolving the discrepancy between the high feature dimension and low rank following the first embedding layer. Specifically, for positional encodings, we show that performance is primarily governed by the trade-off between the stable rank of the embedding matrix and the preservation of distances between embedded coordinates. Motivated by this insight, we propose a novel framework capable of incorporating arbitrary continuous signals as potential embedding functions under mild constraints. Additionally, we introduce a positional encoding scheme that significantly accelerates convergence, allowing a single linear network to reconstruct signals with high fidelity.

Furthermore, we theoretically and empirically show that an untrained positional embedding can serve as a robust surrogate for learned per-point embeddings in point cloud processing. We also find that per-point embeddings should be both band-limited and smooth to ensure robustness against out-of-distribution (OOD) corruptions such as noise and outliers.

For alternative activation functions, our analysis reveals that modifying the activation in the first layer alone can yield performance comparable to modifying all activation layers. To support this claim, we design a novel initialization strategy for the first-layer input embedding under ReLU activation. Without employing any additional architectural changes, our initialization—by increasing the rank of the input embedding—significantly outperforms the default initialization, which tends to produce a low-rank embedding.

In the context of Vision Transformers (ViTs), we propose a new initialization strategy to address their training challenges on small-scale datasets. Our approach draws inspiration from the inductive bias of convolutional architectures but does not require any structural modifications. Through a careful investigation into why random spatial convolution filters work effectively, we initialize self-attention maps as random

impulse convolution filters. This design integrates the inductive bias of convolution while preserving the flexibility of transformer architectures. Notably, our structured initialization also increases the rank of the attention maps, further enhancing performance.

6.1 Limitations and Future Work

Although this thesis takes a step forward in understanding the conditions under which an MLP is effective, a comprehensive understanding of the optimization dynamics, the influence of initialization on training, and the relationship between network architecture and training behavior remains an open challenge.

For MLPs, this thesis demonstrates that the rank of the first-layer embedding plays a crucial role in determining network performance. However, the role of network depth—even with only two to five layers, rather than tens—is still not fully understood. In particular, how the NTK evolves across multiple layers and what constitutes the optimal initialization strategy for the subsequent layers remain open questions that require further investigation.

Although Transformers have been widely adopted across language, vision, and other modalities, the underlying mechanism of attention is still not fully understood. In particular, the initialization of the weights in the query and key layers typically follows the same approach as standard linear layers, despite their distinct architectural roles. This thesis offers a more theoretical perspective on initialization strategies, moving beyond conventional heuristic methods. In the future, a deeper understanding of the training dynamics within self-attention may lead to more effective and tailored initialization schemes. These findings may further contribute to the development of improved initialization methods for standard linear layers in MLP architectures.

Bibliography

Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017. 79

Jean-Marc Andreoli. Convolution, attention and structure embedding. *arXiv preprint arXiv:1905.01289*, 2019. 87

Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7163–7172, 2019. 37

Sanjeev Arora, Simon Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019. 10, 40

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR, 2020. 7

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. *arXiv preprint arXiv:2103.13415*, 2021. 7

Lucas Beyer, Xiaohua Zhai, and Alexander Kolesnikov. Better plain vit baselines for imagenet-1k. *arXiv preprint arXiv:2205.01580*, 2022. URL https://arxiv.org/abs/2205.01580. viii, 84

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *arXiv preprint arXiv:1905.12173*, 2019. 10, 40

Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer, 2014. 92

Christian Bueno and Alan Hylton. On the representation power of set pooling networks. *Advances in Neural Information Processing Systems*, 34:17170–17182, 2021. 40

Zhicheng Cai, Hao Zhu, Qiu Shen, Xinran Wang, and Xun Cao. Batch normalization alleviates the spectral bias in coordinate networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 25160–25171, 2024. 3

George Cazenavette, Joel Julin, and Simon Lucey. Rethinking the role of spatial mixing. *arXiv preprint arXiv:2503.16760*, 2025. 4, 85, 87, 89

Yang Chen, Ruituo Wu, Yipeng Liu, and Ce Zhu. Hoin: High-order implicit neural representations, 2024. URL https://arxiv.org/abs/2404.14674. 68

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011. 92

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *Eighth International Conference on Learning Representations-ICLR 2020*, 2020. 87

Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 3, 85

Z Dai, Z Yang, Y Yang, J Carbonell, QV Le, and R Transformer-XL Salakhutdinov. Attentive language models beyond a fixed-length context. published online june 2, 2019. doi: 10.48550. *arXiv*, 1901. 9

Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. *Advances in neural information processing systems*, 34:3965–3977, 2021. 3, 4, 85, 86

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 7

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 3, 85, 87

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018. 10, 40

Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, pages 2286–2296. PMLR, 2021. 86

Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank diminishing in deep neural networks. *Advances in Neural Information Processing Systems*, 35:33054–33065, 2022. 88

Xinlong Feng, Yinnian He, and Zhinan Zhang. The numerical rank of a matrix and its applications. *Applied Mathematics and Computation*, 196:416–421, 02 2008. doi: 10.1016/j.amc.2007.06.005. 69

Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. *arXiv preprint arXiv:2012.03065*, 2020. 7

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR, 2017. 7, 9

Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks: A new hope. *arXiv preprint arXiv:1909.04866*, 2019. 40

Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2): 187–199, 2021. 2, 37, 40, 46, 48, 51

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 85

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020. 9

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018. 3, 10, 40, 67, 69

Yiping Ji, Hemanth Saratchandran, Peyman Moghaddam, and Simon Lucey. Always skip attention. *arXiv preprint arXiv:2505.01996*, 2025. 88

Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. *arXiv preprint arXiv:2006.15595*, 2020. 9

Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*, 2018. 9

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 92

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019. 10, 40, 69

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019. 7

Alex Li, Yuandong Tian, Beidi Chen, Deepak Pathak, and Xinlei Chen. On the surprising effectiveness of attention transfer for vision transformers. *Advances in Neural Information Processing Systems*, 37:113963–113990, 2024. 87

Kunchang Li, Yali Wang, Junhao Zhang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unifying convolution and self-attention for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 3, 4, 85, 86

Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Pointnetlk revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12763–12772, 2021. 37, 49, 52

Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020. 7

Xuanqing Liu, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. Learning to encode position for transformer with continuous dynamical model. In *International Conference on Machine Learning*, pages 6327–6335. PMLR, 2020. 9

Yahui Liu, Enver Sangineto, Wei Bi, Nicu Sebe, Bruno Lepri, and Marco Nadai. Efficient training of visual transformers with small datasets. *Advances in Neural Information Processing Systems*, 34:23818–23830, 2021a. 3, 85

Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. 46, 48, 51, 58

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021b. 97

Simon Lucey. Gradient descent as a shrinkage operator for spectral bias. *arXiv preprint arXiv:2504.18207*, 2025. 1, 71

Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020. 7

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 1, 7, 10, 40

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 3, 37, 38, 40

Ayman Mukhaimar, Ruwan Tennakoon, Chow Yin Lai, Reza Hoseinnezhad, and Alireza Bab-Hadiashar. Pl-net3d: robust 3d object class recognition using geometric models. *IEEE Access*, 7:163757–163766, 2019. 40

Ayman Mukhaimar, Ruwan Tennakoon, Chow Yin Lai, Reza Hoseinnezhad, et al. Robust pooling through the data mode. *arXiv preprint arXiv:2106.10850*, 2021. 40

Alireza Naderi, Thiziri Nait Saada, and Jared Tanner. Mind the gap: a spectral analysis of rank collapse and signal propagation in transformers. *arXiv preprint arXiv:2410.07799*, 2024. 88

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015. 10, 40

Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3504–3515, 2020. 10

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008. 92

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse. *Advances in Neural Information Processing Systems*, 35:27198–27211, 2022. 88

Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. *arXiv preprint arXiv:2011.10379*, 2020. 7

Xuran Pan, Chunjiang Ge, Rui Lu, Shiji Song, Guanfu Chen, Zeyi Huang, and Gao Huang. On the integration of self-attention and convolution, 2021. 86

Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 7

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 92

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017a. 2, 37, 39, 40, 45, 46, 48, 51, 57, 58

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018. 37

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b. 37, 40, 46, 48, 51, 58

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019. 9

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019. 7

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007. 10

Sameera Ramasinghe and Simon Lucey. Beyond periodicity: Towards a unifying framework for activations in coordinate-mlps. In *European Conference on Computer Vision*, pages 142–158. Springer, 2022. 3

Esteban Real, Jonathon Shlens, Stefano Mazzocchi, Xin Pan, and Vincent Vanhoucke. Youtube-boundingboxes: A large high-precision human-annotated data set for object detection in video. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5296–5305, 2017. 21

Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*. John wiley & sons, 2005. 40

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE, 2007. 69

Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *Journal of the ACM (JACM)*, 54(4):12–es, 2007. 12, 69

Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2304–2314, 2019. 10

Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, Fartash Faghri, Devang Naik, Oncel Tuzel, and Mohammad Rastegari. Weight subcloning: direct initialization of transformers using larger pretrained ones. *arXiv preprint arXiv:2312.09299*, 2023. 87

Mohammad Samragh, Iman Mirzadeh, Keivan Alizadeh Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. Scaling smart: Accelerating large language model pre-training with small model initialization. *arXiv preprint arXiv:2409.12903*, 2024. 87

Aditya Sanghi and Pradeep Kumar Jayaraman. How powerful are randomly initialized pointcloud set functions? *arXiv preprint arXiv:2003.05410*, 2020. 40, 41

Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949. 42

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 9

Vighnesh Shiv and Chris Quirk. Novel positional encodings to enable tree-based transformers. *Advances in Neural Information Processing Systems*, 32:12081–12091, 2019. 9

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019. 10

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 3, 67, 68

Maximilian Soelch, Adnan Akhundov, Patrick van der Smagt, and Justin Bayer. On deep set learning and the choice of aggregations. In *International Conference on Artificial Neural Networks*, pages 444–457. Springer, 2019. 38, 40

Kenneth O Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic programming and evolvable machines*, 8(2):131–162, 2007. 10, 40

Jiachen Sun, Qingzhao Zhang, Bhavya Kailkhura, Zhiding Yu, Chaowei Xiao, and Z. Morley Mao. Benchmarking robustness of 3d point cloud recognition against common corruptions. *arXiv preprint arXiv:2201.12296*, 2022. 2, 37, 38, 39, 44, 45, 50, 51, 60, 63, 64

Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 2, 3, 7, 10, 14, 18, 19, 20, 22, 40, 42, 67

Davoud Ataee Tarzanagh, Yingcong Li, Christos Thrampoulidis, and Samet Oymak. Transformers as support vector machines. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2023. URL https://openreview.net/forum?id=gLwzzmh79K. 90

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 37

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 3, 85, 92, 100, 101

Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022. viii, 4, 83, 84, 85, 102

Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. *arXiv preprint arXiv:2305.09828*, 2023. viii, 3, 84, 85, 87, 92, 93, 94, 95, 96, 99, 104

Asher Trockman, Devin Willmott, and J Zico Kolter. Understanding the covariance structure of convolutional filters. *arXiv preprint arXiv:2210.03651*, 2022. 87

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 2, 7, 9, 37, 38, 40, 87

Benyou Wang, Donghao Zhao, Christina Lioma, Qiuchi Li, Peng Zhang, and Jakob Grue Simonsen. Encoding word order in complex embeddings. *arXiv preprint arXiv:1912.12333*, 2019a. 9

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019b. 37, 45, 46, 48, 51, 58

Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. viii, 92, 93, 95, 96, 100, 104

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22–31, 2021. 3, 85, 86

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019. 37, 58

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 39, 44, 45

Shaowen Xie, Hao Zhu, Zhen Liu, Qi Zhang, You Zhou, Xun Cao, and Zhan Ma. Diner: Disorder-invariant implicit neural representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6143–6152, June 2023. 3

Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. Initializing models with larger ones. In *International Conference on Learning Representations (ICLR)*, 2024. viii, 4, 85, 87, 91, 92, 94, 97, 99, 101

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019. 7

Kentaro Yoshioka. vision-transformers-cifar10: Training vision transformers (vit) and related models on cifar-10. https://github.com/kentaroy47/vision-transformers-cifar10, 2024. 97, 102

Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 579–588, 2021. 3, 4, 85, 86

Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19228–19238, 2022. 68

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 3, 85

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017. 39

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 77

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022. 3, 85, 87

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021. 37

Jianqiao Zheng, Sameera Ramasinghe, and Simon Lucey. Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*, 2021. 40, 42, 43, 45

Jianqiao Zheng, Sameera Ramasinghe, Xueqian Li, and Simon Lucey. Trading positional complexity vs deepness in coordinate networks. In *European Conference on Computer Vision*, pages 144–160. Springer, 2022. 3, 38, 40, 42, 58

Ellen D Zhong, Tristan Bepler, Joseph H Davis, and Bonnie Berger. Reconstructing continuous distributions of 3d protein structure from cryo-em images. *arXiv preprint arXiv:1909.05215*, 2019. 2, 7, 10, 40