# Constructive Spiking Neural Networks for Simulations of Neuroplasticity


Toby Asher Lightheart
The University of Adelaide
School of Mechanical Engineering

A thesis submitted for the degree of
*PhD in Engineering*
February, 2018

# Contents

# Abstract

Artificial neural networks are important tools in machine learning and neuroscience; however, a difficult step in their implementation is the selection of the neural network size and structure. This thesis develops fundamental theory on algorithms for constructing neurons in spiking neural networks and simulations of neuroplasticity. This theory is applied in the development of a constructive algorithm based on spike-timing-dependent plasticity (STDP) that achieves continual one-shot learning of hidden spike patterns through neuron construction.

The theoretical developments in this thesis begin with the proposal of a set of definitions of the fundamental components of constructive neural networks. Disagreement in terminology across the literature and a lack of clear definitions and requirements for constructive neural networks is a factor in the poor visibility and fragmentation of research. The proposed definitions are used as the basis for a generalised methodology for decomposing constructive neural networks into components to perform comparisons, design and analysis.

Spiking neuron models are uncommon in constructive neural network literature; however, spiking neurons are common in simulated studies in neuroscience. Spike-timing-dependent construction is proposed as a distinct class of constructive algorithm for spiking neural networks. Past algorithms that perform spike-timing-dependent construction are decomposed into defined components for a detailed critical comparison and found to have limited applicability in simulations of biological neural networks.

This thesis develops concepts and principles for designing constructive algorithms that are compatible with simulations of biological neural networks. Simulations often have orders of magnitude fewer neurons than related biological neural systems; therefore, the neurons in a simulation may be assumed to be a selection or subset of a larger neural system with many neurons not simulated. Neuron construction and pruning may therefore be reinterpreted as the transfer of neurons between sets of simulated neurons and hypothetical neurons in the neural system. Constructive algorithms with a functional equivalence to transferring neurons between sets allow simulated neural networks to maintain biological plausibility while changing size.

The components of a novel constructive algorithm are incrementally developed from the principles for biological plausibility. First, processes for calculating new synapse weights from observed simulation activity and estimates of past STDP are developed and analysed. Second, a method for predicting postsynaptic spike times for synapse

weight calculations through the simulation of a proxy for hypothetical neurons is developed. Finally, spike-dependent conditions for neuron construction and pruning are developed and the processes are combined in a constructive algorithm for simulations of STDP.

Repeating hidden spike patterns can be detected by neurons tuned through STDP; this result is reproduced in STDP simulations with neuron construction. Tuned neurons become unresponsive to other activity, preventing detuning but also preventing neurons from learning new spike patterns. Continual learning is demonstrated through neuron construction with immediate detection of new spike patterns from one-shot predictions of STDP convergence.

Future research may investigate applications of the developed constructive algorithm in neuroscience and machine learning. The developed theory on constructive neural networks and concepts of selective simulation of neurons also provide new directions for future research.

# Declaration

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 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.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

---

Toby Asher Lightheart

# Acknowledgements

First, I would like to acknowledge and thank my principal supervisor, Steven Grainger. Without his support I would not have been able to start or complete my doctoral studies. Steven has given me guidance, but also freedom to follow my interests and creatively pursue my ideas to their limits. This freedom is uncommon in the modern academic environment, so I am very grateful.

I would like to thank my co-supervisor, Tien-Fu Lu, for his perspective and recommendations. I would also like to thank the reviewers of my work submitted to journals and conferences; although inconsistent at times, this feedback had a strong influence on the path my research has taken. Similarly, I am thankful for the open-mindedness of the examiners of this thesis and for their comments and suggestions. I would like to acknowledge the financial and material support of The School of Mechanical Engineering and The University of Adelaide. I thank the School staff, postgraduate coordinators, and Heads of School for their assistance and patience.

I am thankful for the researchers, engineers and developers, paid and unpaid, who have contributed to the science and technology that made this work possible. I acknowledge the use of MATLAB in the experimental simulations performed in this thesis. I would also like to thank the developers and contributors to open source projects, particularly LaTeX and Inkscape, which have been used to typeset this thesis and create diagrams.

The extended duration of my study has meant that I have seen many colleagues come and go and I have made lots of friends. I cannot list them all, but I feel particularly grateful for the friendship of Will, Zebb, Richard, Erwin, Cristobal, Ladan, John, Michael, Eyad, Christian, Malou, Macarena, Jasmin, Claire, Kristy, Branko, Marcus, Tim, Hywel, Isaac, Nick, Stuart, Elias, Mohsen, Danielle, and Jayesh.

I would like to acknowledge the role of basketball in bringing me to Adelaide. My doctoral studies won over my basketball aspirations, but basketball has continued to provide physical and mental exercise and respite from thesis revisions. I would like to thank Scott; he took a chance on me and helped me move across the country. I would also like to thank the friends I have made at the Adelaide University Basketball Club, in particular Daniel, Chris, Kelly, Alex, and Matthew.

Most importantly, I would like to acknowledge the support of my family and my partner. I am thankful for the love and support of my brother Robert and sister Alana, my aunts Anne, Jenny, and Kathryn, my stepdad Peter, my sister-in-law Joanna, my

uncles Paul and Mike, my cousins and nephews, and the loving family of my partner. I would like to give a special thanks to my brother Jonathan. His company and friendship made Adelaide a home and helped me stay to complete my thesis. I feel tremendously lucky to have such loving, supportive and interested parents, Andrej and Edwina. Mum and Dad, I can only hope that you both feel the depth of my gratitude and love.

To my partner, Ruth, your love, friendship, fun, encouragement and generosity has an inestimable significance in my life. I am sincerely grateful for everything and I love you dearly. Our next adventures await!

*For my Grandpa George.*
*I remember your pride and happiness when I started.*
*I wish this was completed before you left.*

# Chapter 1

# Introduction

Artificial neural networks are developed to perform machine learning and to model biological neural systems for neuroscientific study. Criteria for the effectiveness of an artificial neural network (ANN) depend on the research field and application. An ANN may be considered an effective model in computational neuroscience if it sufficiently describes and predicts some behaviour of a biological neural system or represents an underlying computational process. The effectiveness of an ANN in machine learning and artificial intelligence depends on its performance in a given task, such as the detection or classification of patterns. Many machine learning tasks can be described in general terms as the prediction of output values from input data. In both fields, open-ended research questions in the development and application of ANNs include:

1. What are effective models of network components?
2. What are effective neural network sizes and structures?

In the field of machine learning, constructive algorithms and pruning algorithms have been developed to adapt the neural network size by adding or removing neurons and synapses. However, past developments of constructive algorithms have lacked a standard theoretical foundation, resulting in diffuse literature and varied terminology. Furthermore, the past development of constructive algorithms has not considered compatibility or effectiveness in simulations of biological neural networks. This thesis develops theory and methods for the automatic construction and structural adaptation of neural networks for computational neuroscience and machine learning.

Algorithms and theory for the automatic construction of spiking neural networks are developed in this thesis for compatibility with simulations of biology. The developed constructive algorithms incorporate models of spike-timing-dependent plasticity (STDP) and successfully reproduce learning capabilities for the detection of hidden spike patterns. Neuron construction is demonstrated to be capable of performing one-shot learning and avoids overwriting connections and prior learning. In addition to applications in computational neuroscience, the theory and algorithms developed may be applied to machine learning. This introduction provides a brief background of important concepts and terminology and an overview of the thesis structure.

## 1.1 Background

The developments presented in this thesis draw from topics in machine learning and computational neuroscience. An overview of these topics, models, concepts and terminology is provided here to give context for the contributions of this thesis.

### 1.1.1 Machine Learning and Computational Neuroscience

Machine learning and computational neuroscience have different goals and requirements that guide the selection and design of neural network models. Task performance is a primary concern in machine learning research (Mitchell et al., 1990). Purely theoretical developments and models inspired by biology may be of interest to the research community; however, these developments are unlikely to have a significant influence in machine learning research or practice unless they produce improvements task performance. Development of ANNs for machine learning is often done without concern for biologically plausibility and, therefore, may have limited or no applicability in neural network simulations for studies in neuroscience.

Computational neuroscience is principally concerned with the identification and reproduction of the computational processes of biological nervous systems. This may be approached through the development and simulation of models representing networks of biological neurons and the investigation of the computational processes that result. In this case, it is desirable that the models simulated are biologically plausible; however, it is also desirable to find the minimal set of model features to reproduce the computational capabilities of the biological system. It remains an ongoing effort in neuroscience to determine what aspects of the function and structure of biological neural networks are necessary to perform equivalent computational processes and what may be omitted in models.

Despite different goals, a finding in one field may encourage an investigation in the other and there are cases of convergent findings. For example, low-level features in artificial neural networks for computer vision may resemble the features detected by biological neurons in the primary visual cortex (Serre, Wolf, Bileschi, Riesenhuber, & Poggio, 2007). Nevertheless, the difference in goals in the fields of machine learning and computational neuroscience can lead to a divergence in approaches. For example, error backpropagation is a standard method for supervised machine learning (Rumelhart, Hinton, & Williams, 1986; Schmidhuber, 2015); however, backprop is not widely considered to be biologically plausible and is not commonly implemented in simulations for neuroscience.

### 1.1.2 Neural Networks

While ANNs are used in machine learning and in computational neuroscience, the features and terminology may vary. This section describes the basic biological structure and function of neurons in networks and some standard terminology in computational neuroscience and machine learning.

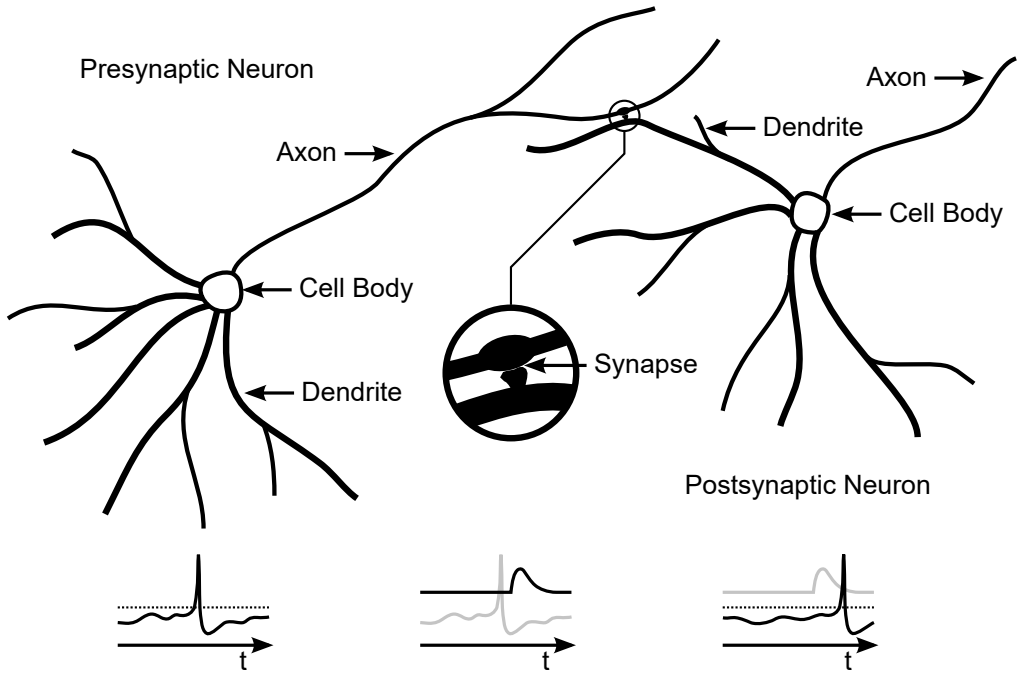


**Figure 1.1:** A simplified diagram of connected biological neurons and spike transmission. An increase in the presynaptic neuron charge can cause a sharp spike in the neuron membrane voltage (left). This voltage spike is transmitted down the axon and triggers a release of neurotransmitters across the synapse (middle). The synaptic transmission to the postsynaptic neuron may produce an increase in the membrane voltage that may lead to a postsynaptic spike (right). (See Gerstner & Kistler, 2002, for more detailed descriptions.)

The fundamental components of a neural network model are the neurons and the connections between neurons. Neurons transmit signals to neurons down branching structures called axons and commonly receive signals from many other neurons through branching structures called dendrites (Figure 1.1). The connection of an axon to any part of a neuron is a synapse. The build up of charge in a neuron can trigger a short spike (around 1-2 ms) in the neuron potential that triggers the release of neurotransmitters at the axon terminals (Gerstner & Kistler, 2002). The neurotransmitters are received or taken up by the dendrite, eliciting a change in the electrical charge of the receiving neuron. In artificial neural networks, the chemical or electrical properties of the synapse are often simplified to a single factor of amplification or attenuation called a weight.

Neurons are often identified with respect to their relationship with the synapse as either being the presynaptic neuron (emitting neurotransmitters) or the postsynaptic neuron (receiving neurotransmitters). A presynaptic neuron spike may induce an excitatory postsynaptic potential (EPSP), increasing the chance of a postsynaptic spike, or an inhibitory postsynaptic potential (IPSP), decreasing the chance of a postsynaptic spike, depending on the neurotransmitter that the presynaptic neuron releases (Dayan & Abbott, 2001; Gerstner & Kistler, 2002).

Biological brains have many regions with order in the spatial distribution and function of neurons and connections (Blumenfeld, 2010); nuclei (groups of similar neurons) and layers are common neural structures. Neuron connections may progress in a forward stream from one layer or group to another. Neurons can also have recurrent connections, that is, connections backwards to neurons in earlier layers or lateral connections to neurons in the same layer or group.

Spiking neuron models may have varying levels of detail from modelling of ion flow to simplified representations of neuron cell membrane voltage (Gerstner & Kistler, 2002; Izhikevich, 2004). Increasing neuron model detail, however, results in significant computational expense and may decrease the number of neurons that it is feasible to simulate. Nevertheless, with continued improvements in computing power, it has become possible to simulate neural networks composed of spiking neuron models, and these simulations are now commonplace in computational neuroscience research.

Research has also applied spiking neural networks and learning algorithms in robotics (Arena, Fortuna, Frasca, & Patané, 2009) and machine learning (Lee, Delbruck, & Pfeiffer, 2016). Theoretical findings have shown that networks of spiking neurons have an advantage in computational power over networks of time-independent sigmoid neurons (Maass, 1997). Time-dependent neuron and network models can intuitively be expected to have intrinsic capabilities for solving time-dependent learning tasks. Despite these potential advantages, spiking neural networks have received significantly less attention than networks of time-independent neurons from the machine learning community.

Machine learning is largely dominated by time-independent neuron models which are computationally inexpensive, have well-studied and effective learning algorithms, and form the basis of many state-of-the-art learning systems (Schmidhuber, 2015). Recently, many advances in machine learning performance have been the result of a combination of improved computing power and deep neural networks, that is, networks with many layers, neurons and connections (Figure 1.2). Spiking neural networks lack similar standards for models and training algorithms but may be suitable for tasks outside of the present standards of machine learning.

### 1.1.3 Training and Plasticity

Iterative training algorithms make incremental changes to the network parameters to detect patterns in data. Training algorithms are a central component in many machine learning implementations of ANNs, including deep neural networks (Schmidhuber, 2015). Different methods of ANN training are available depending on the type of learning task: two broad learning task categories are supervised learning and unsupervised learning.

Supervised learning tasks have training input samples paired with desired outputs, allowing the calculation of the network output error. The aim of supervised learning is to correctly predict desired outputs in training and generalise to make successful output predictions for new input samples. Many supervised learning methods are variants of error backpropagation (Hagan & Menhaj, 1994; Rumelhart et al., 1986; Schmidhuber, 2015), often referred to as backprop. Backprop begins with the forward calculation of

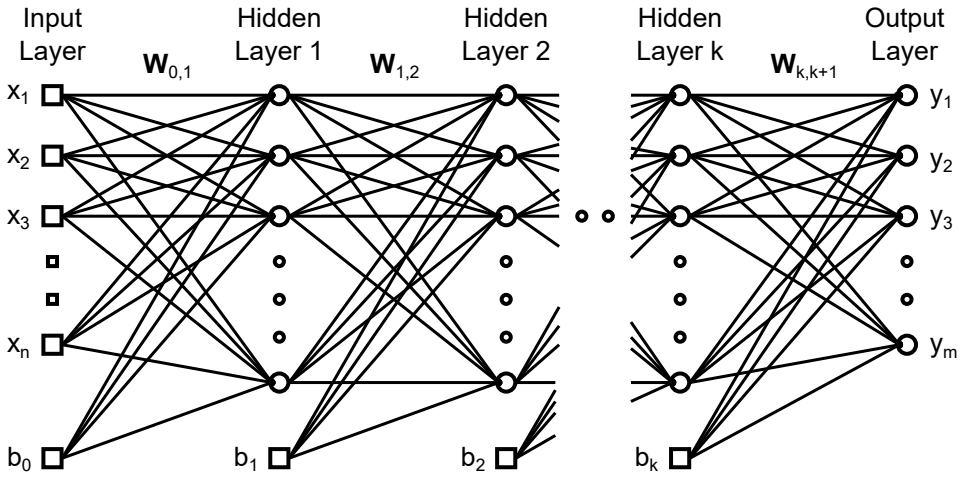


**Figure 1.2:** A general diagram for the architecture of a deep feed-forward neural network or multilayer perceptron. The input layer receives a vector of numerical values ($\mathbf{x} \in \mathbb{R}^n$). The output of each subsequent node is a function of the sum of weighted inputs to that node; each layer may have a different number of nodes. Upon computing the output values of all of the nodes, a vector of numerical values ($\mathbf{y} \in \mathbb{R}^m$) can be read from the output layer. Learning algorithms are used to adjust the matrices of weights ($\mathbf{W}$) between layers that includes weighted input from a constant value bias node ($b$).

neuron activations, network outputs and the cost of output errors. Then calculations to estimate the change in error from changing individual connection weights are performed backwards through the network from the output neurons to the input. The connection weights are then updated using these estimates and the next forward calculations are performed.

There are many approaches to performing backprop training (Montavon, Orr, & Müller, 2012). Estimating changes in error from changes in connection weights may use gradients (first-order derivatives) or include an estimate or calculation of the Hessian matrix (second-order derivatives). Training algorithms may require processing the entire training set (batch training) before updating the connection weights or may perform stochastic updates, based on subsets or mini-batches of the training data. Stochastic gradient descent (SGD) has been found to be effective and memory efficient for use in deep neural networks (Goyal et al., 2017; Ioffe & Szegedy, 2015; Montavon et al., 2012; Sutskever, Martens, Dahl, & Hinton, 2013).

Unsupervised learning algorithms detect patterns in input data without any associated desired output data or other external feedback. A collection of unsupervised learning algorithms for ANNs are related to Hebb's Rule, which is often phrased as: neurons that fire together, wire together. Formalisations of Hebbian learning with weight normalisation have been found to perform an equivalent operation to principal component analysis (Oja, 1982). Competitive learning (Kohonen, 1982) is also related to Hebbian learning with the addition of lateral inhibition or depression of neurons that do not have the strongest response. Many unsupervised learning algorithms, including

competitive Hebbian learning (Kohonen, 1982), may be used as models of biological learning. Deep neural networks have also been found to have improved performance outcomes with a pre-training phase of iterative unsupervised learning algorithms for regenerating network inputs (Erhan et al., 2010).

Success in training deep neural networks has resulted in state-of-the-art performance in a wide range of machine learning tasks (Schmidhuber, 2015). Nevertheless, increasing neural network sizes increases the capability of the network to overfit the training data (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). In addition to evaluating training on a validation data set and stopping when overfitting is observed, the need to prevent overfitting has spurred the adoption of processes such as data augmentation and neuron drop-out (Krizhevsky, Sutskever, & Hinton, 2012; Srivastava et al., 2014). Data augmentation is performed on training data samples to create larger and more varied training data sets. Neuron drop-out is a process of randomly selecting neurons to temporarily exclude during training, resulting in improved robustness from neurons distributing the handling of processing.

The improvements in performance from deep neural networks have come with significant computational costs and training times to handle the enormous data sets and to train large numbers of network parameters. Techniques for parallelisation of deep neural network training can spread the computational cost over large numbers of processing units and substantially reduce training times (Goyal et al., 2017). Nevertheless, the computational cost and time to train deep neural networks impede their development.

The low-level features of a trained deep neural network may be capable of being reused or transferred for learning other similar tasks, for example in image recognition (Yosinski, Clune, Bengio, & Lipson, 2014). Once a network is trained for one task, training the network for a second task can overwrite previous training results and cause 'catastrophic forgetting' (Goodfellow, Mirza, Xaio, Courville, & Bengio, 2013). Determining important connections and holding their weights constant can improve retention when sequentially learning tasks (Kirkpatrick et al., 2017); however, the size of the neural network will constrain the total number of tasks that can be learned. Algorithms that increase the number of neurons and connections are developed in this thesis as an approach to continual learning.

Unlike time-independent neuron models used in deep neural networks that are primarily trained through backprop, spiking neural networks do not have a similar standard training algorithm. Supervised training algorithms for spiking neural networks have been developed (Gardner & Grüning, 2016; Kasiński & Ponulak, 2006; Ponulak & Kasiski, 2010). Modifications to allow backprop to be applied to spiking networks have also been developed (Lee et al., 2016); however, the biological plausibility of backpropagation has long been disputed (Lillicrap, Cownden, Tweed, & Akerman, 2016). Given spiking neural networks are often implemented for their greater biological accuracy, biologically plausible training methods are often preferred.

The biological equivalent of training algorithms are processes that modify the efficacy of synaptic transmissions, also referred to as processes of synaptic plasticity or neuroplasticity. Biological observations of spike-timing-dependent plasticity (STDP)

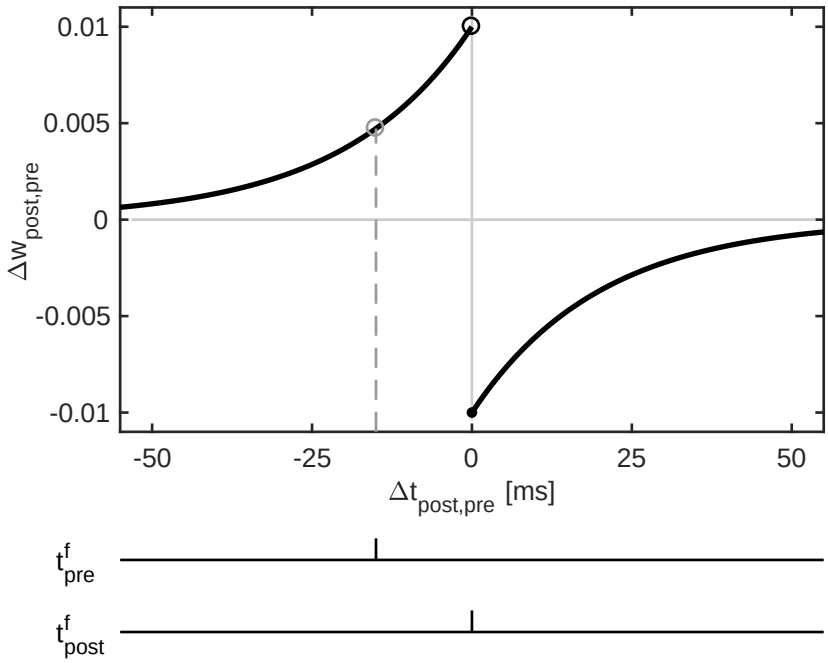


**Figure 1.3:** A graphical representation of a common spike-timing-dependent plasticity curve shape. A presynaptic neuron spike time ($t^f_{\text{pre}}$) and a postsynaptic neuron spike time ($t^f_{\text{post}}$) are represented as spikes on the respective horizontal lines. The synaptic weight update produced by this relative spike timing is indicated with a dashed line. Note that this idealised STDP model has a discontinuity when the presynaptic neuron and postsynaptic neuron spike simultaneously; here, equal spike times produce a negative weight update.

have been the basis for the development of models of unsupervised learning for spiking neural networks (Bi & Poo, 1998; Caporale & Dan, 2008; Morrison, Diesmann, & Gerstner, 2008). As the name implies, the updates to the synapse weight are a function of the relative timing of spikes of the presynaptic neuron and postsynaptic neuron. In the standard model of STDP, synapses are potentiated (increase weight) when the presynaptic neuron spike precedes the postsynaptic neuron spike (Figure 1.3). Synapses are depressed (decrease weight) when the presynaptic neuron spike occurs shortly after postsynaptic neuron spike.

Studies have focused on improving the biological accuracy of STDP models (Morrison et al., 2008; Pfister & Gerstner, 2006) and investigating the learning outcomes and capabilities of these models (Abbott & Nelson, 2000; Legenstein, Naeger, & Maass, 2005; Song, Miller, & Abbott, 2000). Spike-timing-dependent plasticity models are capable of tuning neuron weights to detect the start of repeating spike patterns even when the majority of the input is noise (Masquelier, Guyonneau, & Thorpe, 2008, 2009). The tuning through STDP has also been demonstrated in simulations to spontaneously develop important neural responses in biological vision: motion- or direction-selective cells (Wenisch, Noll, & Hemmen, 2005) and orientation-selective cells (Masquelier, 2012).

Models of STDP have been implemented to perform learning for robotics and artificial intelligence tasks (Arena et al., 2009; Bouganis & Shanahan, 2010; Masquelier & Thorpe, 2007). The demonstrated learning capabilities and simplicity of STDP have been strong factors in its adoption as a model of biological learning.

Developing successful demonstrations of biological learning models or machine learning systems may be informed by past developments, but ultimately, new developments often appear to be a matter of intuition and trial-and-error. The training algorithm or model of plasticity is a part of that development; however, the learning performance is also intrinsically linked to the size and structure of the neural network. Reports of successful novel deep neural network designs often provide a thorough description of the neural network structure, but the design process that led to that structure is often not communicated. For complex perception tasks, there is still insufficient understanding to accurately predict the performance of a given network size and structure prior to training. Known structures in biological neural networks may be used to guide the arrangements and ratios of neurons and connections in models of biology, but the choice of the number of neurons and connections to simulate may remain a matter of prediction, estimation, and trial-and-error refinements.

In the past, methods have been developed for automating the selection of the neural network size by incrementally increasing the number of neurons and connections in an artificial neural network (Ash, 1989; Fahlman & Lebiere, 1990; Fritzke, 1995). Constructive algorithms have shown some success and promise in recent developments (Fukushima, 2014; Schliebs & Kasabov, 2013); however, constructive neural networks have had limited adoption in machine learning and computational neuroscience.

### 1.1.4 Constructive Neural Networks and Growth

A constructive neural network incorporates algorithms that can add neurons and connections into the operation of an artificial neural network. Algorithms for constructing neural networks have been applied to machine learning tasks to automatically find a network structure that meets performance requirements (Nicoletti, Bertini, Elizondo, Franco, & Jerez, 2009). Early constructive neural network developments were spurred by the challenge of selecting hidden layer sizes for multilayer perceptrons. Figure 1.4 gives a simplified diagrammatic representation of Dynamic Node Creation (Ash, 1989), an early constructive algorithm for multilayer perceptrons.

Despite common objectives, the constructive neural network literature is diffuse and without a common theoretical foundation. This is apparent in the range of terms used in literature, where networks that add neurons and connections are also commonly referred to as growing neural networks (Fritzke, 1995; Huang, Saratchandran, & Sundararajan, 2005) and evolving connectionist systems (Watts, 2009). The terms ‘growing’ and ‘evolving’ have biological interpretations that can introduce additional confusion in the contexts of neural networks and computer science. Therefore, this thesis will use the phrases ‘constructive algorithm’ and ‘constructive neural network’.

Constructive algorithms for deep neural networks have received little attention (exceptions include: Fukushima, 2014; Mundt, Weis, Konda, & Ramesh, 2017). Deep

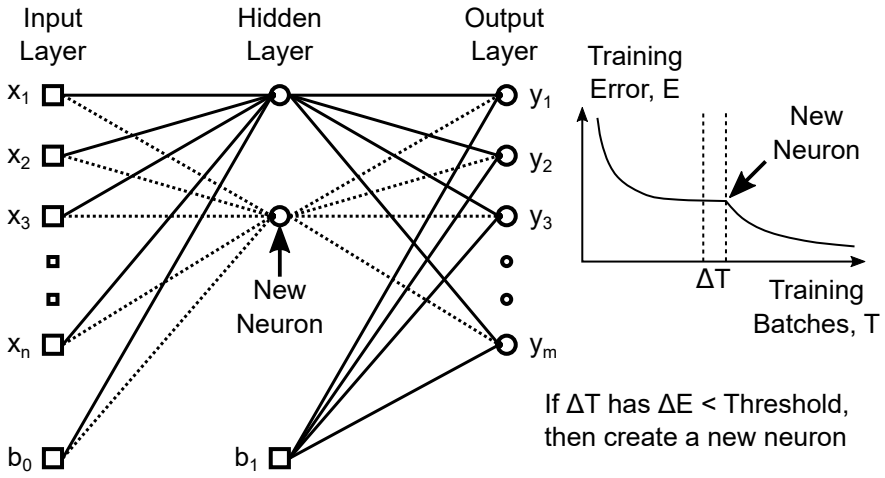


**Figure 1.4:** A simplified diagram of the process of Dynamic Node Creation (Ash, 1989). A multilayer perceptron is trained using backprop until a stopping condition. If the change in error ($\Delta E$) falls below a threshold then a new neuron is added to the hidden layer with full connections from the input layer and to the output layer.

neural networks are often large enough to contain superfluous connections after training; as a result, algorithms for compressing and pruning neurons and connections have received attention for reducing memory and computational costs of operating trained deep neural networks (Han, Mao, & Dally, 2015). However, pruning algorithms do not, in general, aid in selecting efficient initial neural network sizes or speed up training.

Constructive neural networks have strengths that may complement the iterative training of connection weights. Constructive algorithms may speed up training by immediately correcting poor performance through the construction of new neurons and connections. Network construction can be performed without modifying existing connection weights; therefore, construction could be expected to have a reduced likelihood of causing interference or catastrophic forgetting. If constructive algorithms can be successful demonstrated to achieve these capabilities, then these algorithms may contribute to the development of systems for on-line continual learning.

Constructive algorithms can provide strict rules for creating neurons and synapses based on the performance of the network; therefore, the burden of hand-designing neural network structures may be significantly reduced and replaced with the design of constructive algorithms. Automated approaches to selecting neural network sizes can also provide estimates of task complexity (Mundt et al., 2017). At present however, the lack of standard terminology for constructive neural networks also manifests as a lack of proposals for general design methodologies and standard processes of constructive algorithms. As a result, literature often presents constructive algorithms without a clear description of the similarity of algorithm processes to those of other algorithms or providing rigorous analysis of algorithm variations.

A number of constructive neural networks have been developed to operate with specific spiking neuron models (these will be examined in detail in the Chapter 2).

However, the constructive algorithms for spiking neural networks found in literature (Atsumi, 2005; Takita & Hagiwara, 2005; Wysoski, Benuskova, & Kasabov, 2006) were developed in the context of machine learning and, therefore, have been developed with little or no concern for their use in simulations of biology. No investigations or proposals for constructive algorithms compatible with biologically plausible simulations of neural networks have been found in prior literature.

Biological mechanisms for growing new neurons (neurogenesis) and new synapses (synaptogenesis) have been observed in mammalian brains and have been modelled (Aimone et al., 2014; Butz, Lehmann, Dammasch, & Teuchert-Noodt, 2006; Takizawa, Hiroi, & Funahashi, 2012). Detailed models of neurogenesis and synaptogenesis may be incorporated into simulations of biological neural systems. However, neural growth processes in mature neural networks are often slow and small in scale; therefore, models of biological growth processes may have limited applicability in automating the selection and adaptation of neural network sizes and structures.

## 1.2 Summary of Thesis

This thesis makes a number of original contributions in theory, algorithms and experiments using constructive neural networks for simulations of biological neural networks. These contributions and their significance are summarised here, then the structure of the thesis is outlined and publications arising from this work are listed.

### 1.2.1 Contributions and Significance

The first contribution of this thesis is in the analysis and design of constructive neural networks. This thesis defines a set of standard components in constructive neural networks then uses these definitions as the basis of a methodology for the decomposition, comparison and design of constructive algorithms. This methodology is demonstrated in this thesis in the review of literature and the design of novel constructive algorithms. Additionally, spike-timing-dependent construction is proposed as a distinct class of constructive algorithms with processes dependent on the timing of neuron spikes.

This contribution is significant to the research and design of constructive neural networks. Past literature often treats constructive algorithms as indivisible rather than as collections of processes with some standard components. As a result, comparisons of variations in the processes of a constructive algorithm are uncommon in literature. Without adequate decomposition of constructive algorithms into base processes, incremental development and investigation cannot occur. This hinders the discovery of general rules and factors in the performance of constructive algorithms. The identification of standard components in constructive algorithms also provides a guide to improving the consensus on terminology; otherwise, conditions will remain challenging for researchers to maintain awareness of similar research efforts.

The second contribution of this thesis is in the introduction and development of concepts of neural network simulation expansion and contraction. Important distinc-

tions between construction and expansion (and between contraction and pruning) in simulated neural networks are proposed and elucidated. From the basic concepts of simulation expansion and contraction, principles and assumptions for the design of constructive algorithms that are compatible with simulations of biological neural networks are developed.

The concepts of simulation expansion and contraction provide a new approach to the design of algorithms that dynamically select the size of a neural network for machine learning and computational neuroscience. Of additional significance for studies in computational neuroscience, no other constructive algorithms for spiking neural networks found in literature have claimed to be compatible with simulations of biological neural networks. Constructive algorithms can provide an objective approach to neural network size selection and shift the design effort from trial-and-error testing of network sizes to the selection of constructive algorithm processes and parameters. A neural network simulation that incorporates a constructive algorithm may accommodate greater changes in simulation conditions than a non-constructive simulation, without the need to cease and redesign the network structure.

The third contribution is in the development of novel constructive algorithm processes. The constructive algorithms processes are based on the standard components proposed and principles of simulation expansion for compatibility with simulated studies of neuroplasticity. Processes for calculating synapse weights for new neurons are developed from models of spike-timing-dependent plasticity (STDP). A process for evaluating the neural network performance and triggering neuron construction is also developed from the assumption that the simulated neural network exists within a larger hypothetical neural system. A neuron is simulated as a proxy for neurons that are not simulated but exist in the larger neural system. The activity of this neuron is controlled to trigger neuron construction and provide postsynaptic spike time estimates for calculating new synapse weights.

The fourth contribution is in the successful demonstration of the novel constructive algorithms in simulations of pattern detection through STDP. Constructive algorithms are applied to reproductions of a simulated study of STDP that tune neurons to detect a repeating hidden spike pattern (Masquelier et al., 2008) and a study of STDP resulting in neurons competing to detect repeating hidden spike patterns (Masquelier et al., 2009). These past studies present challenging conditions for the constructive algorithm; however, constructed neurons produces comparable rates of learning performance, demonstrating compatibility with the simulated studies of STDP. Extensions to the simulations of competitive pattern detection demonstrate that the constructive algorithm can perform continual learning in long simulations with new hidden spike patterns learned in one shot by constructing neurons.

The results from simulations strengthen the case for the compatibility of constructive algorithms in simulated models of biological learning. The advantages of objective approaches to neural network size selection and dynamic structural responses to changes in simulation conditions are demonstrated: the simulation responds automatically to new spike patterns in a continuous network simulation without interfering with previ-

ously tuned connections. This suggests that constructive algorithms developed using the concepts and approaches presented in this thesis may have broad applications in simulated studies in computational neuroscience.

The success of the developed constructive algorithm in biologically-related learning tasks is expected to be transferable to applications in machine learning. Data values in machine learning tasks may be represented as neuron spike times with similar properties to the spike patterns in presented simulations. The constructive algorithm demonstrated the capability to learn new patterns in one shot without the modification of existing neurons. This indicates that constructive algorithms may be a successful approach to speeding up learning convergence and avoiding catastrophic forgetting.

### 1.2.2 Structure

This thesis is structured to present the theoretical developments in a progressive and logical manner that is then followed by empirical validation and investigations of constructive algorithm performance. The thesis introduction chapter (Chapter 1) is followed by a review of literature (Chapter 2). Definitions and fundamental processes of constructive algorithms are first proposed and then used in the review of constructive spiking neural network literature. The definition of spike-timing-dependent construction is also developed and applied in the examination of constructive spiking neural networks.

Following the review of literature, concepts and principles for neural network simulation expansion and contraction are proposed, developed and contrasted with definitions of network construction and pruning in Chapter 3 (Simulation Expansion and STDP). The assumptions and principles for simulation expansion are applied to the development of STDP estimation methods for calculating synapse weights for new neurons. The developed methods for synapse weight estimation are numerically validated in Chapter 4 (Validation of STDP Estimation). This chapter presents a study that empirically confirms the correctness of STDP estimation equations for the given assumptions and then presents a quantitative analysis of STDP estimate sensitivity to input spike noise.

In Chapter 5 (Spike Prediction with Proxy Neurons) methods are presented for simulating neurons to predict effective construction times and to predict postsynaptic spike times for STDP estimation. Assuming the simulated neural network is a subset of a larger neural system, the neuron simulated to trigger construction is interpreted as a proxy of neurons outside the simulation subset. The results of proxy neuron spike-triggered construction are examined for input spike patterns with varying levels of noise.

Constructive algorithms with a variety of STDP estimation processes are evaluated in simulations based on a past study of hidden spike pattern detection through STDP (Masquelier et al., 2008); details and results of this simulated study are presented in Chapter 6 (STDP Simulation with Neuron Construction). The constructive algorithms are refined from findings and applied to simulations of the competitive detection of multiple hidden spike patterns through STDP (Masquelier et al., 2009); details and

results of this simulated study are presented in Chapter 7 (Continual Learning of Spike Patterns). The thesis concludes with a discussion (Chapter 8), which reviews the original contributions and findings presented in this thesis, discusses the significance and broader context of developments, and proposes future directions of research.

### 1.2.3 Publications

At the time of submission, a number of publications have been produced in the process of developing the work presented in this thesis:

- T. Lightheart, S. Grainger, and T.-F. Lu (2010). A constructive spiking neural network for reinforcement learning in autonomous control, *Australasian Conference on Robotics and Automation 2010*, Brisbane, Australia.
- T. Lightheart, S. Grainger, and T.-F. Lu (2011). Constructive network reinforcement learning for autonomous mobile robots, *International Conference on Mechatronics Technology 2011*, Melbourne, Australia.
- T. Lightheart, S. Grainger, and T.-F. Lu (2013). Spike-timing-dependent construction, *Neural Computation*, vol. 25(10), pp. 2611–2645.
- T. Lightheart, S. Grainger, and T.-F. Lu (2017). Continual one-shot learning of hidden spike-patterns with neural network simulation expansion and STDP convergence predictions, *ArXiv e-prints*, `https://arxiv.org/abs/1708.09072`.

The conference papers 'A constructive spiking neural network for reinforcement learning autonomous control' (Lightheart, Grainger, & Lu, 2010) and 'Constructive network reinforcement learning in autonomous mobile robots' (Lightheart, Grainger, & Lu, 2011) present early conceptual developments that lead to the work presented in this thesis. The contents of the journal paper 'Spike-timing-dependent construction' (Lightheart, Grainger, & Lu, 2013) have been developed and extended into Chapters 2 and 3. The contents of the journal paper 'Continual one-shot learning of hidden spike-patterns with neural network simulation expansion and STDP convergence predictions' have been adapted from Chapter 7.

# Chapter 2

# Theory and Review of Literature

This chapter proposes definitions for the essential components and algorithm steps of constructive neural networks and applies these definitions in the review of constructive spiking neural networks. Spike-timing-dependent construction is proposed as a classification of constructive algorithms that have processes dependent on neuron spike times. Constructive spiking neural networks that perform spike-timing-dependent construction are found in literature and are decomposed into the defined components and processes for detailed comparison. This chapter concludes with a summary of the frontier of research in constructive neural networks and the specific gaps and limitations in current constructive spiking neural networks treated in this thesis.

## 2.1 Constructive Neural Network Theory

This section proposes definitions of constructive neural network components and constructive algorithm steps and processes. Many algorithms result in the change of the network structure; however, past literature often emphasises the differences in constructive neural networks, presenting a diverse range of networks and algorithms with similarly diverse terminology. No past literature has been found that provides general definitions of the essential components, algorithm steps and processes required for a constructive neural network. The comparison, development and analysis of constructive neural networks performed in this thesis is based on the definitions of components and processes proposed here.

### 2.1.1 Definitions of Components and Processes

The phrase 'artificial neural network' is common in machine learning contexts but is less commonly applied to neural network simulations in neuroscience. Here, this term will be used more broadly to include computer-simulations of biological neural networks in neuroscience.

**Definition 1** *An artificial neural network (ANN) is a set of non-biological neurons that receive input signals and produce output signals.*

Non-biological neurons can include those created in a range of media, but this thesis focuses on ANNs implemented and calculated using general purpose computers.

ANNs with algorithms that add neurons and connections have been found in literature under a variety of names, including constructive neural network (Nicoletti et al., 2009), evolving connectionist system (Schliebs & Kasabov, 2013; Watts, 2009), and growing neural network (Fritzke, 1995; Huang et al., 2005). Structural plasticity (Roy & Basu, 2017) and adaptive structure (Wang, Belatreche, Maguire, & McGinnity, 2017) are recent examples of algorithms that add synapses or neurons but do not use common terminology. The use of non-biological terms and phrases when the creation of neurons or synapses does not model a biological process has the advantage of avoiding confusion with the biological meanings of terms, such as growing or evolving. The literature more consistently refers to algorithms that remove neurons and connections as pruning algorithms (Han et al., 2015; Reed, 1993).

Variations in terminology for research with similar methods and objectives reduces the visibility of that research and complicates the comparison of developments and findings. The adoption of standard terminology may improve the organisation and progress of research. In this thesis, the primary phrases used are 'constructive algorithm' and 'constructive neural network'. A broad definition of the phrase constructive algorithm is proposed here:

**Definition 2** *A constructive algorithm is any algorithm that can result in the creation of new synapses or neurons in an artificial neural network.*

This definition of 'constructive algorithm' can be used inclusively to cover algorithms that can remove synapses or neurons as long as they also have processes for creating synapses or neurons, for example, the evolving spiking neural network algorithm (Wysoski, Benuskova, & Kasabov, 2010). Algorithms that only remove neurons or synapses (Reed, 1993) will continue to be referred to as pruning algorithms. This thesis focuses on neuron construction; however, discussions of pruning and merging are also included.

A constructive algorithm, like other algorithms, defines a sequence of steps or processes. The definition of a constructive neural network proposed here is based on the parallel or sequential operation of the constructive algorithm and the ANN:

**Definition 3** *A constructive neural network is an ANN that has processes of a constructive algorithm performed during or between periods of the ANN operation or training.*

Constructive neural networks have periodic or ongoing constructive algorithm operation; therefore, the ANN structure may undergo multiple, incremental modifications. An algorithm that creates a complete neural network in one shot may be described as a constructive algorithm, for example, an Extreme Learning Machine (Huang, Wang, & Lan, 2011); however, unless the constructive algorithm continues to operate the ANN would not fit under this definition of a constructive neural network. The proposed definition does allow the ANN to be adapted through training or operation in addition to the modifications resulting from the constructive algorithm.

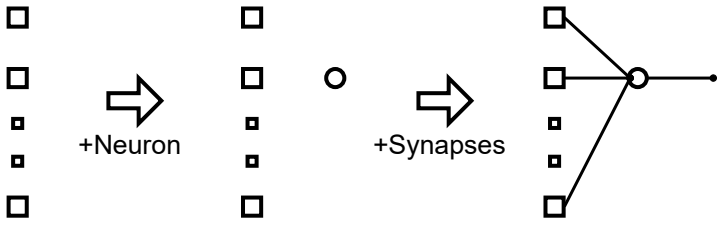


**Figure 2.1:** A diagram depicting the creation of a neuron (circle) and the creation of synapses (lines). Many neuron models require input or synapses to function; therefore, the creation of a neuron typically includes synapses. Synapses may be added as an independent operation. Pruning is the reverse operation: removing synapses or neurons.

Incremental modifications of the ANN structure can take the form of the creation, pruning or merging of synapses or neurons (Figures 2.1 and 2.2). Each of these incremental modifications of the network structure requires one or more sets of processes:

- creating synapses or neurons requires selecting the neurons to connect with new synapses and requires parameter values for new synapses and neurons to be provided;
- pruning requires selecting the synapses or neurons that will be removed;
- merging requires selecting sets of synapses or neurons that will be combined and requires the final parameters values to be provided.

The requirements of these structure modifications can be grouped into sets of fundamental algorithm processes:

- determining when or if the ANN structure will be modified,
- selecting the affected network components, and
- selecting or calculating parameter values (creation and merging).

These sets of processes may be composed of a number of algorithm steps. There is no strict requirement for the order in which the processes or their steps are performed. Examples from literature include: Dynamic Node Creation (Ash, 1989), which first determines when construction should be performed by detecting plateaus in training performance, and then selects new synapse weights; and evolving spiking neural networks (Wysoski et al., 2010), which first calculate parameter values for a new neuron, and then determine whether that new neuron should be added to the network.

The constructive algorithm processes may be independent of the present state or operation of the constructive neural network; however, the high-level aim of implementing a constructive algorithm is to improve the capabilities and performance of the ANN. Performance calculations and criteria are important in the evaluation of ANNs in machine learning and neuroscience. In the context of constructive algorithms, the phrase ‘performance evaluation process’ can be broadly defined:

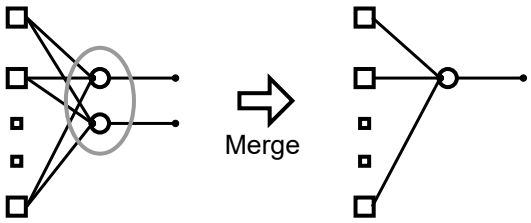


**Figure 2.2:** A diagram depicting the merging of two neurons (circles) into a single neuron. The resulting neuron and synapses have parameter values calculated as a combination of the past values.

**Definition 4** *A performance evaluation process is any set of algorithm steps that produce an assessment of the state or operation of the constructive neural network.*

The constructive neural network includes the ANN and the constructive algorithm; therefore, this definition of a performance evaluation process includes assessments of the ANN structure, parameters, operation and training, and assessments of the results of other constructive algorithm processes. Performance evaluation processes found in literature include conditions on training error (Ash, 1989; Fahlman & Lebiere, 1990), vector distances of parameters (Huang et al., 2005; Platt, 1991), and network component significance (Reed, 1993).

The assessments provided by performance evaluation processes can used in each of the fundamental constructive algorithm processes: determining when the ANN structure should be modified, what components should be affected, and what parameter values should be selected. Given that creation, pruning and merging are binary events (the structure change occurs or does not occur), performance evaluation processes typically culminate with a decision to perform construction using a threshold on performance values. A distinction can be made between performance values of the entire ANN, referred to here as a global performance, and performance values of individual components within the ANN, referred to here as local performance.

Global performance values and conditions do not directly indicate if specific components are deficient; therefore, the selection of locations for new components typically follows a predefined process (this process may be stochastic). This is the case in Dynamic Node Creation (Ash, 1989) and Cascade-Correlation (Fahlman & Lebiere, 1990), which have training error taken as the global performance, and then use a performance threshold and predefined process for determining the location to add neurons and connections. Local performance values and conditions do indicate which components have high or low performance; therefore, components with low performance may be removed or new components added with connections and parameters to improve performance. Performance of individual neurons is calculated in the Growing Neural Gas (Fritzke, 1995) and evolving spiking neural network (Wysoski et al., 2010); both algorithms add or remove neurons on the basis of individual performance calculations.

Parameter values can be predefined or selected using processes independent of the constructive neural network state or operation; however, synapse and neuron construc-

tion may achieve a greater immediate impact on performance with parameter values calculated based on the state or operation of the constructive neural network. The final definition in this section covers the processes for selecting new parameter values:

**Definition 5** *A parameter calculation process is any set of algorithm steps that provides parameter values for new network components in the constructive neural network.*

Given that new synapses and neurons are created to improve the ANN performance, parameter calculation processes can benefit from incorporating available information, including performance values, input data, and existing parameter values. A standard result of construction is the creation of new synapses (including those to or from new neurons) and the standard synapse parameter is its weight. An ANN may also use component models with other functional parameters that can be calculated, such as a neuron activation threshold (Wysoski et al., 2010).

This concludes the definitions of the fundamental processes and standard components of constructive algorithms. In summary, a constructive algorithm can be decomposed into processes: determining when to modify the network, selecting the network components to affect, and selecting parameter values to assign new network components. Performance evaluation processes can be used to provide assessments to indicate when and where to modify the network structure. Parameter calculation processes can provide parameter values for new network components. A constructive algorithm can be decomposed into steps and grouped into these processes for the comparison, design and analysis of constructive neural networks.

The constructive algorithm defines the sequence of steps and processes and must be integrated with the operation and training of the ANN to produce a constructive neural network. General sequences of constructive algorithm processes and approaches to ANN integration are discussed next.

### 2.1.2 Algorithm Sequence and ANN Integration

A constructive neural network has been defined as an ANN that has constructive algorithm processes performed during or between periods of the ANN operation or training. An essential aspect of the constructive neural network is the sequence of the constructive algorithm steps and their integration with the ANN operation or training. Pseudocode representations of the sequence of constructive algorithm processes and the steps of ANN operation or training provide a compact high-level summary of the constructive neural network. Examples of pseudocode summaries of constructive neural networks are provided in Section 2.2.1.

The algorithm sequence may be discussed in terms of the fundamental and standard processes of constructive algorithms identified and defined in the previous section. The fundamental processes of a constructive algorithm identified were: 1) determine when or if the ANN structure will be modified, 2) select the network components to be affected, and 3) select or calculate parameter values for new components. Two basic options for the sequence of these algorithm processes are:

- Determine if the ANN structure will be modified (1), select the network components to affect (2), and then calculate the parameter values (3).
- Select the network components to affect (2), calculate the parameter values (3), and then determine if the ANN structure will be modified (1).

In the first case, a performance evaluation process is performed first to determine if conditions for construction are met. If the conditions are met, then the network components to affect are selected and new parameter values are calculated. In the second case, the constructive algorithm starts with selecting components and calculating new parameter values. Conditions are then tested (and may evaluate new components and calculated parameter values) to determine if construction will be performed.

The evolving spiking neural network (Wysoski et al., 2010) is an example of the second case: parameter values for a potential neuron are calculated for an input, then the parameters of the new neuron are assessed. If the new neuron has sufficiently unique parameter values it is added to the network, otherwise it is merged with the most similar existing neuron. Complex constructive algorithms may have multiple processes for performance evaluation or parameter calculation that can result in a variety of modifications to the ANN structure and algorithm sequences.

Processes for performance evaluation and parameter calculation can have varying degrees of integration with the ANN operation or training. The integration of the constructive algorithm with the ANN can be described in terms of events in operation or training that control or initiate constructive algorithm processes:

- A convergence in ANN operation or training error (Ash, 1989; Fahlman & Lebiere, 1990).
- A number of input samples have been processed (Fritzke, 1995).
- A step in neuron model update or error value calculation (Platt, 1991; Wysoski et al., 2010).

Note that the constructive algorithm processes may be performed at intervals in the ANN operation or training but maintain variables between updates at these intervals. Spiking neuron models provide additional variables and discrete events that may be used as inputs and to control or initiate constructive algorithm processes. Simulations of spiking neurons may be time driven, with neuron variables updated at time steps, or event driven, with neuron variables updated at spike times or other network events (Brette et al., 2007). The development of constructive algorithm processes that depend on the timing of neuron spikes leads to the proposal of spike-timing-dependent construction as a class of constructive algorithms.

### 2.1.3 Spike-Timing-Dependent Construction

Any algorithm or model that takes the timing of neuron spikes as a parameter may be said to be spike timing dependent.

**Definition 6** *A constructive algorithm performs spike-timing-dependent construction (STDC) if the algorithm takes spike times as input variables or to control the algorithm flow.*

Given constructive algorithm processes that are spike timing dependent, spike times and related variables may need to be stored for multiple network update steps or extended periods of the simulated time to perform the constructive processes.

The constructive algorithms found in literature that perform STDC have been provided distinct periods of neural network activity for each training input or observation, on which processes for performance evaluation and parameter calculation are performed (Roy & Basu, 2017; Wysoski et al., 2010). Performing constructive algorithm processes on each simulation update step can be readily applied to indefinite neural network simulations but may introduce additional computational expense. A core focus of this thesis is the development of algorithms processes for STDC; therefore, the next section of this thesis provides a detailed review of the literature.

Note that a spiking neural network is a necessary but insufficient condition for spike-timing-dependent construction. For example, the Dynamic Node Creation algorithm (Ash, 1989) could be applied to the construction of a spiking neural network. No aspect of this constructive algorithm would depend directly on the spike times of neurons:

- Neuron construction occurs in response to error convergence during ANN training.
- Neuron construction occurs in a predefined network location (neurons are added to the single hidden layer).
- Parameters of new network components are initialised stochastically.

This constructive algorithm has all required processes for spiking neural network construction without a direct dependence on neuron spike times.

Spiking neural networks can also have spike rate variables or other rate-based features (Dayan & Abbott, 2001) that can be taken as input variables or used to control the constructive algorithm flow. This would be insufficient to qualify as STDC under a strict application of the definition presented.

The compound adjective 'spike-timing-dependent' is commonly associated with the biological observations and family of plasticity models referred to as spike-timing-dependent plasticity (Caporale & Dan, 2008; Morrison et al., 2008). In these models of plasticity the change in synaptic efficacy or connection weight is dependent on the timing of presynaptic and postsynaptic neuron spikes. A detailed description of two common mathematical models of STDP and their implementation in a simulation is provided in Section 3.2.1.

A constructive spiking neural network may include a model STDP to adapt synapse weights. The implementation of STDP alone would be insufficient for a constructive algorithm to qualify as STDC under a strict application of the definition presented. To qualify as STDC, the constructive algorithm must also use spike times as input variables or to control the algorithm flow. Nevertheless, given that spiking neural networks and

STDP are common models in simulations in computational neuroscience, compatibility with these models is important for the application of constructive algorithms in this field.

The fundamental components and processes of constructive neural networks that have been identified and defined will now be used to give detailed descriptions and perform critical comparisons of existing constructive neural networks.

## 2.2 STDC in Literature

This thesis has proposed spike-timing-dependent construction as a class of constructive algorithms that may have particular relevance to computer-simulations of biological neural networks. Constructive algorithm processes for performance evaluation and parameter calculation that incorporate spike timing have been largely unexplored. Nevertheless, a number of constructive spiking neural networks that can be classified as performing STDC have been identified in the machine learning literature:

1. Refractoriness-based construction (Takita & Hagiwara, 2005)
2. Evolving spiking neural network or eSNN (Schliebs & Kasabov, 2013; Wysoski et al., 2010) and constructive spiking neural networks based on or strongly influenced by the eSNN:
   (a) Dynamic evolving spiking neural network or deSNN (Kasabov, Dhoble, Nuntalid, & Indiveri, 2013)
   (b) Basis coupled evolving spiking neural network or BCESNN (Shirin, Savitha, & Suresh, 2013)
   (c) Growing-pruning spiking neural network or GPSNN (Dora, Sundaram, & Sundararajan, 2015)
   (d) Sequential learning spiking neural classifier or SLSNC (Dora, Suresh, & Sundararajan, 2015)
   (e) Self-regulating evolving spiking neural classifier or SRESN classifier (Dora, Subramanian, Suresh, & Sundararajan, 2016)
   (f) Radial-time basis function evolving spiking neural network (Wang, Belatreche, Maguire, & McGinnity, 2014)
   (g) Evolving Spiking Neural Classifier or ESNC (Wang, Belatreche, Maguire, & McGinnity, 2015a)
   (h) SpikeComp (Wang, Belatreche, Maguire, & McGinnity, 2015b)
   (i) SpikeTemp (Wang et al., 2017)
3. Offline synaptic structural plasticity algorithm (Roy, San, Hussain, Wei, & Basu, 2016)
4. Online synaptic structural plasticity algorithm (Roy & Basu, 2016, 2017)

The review of these constructive spiking neural networks starts with overviews of the spiking neural network models, the sequence of constructive algorithm processes and the events that trigger them. Pseudocode descriptions and diagrams of the constructive spiking neural networks provide a high-level overview and context for a detailed review of the constructive algorithm processes.

Following the high-level description, the review of the constructive algorithms is broken into sections for the two standard types of constructive algorithm processes: performance evaluation and parameter calculation. Applications of the constructive spiking neural networks have been largely restricted to machine learning; however, the constructive spiking neural networks have not been applied to comparable tasks. As a result, direct comparisons of performance on the same tasks have not been performed. Nonetheless, the applications of the constructive spiking neural networks are summarised and discussed.

### 2.2.1 Algorithms and ANN Integration

Constructive algorithms that perform STDC found in literature are integrated with spiking neural networks. This integration of constructive algorithms includes the use of events in neural network operation or training to control the algorithm flow. Diagrams of the spiking neural network architectures and high-level summaries of the algorithm pseudocode are presented, then the constructive algorithm flow and network architectures are critically discussed.

The refractoriness-based constructive algorithm (Takita & Hagiwara, 2005) is closely integrated with the operation of the spiking neural network. The constructive algorithm introduces a leaky integrate-and-fire spiking neuron model with a refractoriness variable that inhibits neuron activation. The refractoriness variable decays at a given rate in each time step and can be directly substituted for the neuron spike time. Algorithm processes are triggered by neuron spikes and depend on individual neuron spike times; therefore, this constructive algorithm performs spike-timing-dependent construction.

The constructive spiking neural network for refractoriness-based construction (Figure 2.3) is designed to perform reinforcement learning: the first hidden layer ($H_1$) detects environment states from input neuron activation, the second hidden layer ($H_2$) detects transitions in the environment state, and the activation of output layer neurons corresponds to selecting actions. The neural network receives reward feedback based on task performance that is used to update connections and in the calculation of performance for neuron construction.

The refractoriness-based constructive algorithm has two sets of constructive algorithm processes:

1. The ‘simple’ process creates neurons to detect the current states from the combination of active input neurons;

2. The ‘complex’ process creates neurons to correct poor reward feedback performance from the inadequate detection of state transitions.

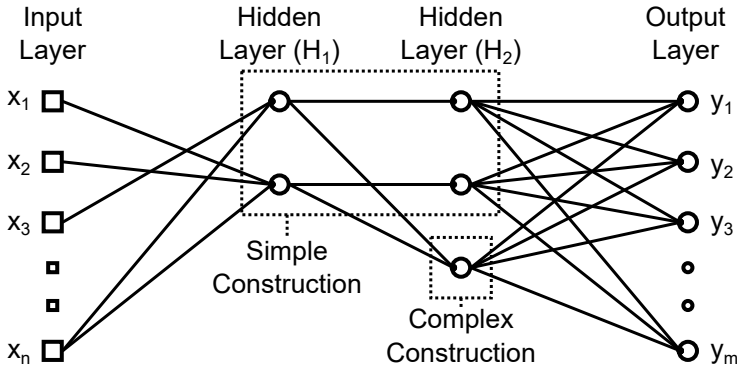


**Figure 2.3:** A simplified diagram of the process of refractoriness-based construction (Takita & Hagiwara, 2005). A multilayer perceptron with two hidden layers is created through a 'simple' construction process and a 'complex' construction process. Simple construction creates neurons in $H_1$ and $H_2$ to recognise current input states and select actions. Complex construction creates neurons in $H_2$ to correct poor performance by detecting sequences of state transitions (sequences of spikes in $H_1$).

An example of the general network architecture is given in Figure 2.3 and pseudocode for the simple construction and complex construction are provided in Algorithms 1 and 2, respectively.

**Algorithm 1** Refractoriness-based construction: simple

1: **for each** time step, $t \leftarrow 1, 2, 3, \ldots, T$ **do**
2: $I_{\text{new}} \leftarrow$ the set of input neurons with refractoriness above $\theta_R$
3: **if** no neuron in $H_1$ has connections from input neurons equal to $I_{\text{new}}$ **then**
4: create a neuron in $H_1$ with input connections from $I_{\text{new}}$
5: create a neuron in $H_2$ with a connection from the new neuron in $H_1$
6: create connections from the new neuron in $H_2$ to the output neurons
7: **end if**
8: **end for**

The simple construction process evaluates the performance of the network in each time step. The condition that triggers construction in the simple process is the lack of a neuron in $H_1$ (hidden layer 1) that has input connections corresponding to the exact set of recently active input neurons. The recent activity of neurons is determined by a threshold on the refractoriness of input neurons. The locations of construction are predefined: new neurons are created in $H_1$ and $H_2$. Synapse weights to the new neuron in $H_1$ are calculated by evenly dividing a constant. The connection weights from the new neuron in $H_1$ to the new neuron in $H_2$ and from the new neuron in $H_2$ to the outputs are predefined.

The complex construction process uses reward feedback to update individual neuron performance variables in each time step, attributing reward feedback to neurons in $H_2$ that are responsible for selecting actions. At predefined intervals in operation the

**Algorithm 2** Refractoriness-based construction: complex

**initialise** $k_{\min} \leftarrow \varnothing$
**for each** time step, $t \leftarrow 1, 2, 3, \ldots, T$ **do**
    **if** $t \mod 10000 = 0$ **then**
        $k_{\min} \leftarrow$ neuron in $H_2$ with the minimum performance
    **end if**
    **if** $k_{\min}$ spikes **then**
        $J_{\text{new}} \leftarrow$ the set of $H_1$ neurons connected to $k_{\min}$
        $J_{\text{new}} \leftarrow J_{\text{new}} \cup H_1$ neuron with highest refractoriness not connected to $k_{\min}$
        **if** no neuron exists in $H_2$ with input connections equal to the set $J_{\text{new}}$ **then**
            create a new neuron in $H_2$ with the connections $J_{\text{new}}$
            calculate synapse weights using neuron refractoriness
            $k_{\min} \leftarrow \varnothing$
        **end if**
    **end if**
    update performance of neurons in $H_2$
**end for**

worst performing neuron in $H_2$ is marked as a trigger for future construction. A spike of the marked neuron triggers the selection of a set of neurons in $H_1$: the $H_1$ neurons connected to the marked $H_2$ neuron plus the $H_1$ neuron with the highest refractoriness not connected to the marked $H_2$ neuron. Construction proceeds if no existing neuron has input connections exactly from that set of selected neurons. If these conditions are met, synapse weights are calculated and the new neuron is added to $H_2$.

The evolving spiking neural network or eSNN (Wysoski et al., 2010) uses the concept of rank order coding to calculate synapse weights and output neuron activation thresholds. Presynaptic neuron spikes are ranked in order of their spike time and are given postsynaptic potential factors that decrease geometrically in order (the earliest neuron spike has the largest postsynaptic potential factor). In the eSNN this ranking process is also applied to the calculation of synapse weights. Connections from the earliest spiking presynaptic neuron are given the highest synapse weight and each subsequent presynaptic neuron that spikes receives a progressively lower synapse weight. Given that the ranking of spikes is based on spike timing, this constructive algorithm is considered to perform STDC in this thesis.

The eSNN is primarily applied to supervised pattern recognition and classification: distinct input samples are paired with desired output classes. Neuron construction occurs in a single layer of the spiking neural network (Figure 2.4); however, layers or reservoirs of spiking neurons are often used to pre-process spatio-temporal data such as images and audio samples (Schliebs & Kasabov, 2013; Wysoski et al., 2010). The processes of the constructive algorithm are performed for each training sample and begin with a calculation of synapse weights for a new neuron (Algorithm 3).

There are a number of variations on this base eSNN algorithm and spiking neural network that will be reviewed in detail in following sections. All variations have the

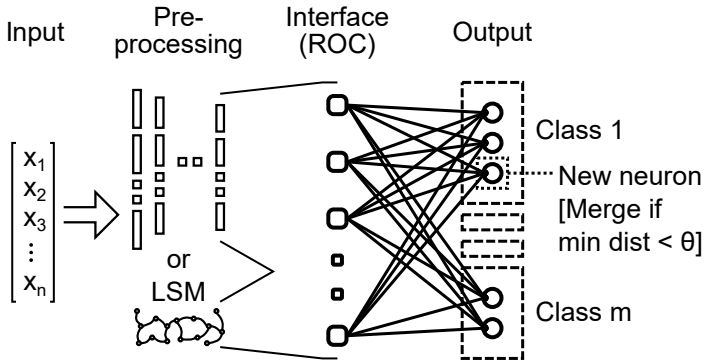


**Figure 2.4:** A diagram of an evolving spiking neural network or eSNN (Wysoski et al., 2010). The input is pre-processed with spiking neuron layers or a Liquid State Machine. The order that interface neurons spike (rank order coding) is used to calculate a weight vector to a new output neuron for the associated class. If any same-class neuron has a weight vector with Euclidean distance to the new weight vector below a threshold, the new weight vector is merged with nearest and no neuron is inserted. Otherwise the new output neuron is constructed.

same underlying algorithm sequence and neural network integration: calculating parameters for the proposed neuron and then accepting or merging the neuron parameters based on their similarity to existing neurons.

**Algorithm 3** Evolving spiking neural network

**for each** sample in training data set, $(\mathbf{x}, y) \in X$ **do**
    simulate the spiking neural network for sample input $\mathbf{x}$
    rank interface neuron spikes in order of spike time
    calculate weight vector (from ranks) and neuron threshold
    $d_{\min} \leftarrow$ minimum Euclidean distance to weight vectors of class $y$ neurons
    **if** the minimum distance is less than the threshold, $d_{\min} < \theta$ **then**
        merge the new weight vector with the nearest neuron
        merge the new threshold with the nearest neuron
    **else**
        add the new neuron to the output neurons for class $y$
    **end if**
**end for**

The offline and online algorithms for structural plasticity (Roy & Basu, 2017; Roy et al., 2016) create and prune synapses on neurons with non-linear dendrites (Figure 2.5). The neurons with non-linear dendrites or NNLD each have a number of dendrite branches with their own synapses and each dendrite has a non-linear contribution to the neuron activation. The structural plasticity algorithm incrementally replaces poor performing synapses to perform learning (Algorithms 4 and 5). When a synapse is pruned another is created on the same dendrite to maintain a fixed number

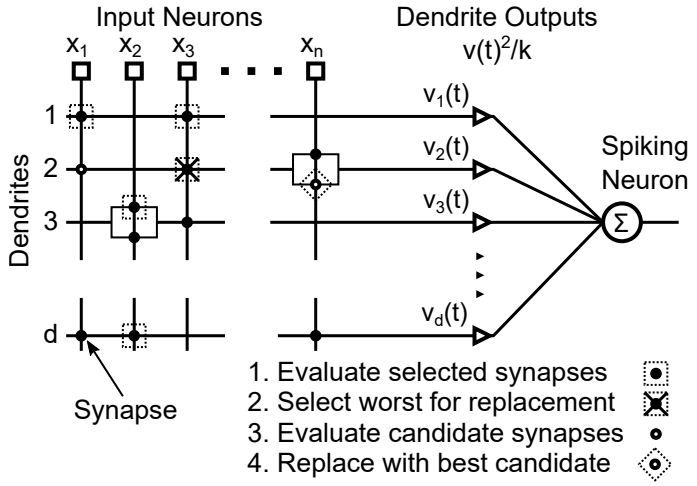


**Figure 2.5:** A simplified diagram of the synaptic structural plasticity algorithm (construction and pruning) for a neuron with non-linear dendrites (Roy & Basu, 2017). The structural plasticity algorithm evaluates the performance of a random selection of synapses. The worst performing synapse is marked for replacement. Candidate synapses on the same dendrite are evaluated. The marked synapse is replaced by the best performing candidate synapse. Note that it is possible for multiple binary synapses to form between an input neuron and a dendrite.

overall. All synapses have a weight of 1, but each dendrite is allowed to have multiple synapses from the same input neuron.

The offline and online structural plasticity algorithms have similar sequences of processes and similar approaches to integration with the neural network. The training has two distinct phases of performance evaluation. The first phase of training evaluates the performance of existing synapses to identify the worst performing synapse and mark it for replacement. The second phase of training evaluates the performance of a set of candidate replacement synapses. In each training phase, the full set of training samples is applied (offline algorithm) or one input spike pattern is applied (online algorithm) to the network of neurons.

These constructive algorithms perform STDC as both the offline and online structural plasticity algorithms have the calculations of synapse performance dependent on the timing of neuron spikes. Performance calculations in the offline algorithm are based on the dendrite and synapse contribution at the time of the maximum postsynaptic neuron potential. The synapse contribution is dependent on the spike time of the presynaptic neuron. Given that the offline and online algorithms only allow synapses to have a weight of 1, there is no need for parameter calculations for construction.

The online algorithm calculates performance using a rule based on STDP, with performance values for each synapse updated at presynaptic and postsynaptic spike times. Mirroring standard STDP, a postsynaptic neuron spike after a presynaptic neuron spike produces a positive change in performance. A presynaptic neuron spike after a post-

**Algorithm 4** Structural plasticity for NNLD (offline)

**while** any misclassifications **and** iteration $<$ maximum iteration **do**
    select synapses at random for performance evaluation
    **for each** sample in training data set, $(\mathbf{x}, y) \in X$ **do**
        simulate the spiking neuron and predict class of sample input $\mathbf{x} \rightarrow \hat{y}$
        **if** sample misclassified, $\hat{y} \neq y$ **then**
            update performance of selected synapses
        **end if**
    **end for**
    select the synapse with the worst performance
    select random input neurons as candidates for a replacement synapse
    **for each** sample in training data set, $(\mathbf{x}, y) \in X$ **do**
        simulate the spiking neural network for sample input $\mathbf{x}$
        **if** sample misclassified, $\hat{y} \neq y$ **then**
            update performance of replacement synapse candidates
        **end if**
    **end for**
    replace worst performing synapse with best performing candidate
**end while**

**Algorithm 5** Structural plasticity for NNLD (online)

**for each** spike pattern in the pattern set, $p \in P$ **do**
    **for each** time step (or spike time) during the pattern, $t \in [0, T_p]$ **do**
        update the neural network simulation
        update performance of synapses using STDP-based rule
    **end for**
    **for each** postsynaptic neuron that spiked $n \in f_p$ for pattern $p$ **do**
        mark the synapse $s$ to $n$ with the worst performance for replacement
        select replacement synapse candidates, $S_{n,\text{can}}$
    **end for**
    **for each** time step (or spike time) during the pattern, $t \in [0, T_p]$ **do**
        update the neural network simulation
        update performance of $S_{n,\text{can}}$, $n \in f_p$, using STDP-based rule
    **end for**
    replace the worst performing synapses with the best performing candidates
**end for**

synaptic neuron spike produces a negative change in performance. Lateral inhibition between postsynaptic neurons produces competition, tuning neurons for patterns or specific spike latencies within a pattern.

The refractoriness-based construction, the evolving spiking neural network, and the structural plasticity algorithm are the primary constructive algorithms found in literature that perform spike-timing-dependent construction. Each of these constructive algorithms has been described here and shown to have a different approach to integrating the constructive algorithm steps and processes with the neural network training or operation. The eSNN and NNLD constructive algorithms, as communicated, require a distinct period of spiking neural network activity for each training example.

The eSNN calculates parameters for new neurons by ranking input neurons in order of their spiking. This ranking requires a defined start time for each input pattern. Furthermore, the basic eSNN algorithm does not accommodate patterns in spikes after the first spike of each input neuron. The NNLD constructive algorithms also require defined input spike pattern periods. These algorithms also specify that the input spike patterns be re-simulated to calculate the performance of replacement synapse candidates. Both the eSNN and NNLD constructive algorithms would need to be modified to be applied to continuous simulation of a spiking neural network with multiple patterns.

The refractoriness-based constructive algorithm is designed to operate with a continuous simulation of a spiking neural network without defined start or end times for input patterns. In part, this may be a result of the refractoriness-based algorithm being developed for reinforcement learning tasks that can have extended durations. The simple construction process in refractoriness-based construction evaluates performance in each time step and performs construction if the recent input pattern is not represented in the network. The complex construction process has a combination of performance calculations and conditions that also operate during the spiking neural network simulation.

The three classes of STDC algorithms described here were designed for specific neuron models: refractoriness-based construction used neurons with a refractoriness variable (Takita & Hagiwara, 2005); the eSNN used rank order coding to update neurons and calculate parameters (Wysoski et al., 2010); and the structural plasticity algorithm reviewed was developed for a neuron with non-linear dendrites (Roy et al., 2016). Each of these constructive algorithms have processes that are generally applicable to networks of neurons using other spiking models. The refractoriness value can be substituted for a spike time to allow a number of the constructive algorithm processes to generalise across a wide range of spiking neuron models. The eSNN and NNLD constructive algorithms have been applied to a range of similar spiking neuron models (Kasabov et al., 2013; Roy & Basu, 2016; Wang et al., 2014); however, these constructive algorithms disrupt the simulation for each input pattern and may interfere with the behaviour of the neuron and network models.

The compatibility of existing STDC algorithms with other spiking neuron and network models may be better understood with a detailed description and discussion of the algorithm processes for parameter calculation and performance evaluation.

### 2.2.2 Parameter Calculation

Parameter calculation processes can be used in the creation of new network components and to merge new or existing network components. Two of the constructive spiking neural networks described have parameter calculation processes: the refractoriness-based construction (Takita & Hagiwara, 2005) has distinct parameter calculations for its simple and complex construction processes; the evolving spiking neural network and its variants (Dora et al., 2016; Schliebs & Kasabov, 2013; Wang et al., 2017; Wysoski et al., 2010) have synapse weights and neuron parameter calculations that are used as inputs in performance calculations. The structural plasticity algorithm for neurons with non-linear dendrites (Roy & Basu, 2016, 2017; Roy et al., 2016) creates a synapse with the same parameters as the pruned synapse and does not create new neurons; therefore, parameter calculations for new network components are not required.

The refractoriness-based algorithm (Takita & Hagiwara, 2005) has two sets of constructive processes: 'simple' construction and 'complex' construction (Figure 2.3). The simple construction process results in a new neuron, $j_{\text{new}}$, in the first hidden layer, $H_1$, with connections from active input neurons, $I^A$, and a new neuron, $k_{\text{new}}$, in the second hidden layer,$H_2$, with a connection from $j_{\text{new}}$ and connections to all neurons in the output layer, $l \in L$. The parameters for the neurons and many of the connections are predefined. The weights of connections from the active input neurons, $i \in I^A$, to the new hidden neuron $j_{\text{new}}$ need to be calculated,

$$w_{i \in I^A, j_{\text{new}}} = w_0 / |I^A|, \tag{2.1}$$

where $w_0 = \phi + \gamma$, that is, the neuron threshold for spiking $\phi$ plus a positive margin $\gamma$, and $|I^A|$ is the number of active input neurons. The connection between the new neurons in each hidden layer, $w_{j_{\text{new}},k_{\text{new}}}$, and the new neuron in the second hidden layer and the output neurons, $w_{k_{\text{new}},l \in L}$, are set to the constant $w_0$. The probability of output connections passing a signal is initialised with a base value of 0 with outputs initially selected with an equal random chance.

The parameter calculations for the complex construction process (Takita & Hagiwara, 2005) are designed to create a neuron in $H_2$ that detects a longer sequence of states or activations in the neurons in $H_1$ and can correct the poor output selections of another $H_2$ neuron. Performance evaluation processes select the worst performing neuron in $H_2$. This worst performing neuron, $k_{\min}$, has connections from the set of $H_1$ neurons, $J_{k_{\min}}$. Parameter calculation occurs when $k_{\min}$ spikes, triggering the construction of a neuron, $k_{\text{new}}$, in $H_2$. At the time of this spike, the set of $H_1$ neurons, $J_{\text{new}}$, that will connect to $k_{\text{new}}$ are found: $J_{k_{\min}}$ plus the $H_1$ neuron with the highest refractoriness not already in $J_{k_{\min}}$.

Connection weights are calculated if no neuron in $H_2$ already has connections from the exact set of neurons in $J_{\text{new}}$. For the purposes of the connection weight calculation, these neurons are assigned indices $m = 1, \ldots, |J_{\text{new}}|$, in order of increasing refractoriness

(least recent to most recent spike). The weight of each connection is calculated as

$$w_{m,k_{\text{new}}} = w_0 \cdot m/|J_{\text{new}}| - \sum_{p=1}^{p<m} w_{p,k_{\text{new}}} \cdot (1 - d_{k_{\text{new}}})^{(t_m^f - t_p^f)}. \tag{2.2}$$

The first component of the equation, $w_0 \cdot m/|J_{\text{new}}|$, increases in order of the least recent to most recent neuron to spike. The second component, $-\sum_{p=1}^{p<m} w_{p,k_{\text{new}}} \cdot (1 - d_{k_{\text{new}}})^{(t_m^f - t_p^f)}$, subtracts a sum based on the weights already calculated. The contribution of previously calculated weights, $w_{p,k_{\text{new}}}$, is reduced by the refractoriness decay factor, $(1 - d_{k_{\text{new}}})$, for neurons in $H_2$. The exponent of the decay factor is the difference in the spike time of the neuron $m$, denoted $t_m^f$, with synapse weight presently being calculated, and the times of earlier spikes, $t_p^f$, from neurons in $J_{\text{new}}$.

The authors of the refractoriness-based constructive algorithm (Takita & Hagiwara, 2005) state that this complex parameter calculation process results in neurons that only spike in response to the same pattern presented in full; however, this is not demonstrated experimentally. The proposal of parameter calculation processes without presenting deeper investigations of their performance is prevalent in the constructive neural network literature.

The evolving spiking neural network (Wysoski et al., 2010) calculates synapse weights for a new neuron on each training input (Figure 2.4). The connection weights are determined by the order of presynaptic neuron spikes,

$$w_{i,j} = \text{mod}^{\text{order}(i,j)}, \tag{2.3}$$

where mod $\in (0, 1)$ is described as a modulation factor and order$(i, j)$ is a function that returns the arrival number (increasing integers starting at zero) of the spike from presynaptic neuron $j$ to the new postsynaptic neuron $i$. This produces connection weights that descend exponentially in the order that each presynaptic neuron spikes, from the first to the last. This result is approximately the inverse of the complex refractoriness-based parameter calculation (Takita & Hagiwara, 2005), which has a component that increases the synapse weights from neurons that spike later.

The rank-order-coding spiking neuron model (Thorpe, Delorme, & Rullen, 2001) was developed as a model capable of producing the rapid visual processing of the biological brain. Given that the postsynaptic spike time will be after a number of presynaptic neuron spikes and that the trend of weight calculations is for presynaptic neurons that spike later to be assigned lower weights, this process of calculating synapse weights is the opposite of the modifications produced by the standard model of STDP.

Alternative approaches to calculating the neuron parameters in the eSNN-based algorithms have been investigated. The growing-pruning spiking neural network or GPSNN (Dora, Sundaram, & Sundararajan, 2015) and the self-regulating evolving spiking network (Dora et al., 2016) use the eSNN rank order weight calculation rule; however, the rule is applied separately to sets of input neurons grouped for each input feature (each element of the input vector).

The basis-coupled rank-order learning (Shirin et al., 2013) and Sequential Learning Spiking Neural Classifier (Dora, Suresh, & Sundararajan, 2015) perform weight calculations using input feature values and Gaussian receptive field parameters. Each input feature, $b$, has a set of input neurons, $j \in J$, that have spike times calculated from Gaussian receptive fields (basis functions). Synapse weights to a new postsynaptic neuron, $i$, are calculated based on the order of spikes for that input feature and the distance of the input feature value, $x_b$, to the input neuron receptive field centre, $\mu_{j,b}$,

$$w_{i,j,b} = \text{mod}^{\text{order}(i,j)}/(1 + |x_b - \mu_{j,b}|). \tag{2.4}$$

The result is that the calculated synapse weight is inversely proportional to the Euclidean distance of the feature input value and the associated receptive field centre. This constructive process produces finer variations in synapse weights than the standard eSNN, which only uses the ranking process.

The evolving spiking neural classifier or ESNC (Wang et al., 2015a), the SpikeComp algorithm (Wang et al., 2015b) and the SpikeTemp algorithm (Wang et al., 2017) adopt an approach close to the eSNN; however, rather than synapse weights being calculated based on spike order, synapse weights are calculated using an exponential decay function,

$$w_{i,j} = w_0 + \exp(-t_j^f/\tau), \tag{2.5}$$

with a constant, $w_0$, presynaptic neuron spike time, $t_j^f$, and time constant, $\tau$. The constructive algorithms may add neurons to a hidden layer in a three-layer network (input layer, hidden layer, output layer) or add neurons to an output layer in a two-layer network (input layer, output layer). Neurons added to a hidden layer require connection weights to the preceding and succeeding layer: different constant values can be selected, for example (Wang et al., 2015a). A survey of the literature (Schliebs & Kasabov, 2013) suggests that the weight constant is not present in the original eSNN algorithm.

Two of the constructive spiking neural networks for STDC found in literature include a synaptic plasticity model: the dynamic evolving spiking neural network or deSNN (Kasabov et al., 2013) and an eSNN-related algorithm described as structure learning (Wang et al., 2014). Both the deSNN and the structure learning algorithm incorporate the plasticity models into the calculation of parameters for neuron construction to perform classification tasks. Nevertheless, synaptic plasticity models are an important feature in many simulations in neuroscience. This indicates a potential for constructive spiking neural networks to be developed for simulations of biological neural networks for neuroscientific studies.

The deSNN (Kasabov et al., 2013) extends the weight calculation of the original eSNN to include spike-driven synaptic plasticity. Synapse weights are first calculated using the original rank order learning method for the first spike of each presynaptic neuron. Initialised synapse weights then drift in each subsequent time step according

to whether the presynaptic neuron spikes,

$$\Delta w_{i,j}(t) = \begin{cases} D_{\text{up}} & \text{if } j \text{ spikes at time step } t, \\ D_{\text{down}} & \text{otherwise,} \end{cases} \tag{2.6}$$

where $D_{\text{up}}$ is positive and $D_{\text{down}}$ is negative.

Two approaches to implementing a deSNN for classification are described (Kasabov et al., 2013) with different requirements for the storage of calculated parameters. The first approach (referred to as deSNNm) starts the classification of an input pattern with all synapse weights initialised to the rank order learning values. The synapse weights are adapted using the drift equation until the first postsynaptic neuron spike provides a prediction of the pattern class. This approach only requires the synapse weights from the the rank order calculation to be stored. The second approach (referred to as deSNNs) starts the classification of an input pattern with a new calculation of synapse weights using the rank order learning and drift equation. The pattern class is predicted through a comparison of the initial rank order learning weight vector and the final weight vector of the new neuron with the stored vectors for existing neurons.

The description of the synapse-driven synaptic plasticity rule implemented in the deSNN is a simplification of the original spike-driven synaptic plasticity rule (Brader, Senn, & Fusi, 2007; Fusi, Annunziato, Badoni, Salamon, & Amit, 2000). The original plasticity model has a bistable dynamic variable, $x_{i,j}(t)$, for each synapse that determines if the synapse weight is potentiated, $w_{i,j}(t) = w_+$ if $x_{i,j}(t) > \theta_x$, or depressed, $w_{i,j}(t) = w_-$ if $x_{i,j}(t) < \theta_x$. The synapse variable $x_{i,j}(t)$ decays to a minimum when below the threshold, $x'_{i,j}(t) = -\beta$ if $x_{i,j}(t) \leq \theta_x$, and rises to a maximum when above the threshold, $x'_{i,j}(t) = \alpha$ if $x_{i,j}(t) > \theta_x$. The synapse variable $x_{i,j}(t)$ also experiences step changes at presynaptic spike times,

$$x_{i,j}(t_j) \leftarrow \begin{cases} x_{i,j}(t_j) + a & \text{if } V_i(t_j) > \theta_V \text{ and } \theta^l_{\text{up}} < C_i(t_j) < \theta^h_{\text{up}}, \\ x_{i,j}(t_j) - b & \text{if } V_i(t_j) \leq \theta_V \text{ and } \theta^l_{\text{down}} < C_i(t_j) < \theta^h_{\text{down}}. \end{cases} \tag{2.7}$$

The synaptic variable $x_{i,j}(t)$ changes according to the postsynaptic neuron potential, $V_i(t)$, and the voltage threshold, $\theta_V$. The additional variable $C_i(t)$ represents calcium in the cell that changes over time, $C'_i(t) = (-1/\tau_C) \cdot C_i(t) + J_C \sum_{t_i \leq t} \delta(t - t_i)$. Presynaptic neuron spikes contribute a constant value $J_C$ to the postsynaptic calcium variable that decays with time constant $\tau_C$.

The structure learning algorithm also incorporates synaptic plasticity models (Wang et al., 2014) and has a similar algorithm organisation to the eSNN but a distinctly different approach to the calculation of synapse weights. In the event the performance threshold is not met, the weights of the new neuron are created from the weights of the winning existing neuron. The weights of the winning neuron are scaled up by a constant factor (1.005) and then the new neuron is trained on the current input using rules derived from STDP models. The STDP curve applied to the synapse weights from the input neurons to the constructed layer is a Gaussian function that is shifted to increase weights when the presynaptic neuron spikes within a small range before the

postsynaptic neuron and decrease the weight outside of this range,

$$\Delta w_{i,j} = \eta \cdot \Big[(1-b) \cdot \exp\Big(\frac{-(\Delta t_{i,j} - p)^2}{\beta^2}\Big) + b\Big]. \tag{2.8}$$

In the reported Gaussian STDP curve, $\eta$ is the learning rate (0.0025), $b$ is a bias ($-0.2$), $\beta$ controls the width of the Gaussian (0.6 ms), and $p$ is the shift of the centre ($-2.85$ ms). A region of the standard exponentially-decaying STDP curve is applied to the synapses from the constructed layer to the output; however, supervised learning is performed to switch the curve between negative and positive depending on the desired output class,

$$\Delta w_{j,k} = \begin{cases} A_p \cdot \exp(-\Delta t_{j,k}/\tau_p) & \text{if } k \in c \text{ and } \Delta t_{j,k} > 0, \\ -A_n \cdot \exp(-\Delta t_{j,k}/\tau_n) & \text{if } k \notin c \text{ and } \Delta t_{j,k} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

Note that no update occurs if the presynaptic neuron spikes at the same time or after the postsynaptic neuron (that is, $\Delta t_{j,k} \leq 0$) and each output neuron, $k$, is associated with a specific output class, $c$. This supervised switching of STDP polarity is not justified with any reference to biological observations.

The eSNN-related constructive spiking neural networks typically include processes for calculating activation thresholds for new neurons. The standard approach (see Dora et al., 2016; Dora, Sundaram, & Sundararajan, 2015; Dora, Suresh, & Sundararajan, 2015; Kasabov et al., 2013; Shirin et al., 2013; Wang et al., 2015a, 2015b, 2017; Wysoski et al., 2010) is to set the threshold to some fraction, $b \in (0, 1)$, of the maximum postsynaptic potential ($PSP$) value recorded for an input sample,

$$PSP_{i,\mathrm{Th}} = b \cdot \max_t PSP_i(t). \tag{2.10}$$

The calculation of neuron threshold ($PSP_{i,\mathrm{Th}}$) depends on the calculations of synapse weights and the neuron models used in the constructive spiking neural network.

The original eSNN (Wysoski et al., 2010) uses a rank order coding neuron model that calculates the postsynaptic potential induced by each presynaptic neuron spike using the same mod and order functions,

$$PSP_i(t) = \sum_{j \in J,\, t_j^f \leq t} w_{i,j} \cdot \mathrm{mod}^{\mathrm{order}(i,j)}. \tag{2.11}$$

Note that this results in the postsynaptic potential being the result of two spike order calculations: one for the weight at the time of construction and one for the current input spikes.

The growing-pruning spiking neural network (Dora, Sundaram, & Sundararajan, 2015) uses the eSNN rank order learning rule for calculating input weights to new neurons; however, the threshold is calculated specifically to produce a desired spike time. The calculation of the threshold is designed to accommodate a spike response

neuron model (Gerstner & Kistler, 2002) and produce a spike at a predefined ideal time, $T_0$,

$$PSP_{i,\mathrm{Th}} = \int_0^{T_0} \sum_{j \in J,\, t_j < T_0} w_{i,j} \cdot \epsilon(t - t_j).\mathrm{d}t. \tag{2.12}$$

The transmission of postsynaptic potential is given by the alpha-function kernel $\epsilon(t) = \frac{t}{\tau} \cdot \exp(1 - \frac{t}{\tau})$. An integral of this form is also used to calculate the membrane potential of the spiking neuron model in this constructive spiking neural network. The authors use a neuron model and threshold calculation rule without integration in subsequent work (Dora et al., 2016),

$$PSP_{i,\mathrm{Th}} = \sum_{j \in J,\, t_j < T_0} w_{i,j} \cdot \epsilon(T_0 - t_j). \tag{2.13}$$

Note that these calculations of the threshold to produce the given spike time $T_0$ were not shown to prevent the neuron potential exceeding this threshold value at times prior to $T_0$ for other input values.

The eSNN-related algorithms often have rules for merging the parameters calculated for a new neuron with those of an existing neuron if performance conditions are satisfied. This merging process is typically implemented as a weighted average (Schliebs & Kasabov, 2013), for example,

$$w_{m,j} = \frac{w_{i,j} + X_m \cdot w_{m,j}}{1 + X_m}, \tag{2.14}$$

where each synapse weight of the new neuron, $w_{i,j}$, is used to update the respective synapse weight of the existing neuron, $w_{m,j}$, weighted by the number of past weight vectors merged with the existing neuron, $X_m$.

The adaptive structure algorithm (Wang et al., 2014) also merges the pair of existing neurons that have nearest average winning spike times if the difference is within a threshold; the resulting parameters follow a similar pattern:

$$w_{c,j} = \frac{X_a \cdot w_{a,j} + X_b \cdot w_{b,j}}{X_a + X_b}. \tag{2.15}$$

The spiking neural network with adaptive structure (Wang et al., 2014) includes STDP-based rules for plasticity and rules for neuron merging; however, their contributions to learning performance are not sufficiently disentangled to make a clear attribution.

Later constructive algorithm developments (Dora et al., 2016; Dora, Sundaram, & Sundararajan, 2015; Dora, Suresh, & Sundararajan, 2015) reinterpret the eSNN neuron merging process as an adaptation process with a non-decaying learning rate, for example,

$$w_{m,j} = w_{m,j} + \eta \cdot (w_{i,j} - w_{m,j}), \tag{2.16}$$

where $\eta$ is a constant or adaptive learning rate.

Within the STDC literature, constructive algorithms have been found that calculate and store parameters that encode neuron spike times (Wang et al., 2015a, 2014). These

parameters are utilised in the calculation and evaluation of performance but do not have a direct function in the neuron or network models; therefore, these parameters will be discussed in the next section.

### 2.2.3 Performance Evaluation

Algorithms found in literature that perform STDC present a number of approaches to the calculation and evaluation of performance. Performance calculations can be performed directly on spike timing (Roy & Basu, 2017), operate on alternative representations of neuron spike timing (Takita & Hagiwara, 2005), or be performed on parameters calculated from neuron spike timing (Wysoski et al., 2010). Constructive algorithms can still be classified as performing STDC if performance calculations do not depend directly or indirectly on spike timing, but another process of the constructive algorithm does depend on spike timing.

The refractoriness-based constructive algorithm (Takita & Hagiwara, 2005) has two constructive processes and two sets of performance conditions for initiating construction. The 'simple' construction process is designed to create neurons in $H_1$, the first hidden layer, that detect the environment state from combinations of active input neurons. At each time step the simple construction process identifies the subset of recently active input neurons, $I_A \subseteq I$, through finding neurons with refractoriness, $R_i(t)$, above a threshold, $\theta_R^A$,

$$I_A = \{i | R_i(t) > \theta_R^A,\ i \in I\}. \tag{2.17}$$

If there is no neuron with that set of connections, $I_A \notin I_{j \in H_1}$, then a neuron with those connections is constructed. This constructed neuron detects when that set of input neurons spike together, representing a unique environment state in the reinforcement learning task.

The 'complex' construction process is designed to create neurons in $H_2$, the second hidden layer, to detect transitions between environment states and correct poor reinforcement learning performance. The performance is calculated and evaluated in a number of steps that include incorporating a reward feedback signal, $r(t)$, from the reinforcement learning task. In each time step each neuron in the second hidden layer, $k \in H_2$, has an internal performance variable updated,

$$P_k(t+1) = \begin{cases} (1 - d_P) \cdot P_k(t) & \text{if } R_k(t) < \theta_R^B, \\ (1 - d_P) \cdot \left( P_k(t) + \dfrac{r(t) \cdot R_k(t)}{\sum_{k \in H_2,\ R_k(t) \geq \theta_R^B} R_k(t)} \right) & \text{otherwise.} \end{cases} \tag{2.18}$$

The update of performance, $P_k(t)$, that neuron $k$ receives depends on its refractoriness, $R_k(t)$, and the given refractoriness threshold, $\theta_R^B$. The next step in the performance evaluation occurs in longer intervals in the neural network operation. The neuron in the second hidden layer with the lowest performance value, $k_{\min} = \arg\min_k P_k(t)$, is determined and marked for counteraction. The next step in performance evaluation occurs when the marked neuron spikes.

The spike of the marked neuron triggers the comparison of $H_1$ neuron refractoriness values to determine the most recent spiking neurons. The set of neurons, $J_B$, to connect to a new neuron in $H_2$ are found from the combination of neurons connected to $k_{\min}$ and the most recent spiking neuron that is not connected to $k_{\min}$. If no neuron in $H_2$ already exists with input connections from that exact set of neurons, $J_B \notin J_{k \in H_2}$, construction is performed.

An important feature in the refractoriness-based constructive algorithm is the prevention of construction in the event that a neuron already exists with the proposed set of connections. A conceptually similar performance criterion is used in the standard eSNN (Wysoski et al., 2010) and a number of the related constructive algorithms (Dora, Suresh, & Sundararajan, 2015; Shirin et al., 2013; Wang et al., 2017). The eSNN calculates parameters for a new neuron and then determines if an existing neuron adequately represents the new parameter values. This performance calculation is based on a threshold of the minimum Euclidean distance between the weight vectors $\mathbf{w}_i$, $i \in I$, and $\mathbf{w}_{\text{new}}$, that is, the weight vectors of existing neurons and the proposed neuron is cancelled if

$$\min_{i \in I} ||\mathbf{w}_i - \mathbf{w}_{\text{new}}|| < \theta_d. \tag{2.19}$$

The eSNN literature often describes this condition as a threshold on the similarity between weight vectors, calculated as the inverse of the Euclidean distance. If the proposed parameter values are not sufficiently unique (an existing neuron corresponds sufficiently to the input sample) the parameters are merged with the nearest neuron. Given that synapse weights are calculated from the order of neuron spikes, this performance condition is indirectly dependent on spike timing.

The activity of a spiking neuron model is time dependent and may have a complex relationship to input activity and synapse weights. Therefore, the effectiveness of comparing synapse weights to determine the similarity of neuron activity should receive a deeper investigation before being accepted. In spiking neural networks that use spike timing to represent values (that is, use time coding), performance could be compared directly on the basis of neuron spike timing. A number of eSNN-related constructive algorithms evaluate the neural network performance based directly on neuron spike timing (Dora et al., 2016; Dora, Sundaram, & Sundararajan, 2015; Wang et al., 2015a, 2014).

The Growing-Pruning Spiking Neural Network (Dora, Sundaram, & Sundararajan, 2015) has two spike-timing-dependent performance conditions that result in the addition of a neuron. The GPSNN constructs neurons in a hidden layer that are then connected to output neurons. The first condition responds to correct classifications that occur later than a threshold time,

$$(c_d = c_f) \wedge ((t_f - T_0) > T_C), \tag{2.20}$$

where $c_d$ is the desired class for the input, $c_f$ is the class of the first neuron to spike, $t_f$ is the first neuron spike time, $T_0$ is a predefined ideal spike time, and $T_C$ is a classification time threshold. The $\wedge$ is used here to denote a logical AND. A correct classification

that occurs late is not robust. Parameter adaptation could reduce the winning neurons association with past input patterns with distinct features; therefore, a new neuron is created to detect the present input pattern.

The second condition responds to the case when the first neuron to spike is a misclassification and the spike time of that neuron that exceeds a threshold range,

$$(c_d \neq c_f) \wedge ((t_f - T_0) > T_M), \tag{2.21}$$

where $T_M$ is a misclassification time threshold. The GPSNN (Dora, Sundaram, & Sundararajan, 2015) does not include a process for depressing the weights of winning neurons that respond to the wrong input class. A late misclassification also indicates that there is no early correct neuron spike; therefore, the classification can be corrected quickly through construction. An early spike that is a misclassification, however, may have beaten an early spike from a neuron of the correct class that could be improved with a parameter update.

The Self-Regulating Evolving Spiking Neural classifier or SRESN classifier (Dora et al., 2016) reproduces the two time-dependent conditions for construction in the GPSNN (Dora, Sundaram, & Sundararajan, 2015) and adds a third condition to handle class overlap. The SRESN network omits the hidden layer of the GPSNN and performs construction directly in the output layer with each constructed neuron assigned a class. This third condition triggers construction in the event that the first output neuron spike is the incorrect class and is within a time threshold, and the time from the first (incorrect) spike to the first correct-class neuron spike is above a threshold,

$$(c_d \neq c_f) \wedge (t_f - T_0 < T_M) \wedge ((t_{fd} - t_f) > T_U/2). \tag{2.22}$$

Here $t_{fd}$ is the first spike time a desired or correct-class output neuron, $t_f$ is the first spike time of any output neuron, and $T_U/2$ is the threshold for updating. This condition results in the construction of a new neuron and weight updates to increase the spike latency of the incorrect-class neuron that spiked first.

The performance conditions reported in the spiking neural network with adaptive structure (Wang et al., 2014) and the SRESN (Dora et al., 2016) have been drawn on in the development of the SpikeComp constructive algorithm (Wang et al., 2015b). Construction is performed automatically for any output class that does not have an associated neuron. The spike times of the winning neuron of the correct class, $t_{fd}$, and the winning neuron of all other classes, $t_m$, are calculated for the input and evaluated against respective thresholds, $\theta_{fd}$ and $\theta_m$. Thresholds are calculated from the average time of winning spikes (centres) for those neurons, $C_{fd}$ and $C_m$ respectively, and given radius constants, $R_{fd}$ and $R_m$, for the correct class and for other classes, $\theta_{fd} = C_{fd}+R_{fd}$ and $\theta_m = C_m + R_m$ respectively. If any of the following spike-time conditions are met then construction is initiated,

$$(t_{fd} < t_m) \wedge (t_{fd} > \theta_{fd}), \text{ or} \tag{2.23}$$

$$t_{fd} > t_m, \text{ or} \tag{2.24}$$

$$(t_{fd} < t_m) \wedge (t_m < \theta_m). \tag{2.25}$$

In the third case, neuron addition is accompanied by modification of the incorrect class neuron to delay the future spike times for similar input patterns.

The GPSNN (Dora, Sundaram, & Sundararajan, 2015), SRESN (Dora et al., 2016), and SpikeComp (Wang et al., 2015b) demonstrate evaluations of performance using neuron spike times. However, like other eSNN-related constructive algorithms, each input sample is treated as a separate period of network activity. These constructive algorithms would need to be revised to be compatible with an ongoing spiking neural network simulation. Identifying the first neuron spike and an ideal postsynaptic neuron spike time may be complicated if the spiking neural network simulation does not have separable activity and sample start times.

The algorithm for spiking neural networks with adaptive structure (Wang et al., 2014) demonstrates another spike-timing-dependent approach to performance evaluation within an eSNN-related algorithm. The performance of constructed neurons is calculated as a weighted average of its spike times for input samples that it wins (spikes first). This average winning spike time, $C_i$, of neuron $i$ is interpreted as the centre of a radial basis function. The existing neural network performance is considered satisfactory if the spike time, $t_f$, of the winning neuron $f$ is within a ratio threshold of the average winning time of that neuron,

$$\frac{|t_f - C_f|}{C_f} \leq \theta_C. \tag{2.26}$$

The threshold increases as current training sample number increases,

$$\theta_C = \theta_0 \times \lfloor x/U + 1 \rfloor, \tag{2.27}$$

where $\theta_0$ indicates the initial threshold, $\lfloor \cdot \rfloor$ indicates the floor operation returning the nearest integer lower than or equal to the enclosed value, $x$ is the current training sample number, and $U$ is a constant threshold update period. If the spike time does not satisfy the threshold, a new neuron is constructed.

In the event that the winning neuron has an acceptable spike time, the average winning time of that neuron is updated,

$$C_f \leftarrow \frac{t_f + X_f \cdot C_f}{1 + X_f}, \tag{2.28}$$

with the present value weighted by the number of input samples that neuron has won, $X_f$.

This algorithm for adapting the structure (Wang et al., 2014) also merges the neurons with the nearest average winning spike times if they are within a margin,

$$\min_{i,m \in I,\, i<m} |C_i - C_m| < \theta_d. \tag{2.29}$$

This combination of performance conditions aims to produce a different spike time for each hidden neuron. The value of interpreting the average spike time of a neuron as the centre of a single radial basis function is unclear. Intuitively, it might be expected that

weights would be sufficient to differentiate classes with neurons and that constructing and training neurons to spike at different times unnecessarily reduces the number of classes that can be stored. Future work by the authors (Wang et al., 2015a) extends this approach with neurons storing a spike delay for each input neuron.

The Evolving Spiking Neural Classifier or ESNC (Wang et al., 2015a) considers the difference in spike times of the new neuron with each presynaptic neuron,

$$\mathbf{c}_i = t_i - \mathbf{t}_J. \tag{2.30}$$

The vector $\mathbf{c}_i$ (a multidimensional 'centre') stores the difference in the postsynaptic neuron spike time, $t_i$, relative to the vector of presynaptic neuron spike times, $\mathbf{t}_J$. The constructive algorithm calculates performance based on the minimum distance between the new vector of spike time differences, $\mathbf{c}_{\text{new}}$, to the vectors of spike time differences of existing neurons, $\mathbf{c}_{i\in I}$,

$$\min_{i\in I} ||\mathbf{c}_{\text{new}} - \mathbf{c}_i|| \leq \theta_d. \tag{2.31}$$

(The authors report this condition as a threshold on the inverse of the distance.) If the condition is satisfied the proposed neuron is merged with the nearest neuron; otherwise, the neuron is added to the neural network. This differentiation between neurons based on the difference in presynaptic and postsynaptic spike times is an approach to performance evaluation that may be generally applicable to constructive spiking neural networks. This may have a strong synergy with network models that have variable transmission delays (Izhikevich, 2006).

In the offline structural plasticity algorithm for learning spike time codes (Roy et al., 2016) the performance of individual synapses is calculated with a cost function,

$$E = \begin{cases} V_{\text{thr}} - V(t_{\max}), & \text{if } P^+, \\ V(t_{\max}) - V_{\text{thr}}, & \text{if } P^-. \end{cases} \tag{2.32}$$

The cost function depends of whether the spike pattern presented is one that should elicit a postsynaptic neuron spike, $P^+$, or one for which the postsynaptic neuron should remain silent, $P^-$. The threshold membrane voltage, $V_{\text{thr}}$, is a learned parameter independent of the synapses weights. The maximum voltage, $V(t_{\max})$, is found from the neuron membrane voltage simulation,

$$V(t) = \sum_{j=1}^{d} b(v_j(t)). \tag{2.33}$$

The neuron potential is the sum of $d$ non-linear dendrites (Figure 2.5) that each contribute,

$$\begin{aligned} b(v_j(t)) &= b\Big(\sum_{i=1}^{n}\Big[w_{i,j}\sum_{t_i^f<t}\epsilon(t-t_i^f)\Big]\Big), \\ &= \Big(\sum_{i=1}^{n}\Big[w_{i,j}\sum_{t_i^f<t}\epsilon(t-t_i^f)\Big]\Big)^2/k, \end{aligned} \tag{2.34}$$

where dendrites sum signals from the $n$ input neurons that induce postsynaptic potential in the form of an alpha-function given as $\epsilon(t - t^f) = V_0 \cdot [\exp(-[t - t^f]/\tau) - \exp(-[t - t^f]/\tau_s)]$ and the constant $k$ is used to control the rate of dendrite voltage increase (a graphical depiction of an alpha function is provided in Section 5.4.3).

The performance of the individual synapses is derived from the gradient-descent of the error. Taking the case of $P^+$,

$$\begin{aligned}
\Delta w_{i,j} &= -\frac{\partial E}{\partial w_{i,j}} \\
&= -\frac{\partial\big(V_{\text{thr}} - \sum_{y=1}^{d} b(v_y(t_{\max}))\big)}{\partial w_{i,j}} \\
&= \frac{\partial b(v_j(t_{\max}))}{\partial w_{i,j}} \\
&= \frac{\partial([\sum_{x=1}^{n} w_{x,j} \sum_{t_x^f < t_{\max}} \epsilon(t_{\max} - t_x^f)]^2/k)}{\partial w_{i,j}} \\
&= \frac{2 \cdot [\sum_{x=1}^{n} w_{x,j} \sum_{t_x^f < t_{\max}} \epsilon(t_{\max} - t_x^f)]}{k} \cdot \sum_{t_i^f < t_{\max}} \epsilon(t_{\max} - t_i^f)
\end{aligned} \tag{2.35}$$

Note that the gradient has been simplified due to each synapse weight only influencing the voltage of the one dendrite and the final step is determined using the chain rule for derivatives. Given that the model only uses binary weights, $w_{i,j} \in \{0, 1\}$, the change in weight, $\Delta w_{i,j}$, is instead interpreted as a correlation and used to determine the performance of the synapse,

$$c_{i,j} = \sum_{\mathbf{x} \in X} \Delta w_{i,j}, \tag{2.36}$$

for the training set, $\mathbf{x} \in X$. The minimum correlation indicates the synapse with the worst performance; this synapse is selected for replacement. A random set of new synapses are proposed on the dendrite that has the synapse being removed. The proposed synapses do not add voltage to the dendrite during the evaluation. The same correlation calculation is performed for these proposed synapses over the complete training data set. The proposed synapse with the highest correlation is selected to replace the worst performing synapse.

In later developments for the structural plasticity algorithm (Roy & Basu, 2016, 2017) the synapse performance or correlation is calculated directly from the relative timing of presynaptic and postsynaptic spikes. The change in correlation at presynaptic neuron spike times is

$$\Delta c_{i,j}(t_j) = -\epsilon(t_j - t_i), \tag{2.37}$$

where $t_j$ is the spike time of presynaptic neuron $j$, $t_i$ is the nearest spike time ($t_i < t_j$) of postsynaptic neuron $i$, and $\epsilon(\cdot)$ is the alpha-function (also seen in Equation 2.35). The change in correlation has the opposite magnitude and relationship to relative spike

times at postsynaptic neuron spike times,

$$\Delta c_{i,j}(t_i) = \epsilon(t_i - t_j), \tag{2.38}$$

giving a positive update when the postsynaptic neuron spikes shortly after a presynaptic neuron spike. This process for determining synapse performance has a strong resemblance to spike-timing-dependent plasticity, which can be implemented to produce positive weight changes when the postsynaptic neuron spikes shortly after the presynaptic neuron and negative weight changes when the presynaptic neuron spikes shortly after the postsynaptic neuron.

The calculation of the correlation or performance of synapses is applied in a similar algorithm sequence. Calculations of the correlation are performed for each given pattern duration. The worst performing synapse is selected for replacement and candidates for construction are randomly placed on the same dendrite. The calculation of correlation is repeated for the pattern and the best performing candidate is selected as the replacement.

This approach to performing synaptic structural plasticity has some limitations. The need to repeat the presentation of the pattern requires storage of the incoming pattern. The cost of this additional storage and the re-simulation should be compared with the benefit of evaluating fewer candidate synapses. The training procedure supplies an input pattern duration and would require modification to be compatible with ongoing simulations that have unknown pattern durations. Lastly, this constructive algorithm does not change the number of neurons or the total number of synapses, so it is limited in its ability to function as a method to automatically select neural network structures.

### 2.2.4 Applications

The constructive spiking neural networks that perform STDC found in literature have been applied to a range of machine learning tasks. The refractoriness-based constructive algorithm (Takita & Hagiwara, 2005) was applied to reinforcement learning tasks, including the cart-pole balancing problem and a task of duelling mobile agents. The majority of the algorithms that perform STDC are related to the evolving spiking neural network and have been applied to a wide range of classification tasks (Schliebs & Kasabov, 2013).

The early work on the eSNN (Wysoski et al., 2006; Wysoski, Benuskova, & Kasabov, 2008) applied the constructive algorithm to image-based face identification. The first experiment with face images reported (Wysoski et al., 2006) used a database of multi-angle views from an earlier study of rank order coding (Delorme & Thorpe, 2001). Neuron construction was performed in the final neuron layer after two layers of non-adapting neurons processed images in steps modelling the on- and off-centre cells in the retina and the orientation-selective cells in the primary visual cortex. This work was extended (Wysoski et al., 2010) to performing audio-visual identification of speakers from the VidTIMIT data set (Sanderson & Paliwal, 2004). Non-adaptive neuron layers

were designed to perform pre-processing of data and extract features in the images and audio signals.

The eSNN has also been implemented in combination with a neuron reservoir or liquid state machine to perform sign-language recognition (Schliebs, Hamed, & Kasabov, 2011). Evolving spatio-temporal data machines (Kasabov et al., 2013, 2016) have been developed to employ an eSNN or deSNN with a neuron reservoir referred to as a NeuCube to perform analysis, detection and prediction of events in EEG data, fMRI data, and data sets from other fields including ecology.

Developments related to the eSNN by other research groups (Dora et al., 2016; Wang et al., 2017) have applied their constructive spiking neural networks to standard data sets from the UCI Machine Learning Repository (Lichman, 2013), including Iris, Breast Cancer (Wisconsin), Image Segmentation, Abalone, Pima Diabetes, Liver Disorders (BUPA), Ionosphere, Yeast, and EEG eyeState.

A motivation of the development of the structural synaptic plasticity algorithms with binary synapses was the low resolution of synapses improving the applicability in neuromorphic hardware (Roy & Basu, 2016, 2017; Roy et al., 2016). These algorithms have been primarily demonstrated in the classification of abstract spike patterns, that is, the spike patterns are randomly generated and not associated with sensory input or real-world datasets.

The offline structural plasticity algorithm (Roy et al., 2016) was first applied to two supervised classification tasks reproduced from an earlier neuron learning model referred to as the tempotron (Gutig & Sompolinsky, 2006). The first task was the supervised learning of patterns of random spike latencies for a neuron to give an associated positive-response (spike) or negative-response (no spike). Patterns in the generated data set had each presynaptic neuron produce a single spike at a random time (uniform distribution) in the input sample duration. The second task was the supervised learning of pairwise synchrony in patterns. Input neurons were grouped into pairs that fire synchronous spikes; these pairs were different for positive-response patterns and negative-response patterns. Although the pairs were synchronised, the timing of the pair of spikes was randomised. The final experiment (Roy et al., 2016) applied the offline structural plasticity algorithm to classification of signals from a tactile sensor array contacted by two spheres of different sizes.

The first application of the online spike-timing-dependent structural synaptic plasticity algorithm (Roy & Basu, 2016) was in the training of a liquid state machine applied to classification of abstract input spike patterns. This online algorithm was then applied to the case of neurons with non-linear dendrites for unsupervised detection of random spike patterns (Roy & Basu, 2017). The unsupervised learning task was inspired by a demonstration of STDP with lateral inhibition producing competitive pattern detection (Masquelier et al., 2009). Structural plasticity was applied to multiple neurons with non-linear dendrites with lateral inhibition and were trained on simulations with 2 patterns, 4 patterns, and 6 patterns of 0.5 s duration.

Earlier studies of STDP (Masquelier et al., 2008, 2009) demonstrated the capability of the learning model to achieve detection of abstract spike patterns. The stochastically

generated repeating spike patterns were produced to provide a challenging learning environment resembling the activity in a biological neural network. The ability of a learning model or constructive algorithm to detect randomly generated spike patterns may be adapted to learning tasks where spike patterns are generated from models of sensory input or data values.

## 2.3 Research Frontiers, Limitations and Gaps

The frontier of research in constructive neural networks could be described as an exploration of a space of designs. This chapter has proposed a set of constructive algorithm components and processes (Section 2.1):

- Parameter calculation processes
- Performance evaluation processes and conditions
- Algorithm sequences and ANN integration

Each of these aspects of the constructive neural network design could be the subject of focused research and development. This thesis aims to contribute to the theory of constructive neural networks generally and also contribute to the study of specific constructive algorithm components and processes.

The literature review has focused on spike-timing-dependent construction, proposed as a class of constructive algorithm that has neuron spike timing taken as an input parameter or an event controlling the algorithm flow. The space of designs in STDC is narrower; however, the limited number of constructive algorithms in the literature leaves this space largely unexplored.

The algorithms for STDC found in literature and reviewed were applied to machine learning tasks. Despite the incorporation of spiking neuron models, the applicability of these constructive spiking neural networks to simulating biological neural networks in neuroscience was not discussed in the prior literature. A reason for this may be the intuitive implausibility of neurons and synapses being instantaneously created or removed from a biological neural network. Other incompatibilities and limitations may arise from specific features and components of constructive algorithms. Reviewed literature does not address the compatibility of constructive algorithm processes with other network models and simulation procedures (Brette et al., 2007; Izhikevich, 2004).

The majority of the STDC algorithms reviewed are related to the evolving spiking neural network (Wysoski et al., 2010). Although the overall design of the eSNN and related constructive algorithms has been successfully applied to a range of classification tasks, the standard eSNN algorithm has limited compatibility with continuous streams of input. The parameter and performance calculations require a reference or start time. A start time has typically been provided with the spiking neural network simulation restarted for each separate input data sample. Spiking neural network simulations in neuroscience and machine learning tasks may require continuous input to capture different timescales of activity and concealed patterns.

Processes of the simple refractoriness-based construction (Takita & Hagiwara, 2005) produce neurons based on recent input activity and are applicable to continuous input streams. These constructive algorithm processes have not been investigated outside of the context of the reinforcement learning applications of the original constructive spiking neural network.

Two algorithms for STDC reviewed incorporated synaptic plasticity (Kasabov et al., 2013; Wang et al., 2014); however, both of these constructive spiking neural networks depart from the original models of biological synaptic plasticity. There is a wide variety of models of synaptic plasticity (Graupner & Brunel, 2012; Morrison et al., 2008) and learning rules for spiking neurons (Ponulak & Kasiski, 2010) that have yet to be investigated for application in constructive spiking neural networks. Given that constructive spiking neural networks have not been developed as simulations in neuroscience, the potential for constructive spiking neural networks to be developed as simulations to study neuroplasticity has not been explored.

A challenge that appeared to be largely ignored in STDC literature is the possibility of the input being pure noise. Continuous input may include periods of activity where there are no features or patterns to be learned. A training data set that is not well processed may also have examples that do not contain the desired features or patterns. A constructive algorithm that creates a new neuron for each input sample, such as eSNN variants, may produce many extraneous neurons or connections. Past algorithms that perform STDC have not included features to address these conditions nor has their performance been studied with pure input noise.

This thesis develops theory and constructive algorithm processes to treat the limitations and gaps identified. The contributions of this thesis to address these limitations and gaps include:

1. A design methodology for constructive neural networks suitable for the incremental development and analysis of constructive algorithms.

2. Theory for the development of constructive algorithms that are compatible with simulations of biological neural networks.

3. Development of constructive algorithm processes:

   (a) Compatible with continuous spiking neural network simulations.

   (b) Compatible with simulated studies that include STDP models.

   (c) Compatible with highly noisy conditions and periods of pure noise.

   (d) To reduce and remove extraneous constructed neurons.

4. Study of the behaviour, learning performance, and compatibility of constructive algorithms in simulations of STDP.

This chapter has proposed a set of definitions of constructive neural network components. Chapter 3 develops an interpretation of neuron construction and pruning and

principles for algorithms compatible with simulated spiking neural networks in neuroscience. From these conceptual developments, base constructive algorithm components are incrementally developed, starting with parameter calculation processes based on standard models of STDP. Chapter 4 investigates the performance of the parameter calculation processes under a range of presynaptic activity conditions. Chapter 5 develops performance evaluation processes based on the prediction of postsynaptic neuron spikes using a proxy neuron.

Chapter 6 combines developed parameter calculation and performance evaluation processes into a basic constructive algorithm that is implemented in a simulation of STDP tuning neurons to detect patterns concealed in high levels of noise. Chapter 7 further develops the constructive algorithm: synapse weight calculations based on predictions of STDP-convergence are investigated and processes for neuron pruning are developed. The constructive algorithm is applied to simulations with multiple hidden spike patterns and demonstrates successful one-shot detection of concealed spike patterns and capabilities for continual learning.

# Chapter 3

# Simulation Expansion and STDP

This chapter introduces and defines the concepts of simulation expansion and contraction and develops theory for designing constructive algorithms that are compatible with simulations of biological neural networks. These conceptual and theoretical developments form important foundations for the novel constructive algorithm processes developed in this thesis. This chapter then presents novel constructive algorithm processes for calculating synapse weights from standard mathematical models of spike-timing-dependent plasticity (STDP).

## 3.1 Simulation Expansion and Contraction

This section develops definitions of the processes of simulation expansion and contraction and contrasting definitions of construction and pruning. Simulation expansion and contraction are described in terms of the transfer of neurons and synapses between sets: the sets of simulated neurons and synapses and the sets of surrounding neurons and synapses. This interpretation of adding and removing neurons from an ANN can avoid the biological implausibility of instantaneously creating or pruning neurons, even when the existence of neurons is hypothetical. The definitions of expansion and construction overlap when the transfer occurs with sets of hypothetical neurons and synapses. Principles and assumptions for performing neuron construction and pruning that are equivalent to expansion and contraction are developed.

### 3.1.1 The Simulated and Surrounding Network

Mammalian brains may be comprised of hundreds of millions to over one hundred billion neurons (Herculano-Houzel, 2009). However, simulations of biological neural networks, and artificial neural networks more generally, often have many orders of magnitude fewer neurons than even small biological brains. In this context, an ANN may be viewed as a small subset within a larger neural system (Figure 3.1). A set-theory perspective of ANNs can be used to define a set of neurons that are simulated and a set of neurons that are not simulated.

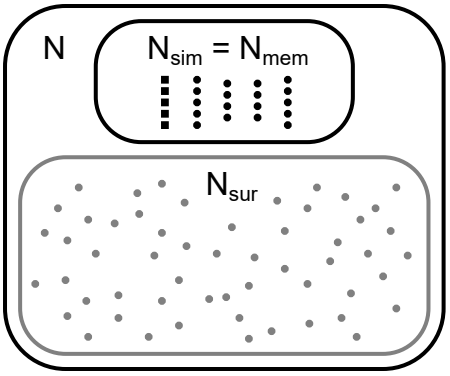


**Figure 3.1:** Diagram of the relationship of the set of simulated neurons in an ANN, $N_{\text{sim}}$, and the set of all neurons in the large neural system, $N$. The surrounding neurons, $N_{\text{sur}}$, are the neurons in $N$ that are not simulated. Here the set of neurons in computer memory, $N_{\text{mem}}$, is equal to the simulated set. The diagram omits synapses for simplicity.

The neurons and synapses in the ANN can be defined as members of sets of simulated neurons and simulated synapses.

**Definition 7** *The sets of simulated neurons, $N_{sim}$, and simulated synapses, $S_{sim}$, are the neurons and synapses stored in memory and participating in the ANN operation.*

Assuming that the ANN is comprised of sets of neurons and synapses that exist within a large neural system with many other neurons and synapses, a pair of sets may be defined for the large neural system.

**Definition 8** *Sets $N$ and $S$ represent the sets of all neurons and all synapses in a neural system and may include hypothetical neurons and synapses as well as those in memory.*

The neurons and synapses that are in the large neural system but not in the simulated sets may also be defined.

**Definition 9** *The sets of surrounding neurons, $N_{sur}$, and surrounding synapses, $S_{sur}$, are any neurons and synapses in $N$ and $S$ that do not participate in the ANN operation.*

An important point in differentiating between definitions of neuron construction and pruning and simulation expansion and contraction is the existence of neurons in memory and the assumed existence of hypothetical neurons.

**Definition 10** *The sets of neurons in memory, $N_{mem}$, and synapses in memory, $S_{mem}$, are the neurons and synapses with representations stored in the computer memory.*

The simulated neurons and synapses, $N_{\text{sim}}$ and $S_{\text{sim}}$, necessarily have some representation in memory to participate in the operation of the ANN. The surrounding neurons and synapses, $N_{\text{sur}}$ and $S_{\text{sur}}$, do not participate in the ANN operation and therefore are

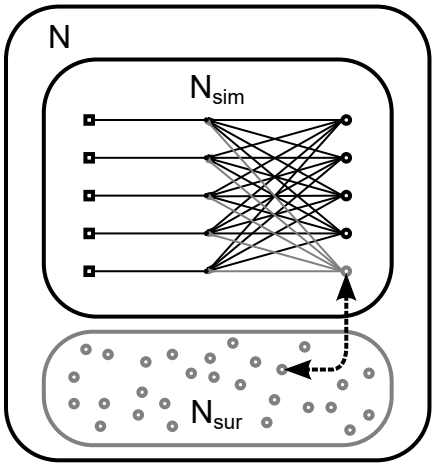


**Figure 3.2:** Simulation expansion represented as a process of transferring a neuron from the surrounding neuron set to a simulated neuron set. Simulation contraction occurs when neurons and synapses are transferred from the simulated sets to the surrounding sets. The arrows with a dashed line represent a neuron transfer. For simplicity, the diagram does not show the transfer of synapses.

not required to be stored in memory. Instead, the surrounding neurons and synapses can be hypothetical constructs that are used to guide the design of the ANN and constructive algorithms. The relationships between the simulated neurons and synapses, the large neural system, the surrounding neurons and synapses, and the computer memory are given in set theory in Appendix A.

### 3.1.2 Transferring Neurons and Synapses

The transfer of neurons and synapses between the simulated sets, $N_{\text{sim}}$ and $S_{\text{sim}}$, and the surrounding sets, $N_{\text{sur}}$ and $S_{\text{sur}}$, does not change the number of neurons or synapses that are assumed to exist in the large neural system, $N$ and $S$. The transfer of neurons and synapses can produce changes in the size and structure of the ANN (Figure 3.2). Set operations that represent the transfer of neurons are provided in Appendix A.

The interpretation of changes in the ANN structure being transfers with sets of neurons and synapses that are not simulated avoids the biologically implausible process of instantaneously creating or deleting neurons and synapses from a neural network. Nevertheless, there are limitations on the connectivity and activity of neurons that can plausibly exist in the surrounding sets; these limitations are discussed in Section 3.1.4. These limitations may prevent many constructive algorithms and pruning algorithms from being reinterpreted as transferring neurons and synapses without other implausible assumptions.

### 3.1.3 Construction and Expansion

Definitions of 'constructive algorithm' and 'constructive neural network' were proposed and 'construction' and 'pruning' were described as types of structural change in Chapter 2. Here, definitions of construction and pruning are proposed based on the sets of neurons and synapses in memory, $N_{\text{mem}}$ and $S_{\text{mem}}$.

**Definition 11** *Construction is the addition of a representation of a neuron or synapse to computer memory.*

**Definition 12** *Pruning is the removal of the representation of a neuron or synapse from computer memory.*

Set notation of these definitions can be found in Appendix A.

The definitions of 'expansion' and 'contraction' proposed here are based on the sets of simulated neurons and synapses, $N_{\text{sim}}$ and $S_{\text{sim}}$, and the sets of surrounding neurons and synapses, $N_{\text{sur}}$ and $S_{\text{sur}}$.

**Definition 13** *Expansion is any transfer of a member to the set of simulated neurons and synapses from the set of surrounding neurons and synapses.*

**Definition 14** *Contraction is any transfer of a member from the set of simulated neurons and synapses to the set of surrounding neurons and synapses.*

The definitions of expansion and construction and the definitions of contraction and pruning coincide when the transfer of the neuron or synapse between the simulated and surrounding sets is implemented as the creation or deletion of the component in memory. This can be the case when the sets of simulated neurons and synapses are held equivalent to the sets of neurons and synapses in memory, $N_{\text{sim}} = N_{\text{mem}}$ and $S_{\text{sim}} = S_{\text{mem}}$.

An algorithm that performs simulation expansion through the construction of neurons and synapses could be referred to as a constructive expansion algorithm. All constructive algorithms developed in this thesis perform constructive expansion; however, to simplify later explanations in this thesis, these algorithms will be referred to as constructive algorithms. A different class of algorithms may be developed in the case of surrounding neurons and synapses also being stored in memory. Algorithms that transfer members between simulated and surrounding sets in memory are discussed in Appendix A.

Assuming the existence of surrounding neurons and synapses that are not simulated and do not exist in memory may appear to be an unimportant trick of interpretation; however, this interpretation can have significant consequences for the design and performance of constructive algorithms. Approaching network construction or pruning as though it were the transfer of neurons and synapses between simulated and surrounding sets may produce constructive algorithms that do not introduce biologically implausible behaviour. A range of assumptions and principles for the development of constructive algorithms compatible with simulations of biological neural networks will now be proposed.

### 3.1.4 Assumptions and Principles

The assumption that an ANN is a relatively small subset of a large biological neural system has been introduced. In terms of the defined sets, this assumption can be stated as having large surrounding sets of neurons and synapses, $N_{\text{sur}}$ and $S_{\text{sur}}$. Therefore, this assumption will be referred to as the *large surrounding network assumption*.

A large hypothetical surrounding network may be considered to contain any biologically plausible neurons and synapses with any plausible combination of parameters. Restrictions on the plausible neurons, synapses and parameters may be provided by the specific properties and characteristics of the biological neural system. A number of general principles for what neurons and synapses may plausibly exist in a hypothetical surrounding network can also be deduced.

The behaviour of the simulated neurons places limitations of the neurons and synapses that could plausibly exist in the surrounding network. The plausibility or probability of a neuron or synapse existing in the surrounding network may be inferred by estimating or recording the change in neuron spike rates or spike latencies that would result from its inclusion in the simulation. The acceptability of changes in the network activity from construction or pruning will depend on the aims, requirements and sensitivity of the model being simulated.

Methods for estimating the probability of specific neurons existing in the surrounding network are not explored in this thesis. Nevertheless, logic dictates that if the addition or removal of a synapse or neuron produces a sudden, large and persistent shift in neuron spike rates or spike latencies, then the plausibility and probability of its existence in the surrounding network is low. This affects what neurons and synapses can be added or removed and be plausibly interpreted as simulation expansion or contraction. Therefore, this thesis proposes that the primary principle for biologically plausible constructive algorithms is that each addition or removal of a neuron or synapses should not introduce a biologically implausible disruption in the behaviour of the ANN. This will be referred to as the *principle of plausible effects*.

Given a neuron that has few connections to other neurons in the simulation, has low activity levels, or both (few connections and low activity), the addition or removal of this neuron is less likely to cause a large change the activity of other neurons. Therefore, in general, plausible simulation expansion and contraction favours the addition and removal of neurons that have few connections and low levels of activity.

Approaches to reducing and explaining changes in the simulation activity from the addition and removal of neurons and synapses may also be explored. For example, models may be developed to include an approximation of the collective effects of surrounding neurons and synapses, which may result in proportionally smaller changes in simulation activity from neuron and synapse addition and removal. Justifications for sudden changes in the overall activity of the neural network may also be investigated. For example, the modelled network may receive inhibition from nearby and distant sources; therefore, the absence of the effects of a surrounding neuron could be justified as the result of the neuron being inhibited. The applicability of these approaches and justifications for simulation expansion and contraction could be investigated for a range

of models; however, this investigation is outside scope of this thesis.

Simulations of neural networks often implicitly assume that the neurons and synapses are fully grown or mature. This assumption may be developed further into a number of proposed explicit assumptions:

1. The neurons and synapses in the simulation are mature and have existed for a long time prior to the time modelled in the simulation.

2. The neurons and synapses have a history of activity and plasticity prior to the simulation.

3. Any pattern of neuron activity observed in the simulation may have occurred multiple times prior to the simulation and caused synaptic plasticity.

These assumptions are collectively referred to in this thesis as the *mature network assumption*.

The large surrounding network and the mature network assumptions have now been introduced and may be used to guide the calculation of synapse weights when constructing neurons. A mature network will have had neuron activity and synaptic plasticity prior to the start of the time modelled in the simulation. If neuron construction is designed to approximate a transfer of a neuron from the surrounding network into the simulation, then the synapse weights of the constructed neuron will be the result of past synaptic plasticity. The next section develops parameter calculation processes for constructive algorithms from models of spike-timing-dependent plasticity (STDP).

## 3.2 STDP and Synapse Construction

Many types of spike-timing-dependent plasticity (STDP) have been observed in nature (Caporale & Dan, 2008). Mathematical models of STDP can often be incorporated in neural network simulations (Brette et al., 2007; Legenstein et al., 2005; Morrison et al., 2008) with implementations mirroring the application of a training algorithm. This section describes implementations of two common models of spike-timing-dependent synaptic plasticity (Morrison et al., 2008): additive STDP and multiplicative STDP. These STDP models are then used to develop novel synapse weight calculation processes for spike-timing-dependent construction.

### 3.2.1 Spike-Timing-Dependent Plasticity

When the efficacy of a synapse in transmitting spikes changes relative to the spike timing of the presynaptic and postsynaptic neuron (Bi & Poo, 1998; Caporale & Dan, 2008), the process is referred to as spike-timing-dependent plasticity (STDP). In computational models of STDP, the relative spike timing of each pair of presynaptic and postsynaptic neurons is used in the calculation of synapse weight adjustments (Morrison et al., 2008). Simulations of spiking neural networks often treat spike duration

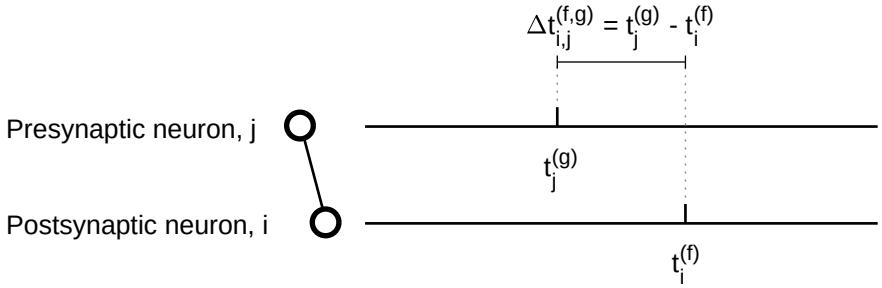


**Figure 3.3:** The presynaptic neuron, $j$, spikes at time $t_j^{(g)}$ and the postsynaptic neuron, $i$, spikes at time $t_i^{(f)}$. Horizontal lines for presynaptic and postsynaptic neurons indicate spike times with vertical marks. The relative spike timing, $\Delta t_{i,j}^{(f,g)}$, is given by the difference in the neuron spike times, $t_j^{(g)} - t_i^{(f)}$. Time units have been omitted; time increases from left to right.

and transmission delay to be constant or negligible, reducing the complexity of analysis and simulation memory requirements (Brette et al., 2007; Izhikevich, 2006). The developments described here use these simplifications.

A formal specification of STDP models requires variables describing the spike times of presynaptic and postsynaptic neurons. The $g^{\text{th}}$ spike of presynaptic neuron $j$ at time $t_j^{(g)}$ and the $f^{\text{th}}$ spike of postsynaptic neuron $i$ at time $t_i^{(f)}$ have relative spike timing

$$\Delta t_{i,j}^{(f,g)} = t_j^{(g)} - t_i^{(f)}. \tag{3.1}$$

A graphical representation of the relative spike timing between a presynaptic and postsynaptic neuron is presented in Figure 3.3.

The relationship between synaptic weight change and the relative spike timing of the presynaptic and postsynaptic neurons may take many forms (Caporale & Dan, 2008; Morrison et al., 2008). Nevertheless, the antisymmetric exponential decay model (for example, see Figure 3.4) is commonly used in simulated studies (Legenstein et al., 2005; Masquelier et al., 2009; Song et al., 2000). Models with this relationship to relative spike timing have changes in synapse weight proportional to the STDP curve described by

$$\Delta w_{i,j}^{(f,g)} = \begin{cases} A_+ \cdot e^{\Delta t_{i,j}^{(f,g)}/\tau_+} & \text{if } \Delta t_{i,j}^{(f,g)} < 0 \\ -A_- \cdot e^{-\Delta t_{i,j}^{(f,g)}/\tau_-} & \text{if } \Delta t_{i,j}^{(f,g)} \geq 0. \end{cases} \tag{3.2}$$

The amplitude of positive and negative weight updates is influenced by the gains $A_+$ and $A_-$, respectively (note that later equations assume that $A_+$ and $A_-$ are positive). The magnitude of the update diminishes exponentially with time constants $\tau_+$ and $\tau_-$ as the time between presynaptic and postsynaptic spikes increases. This set of equations can be graphically represented as an STDP curve (Figure 3.4).

The anti-symmetric Hebbian STDP rule may be viewed as increasing the strength of synapses with presynaptic neurons that had a role in the activation of the postsynaptic

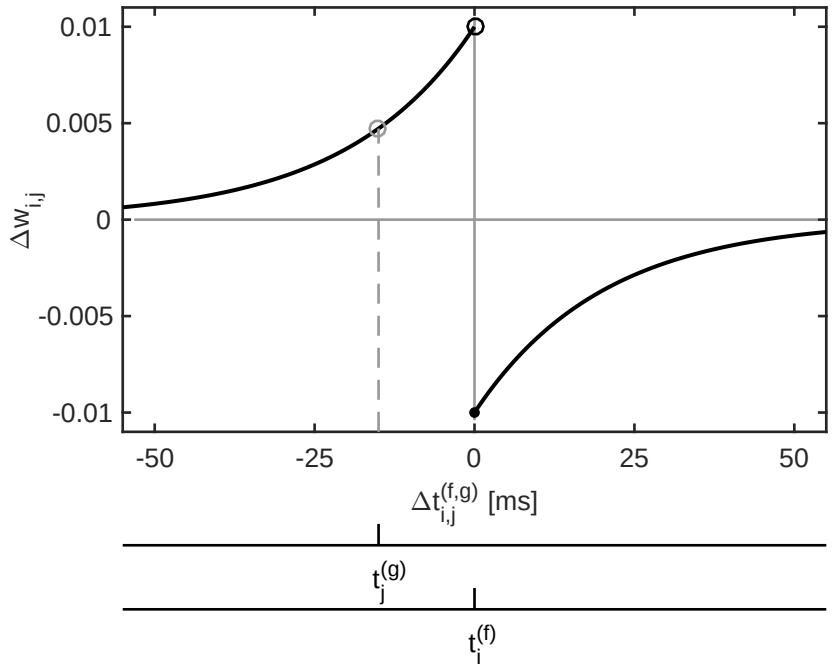


**Figure 3.4:** An antisymmetric spike-timing-dependent plasticity curve. A presynaptic spike shortly before a postsynaptic spike ($t_j^{(g)} < t_i^{(f)}$) causes a positive weight update ($\Delta w_{i,j} > 0$); a postsynaptic spike shortly before a presynaptic spike ($t_i^{(f)} < t_j^{(g)}$) causes a negative weight update ($\Delta w_{i,j} < 0$). Presynaptic neuron spikes that are at the same time as the postsynaptic neuron spike ($\Delta t_{i,j}^{(f,g)} = 0$) are assumed to have not had a causal role in eliciting the postsynaptic spike. Therefore, this relative spike timing is treated as causing a negative weight update. In this example of an STDP curve, $A_+ = A_- = 0.01$ and $\tau_+ = \tau_- = 10\,\text{ms}$.

neuron and reducing the strength of synapses with presynaptic neurons that did not contribute to the activation of the postsynaptic neuron. The STDP curve (Figure 3.4) has a discontinuity where presynaptic neuron spikes at the same time as the postsynaptic neuron, $\Delta t_{i,j}^{(f,g)} = 0$. In biological neural systems the transmission of potential is not instantaneous. The contribution of a presynaptic spike to postsynaptic potential can be modelled as an alpha-function: the difference of two exponential decays with an initial value of zero, a smooth rise to a peak, and a decay back to zero (see Section 5.4.3 for mathematical and graphical depictions). The presence of a transmission delay has not been factored into the calculation of the weight change, except for the STDP curve being defined to result in negative weight updates for $\Delta t_{i,j}^{(f,g)} = 0$.

The methods for calculating synapse weights for construction developed here are based on an online nearest-neighbour STDP implementation (Morrison et al., 2008): 'online' refers to the synapse weight updates occurring during the simulation at the presynaptic neuron and postsynaptic neuron spike times; 'nearest-neighbour' refers to

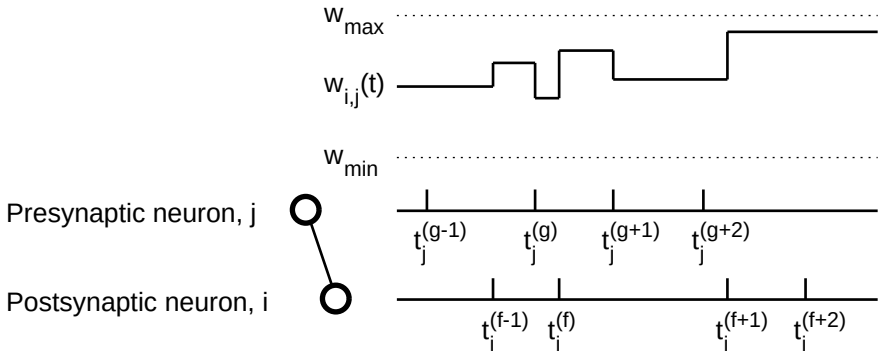


**Figure 3.5:** An example of the change in synapse weights resulting from online nearest-neighbour STDP. The synapse weight, $w_{i,j}(t)$, is updated at spike times, but only for the first postsynaptic spike after a presynaptic spike and for the first presynaptic spike on or after a postsynaptic spike. For this example, postsynaptic spikes at times $t_i^{(f-1)}$, $t_i^{(f)}$ and $t_i^{(f+1)}$ produce positive STDP updates, but not at $t_i^{(f+2)}$ due to it being the second postsynaptic spike after the presynaptic spike. Negative synapse weight updates occur at presynaptic spike times $t_j^{(g)}$ and $t_j^{(g+1)}$, but not at $t_j^{(g+2)}$. (The first presynaptic spike at $t_j^{(g-1)}$ is assumed to have no effect on the weight.) Note that this example has an increased magnitude of weight changes for better visibility.

synapse weight updates occurring only for the first presynaptic neuron spike after a postsynaptic neuron spike and the first postsynaptic neuron spike after a presynaptic neuron spike. A graphical representation of the sequential spiking of a presynaptic and postsynaptic neuron and the corresponding weight updates is given in Figure 3.5. A mathematical description of the implementation of online nearest-neighbour STDP in simulation will now be presented.

The synapses change weight at neuron spike times; therefore, the weight of the synapse between neurons $i$ and $j$ will be represented as a function of time, $w_{i,j}(t)$. To aid the technical specification of this STDP model, consecutive spike times from the same neuron are indicated by incrementing superscripts in parentheses. The times of the consecutive spikes of presynaptic neuron $j$ can be denoted as $t_j^{(g-1)}$, $t_j^{(g)}$, $t_j^{(g+1)}$ and $t_j^{(g+2)}$. The times of the consecutive spikes of postsynaptic neuron $i$ can be denoted as $t_i^{(f-1)}$, $t_i^{(f)}$, $t_i^{(f+1)}$ and $t_i^{(f+2)}$.

Two standard models of STDP are considered: additive STDP and multiplicative STDP. In additive STDP (addSTDP) the magnitude of a synapse weight update is independent of the present synapse weight. The divergence of synapse weights from additive updates is prevented by enforcing a hard limit, $w_{i,j} \in [w_{\min}, w_{\max}]$ where common numerical values of the weight limits are $w_{\min} = 0$ and $w_{\max} = 1$.

Given the relative spike timings displayed in Figure 3.5, the postsynaptic spike at time $t_i^{(f)}$ is the first after a presynaptic neuron spike ($t_i^{(f-1)} \leq t_j^{(g)} < t_i^{(f)} \leq t_j^{(g+1)}$) and

produces a positive weight update in the online addSTDP model,

$$w_{i,j}(t_i^{(f)}) = \min(w_{\max}, w_{i,j}(t_j^{(g)}) + \Delta w_{i,j}^{(f,g)}). \tag{3.3}$$

The $\min(\cdot,\cdot)$ function returns the smallest enclosed element and is used to enforce the upper weight limit, $w_{\max}$. Again referring to Figure 3.5, the presynaptic neuron spike at time $t_j^{(g+1)}$ is the first after a postsynaptic neuron spike ($t_j^{(g)} < t_i^{(f)} \leq t_j^{(g+1)} < t_i^{(f+1)}$) and produces a negative weight update,

$$w_{i,j}(t_j^{(g+1)}) = \max(w_{\min}, w_{i,j}(t_i^{(f)}) + \Delta w_{i,j}^{(f,g+1)}). \tag{3.4}$$

The $\max(\cdot,\cdot)$ function returns the largest enclosed element and is used to enforce the lower weight limit. Note that at presynaptic spike times, the relative spike time to the previous postsynaptic spike is $\Delta t_{i,j}^{(f,g+1)} \leq 0$; therefore, the STDP update (Equation 3.2) produces $\Delta w_{i,j}^{(f,g+1)} \leq 0$.

In multiplicative STDP (mSTDP) the magnitude of a synapse weight update is dependent on the present synapse weight. The online mSTDP model produces a positive weight update,

$$w_{i,j}(t_i^{(f)}) = w_{i,j}(t_j^{(g)}) + (w_{\max} - w_{i,j}(t_j^{(g)})) \cdot \Delta w_{i,j}^{(f,g)}, \tag{3.5}$$

at the postsynaptic neuron spike time $t_i^{(f)}$ if it is the first postsynaptic spike after a presynaptic neuron spike ($t_i^{(f-1)} \leq t_j^{(g)} < t_i^{(f)} \leq t_j^{(g+1)}$). Note the multiplicative factor that includes the current synapse weight in the STDP update and produces a soft maximum limit to the weight: the magnitude of positive updates decay to zero as the weight approaches the maximum, $w_{\max}$. A similar multiplicative factor is in the negative weight update (Equation 3.6), causing the magnitude of negative weight updates to decay to zero as the weight approaches the minimum, $w_{\min}$.

The synapse receives a negative weight update,

$$w_{i,j}(t_j^{(g+1)}) = w_{i,j}(t_i^{(f)}) + (w_{i,j}(t_i^{(f)}) - w_{\min}) \cdot \Delta w_{i,j}^{(f,g+1)}, \tag{3.6}$$

at the presynaptic neuron spike time $t_j^{(g+1)}$ if it is the first presynaptic spike after a postsynaptic neuron spike ($t_j^{(g)} < t_i^{(f)} \leq t_j^{(g+1)} < t_i^{(f+1)}$). Note that the STDP curve value for the change in synapse weight, $\Delta w_{i,j}^{(f,g+1)}$, is negative at the time of presynaptic spikes.

### 3.2.2 Past Neuron Activity and STDP

A model of STDP provides equations for updating synapse weights due to neuron activity; however, the activity of neurons and plasticity prior to the simulation start must be still be estimated. There is evidence that the relative timing between biological neuron spikes can repeat with millisecond precision (Masquelier, 2013). Therefore, a selected pattern of relative spike times in simulated neuron activity may be assumed

to have repeated prior to the simulation with the same relative spike timing some number of times, $M$. The assumption of low or no variability in past repetitions of a selected spike pattern simplifies the estimation of past plasticity. The accuracy of synapse weights calculated based on this assumption will be examined in simulations with models of spike noise in later chapters of this thesis.

A standard result of plasticity or training is that postsynaptic neurons tune to detect patterns or features in the activity of presynaptic neurons. Neuron construction for the task of pattern detection may create postsynaptic neurons to detect patterns in presynaptic neuron activity. The set of presynaptic neurons, $j \in J$, is simulated and the set of postsynaptic neurons, $i \in I$, includes surrounding neurons. The constructed postsynaptic neuron is assumed to be tuned to detect the selected spike pattern and have minimal response to other spike patterns. Therefore, the development of equations to calculate weights for constructed synapses will assume that the STDP resulting from other spike patterns is negligible.

A selected pattern of neuron activity observed in the simulation is assumed to be representative of past neuron activity that has been responsible for past STDP; however, the neuron that will be constructed is not simulated prior to construction. Another process is necessary to predict the activity of the neuron that will be constructed (the surrounding neurons). A method for predicting the spike times of surrounding neurons is developed in Chapter 5. For the purposes of the synapse weight calculations presented in this section, it will be assumed that the relative spike time of the neuron that will be constructed has been provided.

These assumptions may now be developed into equations for calculating synapse weights for new neurons. Using a nearest-neighbour STDP model, the focus can be placed on the first postsynaptic spike after the presynaptic spike and the first presynaptic spike after a postsynaptic spike. Using the nomenclature introduced earlier, the postsynaptic neuron $i$ has a given or predicted spike time $t_i^{(f)}$ and each presynaptic neuron $j \in J$ has the two nearest presynaptic spike times, $t_j^{(g)} < t_i^{(f)} \leq t_j^{(g+1)}$. This gives observed relative spike times $\Delta t_{i,j}^{(f,g)}$ and $\Delta t_{i,j}^{(f,g+1)}$.

In the event that a presynaptic neuron does not spike after the given postsynaptic neuron spike time, the calculation of synapse weights may be delayed. In practice, however, the construction of new neurons and synapses should complete in a finite time. Therefore, in addition to the nearest-neighbour restriction, synapse weight calculations can be restricted to presynaptic spikes within a finite time window.

A time window for eligible presynaptic neuron spikes is defined for the given or predicted postsynaptic neuron spike time: $[t_i^{(f)} - T_+, t_i^{(f)} + T_-]$ for absolute spike times or $[-T_+, T_-]$ for relative spike times (Equation 3.1). Constants $T_+, T_- \geq 0$, with subscripts $+$ and $-$, denote limits on the STDP curve for estimating positive and negative weight updates, respectively. At the conclusion of the construction time window, $t_i^{(f)} + T_-$, the weight calculations can be completed and the postsynaptic neuron can be added to the network (assuming all other conditions are met).

The values of $T_+$ and $T_-$ can be selected as a number several multiples of the STDP decay constant such that the impact of weight updates of more distant spikes can be

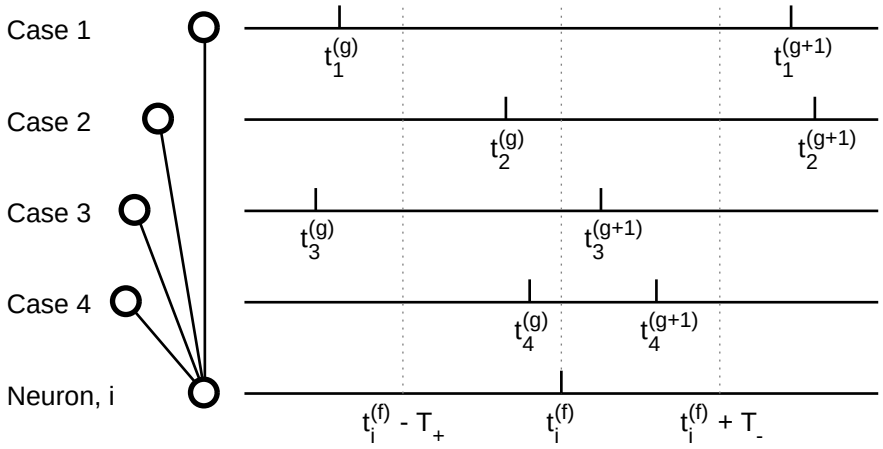


**Figure 3.6:** The four cases of presynaptic neuron activity for nearest-neighbour spikes and an eligibility time window for construction. For a given or predicted postsynaptic spike time $t_i^{(f)}$, only nearest-neighbour presynaptic neuron spikes inside the time range $[t_i^{(f)} - T_+, t_i^{(f)} - T_-]$ are assumed to be correlated with the postsynaptic spike or have a significant effect on the current synapse weight. Presynaptic spikes outside this range are assumed to be uncorrelated or have a negligible effect on the current synapse weight and are excluded from weight calculations for construction.

assumed to be negligible. Alternatively, nearer time-window limits can be selected under the assumption that more distant presynaptic spikes are not correlated with the postsynaptic neuron spike. In either case, it is assumed that the time between the past repetitions of the relative spike timings was sufficient that there was negligible interaction between them. Given a simulation of a mature neural network, the observation of the relative spike times, $\Delta t_{i,j}^{(f,g)}$ and $\Delta t_{i,j}^{(f,g+1)}$, can be assumed to be the $M^{\text{th}}$ occurrence including times prior to the start of the simulation.

The nearest-neighbour presynaptic spike before and after a given or predicted postsynaptic neuron spike time and the time window allows four distinct cases of presynaptic activity that can be observed (see Figure 3.6 for a graphical representation):

1. The presynaptic neuron does not spike within the time window before or after the postsynaptic neuron: $\Delta t_{i,j}^{(f,g)} < -T_+$ and $T_- < \Delta t_{i,j}^{(f,g+1)}$.

2. The presynaptic neuron spikes in the time window before the postsynaptic neuron but not after: $-T_+ \leq \Delta t_{i,j}^{(f,g)} < 0$ and $T_- < \Delta t_{i,j}^{(f,g+1)}$.

3. The presynaptic neuron spikes in the time window after the postsynaptic neuron but not before: $\Delta t_{i,j}^{(f,g)} < -T_+$ and $0 \leq \Delta t_{i,j}^{(f,g+1)} \leq T_-$.

4. The presynaptic neuron spikes in the time window both before and after the postsynaptic neuron: $-T_+ \leq \Delta t_{i,j}^{(f,g)} < 0$ and $0 \leq \Delta t_{i,j}^{(f,g+1)} \leq T_-$.

The detection of eligible presynaptic spikes can be a performance evaluation process in a constructive algorithm. Neuron construction may be triggered at the conclusion of the eligibility time window, that is, $t_c = t_i^{(f)} + T_-$.

The cases of eligible presynaptic activity around a given postsynaptic spike time will now be used in combination with the STDP models and developed assumptions to calculate synapse weights for a constructed postsynaptic neuron.

### 3.2.3 Calculating Synapse Weights from STDP Estimates

This section presents equations for calculating the weights of synapses to a constructed neuron based on the STDP models and assumptions described in the previous section. Four methods for calculating synapse weights are presented here: two based on the additive STDP model and two based on the multiplicative STDP model. Each of the methods presented here come as a set of solutions for the cases of eligible presynaptic neuron spikes (Figure 3.6). Each of these processes for calculating synapse weights are spike-timing-dependent; therefore, constructive algorithms that implement any these processes perform spike-timing-dependent construction (STDC).

Direct equations are found for $M$ past iterations of additive STDP and for $M$ past iterations of multiplicative STDP. Equations to calculate the result of $M$ iterations of the multiplicative STDP can be developed using solutions to the sum of a geometric series. A less computationally expensive alternative to solving the sum to the geometric series is also presented: approximating the change in synapse weight with the number of past iterations of the spike pattern treated as a continuous variable.

Past study of the additive STDP model found that synapse weights frequently converge to bimodal distributions with concentrations at or near the maximum and minimum weight values (Song et al., 2000). Therefore, calculation of new synapse weights in a neural network could assume the past repetition of the observed pattern of spike timings has resulted in the convergence of weights to maximum or minimum values.

The synapse weight between presynaptic neuron $j$ and postsynaptic neuron $i$ will be denoted $w_{i,j}[m]$ with the value $m$ in the square brackets used to indicate the number of discrete repetitions of the given relative spike times. Simulations typically start at time $t = 0$, and in non-expanding simulations the weight at the start of the simulation, $w_{i,j}(0)$, is often initialised randomly without a prediction of past STDP. The initialisation of synapse weights without assuming past plasticity updates could be expressed as $w_{i,j}(0) = w_{i,j}[0]$. The mature network assumption, however, may be used to define the synapse weight at zero STDP updates as being a long time, $t_L$, before the start of the simulation time, $w_{i,j}[0] = w_{i,j}(-t_L)$. It is important to maintain this distinction between the weight as a discrete function of relative spike timing iterations, $w_{i,j}[m]$, and as a continuous function of time, $w_{i,j}(t)$.

#### 3.2.3.1 Specified Iterations: Additive STDP

The new synapse weights between the set of presynaptic neurons, $j \in J$, and the new postsynaptic neuron, $i$, are calculated as a prediction of surrounding synapses at the time of construction, $\hat{w}_{i,j}(t_c)$. Additive STDP weight updates are independent of the current synapse weight; therefore, the total change in weight due to a specified number of iterations, $M$, of a pattern of relative spike times is $M$ times the update values for the spike pairs, $\Delta w_{i,j}^{(f,g)}$ and $\Delta w_{i,j}^{(f,g+1)}$. The final synapse weight can then be calculated by adding the total weight change to an initial weight value, $w_{i,j}[0]$, and lastly clipping the final weight to the limits, $w_{i,j}(t) \in [w_{\min}, w_{\max}]$.

The activity of each presynaptic neuron will satisfy one of the four cases of spike eligibility (Figure 3.6), which will indicate what equation to use to calculate the weight of the constructed synapse. Eligible presynaptic spikes will be used to calculate new synapse weights with the difference in spike times between pre- and postsynaptic spikes being denoted $\Delta t_{i,j}^{(f,g)}$ for the nearest presynaptic spike before the postsynaptic spike and denoted $\Delta t_{i,j}^{(f,g+1)}$ for the nearest presynaptic spike after the postsynaptic spike.

The first of the four cases of past activity has no presynaptic spikes within the observed time window of eligibility. If there is no presynaptic spike in the time window, then all presynaptic activity is assumed to have negligible effect on the synapse weight or cancel out. Therefore, an estimate of the synapse weight, $\hat{w}_{i,j}(t_c)$, is assumed to remain at its initial value, that is,

$$\hat{w}_{i,j}(t_c) = w_{i,j}[0]. \tag{3.7}$$

An STDP curve (Figure 3.4) with a negative bias (larger area in the negative weight update region) can cause the depression of synapse weights between neurons with uncorrelated activity (Song et al., 2000). Therefore, the estimate of the synapse weight could be set to the minimum,

$$\hat{w}_{i,j}(t_c) = w_{\min}. \tag{3.8}$$

The shape of the STDP curve is controlled by the amplitude and time constant values in Equation 3.2. It is assumed that synapses that observe eligible relative spike timings have the STDP from the $M$ iterations of those relative spike timings dominate any influence from biased plasticity and random neuron activations.

The second case of presynaptic neuron activity is a spike inside the eligibility time window prior to the postsynaptic neuron spike, $-T_+ \leq \Delta t_{i,j}^{(f,g)} < 0$, and no eligible presynaptic spike after. This relative spike timing causes a positive weight update, $\Delta w_{i,j}^{(f,g)} > 0$. Assuming a total of $M$ updates of value $\Delta w_{i,j}^{(f,g)}$ and that all other plasticity is insignificant, the synapse weight is calculated,

$$\hat{w}_{i,j}(t_c) = \min(w_{\max}, w_{i,j}[0] + M \cdot \Delta w_{i,j}^{(f,g)}). \tag{3.9}$$

The $\min(\cdot,\cdot)$ function returns the smallest enclosed element and is used to enforce the upper weight limit.

The third case of possible presynaptic neuron activity is a spike inside the eligibility time window after the postsynaptic neuron spike, $0 \leq \Delta t_{i,j}^{(f,g+1)} \leq T_-$. This relative spike timing causes a negative weight update, $\Delta w_{i,j}^{(f,g+1)} < 0$. Assuming all other plasticity is negligible, a total of $M$ updates of value $\Delta w_{i,j}^{(f,g+1)}$ give a synapse weight estimate,

$$\hat{w}_{i,j}(t_c) = \max(w_{\min}, w_{i,j}[0] + M \cdot \Delta w_{i,j}^{(f,g+1)}). \tag{3.10}$$

The $\max(\cdot,\cdot)$ function returns the largest enclosed element and is used to enforce the lower weight limit.

The fourth case of possible presynaptic neuron activity has spikes inside the eligibility time window both before and after the postsynaptic spike, $-T_+ \leq \Delta t_{i,j}^{(f,g)} < 0$ and $0 \leq \Delta t_{i,j}^{(f,g+1)} \leq T_-$. This results in a positive weight update, $\Delta w_{i,j}^{(f,g)} > 0$, and a negative weight update, $\Delta w_{i,j}^{(f,g+1)} < 0$. Under the additive STDP model described, updates are independent of the current weight; therefore, the spike triplet can be considered as independent spike pairs for updates. The order of synapse weight updates is important in the calculation of the maximum final weight as a negative update always occurs last in the repetition of the spike triplet. Combining Equations 3.9 and 3.10 we can estimate the synapse weight resulting from $M$ total repetitions of this neural activity as,

$$\begin{aligned}\hat{w}_{i,j}(t_c) = \min(&w_{\max} + \Delta w_{i,j}^{(f,g+1)}, \\ &\max(w_{\min}, w_{i,j}[0] + M \cdot [\Delta w_{i,j}^{(f,g)} + \Delta w_{i,j}^{(f,g+1)}])).\end{aligned} \tag{3.11}$$

Note that the maximum final weight is one negative STDP update value less than the hard-limit, $w_{\max}$, due to the negative update being the final update.

#### 3.2.3.2 Specified Iterations: Multiplicative STDP

A similar approach may be used to find the synapse weight that results from multiplicative STDP after the $M^{\text{th}}$ repetition of relative spike timings. Each of the four cases of presynaptic and postsynaptic neuron activity may be considered in turn; however, the calculation of final weights is complicated by the dependence of weight updates on the weight of the synapse at that time. The spike-timing-dependent factors for multiplicative STDP (Equation 3.2) are abbreviated in this section, $\Delta w_{i,j}^{(f,g)} = \Delta w_+$ and $\Delta w_{i,j}^{(f,g+1)} = -\Delta w_-$. Note that the negative value of $\Delta w_{i,j}^{(f,g+1)}$ is made more explicit in this abbreviation through the inclusion of the minus sign.

The degenerate case of there being no presynaptic spike inside the time window of eligibility considers all presynaptic activity to have negligible effect on the synapse weight; therefore, the weight is estimated to remain at its initial value,

$$\hat{w}_{i,j}(t_c) = w_{i,j}[0]. \tag{3.12}$$

The multiplicative STDP model attenuates the weight updates as they approach the limits; therefore, unlike addSTDP there is not a strong trend to produce bimodal

distributions of synapse weights or depress synapses with uncorrelated neurons to the minimum weight.

The second case of eligible presynaptic neuron spikes has a spike in the time window prior to the postsynaptic neuron spike. Each of the $M$ updates of mSTDP must consider the weight of the synapse at that time. The weight of the synapse can be treated as a discrete function on the positive integer number of updates from zero to $M$: $w_{i,j}[0], w_{i,j}[1], w_{i,j}[2], \ldots, w_{i,j}[M-1], w_{i,j}[M]$. Using the abbreviation $\Delta w_+$ to represent the positive STDP curve value $\Delta w_{i,j}^{(f,g)}$, successive multiplicative STDP updates up to $M$ can be found and terms collected,

$$\begin{aligned}
w_{i,j}[1] &= w_{i.j}[0] + (w_{\max} - w_{i,j}[0])\Delta w_+, \\
w_{i,j}[1] &= (1-\Delta w_+)w_{i,j}[0] + w_{\max}\Delta w_+, \\
w_{i,j}[2] &= (1-\Delta w_+)w_{i,j}[1] + w_{\max}\Delta w_+, \\
w_{i,j}[2] &= (1-\Delta w_+)^2 w_{i,j}[0] + (1-\Delta w_+)w_{\max}\Delta w_+ + w_{\max}\Delta w_+, \\
w_{i,j}[3] &= (1-\Delta w_+)^3 w_{i,j}[0] + (1-\Delta w_+)^2 w_{\max}\Delta w_+ \\
&\quad + (1-\Delta w_+)w_{\max}\Delta w_+ + w_{\max}\Delta w_+, \\
&\vdots \\
w_{i,j}[M] &= (1-\Delta w_+)^M w_{i,j}[0] + (1-\Delta w_+)^{M-1} w_{\max}\Delta w_+ + \ldots \\
&\quad + (1-\Delta w_+)^2 w_{\max}\Delta w_+ + (1-\Delta w_+)w_{\max}\Delta w_+ + w_{\max}\Delta w_+. \qquad (3.13)
\end{aligned}$$

Within this resulting synapse weight is the sum of a geometric series with a base, $w_{\max}\Delta_+$, and a ratio, $(1-\Delta w_+)$. The sum of this geometric series can be extracted,

$$\begin{aligned}
S_{M-1} &= (1-\Delta w_+)^{M-1} w_{\max}\Delta w_+ + (1-\Delta w_+)^{M-2} w_{\max}\Delta w_+ + \ldots \\
&\quad + (1-\Delta w_+)^2 w_{\max}\Delta w_+ + (1-\Delta w_+)w_{\max}\Delta w_+ + w_{\max}\Delta w_+.
\end{aligned}$$

The solution to the sum of the geometric series may be simplified through an expansion using a factor $[1-(1-\Delta w_+)]$ and then cancelling terms,

$$\begin{aligned}
[1-(1-\Delta w_+)]\cdot S_{M-1} &= (1-\Delta w_+)^{M-1} w_{\max}\Delta w_+ + \ldots \\
&\quad + (1-\Delta w_+)w_{\max}\Delta w_+ \\
&\quad + w_{\max}\Delta w_+ \\
&\quad - (1-\Delta w_+)^M w_{\max}\Delta w_+ \\
&\quad - (1-\Delta w_+)^{M-1} w_{\max}\Delta w_+ - \ldots \\
&\quad - (1-\Delta w_+)w_{\max}\Delta w_+, \\
[1-(1-\Delta w_+)]\cdot S_{M-1} &= w_{\max}\Delta w_+ - (1-\Delta w_+)^M w_{\max}\Delta w_+, \\
S_{M-1} &= \frac{w_{\max}\Delta w_+ - (1-\Delta w_+)^M w_{\max}\Delta w_+}{1-(1-\Delta w_+)}, \\
S_{M-1} &= w_{\max} - (1-\Delta w_+)^M w_{\max}. \qquad (3.14)
\end{aligned}$$

Substituting the simplified sum of the geometric series (Equation 3.14) into the weight after the $M^{\text{th}}$ update (Equation 3.13) gives a direct solution:

$$\begin{aligned} w_{i,j}[M] &= (1-\Delta w_+)^M w_{i,j}[0] + S_{M-1}, \\ w_{i,j}[M] &= (1-\Delta w_+)^M w_{i,j}[0] + w_{\max} - (1-\Delta w_+)^M w_{\max}, \\ w_{i,j}[M] &= w_{\max} + (w_{i,j}[0] - w_{\max})(1-\Delta w_+)^M. \end{aligned} \tag{3.15}$$

This synapse weight resulting from discrete updates can be used as the weight value at the construction time,

$$\hat{w}_{i,j}(t_{\text{c}}) = w_{i,j}[M] = w_{\max} + (w_{i,j}[0] - w_{\max})(1-\Delta w_+)^M. \tag{3.16}$$

This same approach can be taken to solve the weight calculation for a presynaptic neuron spike in the time window after the postsynaptic neuron spike. Using the abbreviation $\Delta w_{i,j}^{(f,g+1)} = -\Delta w_-$, the synapse weight resulting from $M$ updates under the mSTDP model may be calculated directly as,

$$\hat{w}_{i,j}(t_{\text{c}}) = w_{\min} + (w_{i,j}[0] - w_{\min})(1-\Delta w_-)^M. \tag{3.17}$$

See Appendix B.1.1 for the full derivation.

If the presynaptic neuron spikes both before and after the postsynaptic neuron inside the time window, the result of $M$ updates can be estimated using a similar though marginally more complex process. Given that the weight updates produced by the triplet are treated as two interdependent pairs of spikes by the mSTDP model, $M$ iterations of the triplet result in $M$ positive weight updates and $M$ negative weight updates. The order of the updates must also be considered. See Appendix B.1.2 for a full derivation of the mSTDP weight estimate equation for a spike triplet. After $M$ iterations of the spike triplet the weight of the synapse can be estimated as,

$$\hat{w}_{i,j}(t_{\text{c}}) = A_{\text{ST}} + (w_{i.j}[0] - A_{\text{ST}})(1-\Delta w_+)^M(1-\Delta w_-)^M, \tag{3.18}$$

where the mSTDP weight estimate asymptote for the spike triplet is,

$$A_{\text{ST}} = \frac{w_{\max}\Delta w_+(1-\Delta w_-) + w_{\min}\Delta w_-}{1-(1-\Delta w_+)(1-\Delta w_-)}. \tag{3.19}$$

#### 3.2.3.3 Continuous Iteration Variable: Multiplicative STDP

An approximate solution to the synapse weight estimates for multiplicative STDP may be sufficiently accurate and reduce the computational expense of the calculation. Although the formula for multiplicative STDP is applied discretely in weight updates, this formula can be approximated through treatment as a continuous function in the number of mSTDP iterations, $m$. The numerical outcome will be an approximation of a discrete number of mSTDP updates if $m$ is restricted to non-negative integers. In the continuous approximation of multiplicative STDP, the discrete change in weight from

updates (Equations 3.5 and 3.6) is treated as equivalent to the derivative with respect to the number of updates,

$$\dot{\hat{w}}_{i,j}(m) = \begin{cases} [w_{\max} - w_{i,j}(m)] \cdot \Delta w_+ & \text{if } \Delta t_{i,j}^{(f,g)} < 0, \\ [w_{\min} - w_{i,j}(m)] \cdot \Delta w_- & \text{if } \Delta t_{i,j}^{(f,g)} \geq 0. \end{cases} \tag{3.20}$$

Solving the differential equation for both of these cases the weight becomes

$$\hat{w}_{i,j}(m) = \begin{cases} w_{\max} + (w_{i,j}[0] - w_{\max}) \cdot \exp(-m \cdot \Delta w_+) & \text{if } \Delta t_{i,j}^{(f,g)} < 0, \\ w_{\min} + (w_{i,j}[0] - w_{\min}) \cdot \exp(-m \cdot \Delta w_-) & \text{if } \Delta t_{i,j}^{(f,g)} \geq 0. \end{cases} \tag{3.21}$$

See Appendix B.2.1 for the full derivation of this continuous approximation of positive and negative mSTDP. This approximation of the result of discrete updates can be used to calculate constructed synapse weights by setting the continuous variable equal to $M$, the assumed number of iterations of the relative spike timing (a non-negative integer value). The estimated synapse weight at the construction time then becomes

$$\hat{w}_{i,j}(t_c) = \begin{cases} w_{\max} + (w_{i,j}[0] - w_{\max}) \cdot \exp(-M \cdot \Delta w_+) & \text{if } \Delta t_{i,j}^{(f,g)} < 0, \\ w_{\min} + (w_{i,j}[0] - w_{\min}) \cdot \exp(-M \cdot \Delta w_-) & \text{if } \Delta t_{i,j}^{(f,g)} \geq 0. \end{cases} \tag{3.22}$$

The continuous function approximation can also be developed for the case of a pre-post-pre spike triplet. The continuous derivative of the synapse weight with respect to iterations is the combination of positive and negative weight updates,

$$\begin{aligned} \dot{\hat{w}}_{i,j}(m) &= [w_{\max} - w_{i,j}(m)] \cdot \Delta w_+ + [w_{\min} - w_{i,j}(m)] \cdot \Delta w_- \\ &= w_{\max} \Delta w_+ + w_{\min} \Delta w_- - (\Delta w_+ + \Delta w_-) \cdot w_{i,j}(m). \end{aligned} \tag{3.23}$$

Solving the differential equation for an initial weight of $w_{i,j}[0]$ the weight can be found,

$$\hat{w}_{i,j}(m) = A_{\text{STa}} + (w_{i,j}[0] - A_{\text{STa}}) \cdot \exp(-m \cdot [\Delta w_+ + \Delta w_-]), \tag{3.24}$$

where the asymptote for the approximate estimate of weight developed through a spike triplet with mSTDP,

$$A_{\text{STa}} = \frac{w_{\max} \Delta w_+ + w_{\min} \Delta w_-}{\Delta w_+ + \Delta w_-}. \tag{3.25}$$

Therefore, the synapse weight estimate at the time of construction is

$$\hat{w}_{i,j}(t_c) = A_{\text{STa}} + (w_{i,j}[0] - A_{\text{STa}}) \cdot \exp(-M \cdot [\Delta w_+ + \Delta w_-]). \tag{3.26}$$

See Appendix B.2.2 for a derivation of the equations for estimating multiplicative STDP using continuous approximations of the iterations of relative spike timings.

The rate of change of the discrete updates does not change continuously; therefore, the continuous approximation is predicted to have a small under-prediction of the total change in synapse weight from $M$ multiplicative STDP updates. Furthermore, the continuous approximation does not take into account that in a spike triplet the negative update occurs after the positive update. In discrete updates the positive weight update first increases the weight, causing the negative update to have a larger weight-dependent factor and larger negative update. The continuous approximation of iterations does not include this small bias towards a negative weight asymptote for spike triplets.

#### 3.2.3.4 Predicted Convergence: Additive STDP

Given past findings of additive STDP models resulting in convergence to bimodal distributions of weights (Song et al., 2000), it could be assumed that the number of prior repetitions of an observed spike train, $M$, is large and all synapse weights have converged. For a given or predicted postsynaptic neuron spike time, $t_i^{(f)}$, the result of convergence will be different for each of the four cases of presynaptic neuron activity in the eligibility time window.

In the first case the presynaptic neuron does not spike in the eligibility time window; therefore, all activity of that presynaptic neuron is assumed to be uncorrelated with the postsynaptic neuron activity. Given the assumption that convergence has taken place, two responses to this observation are: 1) All past synaptic plasticity is assumed to be negligible; therefore, the synapse weight will be unchanged, remaining at the initial synapse weight, $w_{i,j}[0]$,

$$\hat{w}_{i,j}(t_\mathrm{c}) = w_{i,j}[0]. \tag{3.27}$$

2) The presynaptic neuron activity is assumed to be uncorrelated with the postsynaptic neuron activity and the synapse is depressed to the minimum value,

$$\hat{w}_{i,j}(t_\mathrm{c}) = w_\mathrm{min}. \tag{3.28}$$

The convergence to the minimum weight can be produced for uncorrelated presynaptic and postsynaptic activity if the additive STDP model has a bias towards depression (Song et al., 2000).

In the second case the presynaptic neuron spikes inside the time window prior to the postsynaptic neuron, $-T_+ \leq \Delta t_{i,j}^{(f,g)} < 0$, resulting in a positive weight update. This relative spike timing is assumed to have occurred a large number of times prior to the start of the simulation; therefore, the synapse weight can be estimated as having converged to the maximum weight,

$$\hat{w}_{i,j}(t_\mathrm{c}) = w_\mathrm{max}. \tag{3.29}$$

In the third case the presynaptic neuron spikes inside the time window after (or at) the postsynaptic neuron spike time, $0 \leq \Delta t_{i,j}^{(f,g+1)} \leq T_-$, resulting in a negative weight update. This relative spike timing is assumed to have occurred a large number of times prior to the start of the simulation; therefore, the synapse weight can be estimated as having converged to the minimum weight,

$$\hat{w}_{i,j}(t_\mathrm{c}) = w_\mathrm{min}. \tag{3.30}$$

In the fourth case the presynaptic neuron spikes inside the eligibility time window both before and after the postsynaptic neuron spike. Assuming a sufficient number of past updates for synapse weights to have converged, the total change in weight produced by the positive and negative weight updates, $\Delta w_{i,j}^{(f,g)} + \Delta w_{i,j}^{(f,g+1)}$ may be

positive, negative or zero. Therefore, the synapse weight may be estimated as,

$$\hat{w}_{i,j}(t_c) = \begin{cases} w_{\max} - \Delta w_{i,j}^{(f,g+1)} & \text{if } \Delta w_{i,j}^{(f,g)} > \Delta w_{i,j}^{(f,g+1)}, \\ w_{i,j}[0] & \text{if } \Delta w_{i,j}^{(f,g)} = \Delta w_{i,j}^{(f,g+1)}, \\ w_{\min} & \text{if } \Delta w_{i,j}^{(f,g)} < \Delta w_{i,j}^{(f,g+1)}. \end{cases} \tag{3.31}$$

Note that the maximum final synapse weight is one negative update less than the hard-limit, $w_{\max} - \Delta w_{i,j}^{(f,g+1)}$, due to the final positive weight update (which increases the synapse weight to the maximum, $w_{\max}$) being followed by the final negative weight update.

## 3.3 Discussion

This chapter has introduced simulation expansion and a number of associated concepts, assumptions and processes for adding neurons and synapses to a simulation. These developments are aligned with a major goal of the thesis (Section 2.3): developing constructive algorithms that are compatible with simulations of biological neural systems. The concept of simulation expansion and the related assumptions have been logically constructed to enable the development of constructive algorithms that are compatible with simulations of biological neural systems and models of STDP. However, these assumptions and idealisations may result in brittle performance. As a result, the developed processes for synapse weight calculation (and later performance evaluation) will be tested in a range of conditions in the following chapters.

Four sets of equations for calculating synapse weights have been presented in this chapter, based on either additive or multiplicative models of STDP. The selection of an approach to synapse weight calculation in a constructive algorithm should depend on the conditions and goals of the simulation. Simulated studies of synaptic plasticity may benefit from introducing new neurons with synapses that have experienced a low level of prior plasticity and adapting the constructed synapses. Simulations that aim to study the behaviour and function of mature neural systems may benefit from introducing new neurons with synapses that have weights near or at converged values.

Simulations of biological neural systems may use a variety of plasticity models. Attractive aspects of the additive and multiplicative STDP rules include their simplicity, their wide use and study in spiking neural networks, and successful demonstrations of their learning capabilities (Legenstein et al., 2005; Masquelier et al., 2009). A drawback of these STDP models is that they do not accurately reproduce observations that spike triplets and high spike rates can produce potentiation regardless of the relative timing of individual spikes (Graupner & Brunel, 2012; Pfister & Gerstner, 2006). Equations for synapse construction may be developed from other models of plasticity; however, the synapse weight calculations may require additional assumptions about past activity and may increase in complexity. The development of synapse weight calculations for constructive expansion from alternative models of plasticity is an avenue for future work.

The study of the performance of the constructive processes will progress methodically from the low-level performance of weight calculations to the high-level performance of spike latencies and spike pattern detection. Numerical values of synapse weights produced through constructive calculations and neurons simulated with STDP will be compared for different presynaptic activity conditions (Chapter 4). The comparison of synapse weight calculations with the results of iterative updates through STDP for the ideal conditions assumed will performed to confirm the correctness of developed equations. The same comparisons are performed for different presynaptic spike noise conditions to examine relative sensitivities of synapse weight calculations for construction and the iterative simulation of STDP.

Although plausible synapse weights are important, discrepancies in constructed synapse weights may be acceptable if the resulting spike timing or latency of the simulated neurons is comparable. The spike latencies of constructed postsynaptic neurons will be compared with simulated neurons that have synapses adapted iteratively through STDP (Chapter 5). If constructed and adapted neurons produce similar spike latencies relative to the start of a spike pattern this will be taken as evidence of construction producing a plausible neuron from the proposed assumptions for simulation expansion.

The synapse weight calculation processes developed here require the spike time of a postsynaptic neuron that has not been constructed. There may be contexts where this spike time can be provided externally by the designer, resulting in a type of supervised learning. However, providing spike times to perform neuron construction increases the effort of the designer in producing input data and may be a non-trivial task in continuous simulations of spiking neural networks with complex inputs and activity. Unsupervised methods for the prediction of the spike times of surrounding postsynaptic neurons can be applied to a wider range of tasks and conditions, and may more easily scale to larger and more complex simulations. Chapter 5 presents an unsupervised method for predicting the spike time of surrounding postsynaptic neurons.

# Chapter 4

# Validation of STDP Estimation

This chapter presents a numerical validation and investigation of developed equations for calculating weights for constructed synapses. The previous chapter developed equations assuming idealised past activity and STDP; the initial validation of the developed equations demonstrates the equivalence of constructed synapse weights and synapse weights resulting from simulating the idealised activity with STDP. This is followed by a comparison of the distributions of synapse weights calculated for construction and resulting from STDP simulation under common models of spike noise.

## 4.1 Aims

This thesis aims to produce constructive algorithms that are compatible with common models for simulating biological neural networks. The equations developed in Chapter 3 for calculating weights for constructed synapses aim to produce plausible synapse weight values for simulations that include additive STDP or multiplicative STDP. This chapter presents and compares the weights resulting from calculations for synapse construction and from simulating STDP for ideal and stochastic presynaptic activity.

The first set of investigations presented aim to numerically validate the correctness of developed weight equations for constructed synapses for the idealised activity assumed in development. Neural activity and plasticity meets the assumed ideal under the following conditions:

1. The selected pattern of presynaptic spike times observed in the simulation has occurred a specified number of times, $M$, without variation.

2. A postsynaptic neurons spikes once with the same relative spike time for each of the $M$ occurrences of the selected pattern of presynaptic spikes.

3. Synapse weights are the result of a specified initial weight adapting under the given model of STDP in response to the $M$ repetitions of the spike pattern.

4. All other plasticity that has occurred, including interactions between past repetitions of the observed relative spike times, is negligible.

Conditions that conform to these assumptions only allow deterministic synapse weight outcomes; therefore, the results from ideal conditions do not require statistical analysis.

The initial validation will focus on the performance of estimates of increasing specified numbers of spike pattern repetition, $M$. Synapse weights will be calculated directly using the equations for synapse construction and iteratively using the associated STDP model. The synapse weights are expected to be equal up to the limits of numerical precision.

The calculation of synapse weights that use a continuous variable to find an approximate estimation of multiplicative STDP will also be evaluated. The numerical investigation of the continuous-iteration approximation will quantify the error in estimates for ideal conditions. Synapse weight calculations that assume the convergence of additive STDP represent the final result of plasticity. The converged weight outcomes do not suit a numerical comparison with the result of low numbers of STDP updates that aim to model plasticity that is in progress. The performance of convergence predictions will be revisited in Chapter 7 in the context of pattern detection performance.

Determining the correctness of equations that calculate synapse weights as estimates of past STDP is a preliminary step in assessing the applicability of the processes in simulations of biological neural networks. If the STDP estimates are equivalent to the outcome of iteratively updating weights for STDP, then under these conditions the synapse weight outcomes have equivalent biological plausibility as the base STDP model.

There is evidence that the relative-timings of neuron spikes can be reliably reproduced for a given stimulus (Masquelier, 2013); however, in vivo neuron activity typically appears, and is more safely assumed to be, noisy and variable. The second aim of the numerical evaluations presented in this chapter is to investigate the accuracy of STDP estimates when the presynaptic activity is stochastic. Two types of stochastic presynaptic activity will be examined:

1. Gaussian distributions
2. Poisson processes

Relative spike times are generated to produce a spike pattern or spike train for each iteration. Gaussian distributions are generated through the application of zero-mean Gaussian noise to presynaptic spike times as an approximation of natural variability of time-coded activity. Poisson processes may be used to simulate rate-coded neural activity and the random and independent background activation of presynaptic neurons. Further details of these stochastic processes will be provided in later sections.

As the presynaptic activity is stochastic, the changes in synapse weight through STDP are also expected to be stochastic. As such, Monte Carlo methods (random sampling) will be used to estimate the distribution of adapted and constructed synapse weights that result from Gaussian distributions and Poisson process-based presynaptic spike times. A finding of similar distributions of constructed synapse weights with those resulting from iterative STDP updates is considered evidence that the synapse weight calculations produce plausible outcomes for the STDP model. The qualitative

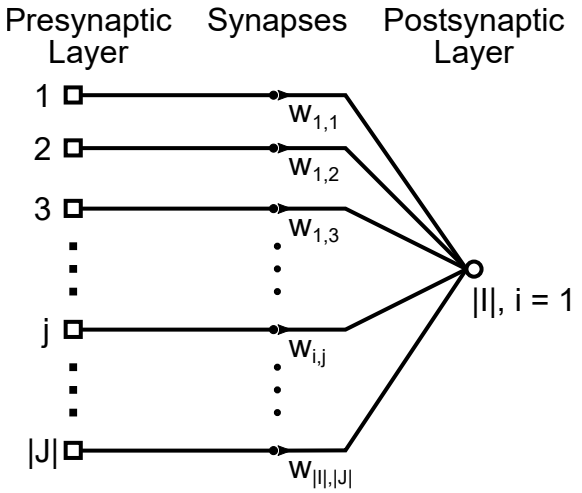


**Figure 4.1:** Numerical evaluations are performed for a network composed of a set of presynaptic neurons (with total number of neurons $|J|$) and a single postsynaptic neuron (a set with a single member, $|I| = 1$).

and quantitative differences in synapse weight distributions resulting from stochastic activity can also be examined as an indication of the limitations of assuming ideal past neural activity.

## 4.2 Models and Methods

The numerical evaluations are performed with a minimal network architecture for STDP: a set of presynaptic neurons and a single postsynaptic neuron (Figure 4.1). The simulations performed in this chapter specify neuron activity directly in terms of relative spike times. Given the direct specification of spike times, neurons do not need additional parameters or variables for simulation. The exact specifications for relative spike times are given in the associated sections for simulation results for ideal neural activity, Gaussian-distributed activity, and Poisson processes.

The synapse weights are not functional in the simulations presented in this chapter, but are maintained and updated for the purposes of evaluating the results of STDP models and synapse construction. A standard model of synapse weights is used, treating each weight as a single real value, $w_{i,j} \in [w_{\min}, w_{\max}] = [0, 1]$.

The models of STDP and the equations for synapse weight calculation have been given in Chapter 3; for brevity, this chapter will point to the sections describing these models and equations rather than restate them. The nearest-neighbour models of additive STDP and multiplicative STDP are described in Section 3.2.1. The equations for synapse weight calculations are provided in Section 3.2.3. The values of parameters for STDP models and synapse construction (summarised in Table 4.1) have been selected to have similar scale as published references (Gerstner & Kistler, 2002; Masquelier et al., 2008). Synapse weight results are collected from STDP model updates and con-

structed synapses for increasing iteration numbers, $M$. Results are truncated to the early stages of synapse weight convergence in additive STDP models and associated construction processes. Multiplicative STDP experiences diminishing updates; nevertheless, the same range of iteration numbers is recorded as for additive STDP.

**Table 4.1:** Parameters for STDP and synapse construction.

| | |
|---|---|
| $A_+$, $A_-$ | 0.015 |
| $\tau_+$, $\tau_-$ | 20 ms |
| $w_{\min}$ | 0 |
| $w_{\max}$ | 1 |
| $w_{i,j}[0]$ | 0.5 |

The results of weights from synapse construction and from iterative STDP updates are first compared directly. The resulting weights can be visualised individually in the case of ideal activity. A quantification of the difference in constructed synapse weights and an equivalent number of STDP iterations for the same relative spike timing can be presented as the residual error,

$$e_{i,j}[m] = w^{\mathrm{c}}_{i,j}[m] - w_{i,j}[m], \tag{4.1}$$

where synapse weights calculated for construction and produced from iterative STDP updates are $w^{\mathrm{c}}_{i,j}[m]$ and $w_{i,j}[m]$, respectively. The number of iterations of STDP is denoted $m = 1, 2, 3, \ldots, M$. Note that Chapter 3 describes constructed synapse weights as estimates and these calculated weights may also be denoted by $\hat{w}_{i,j}[m]$.

The synapse weights that result from stochastic neural activity are better represented as distributions. Histograms are used to represent weight distributions and give a quantitative outcome of the simulations; however, the comparison of resulting distributions is principally qualitative. The distributions of synapse weights resulting from input activity drawn from distributions with the same parameters are compared. Synapse weight distributions produced from neural activity drawn from Gaussian distributions are grouped and compared for equal mean and standard deviation values. The distributions of synapse weights produced from presynaptic activity modelled as Poisson processes may be compared for equivalent spike-rate values.

The parameter values used to generate stochastic presynaptic neural activity are presented in Tables 4.2 and 4.3. Numerical calculations are performed in MATLAB® R2016b and R2017a. The numerical evaluations make use of the `randn()` function for randomly generating numbers from a Gaussian distribution and the `rand()` function to randomly generate numbers from a uniform distribution.

The results of the numerical investigations are divided into sections for ideal activity, activity drawn from Gaussian distributions, and activity generated as Poisson processes. The sections first describe the properties and methods of generating the activity then the resulting synapse weights. The results for each model of neural activity and method of synapse weight generation are compared and discussed progressively

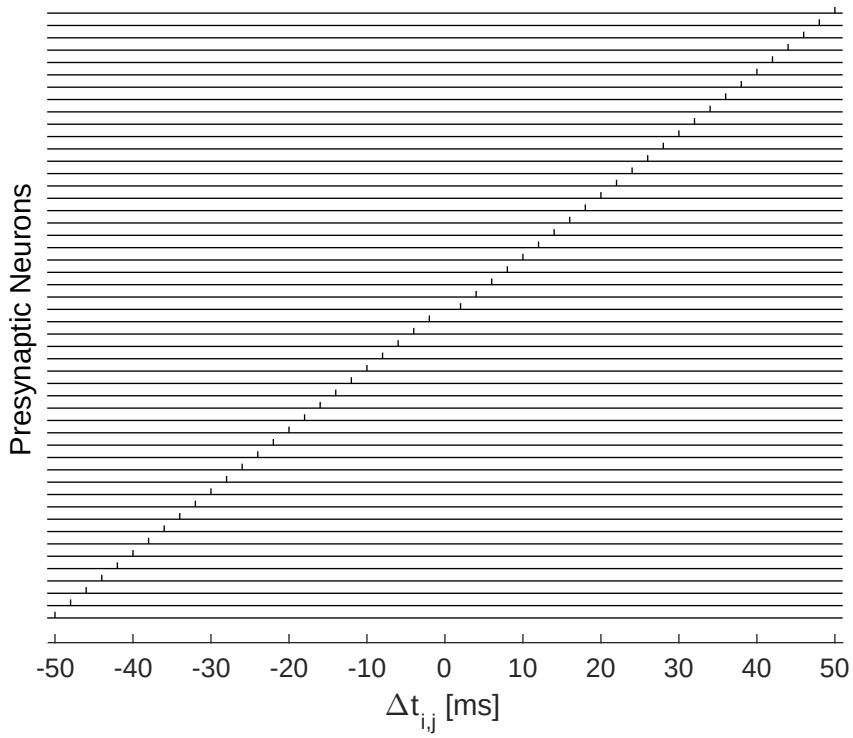


**Figure 4.2:** A graphical representation of presynaptic neuron spike times for ideal spike pairs. The spike times of presynaptic neurons are denoted by vertical marks on an associated horizontal line. In ideal activity, presynaptic neurons repeat the same unique relative spike time in each iteration. Presynaptic neurons are ordered in ascending spike times relative to a postsynaptic neuron spike, with spike times increasing in 2 ms increments from $-50$ ms to 50 ms. At the postsynaptic spike time and STDP curve discontinuity, $\Delta t_{i,j} \neq 0$, there is no presynaptic neuron spike.

with the presentation of results. A high-level discussion of the findings is presented at the end of the chapter.

## 4.3 Idealised Spike Times

The first collection of numerical results are performed for the assumed ideal conditions to confirm the correctness of developed equations. The calculation of synapse weights assume that the relative spike timings prior to construction are the same for all past STDP updates of significance. As such, the presynaptic and postsynaptic activity does not require the simulation of spiking neuron models, only the specification of spike timing for neurons.

Unique spike pair or spike triplet timings are generated for each synapse and repeated up to $M = 50$ times. With the parameters selected for additive STDP (Table 4.1), this number of STDP updates causes synapse weights to begin to saturate at

the maximum and minimum limits. The additive and multiplicative STDP models are evaluated with no other activity causing plasticity and no interaction between repetitions of the spike pairs and triplets. Synapse weights are also calculated for construction for the equivalent range of STDP iterations.

Results have been grouped in terms of presynaptic activity. Numerical evaluations of STDP for ideal activity are presented first for single pairwise interactions: pre-post spiking and post-pre spiking. The results from spike triplets, pre-post-pre spike combinations, are then presented. The cases without any eligible presynaptic spikes are degenerate (produce no change in synapse weights) and have been excluded for brevity.

### 4.3.1 Spike Pairs

Presynaptic neurons have relative spike times ascending from a minimum of $-50\,\text{ms}$ in increments of $2\,\text{ms}$ up to a maximum of $50\,\text{ms}$, with the exclusion of the $\Delta t_{i,j} = 0\,\text{ms}$ (no presynaptic neuron spikes at the postsynaptic spike time). Relative spike times less than $-50\,\text{ms}$ or greater than $50\,\text{ms}$ have a diminishing effect on the synapse weight. Figure 4.2 presents the relative spike timings of pairs used in numerical evaluations.

The results of additive STDP updates and synapse weight construction are presented in Figure 4.3, displaying the changes in synapse weight in 5 iteration increments. Synapses experiencing rapid plasticity are seen to saturate at the maximum or minimum weight thresholds. Under additive STDP, a repeated relative spike timing produces the same synapse weight update in each iteration and can be seen to change linearly for increasing iterations.

The results of repeating additive STDP updates (Equations 3.3 and 3.4) and synapse weight calculations for construction (Equations 3.9 and 3.10) with equivalent numbers of iterations are equal up to numerical precision (Figure 4.3). The residual errors of synapse weight estimates for plotted values fall within the range $e_{i,j}[m] \in [-2.776 \times 10^{-15}, 2.655 \times 10^{-15}]$. This error is due to the limits of numerical precision of the `double` floating-point data type used. Additive STDP residual error is zero once the weights saturate at the minimum or maximum value and are assigned an exact value.

The results of repeating multiplicative STDP updates (Equations 3.5 and 3.6) and synapse weight calculations for construction (Equations 3.16 and 3.17) with equivalent numbers of iterations are equal to numerical precision (Figure 4.4). The residual errors for the constructive weight calculations have recorded values in the range $e_{i,j}[m] \in [-1.277 \times 10^{-15}, 1.332 \times 10^{-15}]$. The change in synapse weights under iterated multiplicative STDP updates show similar trends to additive STDP (Figure 4.3). The multiplicative weight-dependent term, however, causes the rate of change to decelerate as synapse weights approaches the maximum and minimum weight limits.

The residual synapse weight errors for the continuous approximations of $m$ multiplicative STDP (Figure 4.5) fall within the range $e_{i,j}[m] \in [-1.176 \times 10^{-3}, 1.176 \times 10^{-3}]$. The equations for the continuous iteration approximation of multiplicative STDP updates (see Section 3.2.3.3) have a continuously decreasing rate-of-change. The continuous decrease in the rate-of-change results in an under-prediction of the final weight from repeating discrete STDP updates by a small margin.

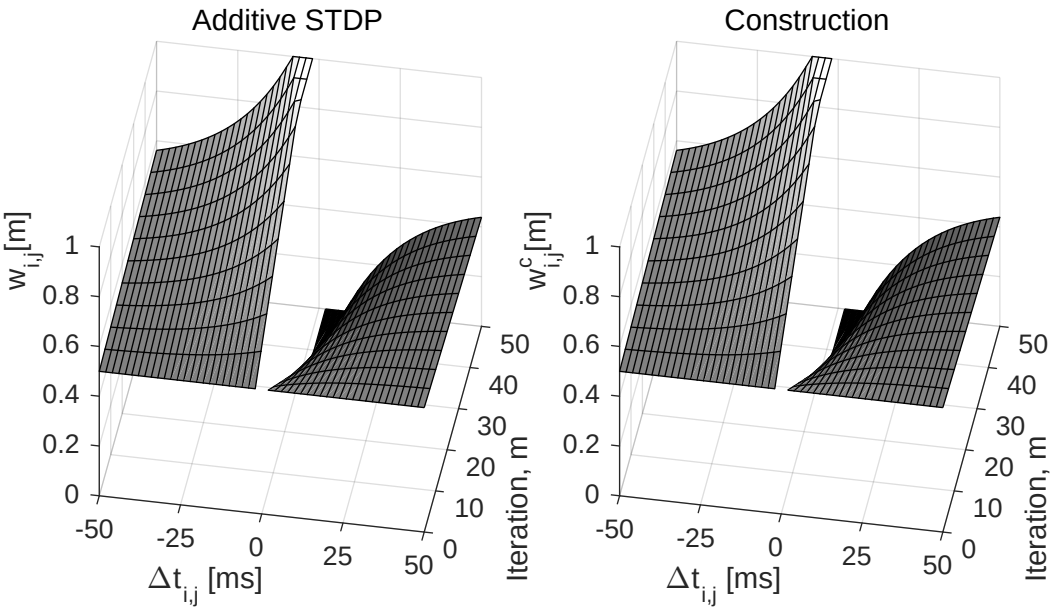


**Figure 4.3:** Left: Synapse weights resulting from additive STDP for $m$ iterations of spike pairs under ideal conditions. Right: Synapse weights resulting from constructive calculations of additive STDP for $m$ iterations of a spike pair.

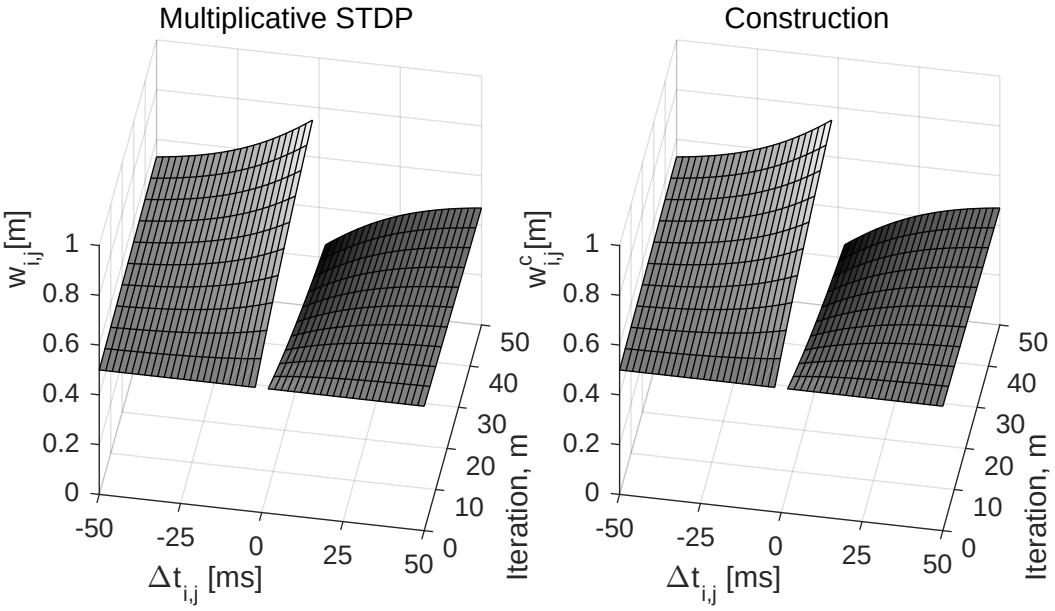


**Figure 4.4:** Left: Synapse weights resulting from multiplicative STDP for $m$ iterations of spike pairs under ideal conditions. Right: Synapse weights resulting from constructive calculations of multiplicative STDP for $m$ iterations of a spike pair.

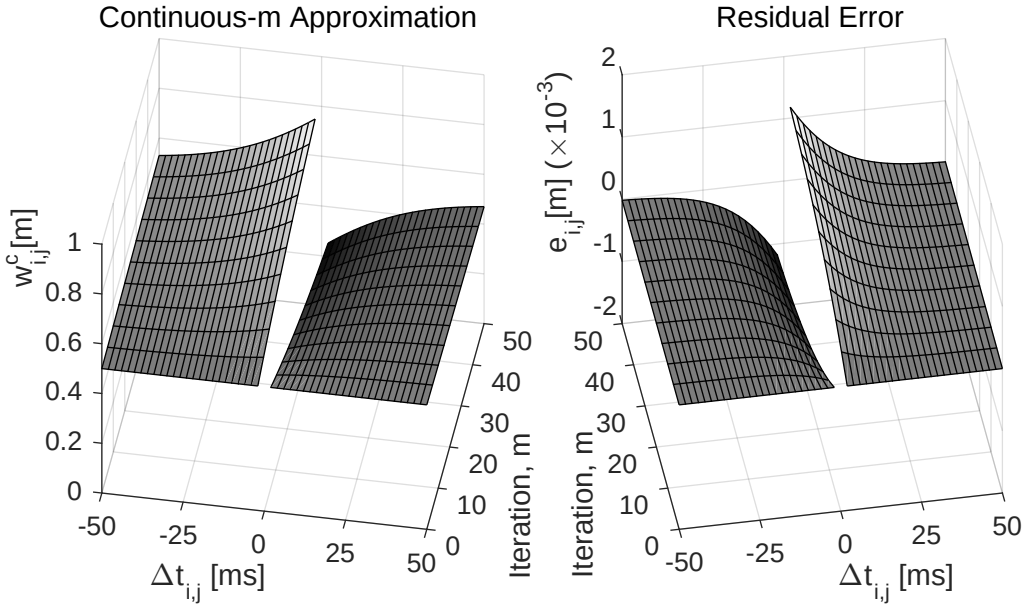


**Figure 4.5:** Left: Synapse weights resulting from constructive calculations using a continuous approximation of multiplicative STDP for $m$ iterations of a spike pair. Right: The residual error of the continuous approximations and equivalent iterative calculations of multiplicative STDP.

### 4.3.2 Spike Triplets

The numerical evaluations of STDP models and estimation methods in response to spike triplets have been performed for a grid of presynaptic neuron spike times before and after the postsynaptic neuron spike. The relative spike time of the first spike of the presynaptic neuron, $\Delta t_{i,j}^{(1)}$, is a value from $-50\,\text{ms}$ to $-2\,\text{ms}$ in increments of $2\,\text{ms}$. The relative spike time of the second spike of the presynaptic neuron, $\Delta t_{i,j}^{(2)}$, is a value from $2\,\text{ms}$ to $50\,\text{ms}$ in increments of $2\,\text{ms}$. Each possible combination of presynaptic spike times is assigned to a unique presynaptic neuron. Figure 4.6 presents a cross-section of the spike triplets used with presynaptic spikes having a constant delay, $t_j^{(2)} - t_j^{(1)} = 60\,\text{ms}$, but with different times relative to the postsynaptic spike.

Under additive STDP, a pre-post-pre spike triplet results in a net change in the synapse weight in the direction of the spike pair update with greater absolute magnitude. This is observed in the final result of $M = 50$ iterative updates and equivalent one-shot calculations of synapse weight for construction (Figure 4.7). The residual errors of the additive STDP estimates of plotted values fall in the range $e_{i,j}[M] \in [-5.329 \times 10^{-15}, 5.329 \times 10^{-15}]$. This residual error is the result of limits in numerical precision. When synapse weights saturate at the maximum or minimum value the equal assignment of weights reduces the error to zero.

Multiplicative STDP shows a similar trend to additive STDP; however, the multiplicative weight term causes the update values to decrease as the synapse weight ap-

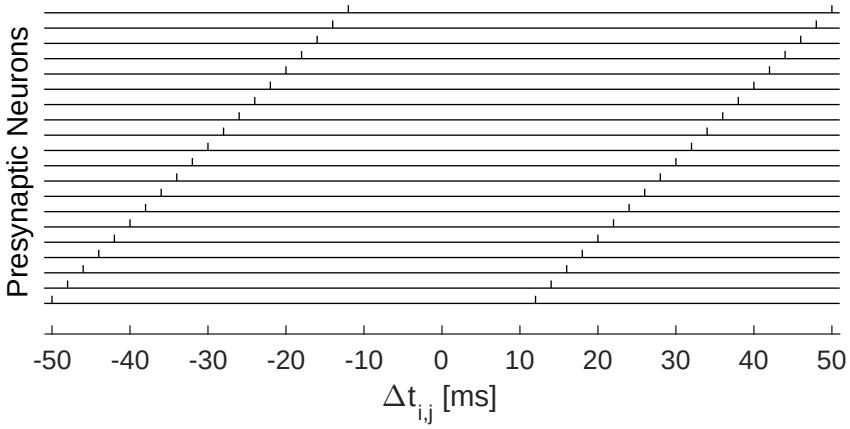


**Figure 4.6:** A graphical representation of presynaptic neurons spiking in pre-post-pre triplets. The presynaptic spike times presented all have a 60 ms delay between the first and second spike but different times relative to the postsynaptic spike.

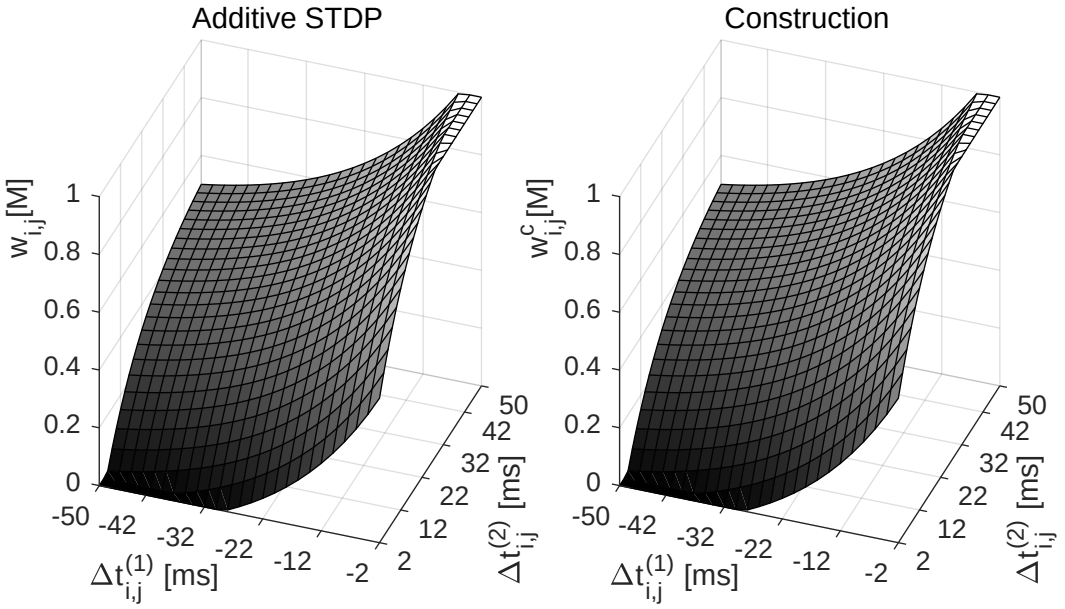


**Figure 4.7:** Left: Synapse weights resulting from additive STDP for $M = 50$ iterations of spike triplets under ideal conditions. Right: Synapse weights resulting from constructive calculations of additive STDP for $M = 50$ iterations of a spike triplet.

proaches the maximum and minimum weight limits (Figure 4.8). The synapse weight estimates for $M$ discrete iterations of multiplicative STDP are indistinguishable from the weights produced from performing $M$ multiplicative STDP updates iteratively. The residual errors of the discrete solution for $M = 50$ iterations of multiplicative STDP have values in the range $e_{i,j}[M] \in [-3.997 \times 10^{-15}, 3.553 \times 10^{-15}]$. These errors are due to limits in numerical precision.

The synapse weight calculations using a continuous variable for iterations is not visibly distinguishable from the weights produced through multiplicative STDP updates (Figure 4.9); nevertheless, there is a greater residual error in synapse weights calculated for construction using this method. The residual errors of the continuous-$m$ approximation of multiplicative STDP fall within the range $e_{i,j}[M] \in [-8.801 \times 10^{-4}, 2.545 \times 10^{-3}]$. The residual error shows the highest positive error when both presynaptic spikes are nearest the postsynaptic spike time.

A significant portion of this error can be attributed to the asymptote value calculated for spike triplet case (Equation 3.26) not taking into account the order of presynaptic spikes. The first spike pair (pre-post) first increases the weight and changes the multiplicative factor for the second spike pair update (post-pre) that depresses the weight. When the spike pairs have opposite relative spike times and the synapse weight is at the range midpoint, the second update will have a larger multiplicative factor, nudging the asymptote towards depression.

The accuracy of constructed synapse weights could be improved by using the asymptote value found for $M$ discrete iterations of multiplicative STDP (Equation 3.19). The difference in calculated asymptote values for the spike triplet with presynaptic spikes at $-2$ ms and 2 ms gives $A_{STa} - A_{ST} = 3.416 \times 10^{-3}$. The discrepancy in asymptote values is greater than the residual error for that spike triplet ($3.416 \times 10^{-3} > 2.545 \times 10^{-3}$). The remaining error may be attributed to the continuous approximation of the rate-of-change of discrete updates, as discussed in relation to the error in spike pair results.

The synapse weight calculations for construction have been found correct up to margins of numerical precision. This confirms the correctness of the equations developed in Chapter 3 for the idealised conditions assumed. The next sections relax the assumption that the same relative spike timings have repeated, investigating the weights resulting from spike times generated using Gaussian and Poisson process models.

## 4.4 Gaussian-Distributed Spike Times

The standard assumption in neuroscience is that neuronal activity is highly noisy (Dayan & Abbott, 2001; Gerstner & Kistler, 2002); however, there have been observations of the relative spike times of biological neurons being repeatable to within 1 ms (Masquelier, 2013). Based on the observations of precisely repeatable relative spike times, the equations developed to calculate new weights for constructed synapses (Section 3.2.3) assume no variation in the relative timing of spikes. Nevertheless, some variability in relative spike times is highly probable in biological systems and noise is a standard feature in simulations. Relative spike times drawn from a Gaussian or normal

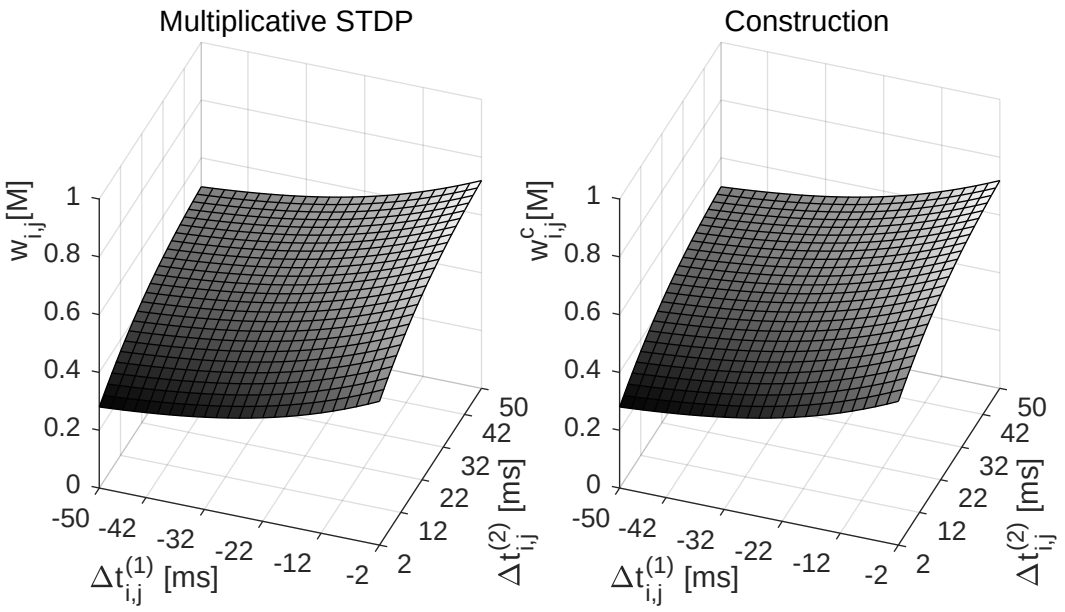


**Figure 4.8:** Left: Synapse weights resulting from multiplicative STDP for $M = 50$ iterations of spike triplets under ideal conditions. Right: Synapse weights resulting from constructive calculations of multiplicative STDP for $M = 50$ iterations of a spike triplet.

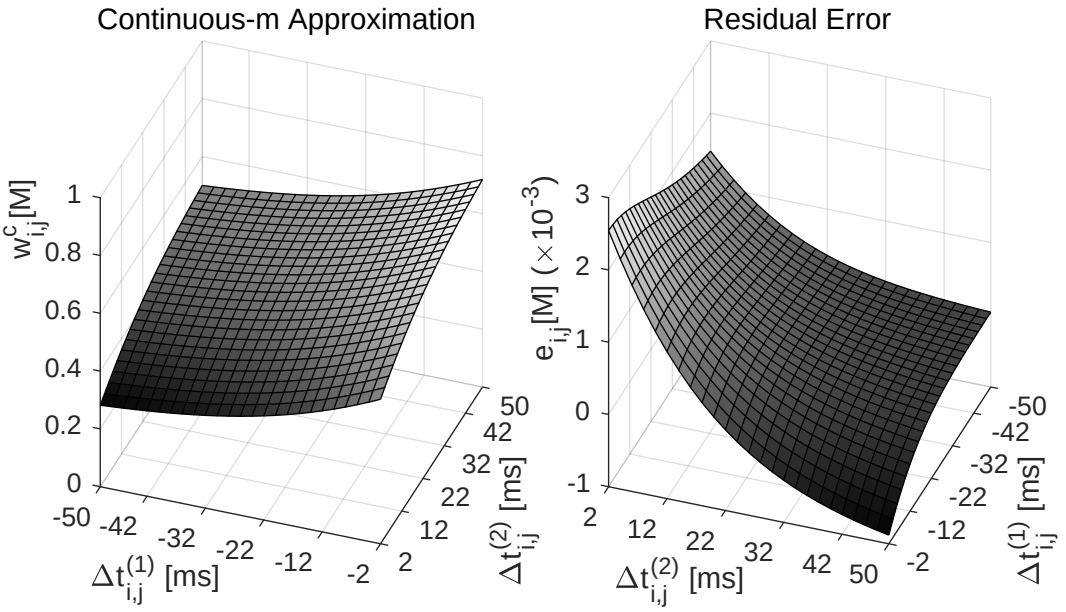


**Figure 4.9:** Left: Synapse weights resulting from constructive calculations using a continuous approximation of multiplicative STDP for $M = 50$ iterations of a spike triplet. Right: The residual error of the continuous approximations and $M = 50$ iterative calculations of multiplicative STDP for the same spike triplets.

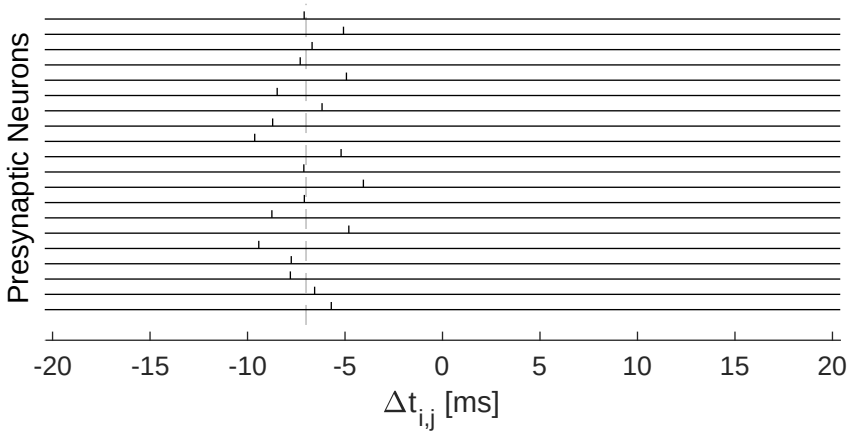


**Figure 4.10:** A graphical representation of relative spike times of presynaptic neurons drawn from a Gaussian probability distribution. Presynaptic spikes (represented as vertical marks) are drawn from a distribution with a mean $\mu = -7\,\text{ms}$ and a standard deviation $\sigma = 2\,\text{ms}$.

probability distribution (Equation 4.2) varying around a fixed mean are used here as a model of repeatable but noisy neural activity (for example, see Figure 4.10). This section presents synapse weight distributions resulting from STDP updates for iterated Gaussian-distributed relative spike times and from synapses constructed estimating an equal number of STDP iterations. Equations for STDP and synapse construction from previous sections are reused.

Presynaptic spike times $t_j$, $j \in J$, are generated using the standard MATLAB function `randn()` to produce spike times drawn from a normal distribution,

$$N(t, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big( - \frac{(t-\mu)^2}{2\sigma^2} \Big), \tag{4.2}$$

with standard deviation $\sigma$ (variance $\sigma^2$) around the mean relative spike time $\mu$. Gaussian distributions can be defined by their mean and standard deviation; therefore, these are treated as the variables of interest for the examination of weight distributions developed through iterative STDP updates and calculations for constructed synapses.

Numerical evaluations are performed to collect synapse weight distributions developed from Gaussian presynaptic spike times with standard deviations $\sigma = 1\,\text{ms}$, $2\,\text{ms}$ and $5\,\text{ms}$ and mean relative spike timing from $\mu = -40\,\text{ms}$ to $40\,\text{ms}$ in $1\,\text{ms}$ increments. Numerical evaluations of Gaussian-distributed activity are performed for $|J| = 10^6$ presynaptic neurons. Therefore, $10^6$ spike times are generated for each STDP update iteration (up to $M = 40$). An additional set of spike times is generated once for the calculation of synapse weights for construction.

Numerical evaluations are performed for each combination of mean and standard deviation parameter values: 3 standard deviation values and 81 mean spike times gives 243 combinations of parameter values. The 243 combinations of parameter values are

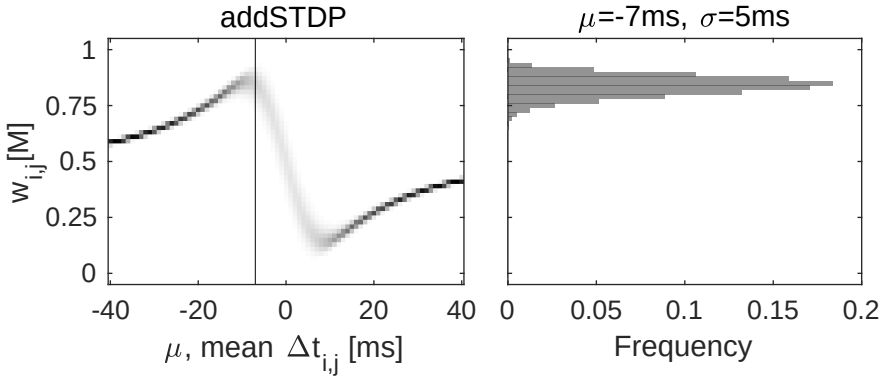


**Figure 4.11:** Example of a grey-scale image representation of synapse weight distributions resulting from Gaussian-distributed presynaptic activity. Left: Grey-scale image representation of synapse weight distributions resulting from additive STDP (addSTDP). Each column of pixels corresponds to a histogram of the synapse weights for that mean relative spike time. The darker pixels in the image correspond to the histogram bins with higher numbers of samples. Right: The histogram of synapse weight values for mean relative spike time $-7$ ms (indicated by the vertical line in the image on the left), a standard deviation of 5 ms and $M = 40$ iterations of additive STDP.

**Table 4.2:** Gaussian parameters for presynaptic activity.

| | Gaussian |
|---|---|
| $\|J\|$ | $10^6$ |
| $M$ | 40 |
| $\mu$ | $-40$ ms to 40 ms in 1 ms increments |
| $\sigma$ | 1 ms, 2 ms and 5 ms |

tested for additive STDP updates (Equations 3.3 and 3.4), additive STDP construction (Equations 3.9 and 3.10), multiplicative STDP updates (Equations 3.5 and 3.6) and multiplicative STDP construction (Equations 3.16 and 3.17), giving a total $972 \times 10^6$ generated synapse weights. The ranges of parameter values have been summarised in Table 4.2. The parameters for initial weights and STDP are unchanged from the idealised activity calculations (Table 4.1). Note that evaluations of continuous-iteration approximations of $M$ multiplicative STDP updates have been omitted for brevity.

A distribution of synapse weights is produced for each combination of parameters for Gaussian-distributed spike times, STDP model and constructive calculations. Frequencies of synapse weight values for each distribution were found by dividing the weight range, $[0, 1]$, into 50 bins with width 0.02; synapse weights at the limits were included in the end bins. The bin counts for a distribution are presented as a column of pixels in a grey-scale image (Figure 4.11); darker pixels in a column indicate a higher relative frequency of synapses in that bin. A grey-scale image grouping the synapse weight distributions has been generated for the results of STDP updates and calculations for synapse construction for each standard deviation value for spike times.

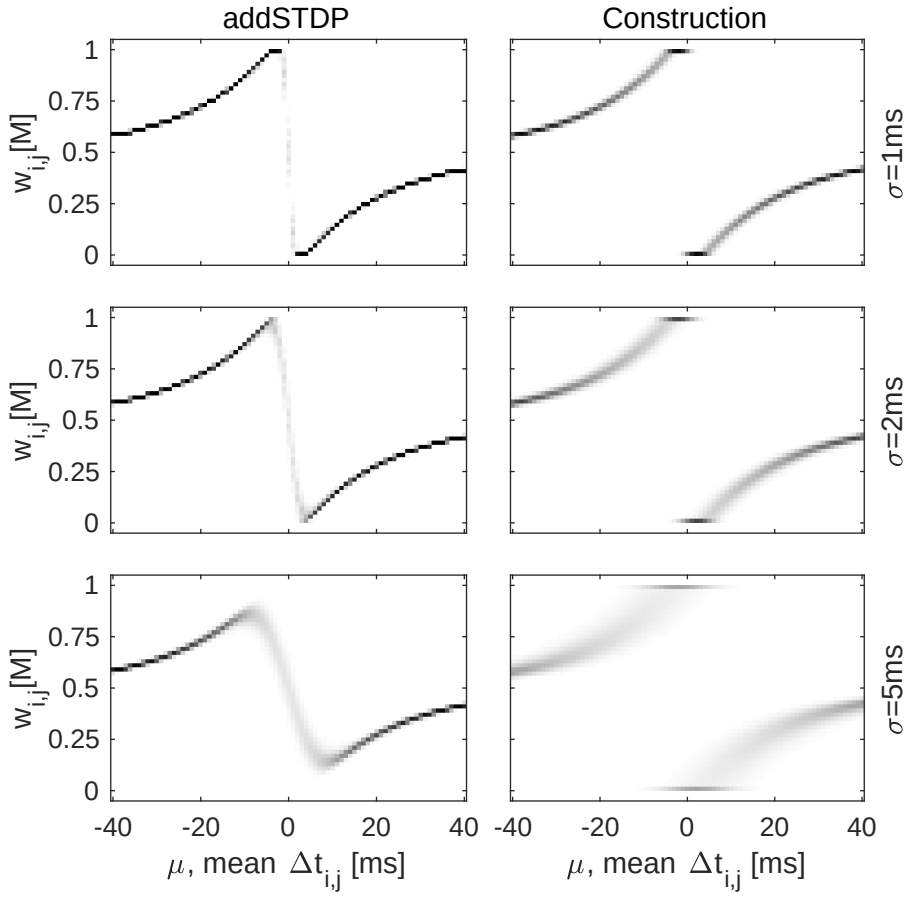


**Figure 4.12:** Weight distributions from additive STDP (left) and constructed synapses (right) for an equivalent number of iterations ($M = 40$) of Gaussian-distributed presynaptic spike times. The synapse weight range, $[0, 1]$, is divided evenly into bins with width 0.02. (See Figure 4.11 for an explanation of grey-scale image representations of synapse weight distributions.)

The distribution of weights developed through iterative calculations of additive STDP updates (Figure 4.12) and multiplicative STDP updates (Figure 4.14) fall into curves with similar shapes to those produced from idealised spike times (Figures 4.3 and 4.4). The results of Gaussian-distributed activity differ most from the ideal around the discontinuity in the STDP curve at $\Delta t_{i,j} = 0$. The synapse weight distributions produced from iterative updates transition from above the initial weight to below initial weight as the mean spike time increases over $\Delta t_{i,j} = 0$. As the variance of spike timing increases, this transition region spreads further from the STDP curve discontinuity. The synapse weight distributions at each mean spike time (represented by pixel columns) also spread more widely for increasing spike time variance. The increase in the spike time variance results in spikes being more likely to occur in both the depressing and potentiating regions of the STDP curve for a wider range of mean spike times.

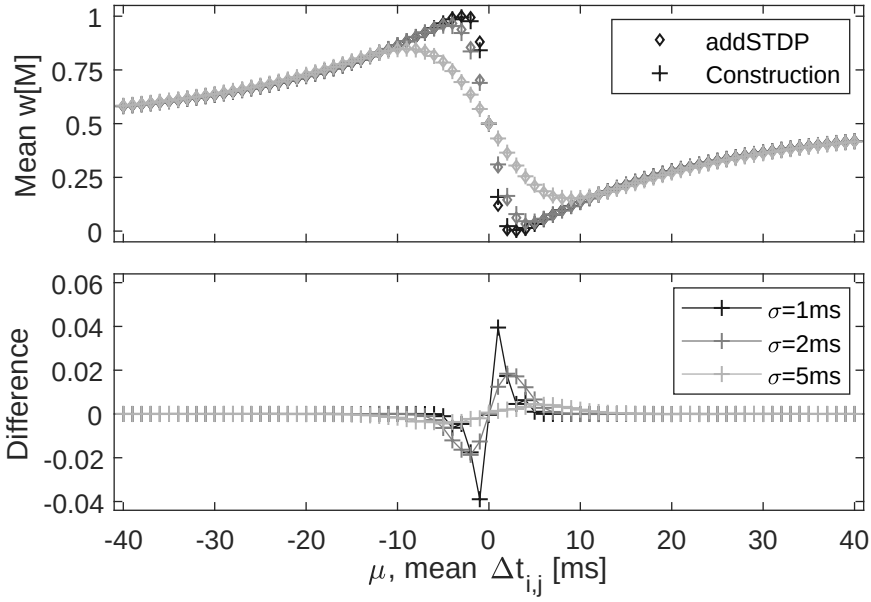


**Figure 4.13:** Top: Mean synapse weights from additive STDP updates and synapse weight construction for Gaussian-distributed presynaptic spike times. Marker darkness indicates the spike time standard deviation value: from darkest to lightest, $\sigma = 1\,\text{ms}$, $2\,\text{ms}$ and $5\,\text{ms}$. Bottom: The difference in mean weight values (mean constructed synapse weight minus mean weight after final STDP update). Markers indicate calculated values; lines are only included for visibility.

The constructed synapse weights for additive STDP (Figure 4.12) and multiplicative STDP (Figure 4.14) produce bimodal distributions around the relative spike time of zero, $\Delta t_{i,j} = 0$. The distributions of relative spike times with a mean near $\Delta t_{i,j} = 0$ can have a substantial frequency of occurrences in both the positive and negative regions of the STDP update curve (Figure 3.4). Weight calculations for synapse construction based on these spike time distributions result in a bimodal distribution of synapse weights. The magnitude of the STDP update value is also largest around the relative spike time of zero, producing synapse weight estimates with the largest possible distance between the positive and negative modes. The additive STDP estimates are limited to the maximum and minimum weight values, clipping the distributions of synapse weights. The constructed weights from estimations of additive STDP are calculated further from the initial weight than multiplicative STDP estimates for the same variance due to the attenuating effect of the multiplicative factor.

These outcomes indicate that the parameter calculation methods developed from STDP models have a heightened sensitivity to spike time noise in time-coded neural activity. The bimodal distributions of constructed synapse weights around the STDP curve discontinuity is significantly different from unimodal distributions produced from iterative updates. The mean of the synapse weights produced through STDP updates and construction at each mean spike time, however, are within a small margin (Fig-

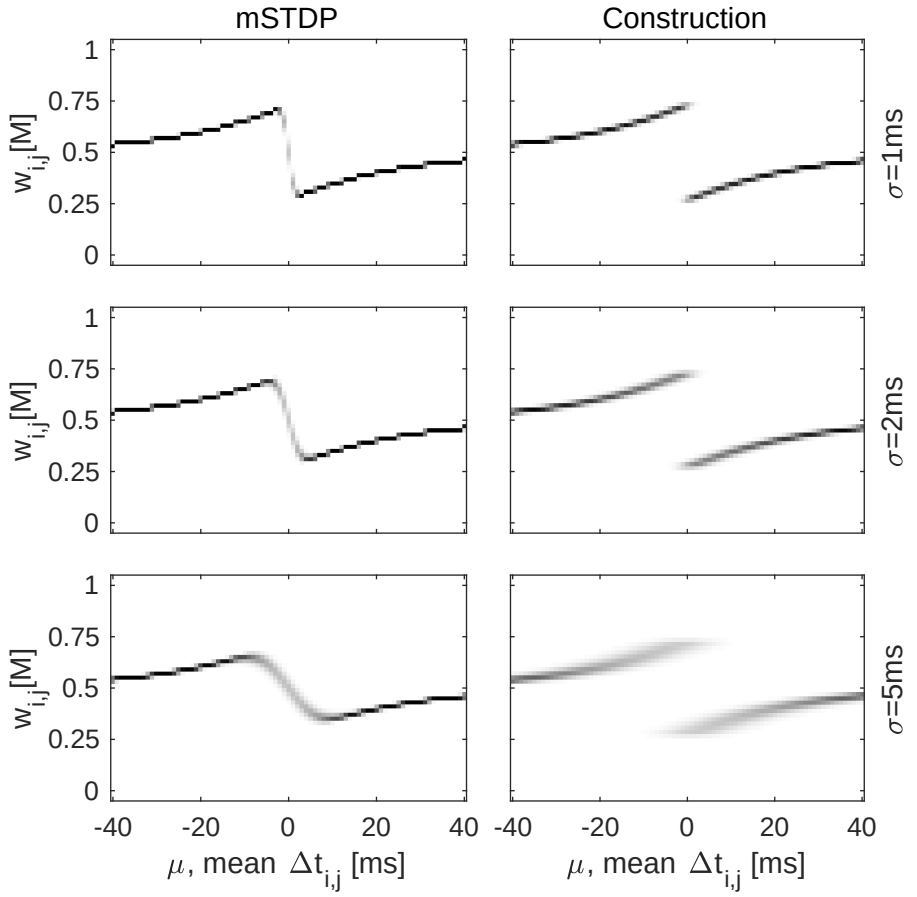


**Figure 4.14:** Weight distributions from multiplicative STDP (left) and constructed synapses (right) for an equivalent number of iterations ($M = 40$) of Gaussian-distributed presynaptic spike times. The synapse weight range, $[0, 1]$, is divided evenly into bins with width 0.02.

ures 4.13 and 4.15). Although the difference in individual synapses may be significant, if the average or total weight of input synapses has a similar value the behaviour of the resulting simulated neuron may be similar. The behaviour of simulated neurons with updated and constructed weights will be investigated in Chapter 5.

## 4.5 Poisson Process Presynaptic Activity

Poisson processes can be used to model rate-coded neural activity or random spontaneous activity in neurons. The nearest-neighbour models of STDP implemented only update a synapse for the last presynaptic spike before and the first presynaptic spike after the postsynaptic spike; therefore, spike times are generated for one presynaptic spike before and one presynaptic spike after the postsynaptic spike time for each it-

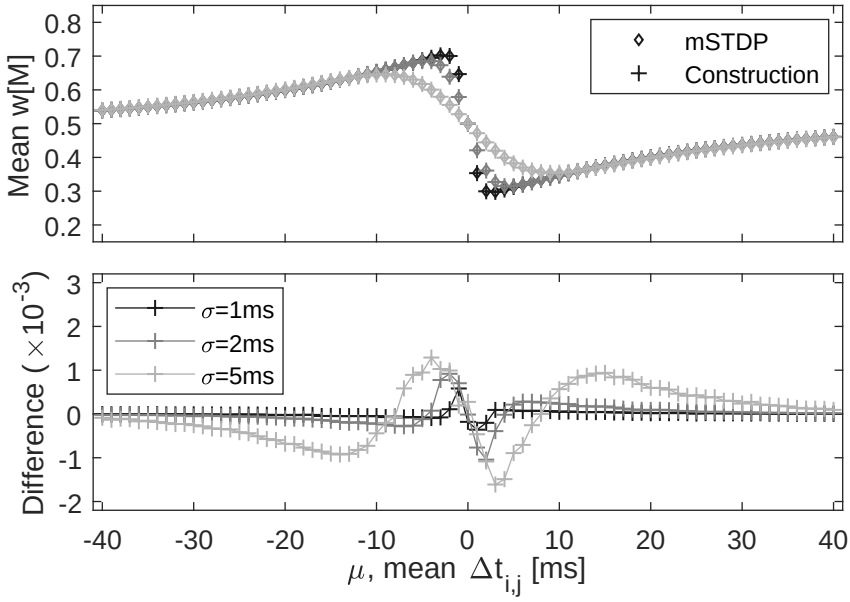


**Figure 4.15:** Top: Mean synapse weights from multiplicative STDP and construction for Gaussian-distributed presynaptic spike times. Marker darkness indicates the spike time standard deviation value: from darkest to lightest, $\sigma = 1\,\text{ms}$, $2\,\text{ms}$ and $5\,\text{ms}$. Bottom: The difference in mean weight values (mean constructed synapse weight minus mean weight after final update). Markers indicate calculated values; lines are only included for visibility.

eration of STDP updates. It is assumed that the time between postsynaptic spikes is long enough that there is no interference between these events. See Figure 4.16 for an example of spike triplets generated as Poisson activity for the numerical investigations presented.

The presynaptic spike times generated as Poisson processes (from exponential probability distributions) are controlled by the rate variable, $\lambda$. Recall that the models of STDP being investigated do not consider the spike rate when calculating the resulting plasticity. Spike times are generated for a large number of presynaptic neurons ($|J| = 10^6$) to improve the accuracy of the histogram in approximating the probability distribution of synapse weights. Construction is performed on distinct presynaptic spike times drawn from the same statistical distributions to minimise any correlation from sharing randomly generated values.

In a Poisson process, the probability of different times between spikes can be represented as an exponential distribution. Spike times may be generated from some manipulation of the exponential cumulative distribution function,

$$f(t,\lambda) = \begin{cases} 1 - \exp(-\lambda \cdot t) & \text{if } t \geq 0, \\ 0 & \text{if } t < 0, \end{cases} \tag{4.3}$$

where the calculated value $f(t,\lambda)$ is the expected percentage of samples with a value less than $t$ given an event rate $\lambda$. A sample time, $t$, can be generated by taking the

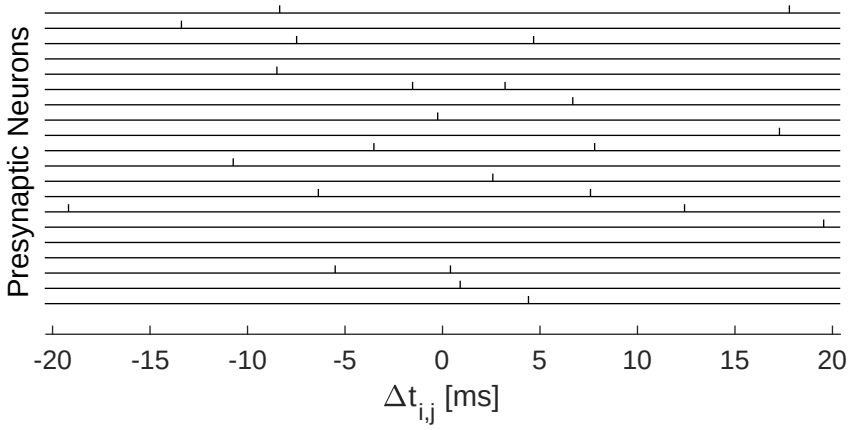


**Figure 4.16:** A graphical representation of presynaptic neuron spike times in pre-post-pre triplets generated using an inverse exponential distribution to approximate a Poisson process with a rate of 20 Hz. All presynaptic neurons have spikes before and after the post-synaptic neuron; however, some presynaptic spikes are outside the time window presented.

inverse of the exponential distribution function and supplying a value drawn from a uniform distribution, $U \in [0, 1]$, in place of the percentage $f(t, \lambda)$,

$$t = \frac{-\ln(U)}{\lambda}. \tag{4.4}$$

The value of $t$ generated can be treated as the time until the next spike; therefore, the next presynaptic spike time may be calculated by adding $t$ to the previous spike time. The only variable that can be controlled in the Poisson process is the spike rate, $\lambda$.

The time from the first presynaptic neuron spike to the postsynaptic neuron spike, $\Delta t_{i,j}^{(1)}$, and time from the postsynaptic neuron spike to the second presynaptic neuron spike, $\Delta t_{i,j}^{(2)}$, are calculated,

$$\Delta t_{i,j}^{(1)} = \frac{\ln(U)}{2 \cdot \lambda} \text{ and} \tag{4.5}$$

$$\Delta t_{i,j}^{(2)} = \frac{-\ln(U)}{2 \cdot \lambda}. \tag{4.6}$$

Note that the time to presynaptic spikes is calculated from the postsynaptic spike time rather than a presynaptic spike time. The absence of a presynaptic spike at $t = 0$ effectively doubles the average time between presynaptic spikes; therefore, to compensate, the rate variable $\lambda$ is multiplied by two.

Numerical evaluations have been performed to investigate the effect of increasing rate on synapse weights developed. For the purposes of this investigation, simulations have been conducted to find weight distributions developed from Poisson processes at 5 Hz, 10 Hz, 20 Hz and 100 Hz. Numerical evaluations of Poisson presynaptic activity are performed for $|J| = 10^6$ presynaptic neurons. Therefore, $2 \times 10^6$ spike times are

**Table 4.3:** Poisson process parameters for presynaptic activity.

| | Poisson |
|---|---|
| $\|J\|$ | $10^6$ |
| $M$ | 40 |
| $\lambda$ | 5 Hz, 10 Hz, 20 Hz and 100 Hz |

generated for each iteration (up to $M = 40$) and for STDP estimates. These numerical studies calculated $10^6$ synapse weights for each plasticity model (additive STDP and multiplicative STDP) and constructed specified iteration at each spike rate value ($\lambda =$ 5 Hz, 10 Hz, 20 Hz and 100 Hz). This gives a total of $16 \times 10^6$ generated synapse weights. The resulting synapse weights are presented as histograms (Figures 4.17 and 4.18).

Histograms of synapse weights produced from iterative updates of additive and multiplicative STDP show the weights cluster around the initial weight, $w[0] = 0.5$, for all spike rates simulated (Figures 4.17 and 4.18). Poisson activity results in spike times occurring in exponential distributions in the positive update and negative update regions. Over the course of multiple STDP updates the positive and negative changes largely cancel, resulting in weights most frequently falling within the centre histogram bin (centred on the initial weight). Asymmetry appears in the histogram results for multiplicative STDP due to negative updates (Equation 3.6) tending to have a larger multiplicative factor from the earlier positive update (Equation 3.5).

Low spike rates give a low likelihood of weight updates of significant magnitude; therefore, there is a low likelihood of large overall changes in the synapse weight. Here, an update is considered to have a significant magnitude if $\tau_+ < \Delta t_{i.j} < \tau_-$. As the spike rates increase a greater number of weight updates have a significant magnitude; therefore, the synapse weights have a higher likelihood of drifting away from the initial value. If the spike rate continues to increase, all weight updates are likely to have a high absolute magnitude and become more likely to cancel overall. This is observed in the synapse weight distributions that cease to widen (Figure 4.17) and start to cluster more tightly around the initial weight (Figure 4.18) from $\lambda = 20$ Hz to 100 Hz.

Constructed synapse weights can be seen to fall into substantially wider distributions (Figures 4.17 and 4.18). The assumption of multiple iterations in construction amplifies the update from the observed spike times; therefore, low spike rates can still result in wide distributions of synapse weights. In the case of additive STDP, the width of the distribution may be limited by hard weight limits (Figure 4.17). In the case of multiplicative STDP, the multiplicative factor keeps the synapse weights more tightly clustered around the initial weight (Figure 4.18). As the spike rate increases from 5 Hz to 20 Hz the likelihood of synapses remaining at approximately the initial weight decreases. However, as the spike rate continues to increase, the likelihood of positive and negative STDP updates for the same synapse approximately cancelling increases. At a spike rate of 100 Hz the distribution of constructed synapse weights has clustered more closely around the initial synapse weight.

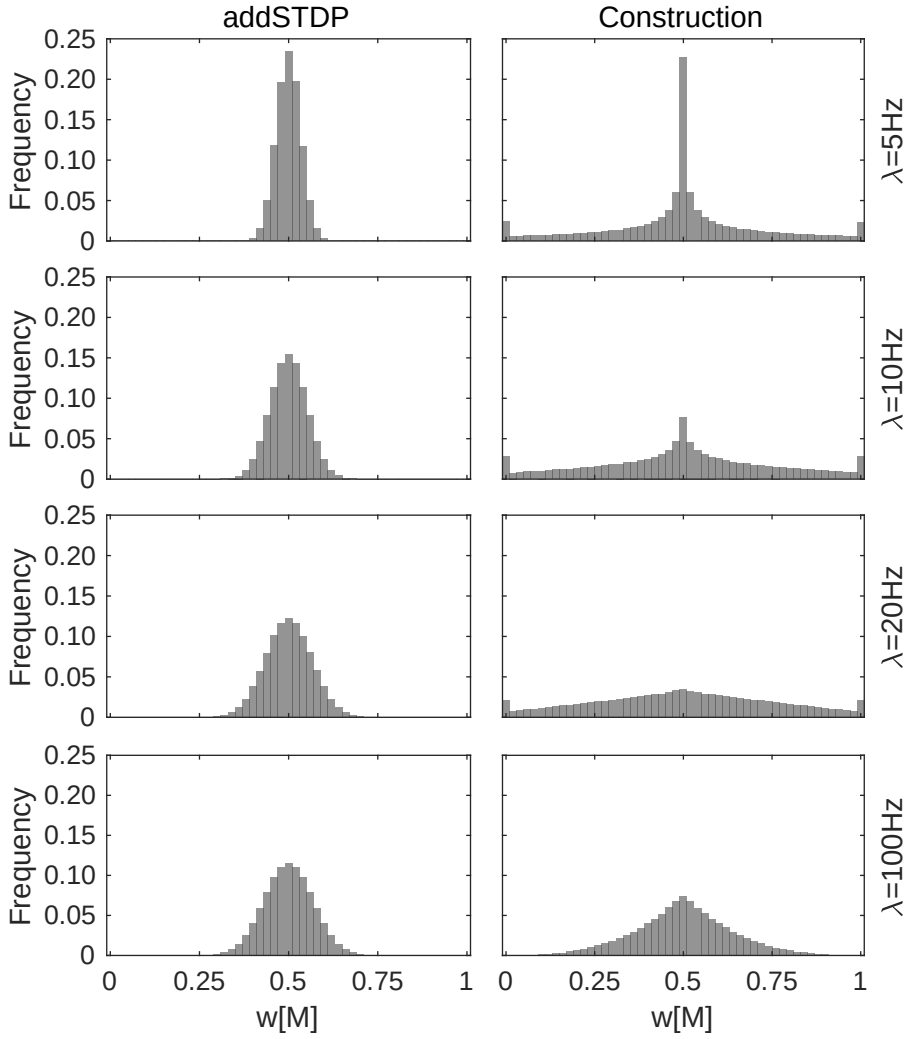


**Figure 4.17:** Weight distributions from additive STDP (left) and constructed synapses (right) for an equivalent number of iterations of Poisson presynaptic spike times. The histogram bin width is 0.02 with the middle of the centre bin on the initial weight, $w[0] = 0.5$.

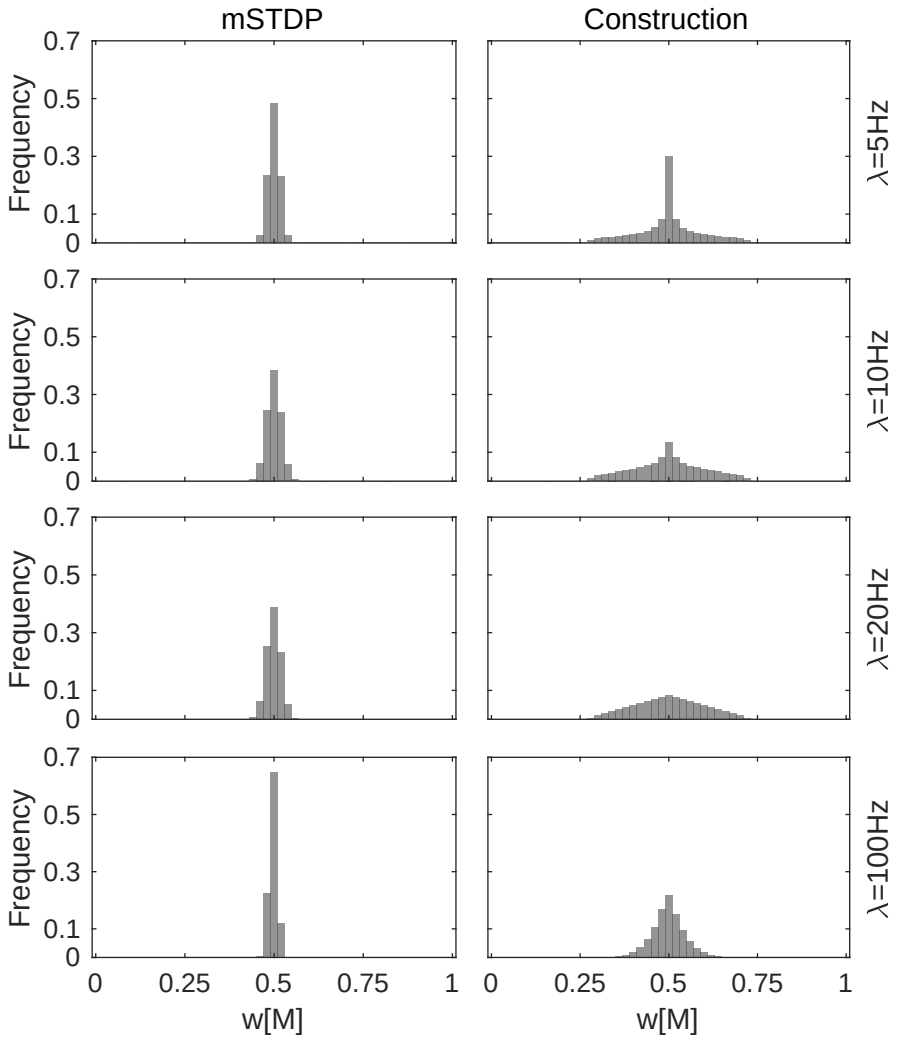


**Figure 4.18:** Weight distributions from multiplicative STDP (left) and constructed synapses (right) for an equivalent number of iterations of Poisson presynaptic spike times. The histogram bin width is 0.02 with the middle of the centre bin on the initial weight, $w[0] = 0.5$.

## 4.6 Discussion

The numerical calculations performed with ideal neural activity (Section 4.3) have produced equal synapse weights (up to the limits of numerical precision) through construction and STDP updates for equivalent iterations. This successful numerical validation confirms that the equations for calculating constructed synapse weight (Section 3.2.3) are correct for the assumed ideal conditions. Stochastic distributions of presynaptic neuron activity have been used to model inter-spike-train variability and random neural activity seen in biological neural networks. Stochastic presynaptic activity results in substantial differences in the distributions of synapse weights resulting from STDP updates and from constructive calculations. The iterative updating of synapses using additive STDP (Equations 3.3 and 3.4) and multiplicative STDP (Equations 3.5 and 3.6) models demonstrates a resistance to the stochastic noise models, whereas synapse weight calculations for construction assume the observed spike times repeat without noise and result in an amplification of any noise.

A simulation of a biological neural network may find the calculated synapse weights sufficiently biologically plausible for inclusion despite the initial distribution of weights not accounting for noise; this is likely to be largely dependent on the specific requirements of the model. This sensitivity to noise may be an issue if an application requires synapse weight distributions that closely match those that result from iterative STDP updates in the presence of stochastic activity. It may be possible to improve the calculations of synapse weights by incorporating models of noise or basing calculations on multiple observations of neuron activity. The development and study of alternative approaches to synapse weight calculation for construction is a significant topic of research in its own right. This thesis will restrict investigations to minor refinements of the parameter calculation methods presented.

Construction may provide significant benefits even if the initial result of construction does not achieve the desired behaviour or learning task. To reiterate earlier points, construction provides an approach to: 1) automatically select an appropriate number of neurons and synapses and 2) allow the neural network to dynamically accommodate changing conditions. The immediate improvement in the model performance from the addition of a neuron or synapses may be treated as a secondary goal. A constructed neuron that has poor initial performance may still produce an improvement after parameter adaptation (models of plasticity or gradient descent). This is studied in Chapter 6 (STDP Simulation with Neuron Construction).

Synapse weight distributions give an indication of the potential error in constructed estimates of past STDP; however, the performance of a constructed neuron in the network is more directly dependent on the neuron activity. When the results are averaged over a large numbers of synapses, the mean or total synaptic weight resulting from STDP updates and constructed estimates match closely. Small peaks in error are observed around the STDP curve discontinuity for Gaussian-distributed spike times (Figures 4.13 and 4.15). In applications where the spike patterns are distributed over a large number of synapses, the similarity in the average synapse weight may result in

similar levels of overall neuron activity.

The simulation of STDP in noisy conditions produced narrow distributions of synapse weights around an average value. If all synapse weights are similar, the neuron may not be selective in its response to any specific combination of input activity. The wider distributions and bimodal distributions produced through constructive synapse weight calculations, however, bias neurons to respond to specific presynaptic neurons with potentiated synapses. The construction of neurons with distinct activity characteristics in noisy conditions may be advantageous. The behaviour of constructed postsynaptic neurons will be investigated further in the remainder of this thesis.

The numerical evaluations performed in this chapter incorporated a known postsynaptic neuron spike time. In unsupervised learning, the postsynaptic spike time will not be provided. The performance evaluation processes to trigger construction may be designed to provide a prediction of the postsynaptic neuron spike time. The next chapter develops a method for evaluating performance and predicting postsynaptic neuron spike times through the simulation of a spiking neuron model. Construction of spiking neurons will be performed and the activity of constructed neurons will be compared with neurons that have had synapses updated through STDP for the same number of postsynaptic spikes.

# Chapter 5

# Spike Prediction with Proxy Neurons

This chapter introduces proxy neurons as a tool for predicting the spike times of surrounding postsynaptic neurons and triggering neuron construction. A proxy neuron can be simulated as a representative of a susbet of the surrounding neurons (Figure 5.1) to provide a prediction of the spike times of those neurons. The prediction of postsynaptic neuron spike times is required for the synapse weight calculation methods developed for neuron construction in Chapter 3. A constructive algorithm that incorporates a proxy neuron to trigger construction is presented and evaluated. The synapse weights of neurons constructed using algorithms based on the spikes of a proxy neuron are compared with synapse weights resulting from neurons simulated with STDP. The spike latencies of constructed neurons are found to closely match those of neurons with synapses updated with STDP for an equivalent number of postsynaptic spikes.

## 5.1 Postsynaptic Spike Predictions

The behaviour of biological neurons can be modelled and predicted through the simulation of spiking neuron models. Many postsynaptic neuron models calculate or predict spike times based on the activity of the presynaptic neurons connected to it (Gerstner & Kistler, 2002). A postsynaptic neuron may be tuned to detect a specific pattern of presynaptic neuron spike times and represent a prediction of the presence of that spike pattern. The initial capacity of a simulated neural network may be insufficient to detect the full number of spike patterns present in presynaptic activity. This thesis explores algorithms for increasing the network capacity through the construction of new postsynaptic neurons.

The processes for calculating synapse weights to a constructed postsynaptic neuron introduced in Chapter 3 require the spike time of the postsynaptic neurons before it is constructed. Despite not being constructed, spike times of a postsynaptic neuron may be predicted from the set of connected presynaptic neurons that are simulated or have spike times provided. This prediction of a postsynaptic spike time may be desired to

coincide with a pattern of spikes from presynaptic neurons. Therefore, the prediction of a postsynaptic spike time for the constructive algorithm may be considered synonymous with the prediction of a presynaptic spike pattern.

A neural network model may provide the desired times of postsynaptic neuron spikes or the times of presynaptic spike patterns. Neuron construction may aim to reproduce desired postsynaptic spike times or aim to detect presynaptic spike patterns. This is a type of supervised learning. The performance of a constructed neuron may be based on its ability to reproduce the desired relative spike time in response to an associated presynaptic spike pattern.

In the event that desired postsynaptic neuron spike times or presynaptic spike pattern times are not provided, unsupervised methods of prediction are required. The presynaptic neurons associated with a spike pattern may have a detectable increase in their activity during that pattern. Unsupervised methods for postsynaptic spike time prediction and presynaptic spike pattern detection may focus on detecting elevations in overall presynaptic activity. A preliminary approach was to place a threshold on the number of presynaptic spikes in a given time window (Lightheart et al., 2013). When the threshold was exceeded, a postsynaptic spike was predicted and construction was performed.

Given the interpretation of neuron construction as a transfer of a neuron from the set of surrounding neurons to the set of simulated neurons (Chapter 3), predictions of postsynaptic spike times are predictions of activity in this surrounding neuron set. The spike times of surrounding postsynaptic neurons could be calculated if those neurons were simulated. A prediction of a spike in a set of neurons may be provided by the simulation of an approximation of the set of neurons. This chapter introduces the approach of predicting spike times in a set of neurons with the simulation of a single spiking neuron as a proxy.

## 5.2 Proxy Neurons

Proxy neurons may be designed to represent hypothetical surrounding neurons or to represent surrounding neurons in memory (Figure 5.1). In this thesis, proxy neurons are simulated to predict if any neuron in a set spikes. A brief discussion of proxy neurons that predict activity in sets of neurons in memory is provided in Appendix A; otherwise, this is not a topic of investigation in this thesis. This section focuses on the design of proxy neurons that approximate hypothetical neurons.

The hypothetical surrounding neurons in a simulation of a biological neural system should be assumed to have the characteristics of the types of neurons that have been experimentally observed in that neural system. Simulations in computational neuroscience may investigate mechanisms of behaviour and adaptation in abstract network models with simplified characteristics. The design of proxy neurons to represent different classes of neurons in specific neural systems is a topic for future research. The development of proxy neurons in this thesis focus on the application in computational studies with homogeneous neuron models.

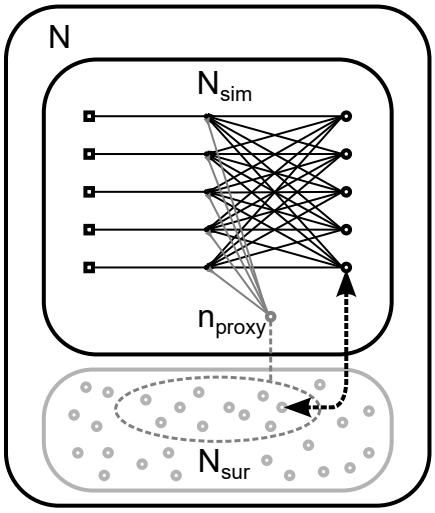


**Figure 5.1:** A proxy neuron is simulated as a representative of another set of neurons and used to predict activity in that set. Proxy neuron spikes can be used as a signal to expand the simulated network through neuron construction.

The surrounding neurons may have characteristics that are required, desired or assumed, and the design of the proxy neuron may aim to represent or reproduce these characteristics. Alternatively, the proxy neuron may be designed to give a desired response and then the plausibility of an associated set of neurons in the surrounding network examined. In this chapter, a proxy neuron is designed to predict the presence of a pattern in presynaptic activity that may repeat.

The proxy neuron development performed here assumes that the onset of a presynaptic spike pattern produces a clear and consistent increase in overall presynaptic activity. This condition allows the proxy neuron synapse weights and threshold to be selected to detect this increase.

Synapse weights quantify the bias or sensitivity of the postsynaptic neuron to detecting the activity of presynaptic neurons. Different presynaptic spike patterns may involve different sets of presynaptic neurons. Therefore, a proxy postsynaptic neuron with non-uniform synapse weights will be biased in the prediction of spike patterns. To reduce this bias, a uniform synapse weight for the proxy neuron is selected. Effective numerical values of the synapse weights and the threshold are dependent on simulation conditions.

In simulations of mature networks, most neurons are assumed to have undergone some prior synaptic plasticity. A proxy neuron might predict the activity of neurons that are mature and have weights resulting from past plasticity. Proxy neurons may also represent the predictions of the past activity of immature neurons. The prediction of past spike times of immature neurons may be used in the synapse weight calculations for neuron construction that assume past STDP. In this case it should not be expected that the constructed neuron reproduces the relative spike time of the proxy neuron;

the constructed neuron should produce the relative spike time of a neuron that has undergone an equivalent number of STDP updates.

## 5.3 Spike-Triggered Construction

A proxy neuron spike taken as a prediction of a surrounding neuron spike time may also be taken as a signal to transfer one or more neurons into the simulation (Figure 5.1). Two standard types of constructive algorithm processes were proposed in Chapter 2: processes for performance evaluation, including conditions for changing the network structure; and processes for parameter calculation and assignment. The proxy neuron is simulated as a process for evaluating the neural network performance; proxy neuron spikes are a condition for neuron construction.

The outline of a constructive algorithm that triggers neuron construction in response to a proxy neuron spike is presented in pseudocode (Algorithm 6). The simulation is described as advancing on update steps; in practice, the simulation of the spiking neural network could be updated in time steps with fixed duration or updated on network events such as neuron spike times. The constructive algorithm is not dependent on whether the implementation of the simulation is time driven or event driven.

**Algorithm 6** Proxy neuron spike-triggered construction

```
1: for neural network update steps, t ← 1, 2, 3, ..., T do,
2:     Update the neural network
3:     Update the proxy neuron
4:     if the proxy neuron spikes then
5:         Calculate new neuron and synapse parameters
6:         Add the new neuron and synapses to the network
7:     end if
8: end for
```

In principle, the constructive algorithm can complete the neuron construction in the same simulation event step as the spike of the proxy neuron (Algorithm 6). Methods for calculating synapse weights as an estimate of past STDP (developed in Section 3.2.3) allow for presynaptic spikes after the predicted postsynaptic spike to be included in calculations. This requires that synapse weight calculations continue until the end of the time window for eligible presynaptic activity or that the neuron construction and calculations are delayed. Delaying construction requires the activity of presynaptic neurons to be recorded in memory until the calculation is completed. Pseudocode for delayed neuron construction is provided in Algorithm 7. The constructive algorithms developed and evaluated in this chapter incorporate this delay to include depressed synapse weights in the parameter calculations.

The constructive algorithm can incorporate additional performance evaluation processes and conditions for construction and pruning. Augmentations of the proxy neuron simulation are investigated in later chapters of this thesis.

**Algorithm 7** Proxy neuron spike-triggered construction with delay

**initialise** construction flag, $c \leftarrow 0$; construction time, $t_c \leftarrow -\infty$
**for** neural network update steps, $t \leftarrow 1, 2, 3, ..., T$ **do**,
    Update the neural network
    **if** neuron construction flag is set, $c = 1$ **then**
        **if** $t > t_c$ **then**
            Calculate new neuron and synapse parameters
            Add the new neuron and synapses to the network
            Clear neuron construction flag: $c \leftarrow 0$
        **else**
            Record eligible relative spike times, $\Delta t_{\text{proxy},j} \leq T_-$
        **end if**
    **else**
        Update the proxy neuron
        **if** the proxy neuron spikes **then**
            Record eligible relative spike times, $\Delta t_{\text{proxy},j} > -T_+$
            Set neuron construction time: $t_c \leftarrow t + T_-$
            Set neuron construction flag: $c \leftarrow 1$
        **end if**
    **end if**
**end for**

## 5.4 Simulated Experiments

Proxy neurons and constructive algorithms that implement them have now been proposed; next, simulated experiments will be described that investigate the performance of these constructive algorithms. Network and neuron models have been adapted from a past study of additive STDP resulting in the detection of hidden spike patterns (Masquelier et al., 2008). Detailed pseudocode of the simulation including the constructive algorithm is provided. Then the constructed synapse weights and postsynaptic neurons are evaluated in comparison with the synapse weights and spike times resulting from neurons simulated with STDP.

### 5.4.1 Aims

Simulations are performed to investigate the results of neuron construction triggered by the spikes of a proxy neuron representing the activity of surrounding postsynaptic neurons. This approach to neuron construction has been developed to be compatible with concepts of simulation expansion and, therefore, compatible with simulations of biological neural networks. An aim of the experiments is to examine the difference in outcomes of neurons constructed assuming past activity and STDP and the result of an equivalent neuron simulation with STDP.

Specifically, the experimental results are collected to compare synapse weights re-

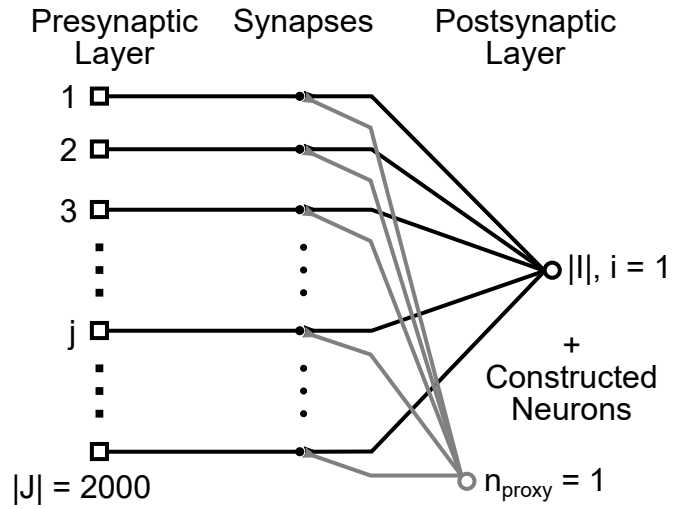


**Figure 5.2:** Network architecture for proxy neuron simulations. Simulations start with one proxy postsynaptic neuron ($n_{\text{proxy}} = 1$) and one standard postsynaptic neuron ($|I| = 1$). The activity of the $|J| = 2000$ presynaptic neurons is pregenerated. The proxy neuron triggers construction and adds neurons to the postsynaptic neuron set ($I$).

sulting from neurons simulation with STDP and neuron construction (extending the investigation of STDP estimation presented in Chapter 4). Potentiation of synapses reduces postsynaptic spike latencies to the same presynaptic activity; therefore, simulated neurons and constructed neurons are also compared for subsequent spike latencies relative to the onset of a repeating pattern. Experiments are repeated for presynaptic spike patterns with increasing background spike rates and the effects on synapse weights and postsynaptic spike times are examined. Sufficient equivalence in the neurons resulting from simulation and construction may also be obtained in the event that there is a low disparity in the spike latency of neurons.

### 5.4.2 Network Structure and Presynaptic Activity

The network (Figure 5.2) has the same general structure as used in the validation of STDP estimates and in the past study of additive STDP (Masquelier et al., 2008). The simulation of the predefined postsynaptic neuron and STDP has been performed concurrently with the proxy neuron simulation and neuron construction; however, the postsynaptic neurons do not interact. The network has $|J| = n_{\text{pre}} = 2000$ presynaptic neurons with half selected to repeat a spike pattern, $n_{\text{pattern}} = 1000$. Simulations are initialised with one predefined postsynaptic neuron ($|I| = n_{\text{post}} = 1$) that is based on the spike response model (Gerstner & Kistler, 2002; Masquelier et al., 2008) and is updated using an event-driven update cycle (Section 5.4.3). Simulations are also initialised with one proxy neuron given the same model and parameters as the predefined postsynaptic neuron in the STDP simulations.

Presynaptic neuron activity is generated stochastically prior to the simulation of

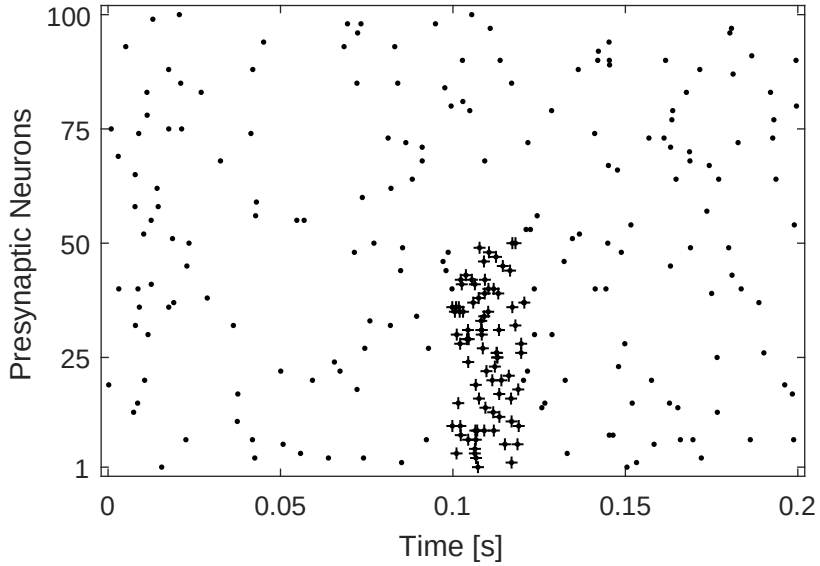


**Figure 5.3:** An example of presynaptic activity in a 200 ms segment. The first 100 ms is background activity (10 Hz in this example), the next 20 ms contains the spike pattern, and the segment ends with another 80 ms of background activity. The figure shows activity for 100 of the 2000 presynaptic neurons to improve the visibility of individual spikes. The neurons displayed are in the same proportion as the overall simulation: 50% of neurons repeat pattern spikes, 50% of neurons only produce noise.

the postsynaptic neuron and STDP. Figure 5.3 presents an example of a segment of the presynaptic activity. Presynaptic neuron activity is generated for 21 repetitions of a 200 ms segment (total simulation time of 4.2 s). The first 100 ms of each segment is pure background activity, the next 20 ms contains the repeating presynaptic spike pattern and the remaining 80 ms is pure background activity. The delay between repetitions of the presynaptic spike pattern reduces the interference of the earlier spike pattern on the STDP.

The number of presynaptic spikes in the repeating pattern was selected to reliably produce a single spike in the postsynaptic neuron. The repeating spike pattern is composed of $G_{\text{pattern}} = 1600$ spikes spread across all neurons involved in the pattern ($n_{\text{pattern}} = 1000$) for the duration of the pattern (0.02 s). The repeating presynaptic spike pattern is generated in a number of steps:

1. Pattern neurons are assigned one spike at a random time (uniform probability) in the 20 ms pattern time (1000 spikes).

2. The remaining 600 pattern spikes are assigned randomly (uniform probability) to the pattern neurons and given a random time in the 20 ms pattern time.

3. These spike times form the base of the spike pattern in each of the 21 segments with zero-mean, 1 ms-standard deviation Gaussian noise added to each spike time.

**Table 5.1:** Network and presynaptic activity parameters.

| | |
|---|---|
| $n_{\text{pre}}$ | 2000 |
| $n_{\text{pattern}}$ | 1000 |
| $n_{\text{post}}$ | 1 |
| $n_{\text{proxy}}$ | 1 |
| $G_{\text{pattern}}$ | 1600 |
| $r_b$ | 0 Hz, 10 Hz and 20 Hz |

The precise repetition of relative spike times has been observed in biological neural networks (Masquelier, 2012); the addition of Gaussian noise to spike times is included to accommodate potential variability.

This process of generating presynaptic spike times is a simplification of the process used in the past study of STDP (Masquelier et al., 2008). The past study generated presynaptic spike times with an average spike rate of 64 Hz over all of the neurons. The repeating spike pattern was concealed to prevent it from producing a consistent change in the overall activity of presynaptic neurons compared to periods of pure noise spikes. The simulations presented in this chapter assume that significant spike patterns are associated with a detectable change in the overall presynaptic activity. Therefore, the addition of background neuron activity is applied as an additive noise.

The random background activity of a presynaptic neuron may be modelled as a Poisson process. Background presynaptic activity at a rate of 25 Hz had a significant probability of causing multiple postsynaptic neuron spikes times during the presynaptic spike pattern. This complicates comparison of the performance of constructed neurons and neurons simulated with STDP; therefore, background activity was limited to 20 Hz. Batches of simulations were performed for background Poisson process rates, $r_b = 0$ Hz, 10 Hz and 20 Hz.

Spike times for background activity were generated in 1ms time steps up to the total simulation time of 4.2 seconds. A discrete approximation of a Poisson process calculates the probability of a neuron spiking in a time step as the neuron rate parameter multiplied by the time step length, $r_b \cdot \Delta t$. A random number, `rand`, with uniform probability in $[0, 1]$ is computer-generated for each neuron in each time step and compared with the calculated probability of a spike, $r_b \cdot \Delta t >$ `rand`. Neurons with a random number that satisfies the comparison emit a spike within that 1ms time step. The exact time of each spike is generated at random with a uniform probability in that step. The number of neurons and parameters for presynaptic activity are summarised in Table 5.1.

### 5.4.3 Synapse and Postsynaptic Neuron Model

Simulations are performed by applying the generated presynaptic activity to the postsynaptic neuron. The postsynaptic neuron model implemented in simulations is a reproduction of the model described in the past study of STDP (Masquelier et al., 2008). The spike response neuron model treats the change in postsynaptic potential in

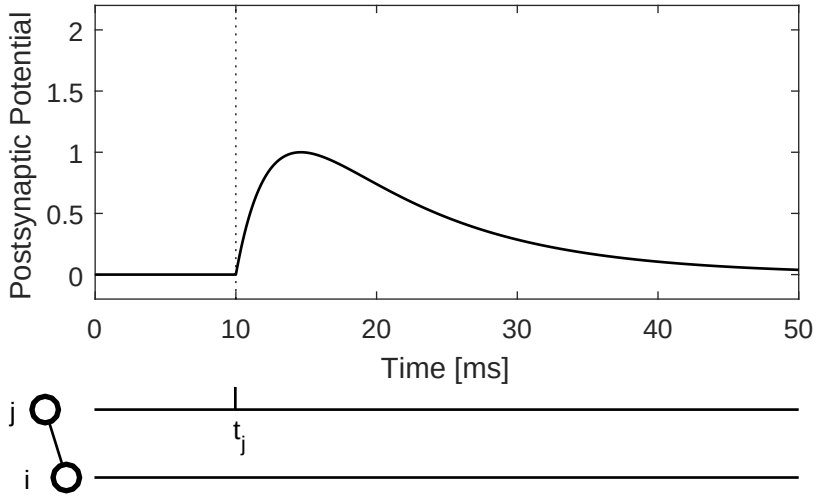


**Figure 5.4:** The function for the excitatory postsynaptic potential (EPSP) in response to a spike from presynaptic neuron $j$. This is a graphical representation of Equation 5.1, the function for $\epsilon(t - t_j)$, with $t_j = 10\,\text{ms}$, $\tau_m = 10\,\text{ms}$, $\tau_s = 2.5\,\text{ms}$ and $K = (4^{4/3})/3$.

response to a presynaptic spike as a time-dependent function,

$$\epsilon(t - t_j) = K \cdot \Big( \exp\Big( - \frac{t - t_j}{\tau_m} \Big) - \exp\Big( - \frac{t - t_j}{\tau_s} \Big) \Big) \cdot \Theta(t - t_j). \tag{5.1}$$

The difference of exponentials with time constants $0 < \tau_s < \tau_m$ produces a smooth rise and fall in the postsynaptic potential (Figure 5.4). The Heavyside step-function, $\Theta(t - t_j)$, sets the value of the function to zero until the time of the presynaptic spike $t_j$. The constant $K$ is selected to ensure $\max_t \epsilon(t - t_j) = 1$. This postsynaptic potential function is multiplied by the synapse weight, $w_{i,j}(t_j) \in [0, 1]$, to determine the maximum contribution of presynaptic spike. The simulations presented in this chapter had all synapse weights initialised with a value of $w_{i,J}(0) = 0.5$.

Postsynaptic spikes occur when a threshold for membrane potential, $\theta$, is exceeded and are described by a similar time-dependent function,

$$\begin{aligned} \eta(t - t_i) = \theta \cdot \Big[ & K_1 \cdot \exp\Big( - \frac{t - t_i}{\tau_m} \Big) \\ & - K_2 \cdot \Big[ \exp\Big( - \frac{t - t_i}{\tau_m} \Big) - \exp\Big( - \frac{t - t_i}{\tau_s} \Big) \Big] \Big] \cdot \Theta(t - t_i). \end{aligned} \tag{5.2}$$

The constants $K_1 = 2$ and $K_2 = 4$, giving this function has an initial value $\eta(0) = 2 \cdot \theta$. Without other input, the postsynaptic potential quickly decays to below the resting threshold (hyperpolarisation) before gradually settling at the resting value (Figure 5.5)

The total postsynaptic potential is a summation of the most recent postsynaptic spike and contributions of presynaptic spikes since the last postsynaptic spike,

$$p(t) = \eta(t - t_i) + \sum_{j \in J,\, t_i < t_j < t} w_{i,j} \cdot \epsilon(t - t_j). \tag{5.3}$$

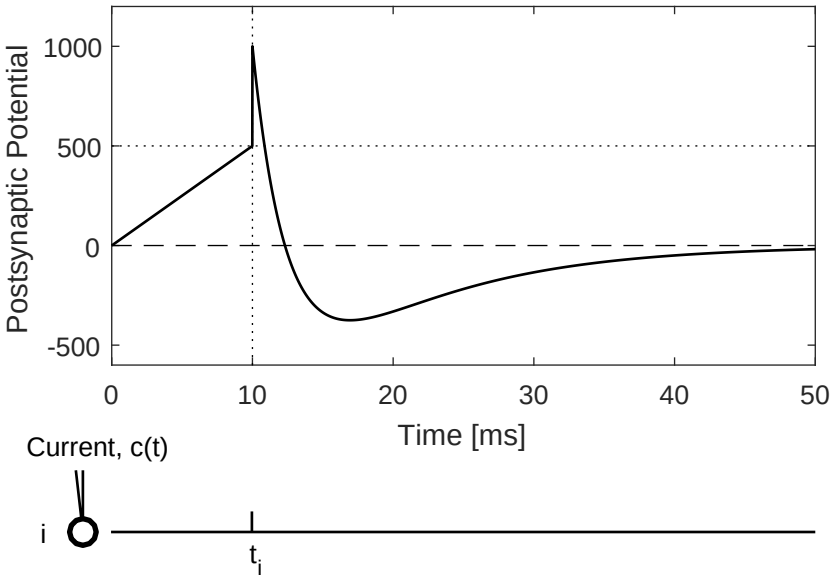


**Figure 5.5:** The postsynaptic potential is increased linearly up to the threshold ($\theta = 500$) then the potential changes according the equation for a postsynaptic spike. The injected current, $c(t)$, is assumed to control the postsynaptic potential from $t = 0\,\text{ms}$ to $10\,\text{ms}$ and then let the potential change freely. The postsynaptic potential from the spike time $t_i = 10\,\text{ms}$ is a graphical representation of Equation 5.2, the function for $\eta(t - t_i)$, with $\tau_m = 10\,\text{ms}$ and $\tau_s = 2.5\,\text{ms}$.

The simulations performed were event driven: the postsynaptic neuron model is updated at presynaptic spike times. The postsynaptic neuron is implemented with two variables for postsynaptic potential, one for each time constant, with their sum giving the membrane potential,

$$p_i(t) = p_{m,i}(t) + p_{s,i}(t). \tag{5.4}$$

The steps in a simulation update cycle are:

1. The next presynaptic spike time is selected and the change in simulation time is calculated (Equation 5.5).

2. Postsynaptic neuron potential variables receive an exponential decay update for the change in time (Equations 5.6a and 5.6b).

3. If the postsynaptic neuron potential is above the activation threshold:

   (a) The postsynaptic neuron has potential variables set to spike values (Equations 5.8a and 5.8b).

   (b) Synapses to spiking postsynaptic neurons receive a positive weight update according to the STDP model (Equation 3.3).

4. Synapses from the neuron with the current presynaptic spike time receive a negative weight update according to the STDP model (Equation 3.4).

5. Postsynaptic neuron potential variables are increased in proportion to the weight of the synapse from the spiking presynaptic neuron (Equations 5.11a and 5.11b).

This sequence is incorporated into the update equations using the infinitesimal time, $\varepsilon$, to denote times before or after the presynaptic neuron spike times, that is, $t_J^{(G)} - \varepsilon$ and $t_J^{(G)} + \varepsilon$. The times $t_J^{(G-1)}$ and $t_J^{(G)}$ are consecutive spikes in the population of presynaptic neurons $J$. The change in simulation time between presynaptic spikes is calculated as

$$\Delta t_J^{(G,G-1)} = t_J^{(G)} - t_J^{(G-1)}. \tag{5.5}$$

After the initial step of selecting the next presynaptic neuron spike as an event time, the potential variables of the postsynaptic neuron undergo exponential decay with time constants $\tau_m$ and $\tau_s$,

$$p_{m,i}(t_J^{(G)} - \varepsilon) = p_{m,i}(t_J^{(G-1)}) \cdot \exp(-\Delta t_J^{(G,G-1)}/\tau_m), \tag{5.6a}$$

$$p_{s,i}(t_J^{(G)} - \varepsilon) = p_{s,i}(t_J^{(G-1)}) \cdot \exp(-\Delta t_J^{(G,G-1)}/\tau_s). \tag{5.6b}$$

The decay of potential is assumed to have occurred in the interval up to an infinitesimal time prior to the presynaptic neuron spike time, $t_J^{(G)} - \varepsilon$. Note that the potential variables are often different polarities and magnitudes; therefore, the exponential decay may result in an increase or decrease in the overall neuron potential (the sum of the potential variables).

After calculating the decay of the potential variables the total potential of the postsynaptic neuron is calculated (Equation 5.4) for $t = t_J^{(G)} - \varepsilon$. A postsynaptic neuron spike occurs if the total potential exceeds the threshold

$$p_i(t_J^{(G)} - \varepsilon) > \theta. \tag{5.7}$$

The threshold value $\theta = 500$ has been replicated from the past study (Masquelier et al., 2008).

Postsynaptic neuron spikes are implemented in simulation by setting the potential variables to values given by constant scaling factors and the threshold value,

$$p_{m,i}(t_J^{(G)}) = p_{m,\text{spike}} = (K_1 - K_2) \cdot \theta, \tag{5.8a}$$

$$p_{s,i}(t_J^{(G)}) = p_{s,\text{spike}} = K_2 \cdot \theta. \tag{5.8b}$$

The scaling factors $K_1 = 2$ and $K_2 = 4$ have values taken from the previous study (Masquelier et al., 2008), giving $p_{m,\text{spike}} = -2 \cdot \theta$ and $p_{s,\text{spike}} = 4 \cdot \theta$. The model in the previous work and the reproduction include an activation refractory period $t_{\text{ref}} = 1\,\text{ms}$ that prevents the spike in postsynaptic neuron potential from immediately triggering another spike.

The nearest-neighbour model of additive STDP described in Section 3.2.1 is implemented here. The amplitude parameters for the positive and negative plasticity curves take values $A_+ = 0.03125$ and $A_- = 0.85 \cdot A_+$, respectively. The rate of curve decay is given by the time constants $\tau_+ = 16.8\,\text{ms}$ and $\tau_- = 33.7\,\text{ms}$. These values have been reproduced from the past study of STDP (Masquelier et al., 2008). For brevity, a detailed description of the additive STDP model is not reproduced here.

If the postsynaptic neuron spikes and it is the first since the last presynaptic spike of neuron $j$ at time $t_j^{(g-1)}$ (it is the nearest neighbour), a positive change in the synapse weight occurs,

$$w_{i,j}(t_J^{(G)}) = \min(w_{\max}, w_{i,j}(t_J^{(G)} - \varepsilon) + \Delta w_{i,j}^{(f,g-1)}). \tag{5.9}$$

If the spike from the presynaptic neuron $j$ at time $t_j^{(g)} = t_J^{(G)}$ is the first since the last postsynaptic neuron spike, a negative change in the synapse weight occurs,

$$w_{i,j}(t_J^{(G)} + \varepsilon) = \max(w_{\min}, w_{i,j}(t_J^{(G)}) + \Delta w_{i,j}^{(f,g)}). \tag{5.10}$$

A spike from presynaptic neuron $j$ makes a contribution to the slow decaying variable ($p_{m,i}(t)$) and the fast decaying variable ($p_{s,i}(t)$),

$$p_{m,i}(t_J^{(G)} + \varepsilon) = p_{m,i}(t_J^{(G)}) + K \cdot w_{i,j}(t_J^{(G)} + \varepsilon), \tag{5.11a}$$

$$p_{s,i}(t_J^{(G)} + \varepsilon) = p_{s,i}(t_J^{(G)}) - K \cdot w_{i,j}(t_J^{(G)} + \varepsilon). \tag{5.11b}$$

The value of $K = (4^{4/3})/3$ can be calculated by solving the postsynaptic potential function $\max_t \epsilon(t) = 1$ for $\tau_m = 10\,\text{ms}$ and $\tau_s = 2.5\,\text{ms}$. This concludes a simulation update cycle. The simulation repeats this update cycle with the next presynaptic spike and continues until all presynaptic spikes have been processed. The parameters for the synapse, postsynaptic neuron, and STDP models are summarised in Table 5.2.

In an event-driven simulation that only updates at presynaptic spike times, it is possible that the postsynaptic potential could rise above the threshold and fall back below the threshold between updates. This is potentially true of time driven simulations as well; in practice, however, the time steps are selected to be short enough that missing a postsynaptic spike in this way has very low likelihood. Similarly, if the time between update events (presynaptic neuron spikes) is short enough, a postsynaptic spike is unlikely to be missed.

The simulations performed have an expected time between presynaptic spikes of 0.0125ms during the spike pattern for zero background activity. The presynaptic activity and postsynaptic neuron variables have been selected to ensure that the postsynaptic neuron will only spike once during each repetition of the presynaptic spike pattern. The probability of an additional postsynaptic spike occurring is considered negligible due to post-spike hyperpolarisation.

### 5.4.4 Proxy Neuron Simulation and Neuron Construction

A proxy neuron is simulated and its spikes are used as the condition to trigger neuron construction and to provide a prediction of a postsynaptic spike time for synapse weight

**Table 5.2:** Summary of synapse, postsynaptic neuron, and STDP model parameter values that appear in equations. These values are reproduced from a past computational study of STDP (Masquelier et al., 2008) and were developed to approximate findings in biological experiments.

| | |
|---|---|
| $w_{\min}$ | 0 |
| $w_{\max}$ | 1 |
| $w_J(0)$ | 0.5 |
| $\tau_m$ | 10 ms |
| $\tau_s$ | 2.5 ms |
| $\theta$ | 500 |
| $K$ | $(4^{4/3})/3$ |
| $K_1$ | 2 |
| $K_2$ | 4 |
| $p_{m,\text{spike}}$ | $-2 \cdot \theta$ |
| $p_{s,\text{spike}}$ | $4 \cdot \theta$ |
| $t_{\text{ref}}$ | 1 ms |
| $A_+$ | 0.03125 |
| $A_-$ | $0.85 \cdot A_+$ |
| $\tau_+$ | 16.8 ms |
| $\tau_-$ | 33.6 ms |

calculations. The proxy neuron is updated in the same simulation cycle as a standard postsynaptic neuron; however, the proxy neuron synapses do not experience STDP and the potential is not updated during neuron construction.

The proxy neuron potential variables receive decay updates and contributions from presynaptic spikes. If the proxy neuron spikes, construction is triggered and the proxy neuron potential variables are reset to the resting values (not the spike values). Neuron construction is delayed to record the activity of presynaptic neurons after the predicted postsynaptic spike time and calculate the depression of those connections (Algorithm 7).

In a biological neural network, spike patterns may coincide or overlap. For the simplification of analysis, the simulations investigate the case of a single repeating spike pattern without overlaps. The simulation of the proxy neuron is paused during construction under the assumption that spike patterns of interest will not overlap and to prevent the simultaneous construction of multiple neurons. Simulation updates of the proxy neuron resume at the completion of weight calculations and neuron construction.

The synapse weights to the proxy neuron are kept constant to prevent spike pattern detection becoming biased to specific presynaptic neurons. Given that the proxy neuron is a generalised model of untuned postsynaptic neurons, the synapse weights are set to the initial value of the predefined postsynaptic neurons, that is, $w_{\text{proxy},J} = 0.5$. All other parameter values of the proxy neuron are equal to those of the predefined postsynaptic neurons (see Table 5.2).

The process for synapse weight calculation implemented is the estimation of spe-

**Table 5.3:** Proxy neuron and constructive algorithm parameters.

| | |
|---|---|
| $w_{\text{proxy},J}$ | 0.5 |
| $M$ | 1, 5, 10, 15 and 20 |
| $T_+$ | 50.4 ms |
| $T_-$ | 67.4 ms |

cific numbers of iterations of additive STDP (Section 3.2.3.1). The window of eligible presynaptic activity is limited to ensure that the delayed construction completes in finite time. Presynaptic spikes must occur within a window around the proxy neuron spike time, $t_J \in [t_{\text{proxy}} - T_+, t_{\text{proxy}} + T_-]$. The values $T_+ = 3 \cdot \tau_+ = 50.4$ms and $T_- = 2 \cdot \tau_- = 67.4$ms have been selected to ensure that the eligibility window ends and construction completes before the start of the next pattern repetition. The calculations of synapse weight are performed for a selection of iteration numbers, $M = 1$, 5, 10, 15 and 20.

The proxy neuron and constructive algorithm parameters are summarised in Table 5.3. A pseudocode summary of the simulation update cycle with proxy neuron simulation and neuron construction is provided as Algorithm 8.

### 5.4.5 Data Collection and Analysis

The delay or spike latency of postsynaptic neurons is measured relative to the start of the repeating presynaptic spike pattern in each segment of presynaptic activity (100 ms into each segment of presynaptic activity, see Figure 5.3). Spike latencies and synapse weights of constructed postsynaptic neuron and predefined postsynaptic neurons updated through STDP are collected and compared. Close agreement between the spike latencies of constructed neurons and the spike latencies of predefined neurons updated through STDP is evidence of the compatibility of neuron construction in simulations of neurons with STDP.

Simulations are performed with three values of background spike activity, $r_b = 0$ Hz, 10 Hz and 20 Hz. For each background activity rate, one hundred sets of presynaptic activity are generated and used to simulate activity in predefined postsynaptic neurons and to construct postsynaptic neurons. Each set of generated presynaptic activity includes 21 repetitions of a spike pattern.

Predefined postsynaptic neurons have synapses updated through STDP with spike latencies and synapse weights recorded at the end of each repetition of the spike pattern. With one predefined postsynaptic neuron in each of the 100 simulations for each rate of background activity, 100 spike latencies are recorded for each number of presynaptic spike pattern repetitions and set of STDP updates. The initial input synapse weights of predefined postsynaptic neurons is provided; synapse weights are recorded after each presynaptic activity segment providing $100 \times 2000 = 2 \times 10^5$ data points for each number of spike pattern repetitions.

Simulations with neuron construction are initialised with one proxy neuron. For each proxy neuron spike, neurons are constructed for each specified number of iterations

**Algorithm 8** Event-driven simulation with spike-triggered construction.

**initialise** construction flag, $c \leftarrow 0$; construction time, $t_\mathrm{c} \leftarrow -\infty$; neural network parameters, $p_{m,i}$, $p_{s,i}$, $p_{m,\mathrm{proxy}}$, $p_{s,\mathrm{proxy}}$, $\theta$, etc
**for** $t \leftarrow t_J^{(1)}, t_J^{(2)}, t_J^{(3)}, \ldots, t_J^{(G-1)}, t_J^{(G)}, \ldots, t_J^{(\Omega)}$ **do**
    Find change in simulation time, $t = t_J^{(G)}$, $\Delta t \leftarrow t_J^{(G)} - t_J^{(G-1)}$
    **if** neuron construction flag is set, $c = 1$ **then**
        **if** $t > t_\mathrm{c}$ **then**
            Calculate synapse weights (Section 3.2.3.1)
            Add the new neuron and synapses to the network
            Clear neuron construction flag: $c \leftarrow 0$
        **else if** current presynaptic spike is nearest, $t_j^{(g-1)} < t_\mathrm{proxy} \leq t_j^{(g)} = t$ **then**
            Record relative spike time: $\Delta t_{\mathrm{proxy},j}^{(g)} \leftarrow t_j^{(g)} - t_\mathrm{proxy}$
        **end if**
    **else**
        Update proxy neuron variable decay (Equations 5.6a and 5.6a)
        **if** proxy neuron spikes, $p_\mathrm{proxy} > \theta$ **then**
            Reset proxy neuron potential: $p_{m,\mathrm{proxy}} \leftarrow 0$ and $p_{s,\mathrm{proxy}} \leftarrow 0$
            Record proxy neuron spike time: $t_\mathrm{proxy} \leftarrow t$
            Record nearest presynaptic spike times: $\Delta t_{\mathrm{proxy},j}^{(g-1)} \leftarrow t_j^{(g-1)} - t$, $\forall j \in J$, $t - T_+ \leq t_j^{(g-1)} < t \leq t_j^{(g)}$
            Set the neuron construction time: $t_\mathrm{c} \leftarrow t + T_-$
            Set neuron construction flag: $c \leftarrow 1$
        **else**
            Add presynaptic spike to proxy neuron (Equations 5.11a and 5.11b)
        **end if**
    **end if**
    Update postsynaptic variable decay (Equations 5.6a and 5.6a)
    **if** postsynaptic neuron spikes, $p_i > \theta$ **then**
        Set postsynaptic potential and refractoriness: $p_{m,\mathrm{spike}}$, $p_{s,\mathrm{spike}}$, and $t_\mathrm{ref}$
        Positive synapse weight update (Equation 5.9)
    **end if**
    Negative synapse weight update (Equation 5.10)
    Add presynaptic spike to postsynaptic potential (Equations 5.11a and 5.11b)
**end for**

of STDP updates used in weight calculations, $M = 1$, 5, 10, 15 and 20. Synapse weights for construction are calculated and recorded in each simulation for each repetition of the presynaptic spike pattern. This provides $100 \times 21 \times 2000 = 4.2 \times 10^6$ recorded synapse weights for each specified number of STDP updates ($M$).

After construction, the postsynaptic neurons are simulated for the next segment of presynaptic activity to obtain a spike latency. The number of spike latencies recorded for each value of $M$ is $100 \times 20 = 2000$. The spike latency of the proxy neuron is also recorded for each repetition of the spike pattern giving $100 \times 21 = 2100$ values. The spike latencies of predefined neurons and constructed neurons are presented and compared using box plots.

The synapse weights produced through STDP updates and neuron construction form irregular distributions; therefore, the distributions of synapse weights have been presented with histograms. For brevity, synapse weight distributions are only presented for synapses constructed with specified iterations, $M = 10$ and 20, and for the synapses of predefined postsynaptic neurons updated from 10 and 20 iterations of the 200 ms segment of presynaptic activity.

The modification of synapse weights due to additive STDP is expected to result in a decrease in the postsynaptic spike time relative to the start of the presynaptic spike pattern (Masquelier et al., 2008). A decreasing postsynaptic spike time may result in a change in the order presynaptic and postsynaptic spikes and affect future STDP updates. Examples of the trajectory of individual synapse weights are presented to demonstrate the impact of changing spike latency on STDP updates (an effect neglected in synapse weight calculations for construction).

Simulations have been performed in MATLAB R2016a.

### 5.4.6 Simulation Results

Postsynaptic neurons demonstrated a decreasing spike time latency relative to the start of the presynaptic spike pattern for increasing STDP updates (Figure 5.6). Increasing background presynaptic spike rates produced an overall reduction in the spike latencies (Figures 5.7 and 5.8). The spike latencies of constructed postsynaptic neurons closely tracked the latencies of predefined neurons that had received an equivalent number of STDP updates. For the background spike rate of 0 Hz the constructed neurons had median, first quartile and third quartile values for spike latency at most 0.5 ms earlier than predefined neurons simulated with STDP (observed at 20 iterations). For background rates of 10 Hz and 20 Hz the majority of differences in spike latency statistics were less than 0.1 ms.

The decrease in postsynaptic neuron spike latency is a result of STDP increasing synapse weights. The decrease in postsynaptic spike time relative to the spike pattern start will cause some presynaptic spikes to transition from occurring before (and the synapse receiving positive updates) to occurring after the postsynaptic spike (and the synapse receiving negative updates). The proxy neuron has constant synapse weights, however, and does not experience a decrease in spike latency over the course of the simulation. This can result in different outcomes for individual synapse weights. Indi-

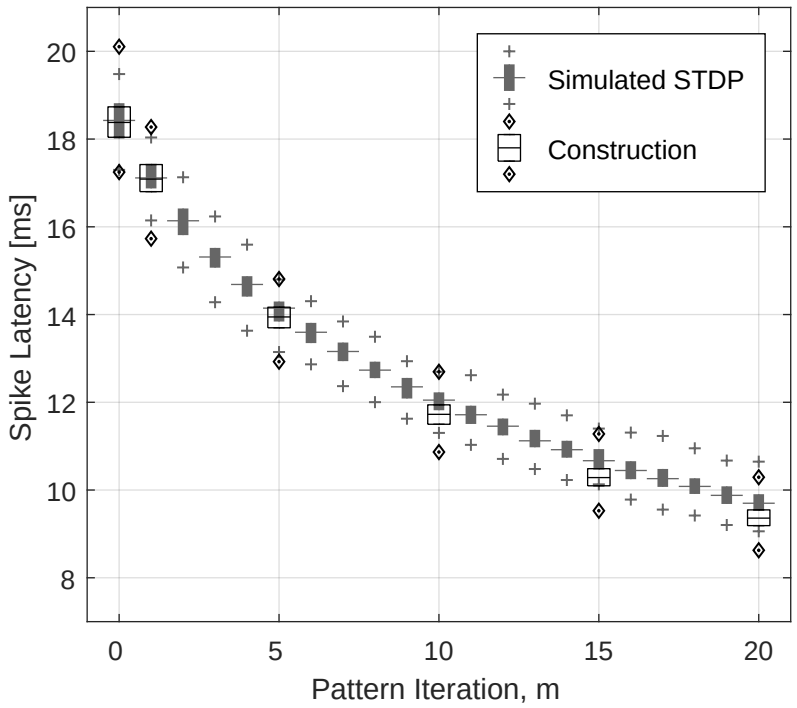


**Figure 5.6:** Box plots of postsynaptic spike latencies relative to the start of the repeating presynaptic spike pattern with zero background activity, $r_b = 0\,\text{Hz}$. Black boxes indicate statistics of the proxy neuron spike times (2100 spikes at 0 STDP updates) and constructed neuron spike times (2000 spikes at each $M = 1$, 5, 10, 15 and 20). The grey bars indicate statistics of the spike times of predefined postsynaptic neurons simulated with STDP (100 spikes at each STDP update value). The box middle lines indicate the median values, the bottom lines (or bar edges) indicate the first quartile values, and the top lines (or bar edges) indicate the third quartile values. The maximum and minimum spike times are indicated with a diamond and centre dot (proxy and constructed neurons) or a cross (neurons simulated with STDP).

vidual synapse weights that result from the simulation of STDP and from construction in a simulation with background activity $r_b = 10\,\text{Hz}$ are presented in Figure 5.9.

In the event that the presynaptic and postsynaptic spikes do not change order, the constructed synapse weights can be expected to have a distribution with the same trend resulting from STDP (for example, Figure 5.9A and B). When STDP results in a change in the order of presynaptic and postsynaptic spikes, the STDP updates become negative while constructed synapse weights remain in a potentiated distribution (for example, Figure 5.9C). When the spike times of the presynaptic neuron and proxy postsynaptic neuron fluctuate around a similar value, the constructed synapse weights can split into potentiated and depressed modes (for example, Figure 5.9D). A postsynaptic neuron simulated with STDP under the same initial condition results in

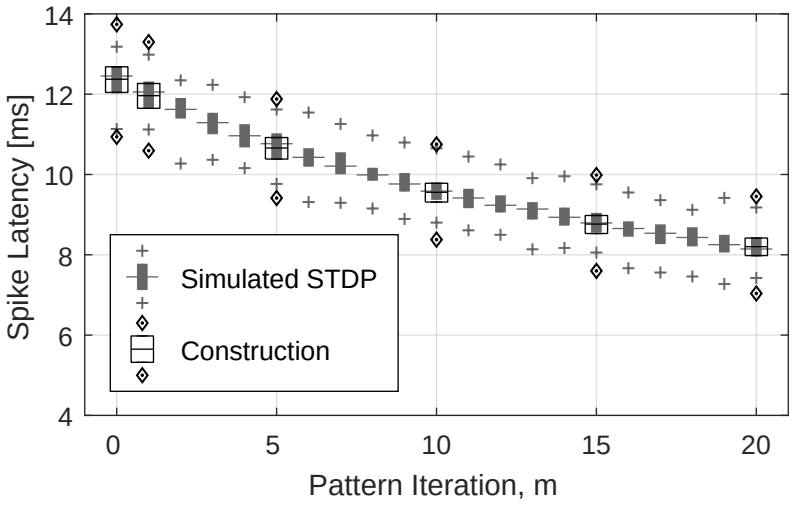


**Figure 5.7:** Box plots of postsynaptic spike latencies relative to the pattern start time for predefined neurons simulated with STDP and neurons constructed with estimated STDP (background activity, $r_b = 10\,\mathrm{Hz}$). See Figure 5.6 for a detailed description of the box plots.

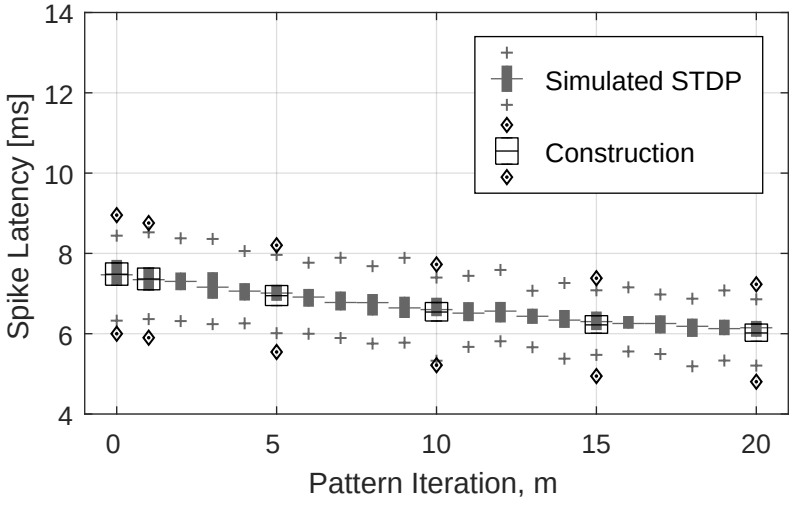


**Figure 5.8:** Box plots of postsynaptic spike latencies relative to the pattern start time for predefined neurons simulated with STDP and neurons constructed with estimated STDP (background activity, $r_b = 20\,\mathrm{Hz}$). See Figure 5.6 for a detailed description of the box plots.

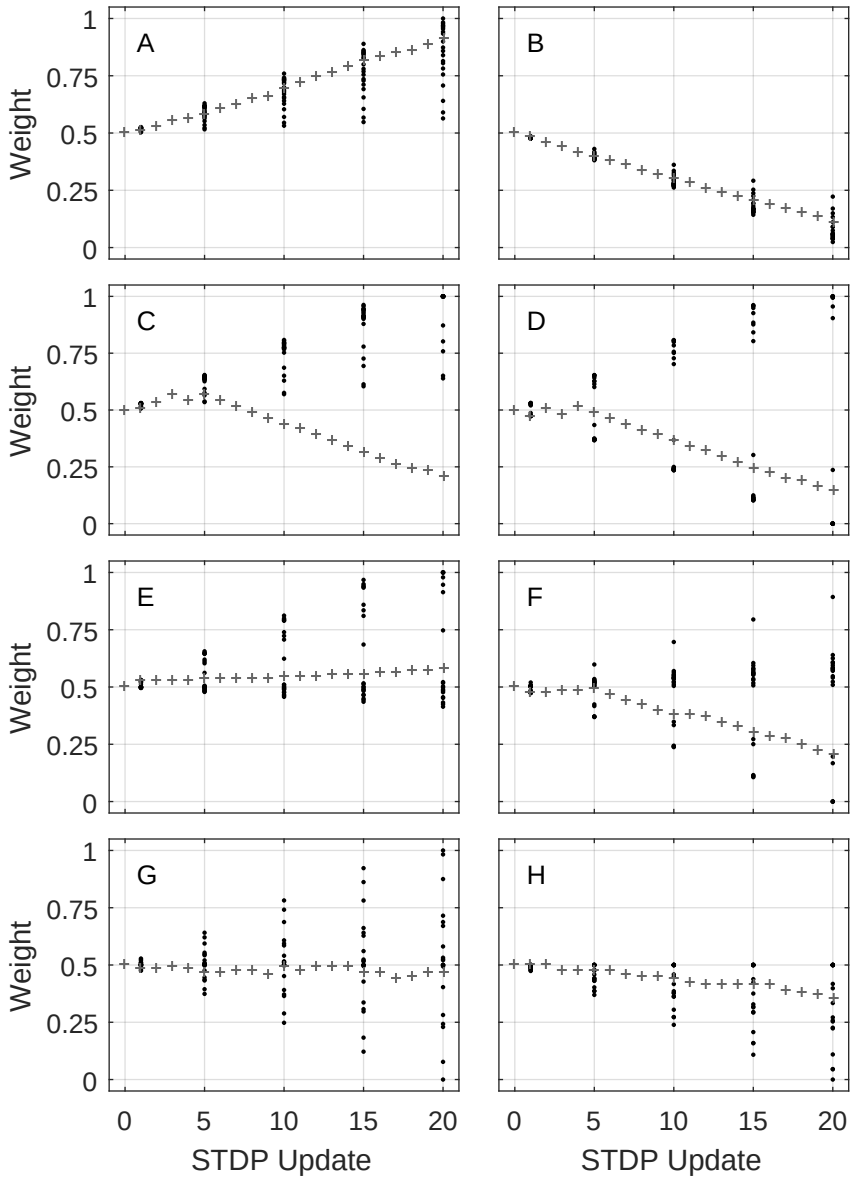


**Figure 5.9:** Examples of individual synapse weights changing during STDP simulation (grey crosses) and constructed synapse weights (black dots) for the same presynaptic neuron activity (including 10 Hz background activity). Synapses from presynaptic neurons involved in the repeating pattern are shown in axes labelled A-F. The change in synapse weights through STDP is affected by the decreasing latency of postsynaptic spikes. In C, D, and F synapse weights with STDP show initial positive or random changes before being depressed. The constructed synapse weights have variation caused by random fluctuations in presynaptic and proxy neuron spike times. Synapses from presynaptic neurons that only produce background activity are shown in axes labelled G and H.

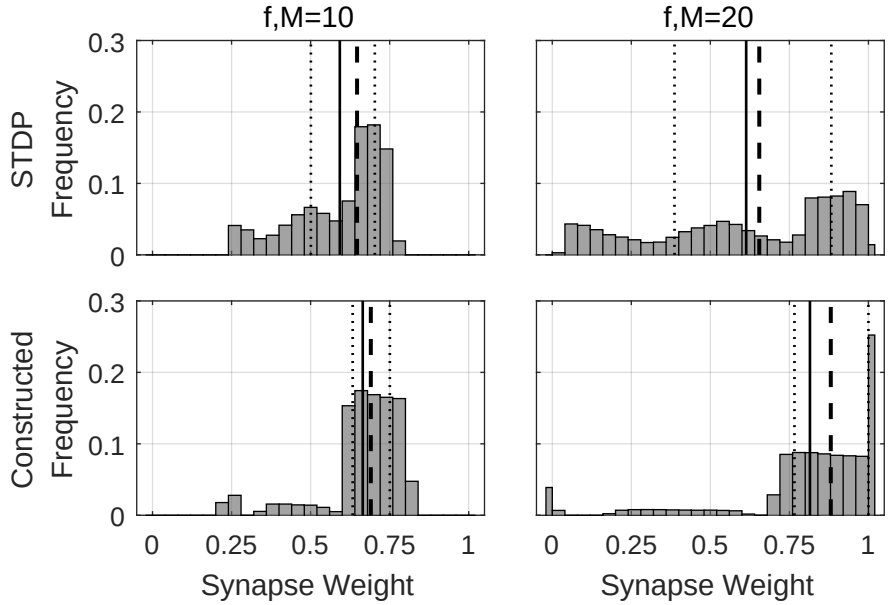


**Figure 5.10:** Histograms of synapse weights from pattern neurons (background spike rate of $r_b = 0\,\text{Hz}$). Distributions of synapse weights are organised into rows for the results of STDP and construction. Columns group equivalent STDP updates ($f = 10$ and 20) and constructed estimates ($M = 10$ and 20). The solid vertical line represents the mean synapse weight, the dashed line represents the median, and the dotted lines represent the first and third quartiles. Histogram bins have width $w = 0.04$. Synapses at minimum and maximum weight limits, 0 or 1, are placed in separate bins.

a decrease in postsynaptic spike latency and eventual depression of the synapse.

When the presynaptic neuron spikes twice, once before and once after the postsynaptic spike, the synapse weight updates may approximately cancel. The constructed synapse weights tend to cluster around the initial value (for example, Figures 5.9E and F). A postsynaptic neuron that has decreasing spike latency from STDP might not change the order of spikes (Figure 5.9E) or may be reduced to consistently occur before both presynaptic spikes and depress the connection (Figure 5.9F).

Presynaptic neurons that are not a pattern neuron (do not participate in the repeating pattern but produce random background activity) can result in constructed synapse weights distributed across all possible values (for example, Figure 5.9G). The STDP model simulated with uncorrelated activity produces a biased random walk. Since construction has a limited window of eligibility for presynaptic spikes, it is possible that a presynaptic neuron that produces background activity will be recorded spiking only before or only after the proxy neuron across multiple observations (for example, Figure 5.9H).

Differences have been observed in individual synapse weight outcomes from construction and STDP; however in neurons with many synapses, the overall distribution of synapse weights may also be significant in determining the postsynaptic neuron ac-

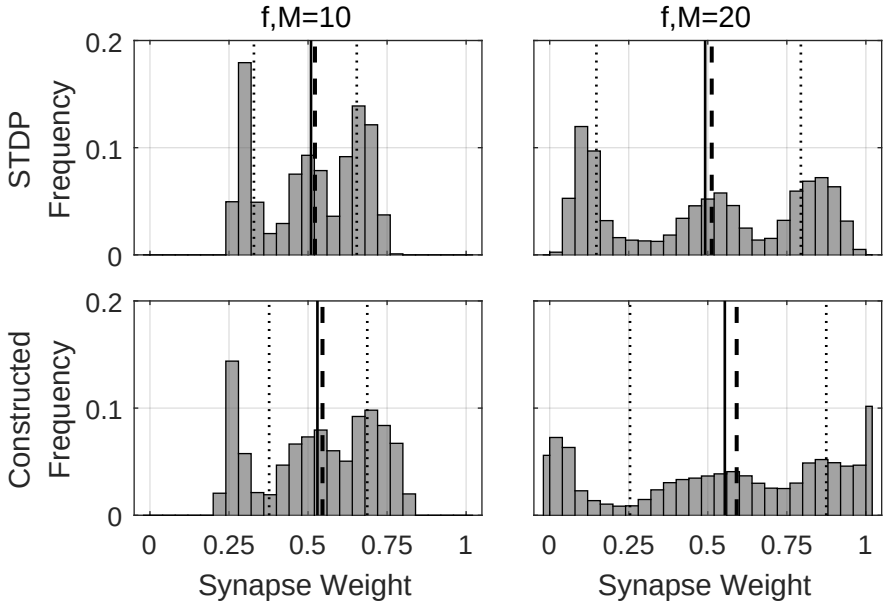


**Figure 5.11:** Histograms of synapse weights from pattern neurons (background spike rate of $r_b = 10\,\text{Hz}$). Further details are provided in the caption of Figure 5.10.

tivity. The distributions of synapse weights resulting from STDP simulation and construction are presented as histograms for synapses from presynaptic pattern neurons for increasing background rates: 0 Hz (Figure 5.10), 10 Hz (Figure 5.11), and 20 Hz (Figure 5.12). The distributions of synapse weights from background presynaptic neurons for background rates 10 Hz (Figure 5.14) and 20 Hz (Figure 5.15)) are presented. When background activity is zero the synapse weights from background presynaptic neurons do not change: these distributions are omitted. The distributions of the residual difference in synapse weights resulting from STDP and construction are also presented for synapses from the same presynaptic patterns neurons (Figure 5.13) and from the same presynaptic background neurons (Figure 5.16).

The distributions of synapse weights (Figures 5.10, 5.11, and 5.12) for equivalent STDP updates and constructed estimates have the same number of clusters, corresponding to presynaptic spikes before the postsynaptic spike (potentiated synapses), after the postsynaptic spike (depressed synapses), and presynaptic spikes before and after the postsynaptic spike (synapses near the initial value). The consistent spike time of the proxy neuron later than neurons simulated with STDP results in higher numbers of constructed synapses in the potentiated cluster and a portion of synapses that saturate at the maximum weight. This effect is visible in the higher mean and median values of synapse weight for constructed neurons.

Increases in the background spike rate results in the central cluster of synapse weights growing, indicating that more STDP updates and constructed synapses are occurring for presynaptic spikes before and after the postsynaptic spike. Simulations of STDP are more successful than construction in filtering the background spike noise

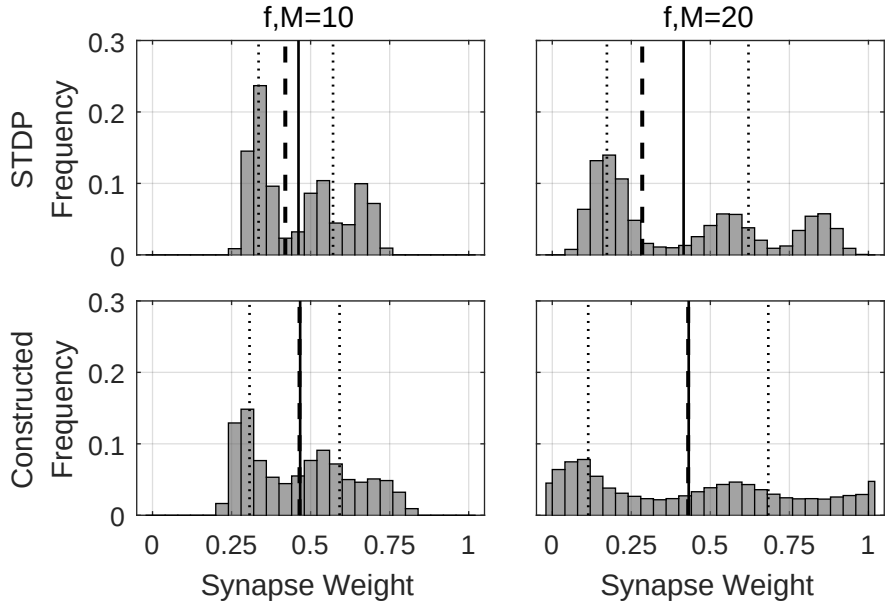


**Figure 5.12:** Histograms of synapse weights from pattern neurons (background spike rate of $r_b = 20\,\text{Hz}$). Further details are provided in the caption of Figure 5.10.

at the rate $r_b = 20\,\text{Hz}$, with clusters of synapse weights for depression, potentiation and spike triplets remaining easily discernible. The distribution in constructed synapse weights is largely flattened at the background spike rate $r_b = 20\,\text{Hz}$.

The difference in individual synapse weights resulting from equivalent STDP updates and construction shows that the central peak of the histogram is on or adjacent to the bin for zero error (Figure 5.13). The tail growing toward the positive limit for increasing update numbers is a result of the postsynaptic neuron simulated with STDP spiking at earlier times and starting to depress synapses that continue to be potentiated in calculations using the proxy neuron spike time. At 20 updates the peak in error shifts one histogram bin negative due to the spike time of the postsynaptic neuron simulated with STDP shifting earlier and producing greater positive updates for earlier presynaptic spikes.

The synapses from presynaptic neurons that only produce background activity follow approximately the same trends as observed in the earlier examination of synapse weight calculations (Section 4.5). Synapse weight calculations amplify the updates for observed presynaptic spike times while STDP updates may approximately cancel out, reducing the overall spread of the synapse weights. The synapse weight errors for neurons that only produce background activity (Figures 5.16) are limited due to synapse weights resulting from simulated STDP remaining clustered around the initial value ($w_j(0) = 0.5$).

The difference in mean synapse weight from pattern neurons between STDP simulation and construction decreases for increasing background spike rates. The difference in mean synapse weight from background activity neurons remains small for the back-

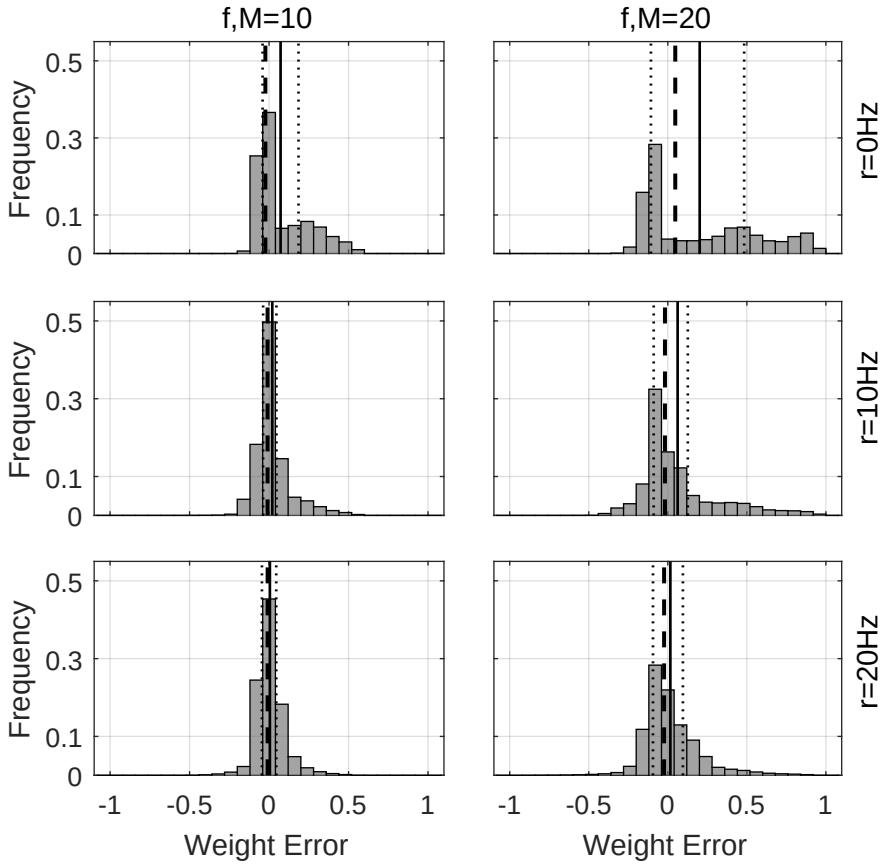


**Figure 5.13:** Histograms of the difference in synapse weight produced from STDP simulation and construction for the same pattern neuron. Columns group equivalent STDP updates ($f = 10$ and 20) and constructed estimates ($M = 10$ and 20). Rows group distributions by background spike rates, $r_b = 0\,\text{Hz}$, $10\,\text{Hz}$ and $20\,\text{Hz}$. The solid line represents the mean weight error, the dashed line represents the median, and the dotted lines represent the first and third quartiles. Histogram bins have width $w = 0.08$.

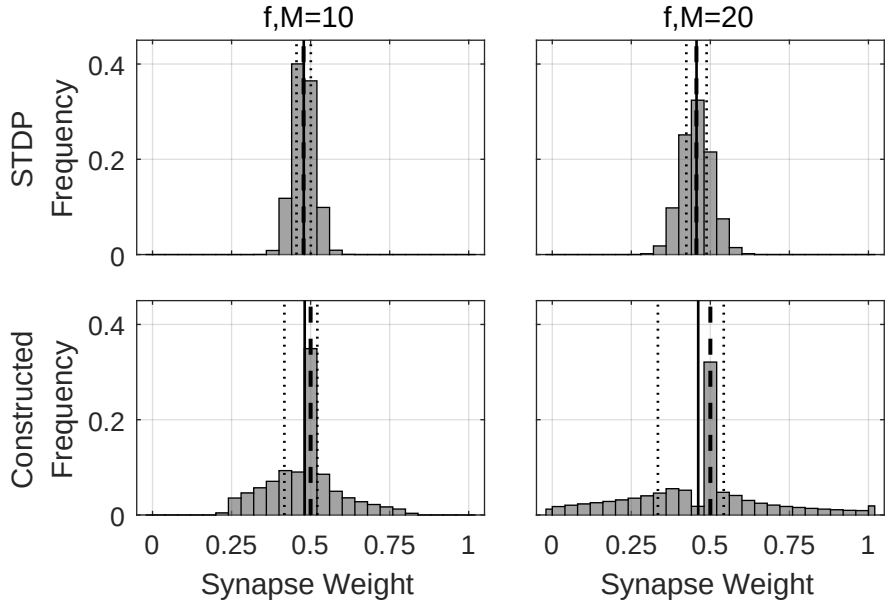


**Figure 5.14:** Histograms of synapse weights from background neurons with spike rate $r_b = 10\,\text{Hz}$. Further details are provided in the caption of Figure 5.10.

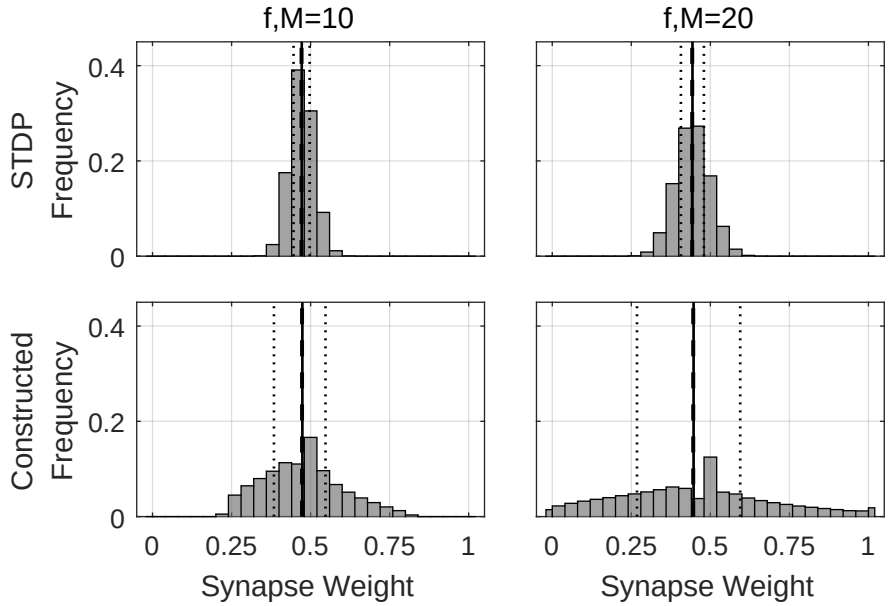


**Figure 5.15:** Histograms of synapse weights from background neurons with spike rate $r_b = 20\,\text{Hz}$. Further details are provided in the caption of Figure 5.10.

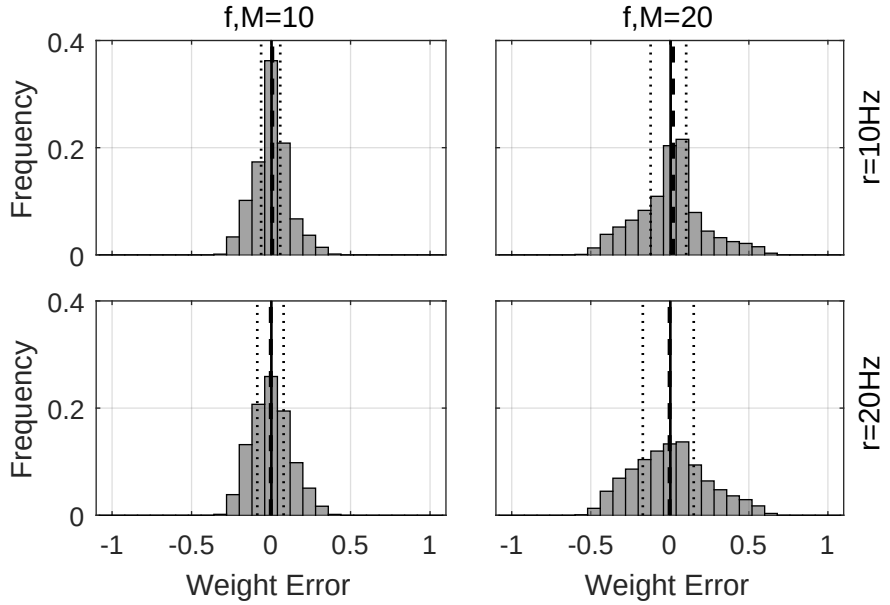


**Figure 5.16:** Histograms of the difference in synapse weight produced from STDP simulation and construction for the same background neuron. Columns group equivalent STDP updates ($f = 10$ and 20) and constructed estimates ($M = 10$ and 20). Rows group distributions by background spike rates, $r_b = 10\,\text{Hz}$ and $20\,\text{Hz}$. The solid line represents the mean weight error, the dashed line represents the median, and the dotted lines represent the first and third quartiles. Histogram bins have width $w = 0.08$.

ground activity rates tested. Neurons with approximately equal numbers of synapses and mean synapse weight will have an approximately equal total synapse weight and similar overall responsiveness to increasing presynaptic activity levels.

The median weight values of synapses constructed from pattern neurons are higher than synapses simulated with STDP and the difference does not diminish for increasing background activity. This indicates that more synapses are depressed in STDP simulations and suggests that the postsynaptic neuron will be responsive to a smaller selection of presynaptic neurons.

## 5.5 Discussion

This chapter has introduced the simulation of proxy neurons as a method for predicting spike times of surrounding postsynaptic neurons and triggering neuron construction. A novel constructive algorithm has been presented that combines the simulation of a proxy neuron with synapse weight calculations derived from STDP models. The presented constructive algorithm has capabilities not found in other constructive algorithms in literature (Section 2.3): compatibility with the continuous simulation of a spiking neural network; and compatibility with simulations that include STDP models. These are qualitative improvements on the past constructive algorithms for spiking

neural networks.

This constructive algorithm has been applied to simulations that repeat a presynaptic spike pattern. Postsynaptic spike latency and synapse weights that result from the simulation of STDP and neuron construction have been presented and compared. Close agreement in postsynaptic spike latencies were observed; however, differences in individual synapse weights and distributions may result in differences in neuron activity under other simulation conditions.

The weight calculations for synapse construction take an assumed number of past repetitions of a spike pattern (and therefore the number of STDP updates) prior to the start of a simulation. In practice, this information is unlikely to be available, but also may not be necessary to provide sufficiently plausible synapse weights. The constructed neurons are not estimates of specific neurons, they are estimates of neurons that may plausibly exist in the surrounding network. If an accurate model of a neuron that has a specific history is required, then it may be more appropriate to simulate that neuron or tailor construction to produce neurons with specific parameter values.

A constructive algorithm may be applied to increase the number of simulated neurons to improve an aspect of the simulation performance. Constructed neurons may immediately improve the performance of the neural network in presynaptic spike pattern detection. If this is not the case, constructed neurons may still be adequate to improve the network performance through subsequent plasticity or parameter tuning mechanisms. If performance value calculations are available, the proxy neuron simulation may be extended with additional performance evaluation processes to determine when to perform construction and whether a constructed neuron should remain in the simulation.

The developments presented in this chapter build on concepts introduced earlier in this thesis: constructive algorithms that perform neuron construction may be interpreted as adding neurons from the hypothetical larger neural system to maintain biological plausibility. The synapse weight calculations for neuron construction are based on assumptions of a mature and large surrounding network; therefore, the parameter calculations for constructed neurons are assumed to be biologically plausible. The activity of a postsynaptic neuron does not affect the activity of other neurons in the simulation; therefore, the addition of new postsynaptic neurons does not introduce implausible effects in the simulated network behaviour.

The presented method of constructing postsynaptic neurons is a generalised process that could take place in any postsynaptic layer of an artificial neural network. Nevertheless, the performance and biological plausibility of applying the developed constructive algorithms to more complex neural network structures requires further consideration and investigation. Chapter 7 introduces lateral inhibition within the postsynaptic layer; however, the performance of developed constructive algorithms in networks with additional layers of postsynaptic neurons is not investigated in this thesis.

The performance of the constructive algorithm has been quantitatively compared with the behaviour and parameters of neurons simulated with STDP. The results provide evidence that the constructive algorithm is compatible with simulations of bio-

logical neural networks; however, the learning capabilities and pattern detection performance of simulations with the constructive algorithm requires further investigation. The conditions investigated in this chapter do not test the ability of constructed neurons to reject other spike patterns. The next chapter further develops the constructive algorithm and applies it to a task of creating neurons that selectively respond to a repeating spike pattern hidden in high levels of background activity (Masquelier et al., 2008).

# Chapter 6

# STDP Simulation with Neuron Construction

This chapter presents an application of neuron construction in a reproduction of a simulated study of STDP (Masquelier et al., 2008). The past study (Masquelier et al., 2008) demonstrated that an additive STDP model can tune synapse weights to produce postsynaptic neuron spikes at the start of a repeating pattern concealed within a high level of stochastic presynaptic activity. This study has been reproduced and extended to include the constructive algorithm introduced in Section 5.4.4. Three processes for calculating synapse weights for neuron construction are evaluated: estimates of additive STDP, Gaussian distributions with parameters matching the estimated additive STDP, and estimates of positive additive STDP with normalisation. The results of simulating STDP with predefined neurons and constructed neurons is investigated.

## 6.1 Background

Spike-timing-dependent plasticity models (Caporale & Dan, 2008; Morrison et al., 2008) provide simple representations of neuroplasticity but may have sufficient accuracy for many computational studies. Models of STDP have been developed and studied for their correspondence to observations of biological neuroplasticity (Caporale & Dan, 2008), to examine the learning capabilities that they provide (Legenstein et al., 2005; Masquelier et al., 2008), and as a component in models of developing neural systems (for example, vision: Masquelier, 2012; Masquelier & Thorpe, 2007). In this thesis, a computer simulation performed to investigate the effects of an STDP model on the behaviours and states of neurons and synapses is referred to as an STDP simulation.

Past studies have shown that models of STDP are capable of tuning synapses to enable neurons to detect spike patterns hidden in high levels of stochastic activity (for example, Masquelier et al., 2008, 2009). The simulation of STDP resulting in a postsynaptic neuron tuned to the start of a hidden repeating pattern (Masquelier et al., 2008) is reproduced in this chapter. The capability of neurons to detect specific patterns in presynaptic activity is likely to be an important feature of the behaviour

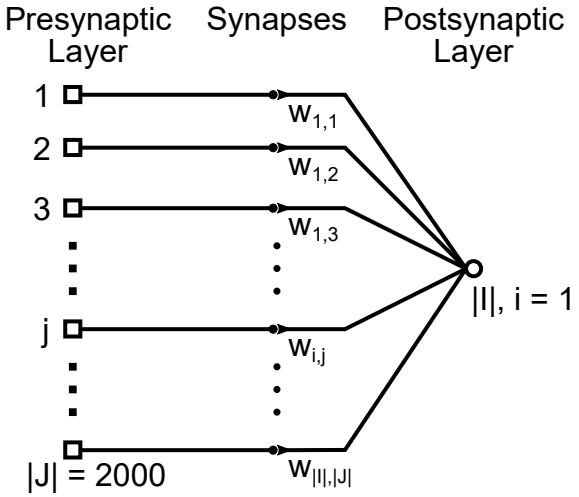


**Figure 6.1:** Simulations of hidden pattern detection without neuron construction are performed for a network with 2000 presynaptic neurons and a single postsynaptic neuron.

of many biological neural systems. The simulation conditions and results of the past study are summarised here.

The past study of STDP (Masquelier et al., 2008) conducted simulations with a single postsynaptic neuron that had synapses from 2000 presynpatic neurons (Figure 6.1). Presynaptic neuron activities were modelled as independent Poisson processes with half of the presynaptic neurons (1000) intermittently repeating a 50 ms segment of their activity (see Figure 6.2). A detailed description of the process and parameters for generating presynaptic neuron spike times is provided in Section 6.3.

Conditions were designed to increase the challenge of pattern detection by generating presynaptic activity with a consistently high average spike rate. The average spike rate across all presynaptic neurons was approximately 64 Hz and did not produce distinct fluctuations in average spike rate when displayed in 10 ms time bins (Figure 6.2).

In each simulation, the postsynaptic neuron was initialised with all synapse weights equal to $w_{i,j}(0) = 0.475$, which resulted in an initial period of indiscriminate spiking before the postsynaptic neuron tuned to the repeating pattern. The latency of the postsynaptic neuron spikes typically converged to an average time of less than 10 ms after the start of a repetition of the presynaptic spike pattern. A graphical representation of the postsynaptic spike latency (Figure 6.3) shows this initial random spiking and eventual tuning to the start of the repeating pattern. The past study (Masquelier et al., 2008) reported that 96 of 100 simulated trials produced a neuron that successfully detected the start of the hidden repeating pattern. The conditions for a neuron successfully tuning to a pattern were defined as having a true positive rate greater than 98%, zero false positives, and an average true positive spike latency of less than 10ms from the start of the pattern. The details of these learning success criteria are provided in Section 6.5.

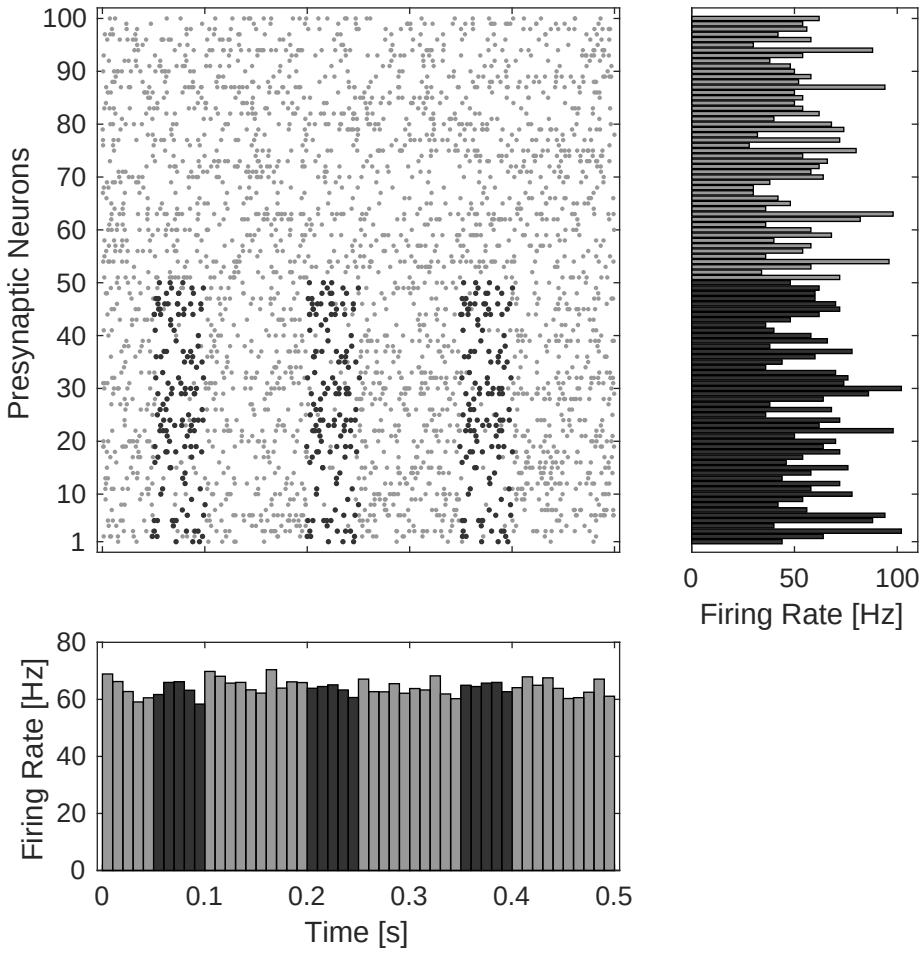


**Figure 6.2:** Example of the presynaptic neuron spike activity generated for the reproduced simulations. Top-left: Spike trains of neurons selected for pattern repetition (Presynaptic Neurons 1 to 50) and only random activity (Presynaptic Neurons 51 to 100). Top-right: Individual neuron spike rate averages over the 0.5 s window of activity. Bottom: Average spike rate over entire population in 10 ms time bins.

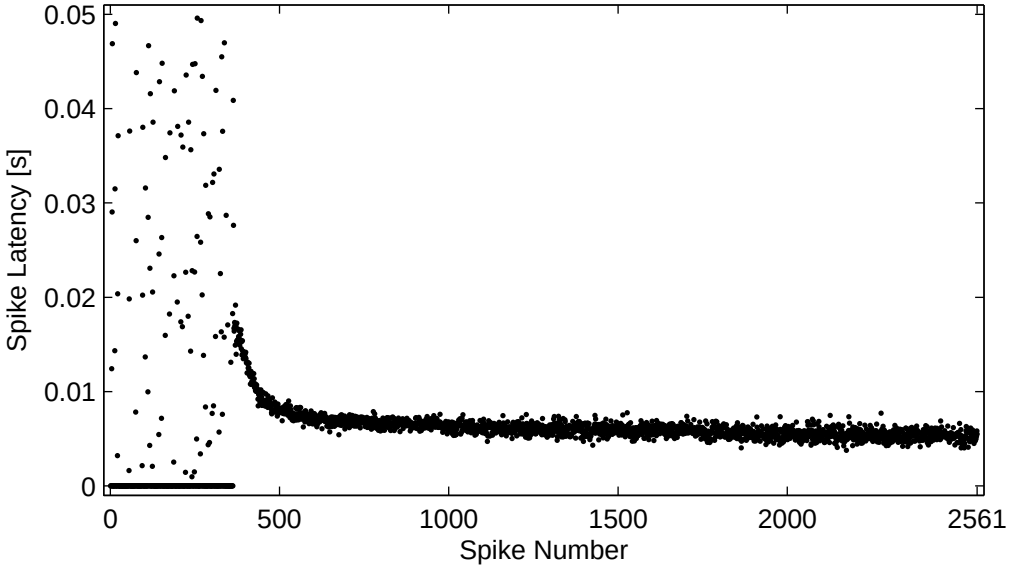


**Figure 6.3:** Example of the change in postsynaptic neuron spike latency relative to the start of the presynaptic spike pattern resulting from the tuning of synapses through additive STDP. Postsynaptic spikes outside of the 0.05 s presynaptic spike pattern time are marked as having a latency of 0 s. The postsynaptic neuron ceases to spike outside the repeating pattern time in fewer than 500 spikes and then the latency decreases to less than 0.01 s as the tuning of synapses converges.

## 6.2 Experiment Aims

A major goal of the developments of constructive algorithms and simulation expansion presented in this thesis is compatibility with computational studies of biological neural networks. The demonstration of simulations with and without neuron construction producing comparable results is evidence of the compatibility of neuron construction with simulations of STDP. Finding neuron construction to be compatible with STDP simulations is a step toward the application of constructive algorithms in models of neural systems involved in perception. The primary aim of the investigation presented in this chapter is to demonstrate simulations of STDP that achieve approximately equivalent results with and without neuron construction. A secondary aim is to investigate whether synapse weight calculation methods for neuron construction developed in this thesis produce an advantage in learning performance: faster learning or higher rates of success.

A past study and simulation of STDP resulting in hidden pattern detection (Masquelier et al., 2008) has been chosen for the present investigation of neuron construction and STDP. A preliminary aim of the presented experiments is to examine the relationship of postsynaptic neuron activation threshold in the STDP simulations to the learning success rate, the final synapse weights and the final latencies of spikes. The past study

(Masquelier et al., 2008) examined the learning success rates for changing initial weights and properties of the presynaptic spike activity, but did not present results from changing the neuron threshold. The results of STDP simulation and convergence are used in the design of synapse weight calculation processes in Chapter 7.

The simulation conditions have consistently high spike rates across the presynaptic neurons; therefore, the proxy neuron is not expected to be successful in discriminating the repeating pattern from background activity. Given a random latency of proxy neuron spikes relative to the repeating pattern, results have been collected to examine the relationship of proxy neuron spike latency to neuron learning success for different synapse weight calculation methods. Proxy neuron spikes during the repeating pattern are expected to potentiate synapses from pattern neurons and give constructed neurons an advantageous bias for detecting and learning a concealed pattern.

A fundamental goal of the development of constructive neural networks is the capability to automatically select and adjust the size of the neural network during operation. In general, the optimal size and structure of the neural network for a computational study is unlikely to be known in advance, for example, the number of concealed patterns can be unknown or change over time. The simulation conditions reproduced in this chapter, however, have a known number of spike patterns to be learned (one). Therefore, the capability of the constructive algorithm to automatically select the network size is not tested in the study presented in this chapter. A study investigating the advantages of using a constructive algorithm to automatically select the neural network size to detect multiple hidden repeating patterns is presented in Chapter 7.

## 6.3 Models and Simulation

Simulations performed with and without neuron construction use the event-driven parameter update process described in Section 5.4.3. The postsynaptic neuron model, synapse model, and additive STDP model implemented to reproduce the past study of STDP (Masquelier et al., 2008) were used in simulations presented in Chapter 5. For brevity, the descriptions of these models given in Chapter 5 are not reproduced here. The only difference from the models in the previous chapter is that the initial weight of synapses is $w_{i,j}(0) = 0.475$.

The presynaptic activity is pregenerated and then used to update the postsynaptic neuron model and synapses at presynaptic spike times. The simulations have 2000 presynaptic neurons spiking at an average rate of 64 Hz (Masquelier et al., 2008), giving an expected mean time between presynaptic neuron spikes of 7.8125 µs. This expected time between presynaptic neuron spikes is several orders of magnitude smaller than the neuron potential time constants ($\tau_m = 10$ms and $\tau_s = 2.5$ms). Therefore, event-driven simulation using presynaptic neuron spike times is assumed to be sufficiently accurate for postsynaptic neuron potential updates and activation.

The process for generating presynaptic neuron activity given in the past study (Masquelier et al., 2008) is reproduced. Independent Poisson processes with fluctuating rates are used to model the presynaptic neuron activity. Each presynaptic neuron, $j \in$

$J$, has an independent Poisson spiking rate, $r_j[t]$, that is time-variant. The generation of neuron spikes is performed in discrete time steps, $\Delta t = 1$ms. The probability of a neuron spiking in each time step is estimated as the neuron spike rate at that time multiplied by the time step length, $Pr(j,t) = r_j[t] \cdot \Delta t$, where the rate $r_j[t]$ is in hertz and the time step length $\Delta t$ is in seconds. In each time step a computer-generated random number with uniform probability is produced for each presynaptic neuron, $u_j[t] \in [0,1]$, and compared with the probability of a spike in that time step, $u_j[t] \leq Pr(j,t)$. Each neuron that returns a true comparison then spikes in that time step with an exact time generated with uniform probability in the 1 ms time step.

This method of generating Poisson neuron activity is performed independently for each of the 2000 presynaptic neurons. In each time step, each neuron spike rate has a randomly generated acceleration, $\Delta\Delta r_j[t]$, with uniform probability in the range $[-360,\ 360]$ Hz s$^{-1}$ ms$^{-1}$. The velocity of the spike rate of each presynaptic neuron is updated by the full value of the acceleration in each time step, $\Delta r_j[t] = \Delta r_j[t - \Delta t] + \Delta\Delta r_j[t]$. Before updating the spike rate, $\Delta r_j(t)$ is restricted to the range $[-1800,\ 1800]$ Hz s$^{-1}$. The spike rate for each neuron is then updated, $r_j[t] = r_j[t - \Delta t] + \Delta r_j[t] \cdot \Delta t$. All neuron spike rate variables are restricted to the range 0 Hz to 90 Hz.

The authors of the past study (Masquelier et al., 2008) chose this update process to produce smooth random fluctuations in the presynaptic neuron spike rates. During presynaptic neuron activity generation, spikes are also generated for any neuron that has not spiked in the past fifty time steps (50 ms). This prevents neurons from remaining silent for more than 50 ms and means that all presynaptic neurons selected for the repeating pattern will have at least one spike in the pattern. This process of varying the spike rate of each neuron and then generating presynaptic neuron spikes is performed for each time step up to 150 s.

The process for creating the final spike activity for presynaptic neurons that includes a repeating pattern operates on the 150 s of generated neural activity. The process for generating the repeating activity pattern can be summarised:

1. Input neuron spikes are generated with randomly fluctuating Poisson rates and a maximum silent period of 50 time steps for 150 s of simulation time;
2. The generated activity is divided into sequential 50 ms segments;
3. One 50 ms segment is selected at random as the base pattern of spikes for repetition;
4. The spike times of half the presynaptic neurons (designated as pattern neurons) in this segment are copied;
5. A 50 ms segment is selected at random, excluding segments that contain or are adjacent to a repetition of the pattern;
6. The selected segment has the spikes of pattern neurons replaced with the copied spike times with zero-mean, 1 ms-standard deviation Gaussian noise applied to

spike times;

7. Steps 5 and 6 are repeated until one quarter of all 50 ms segments have a noisy copy of the spike pattern;

8. An additional 10 Hz Poisson activity is then overlaid on the spike activity all presynaptic neurons;

9. The resulting 150 s of presynaptic activity is repeated two times to form a set of continuous 450 s spike trains.

This process of input neuron activity and hidden pattern generation is performed with different random number generator seed states for each simulation trial to provide variable simulation input conditions. An example of the presynaptic neuron activity generated through this process is provided in Figure 6.2. This generated presynaptic activity is then used to simulate the activity of the postsynaptic neuron.

## 6.4 Constructive Algorithms

The constructive algorithms implemented in simulations presented in this chapter are based on the algorithm for proxy neuron spike-triggered construction (Algorithm 8) presented in Chapter 5. Minor changes have been made to the constructive algorithm and are described here.

The high spike rate of input neurons causes the proxy neuron to spike indiscriminately and continuously trigger the construction of neurons. Multiple neurons are constructed in each simulation to provide more samples for statistical analysis. The maximum number of postsynaptic neurons constructed, $n_{\max} = 101$, was selected to provide samples for statistical analysis and to limit consumption of computational resources and memory. Once the number of postsynaptic neurons reaches 101 the proxy neuron is no longer updated nor spikes.

The performance of constructed neurons is compared for three processes of synapse weight calculation:

1. Specified iterations of additive STDP (Section 3.2.3.1);

2. Gaussian distributions calculated from additive STDP estimates;

3. Specified iterations of positive additive STDP with normalisation.

The first process is reproduced from Section 5.4.4 with the initial synapse weight taken from the proxy neuron synapse weights, $w_{\text{proxy},j} = 0.475$. The weights of synapses constructed during simulations using the first process are recorded, then the mean and variance of constructed synapse weights can be calculated for specified numbers of additive STDP updates.

The second process for calculating new synapse weights uses values of mean and variance resulting from the first process (estimating past additive STDP updates) to

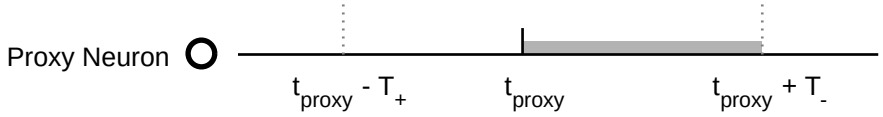


**Figure 6.4:** A proxy neuron spike time, $t_{\text{proxy}}$, provides an eligibility time window, $[t_{\text{proxy}} - T_+, t_{\text{proxy}} + T_-] = [t_{\text{proxy}} - 50.4\,\text{ms}, t_{\text{proxy}} + 67.4\,\text{ms}]$, for selecting presynaptic neuron activity for synapse construction. Neuron construction occurs at the end of the eligibility window, $t_c = t_{\text{proxy}} + 67.4\,\text{ms}$, for construction with synapse weights calculated from specified iterations of additive STDP and selected from Gaussian distributions. Neuron construction occurs at the proxy neuron spike time for construction with synapse weights calculated from specified iterations of positive additive STDP with normalisation. The proxy neuron has potential variables reset to resting values and does not resume parameter updates until after $t = t_{\text{proxy}} + T_-$. This pause in simulation updates of the proxy neuron is indicated by the grey bar between $t_{\text{proxy}}$ and $t_{\text{proxy}} + T_-$.

generate random synapse weights with a normal or Gaussian probability density function. Weights have been generated using the random number generator `randn` in MATLAB and then clipped to the synapse weight maximum and minimum values, $[0, 1]$. Synapse weights drawn randomly with Gaussian probability density functions are used as a control to provide an indication of whether observations can be explained by changes in the mean and variance of constructed synapse weights.

Simulations that calculate synapse weights as estimates of additive STDP or as approximations of Gaussian distributions delay neuron construction to the end of the eligibility window, $t_{\text{c}} = t_{\text{proxy}} + T_-$, $T_- = 2 \cdot \tau_-$ (see Figure 6.4). This ensures that these constructed neurons experience the same initial conditions of presynaptic activity and minimises this as a factor in the differences in outcomes.

The potentiation of synapses from presynaptic neurons with high activity during the repeating pattern increases the likelihood of the postsynaptic neuron responding. High presynaptic spike rates, however, also increase the likelihood of spike triplets, which can result in a net decrease in the synapse weight under additive STDP (when $|\Delta w_{i,j}^{(f,g)}| < |\Delta w_{i,j}^{(f,g+1)}|$ in Equations 3.3 and 3.4). Therefore, construction of synapse weights with an emphasis on potentiation may improve learning success in conditions with high presynaptic spike rates. The third synapse weight calculation process only considers presynaptic spikes eligible for the synapse weight calculation in the potentiating time window, that is, $\Delta t_{\text{proxy},j} \in [-3 \cdot \tau_+, 0)$. All other weights remain at the initial value, $w_{\text{proxy,j}} = 0.475$, until normalisation.

After the calculation of potentiated synapse weights for construction and before limiting the synapse weights to the maximum value, the synapses are normalised multiplicatively to keep the total or sum of input weights at the initial value,

$$w_{i,j}(t_{\text{c}}) = w^*_{i,j}(\Delta t_{\text{proxy},j}) \times 950 / \sum_{j \in J} w^*_{i,j}(\Delta t_{\text{proxy},j}), \tag{6.1}$$

where $w^*_{i,j}(\Delta t_{\text{proxy},j}) = w_{\text{proxy},j} + M \cdot A_+ \cdot \exp(\Delta t_{\text{proxy},j}/\tau_+)$ and 950 is the product of

the initial synapse weight and the number of synapses ($0.475 \times 2000$).

Simulations that calculate synapses weights as the normalised result of positive STDP updates introduce constructed neurons at the times of the proxy neuron spikes ($t_c = t_{proxy}$). Although neuron construction is not delayed, the proxy neuron updates are paused for the same duration as the first two weight calculation processes ($t > t_{proxy} + 2 \cdot \tau_-$).

For positive STDP with normalisation, presynaptic spikes that occur after the postsynaptic neuron are assumed to be insufficiently correlated with the postsynaptic activity to have a significant effect on the final synapse weight. This assumption may be considered an extension of the assumption that presynaptic spikes outside the eligibility window are not correlated with the postsynaptic activity. Models of plasticity may include homeostatic mechanisms (Watt & Desai, 2010); the multiplicative normalisation performed here is not based on a specific mechanism but is a simple process for ensuring the initial postsynaptic spike rate is approximately equivalent to predefined neurons in non-constructive simulations.

See Figure 6.9 in Section 6.6.2 for histograms of the synapse weights resulting from each of the synapse weight calculation methods and a comparison with synapse weights resulting from non-constructive STDP simulations. When neurons are first constructed they are not tuned to the repeating pattern. To investigate the performance of constructed neurons in STDP simulations, the constructed neurons are then simulated with synapses adapted using additive STDP.

## 6.5 Data Collection and Analysis

Data has been collected and analysed to investigate the effects of varying the postsynaptic neuron activation threshold and varying the synapse weight calculation methods for neuron construction. A focus of the data collection and analysis has been the criteria for learning success taken from the past study of STDP (Masquelier et al., 2008). The reproduced criteria for learning success are based on the following definitions:

- A true positive is counted for each repetition of the pattern that has one or more postsynaptic neuron spikes inside the 50 ms pattern duration.
- The true positive percentage is calculated by dividing the number of true positives by the number of pattern repetitions in a given period of the simulation.
- A false positive is counted for every postsynaptic neuron spike outside of the 50 ms duration of any repetition of the pattern.
- Spike latency is measured for true positives as the time from the start of the pattern to the first postsynaptic neuron spike.

A postsynaptic neuron achieved learning success if in the last 150 s of the simulation: the true positive rate was greater than 98%, the mean spike latency was less than 10 ms after the pattern start, and zero false positives were recorded. Postsynaptic neuron spike

times and pattern times have been recorded in all simulations to determine individual neuron learning success.

The first phase of the investigation examined the performance of non-constructive simulations of STDP for varied postsynaptic neuron activation thresholds (an explanation of the neuron threshold is provided in Section 5.4.3). The non-constructive simulations reproduced the network structure of the past study (Figure 6.1) but simulated multiple postsynaptic neurons with thresholds from 500 to 600 in increments of 5. The postsynaptic neuron spike times were recorded for each of 200 sets of generated presynaptic activity and compared with the times of presynaptic spike pattern repetition for the calculation of neuron learning success. Synapse weights were recorded prior to each of the first 21 postsynaptic neuron spikes and at the end of the simulation. The final total input weight, the number of potentiated synapses and the mean spike latencies were found for each postsynaptic activation threshold and examined for trends and correlation with changes in learning performance.

The second phase of the investigation examined the performance of different synapse weight calculation methods in constructive simulations (Section 6.4). The presynaptic activity generated in 100 of the non-constructive simulations was replicated for each set of constructive simulations. A single neuron threshold value, selected from the non-constructive simulations, was used for the proxy neuron and constructed postsynaptic neurons.

The first set of constructive simulations performed used specified iterations of additive STDP ($M$ = 1, 5, 10, 15 and 20) and the presynaptic spike times observed around the proxy neuron spike time to calculate synapse weights (Section 3.2.3.1). The synapse weights recorded from the first set of constructive simulations were used to calculate Gaussian models to produce synapse weights for the second set of constructive simulations. The third set of constructive simulations were performed with the synapse weights calculated using positive STDP with normalisation to the initial total weight (Equation 6.1). In addition to recording the postsynaptic neuron spike times and the synapse weights, the spike times of the proxy neurons (and the times of neuron construction) were recorded.

The overall rates of neuron learning success of constructed neurons were calculated and compared with those of non-constructive simulations performed with the same presynaptic neuron activity. The rates of neuron learning success were also examined for neurons constructed with varying proxy neuron spike times relative to the times of repeating patterns.

The effect of performing neuron construction with synapse weights calculated assuming multiple past iterations of STDP has been speculated to improve the speed of learning. The process of neuron tuning observed in non-constructive simulations begins with a period of rapid indiscriminate postsynaptic spiking prior to the neuron becoming responsive only to the repeating pattern. Therefore, the speed of learning has been related to the number of false positive spikes of the postsynaptic neuron in its early activity. The total number of false positives of postsynaptic neurons is recorded in simulations and compared across the different simulation conditions.

The random generation of presynaptic neuron activity for simulations introduces variability between trials. Statistics of these simulations are presented visually using box plots that indicate the median result and the first and third quartile. Same-trial comparisons of early learning performance and learning success of initial postsynaptic neurons and constructed postsynaptic neurons are also performed to examine inter-trial variability.

The simulations have been performed in MATLAB R2014a and R2016a.

## 6.6 Simulation Results

Simulation results are divided into sections examining:

1. The performance of neurons in non-constructive simulations with varied activation thresholds
2. The constructed synapse weights and Gaussian distributions
3. The learning success rates of constructive and non-constructive simulations
4. The early false positives or speed of learning for increasing $M$ (estimated STDP updates)

### 6.6.1 Neuron Activation Threshold

The reproduction of the original, non-constructive simulation studying STDP was investigated for sensitivity in learning success to the postsynaptic neuron activation threshold. A threshold $\theta = 500$ (see Equation 5.2) was used in the past study (Masquelier et al., 2008) and reported a 96% success rate in 100 simulations. The reproduced simulations found a threshold of 500 gave the success rate 89% in 200 simulations. The rate of learning success increased for increasing neuron activation threshold up to a maximum of 96% at the threshold of 530 (Figure 6.5). Above the threshold of 530 the rate of learning success decreases for increasing neuron potential threshold.

The change in statistics of the total input weights to postsynaptic neurons (Figure 6.6) showed small variations without strong trends for the evaluated neuron thresholds. The initial total input synapse weight for each postsynaptic neuron was $2000 \times 0.475 = 950$. All activation thresholds produced neurons that experienced a significant decrease from the initial total input weight, with median values falling in the range $[363.7, 387.9]$. The total weights were below the activation threshold; therefore, the postsynaptic neurons must require multiple spikes from selected presynaptic neurons in a short space of time.

The statistics for the number of potentiated synapses (weight greater than 0.9; Figure 6.7) also showed small variations without strong trends for the changes in activation threshold. The median numbers of potentiated synapses for each postsynaptic neuron fall in the range $[306, 334]$. Therefore, the majority of the total input weight is held by the small fraction of potentiated synapses. This agrees with past studies (for example,

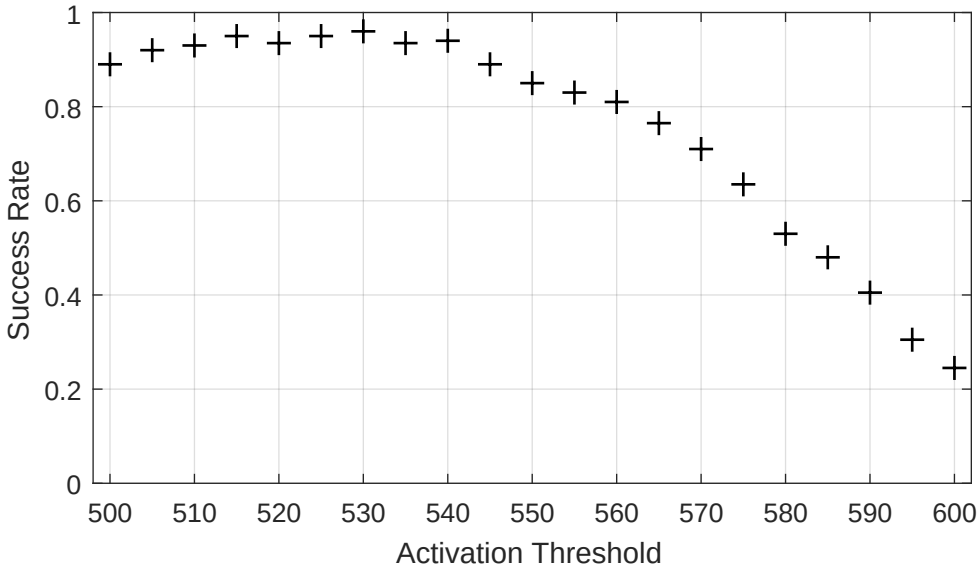


**Figure 6.5:** The rate of postsynaptic neuron learning success for non-constructive simulations with increasing postsynaptic neuron activation threshold. Success-rates are normalised to the range $[0, 1]$ and calculated from 200 simulations.

Song et al., 2000) that found neurons tuned with additive STDP resulted in bimodal distributions of synapse weights with concentrations around the maximum ($w_{\max} = 1$) and minimum ($w_{\min} = 0$).

Phases of postsynaptic neuron activity and learning seen in Figure 6.2 and described in the past study (Masquelier et al., 2008) correspond to changes in the total input weight. The first phase of postsynaptic activity is rapid indiscriminate spiking. This indiscriminate activity is the result of the high initial total input weight, $\sum w_{i,j}(0) = 950$. The parameters of the STDP curve (Table 5.2) have a bias toward synaptic depression, decreasing the total input weight as the postsynaptic neuron spikes.

The second phase of postsynaptic activity occurs when the total input weight has decreased to the point where the postsynaptic neuron ceases to spike in response to the background activity. At this critical point, the weight of synapses from presynaptic neurons that are active in the repeating pattern may be sufficient to activate the postsynaptic neuron, otherwise the postsynaptic neuron will cease to respond to input entirely. The start of selective spiking initiates a phase of decreasing postsynaptic spike latency relative to the start of the repeating pattern. The consistent spiking of the postsynaptic neuron during the repeating pattern results in consistent potentiation of synapses from presynaptic neurons that are active early in the pattern. Synapses from other presynaptic neurons continue to trend towards depression. This produces the shift in synapse weights to a bimodal distribution.

The final phase of postsynaptic activity is seen as a stabilisation in the spike latency. The stabilisation in the spike latency occurs when the synapse weights have settled

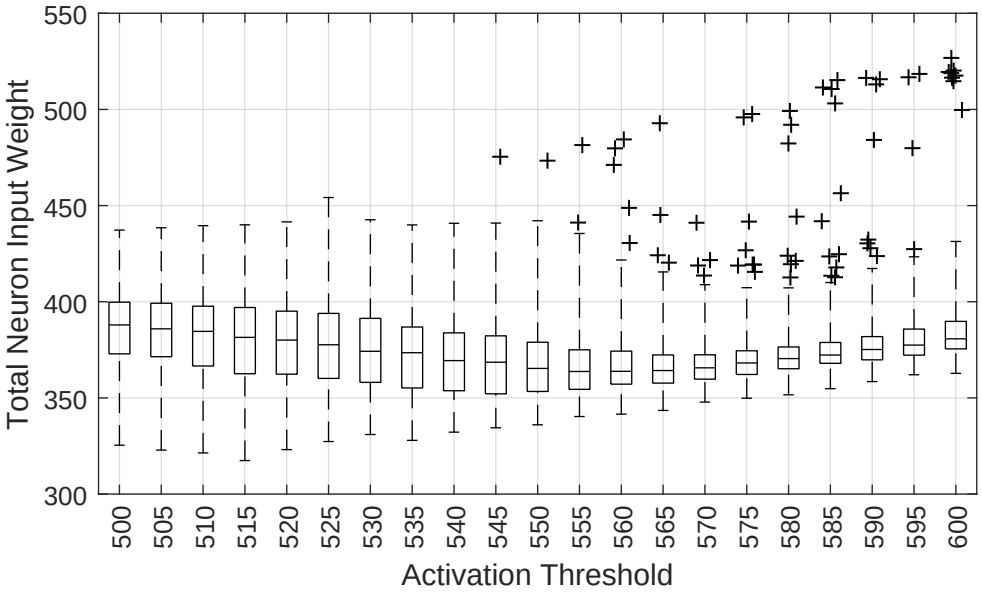


**Figure 6.6:** The final total synapse weight for increasing postsynaptic neuron activation threshold. The box plot represents the first quartile ($q_1$), the median, and the third quartile ($q_3$) as the bottom, middle and top lines of the box respectively. The whiskers extend to the furthest data point within $q_1 - 3 \cdot (q_3 - q_1)$ and $q_3 + 3 \cdot (q_3 - q_1)$. Outliers beyond the maximum whisker range are represented as crosses (+).

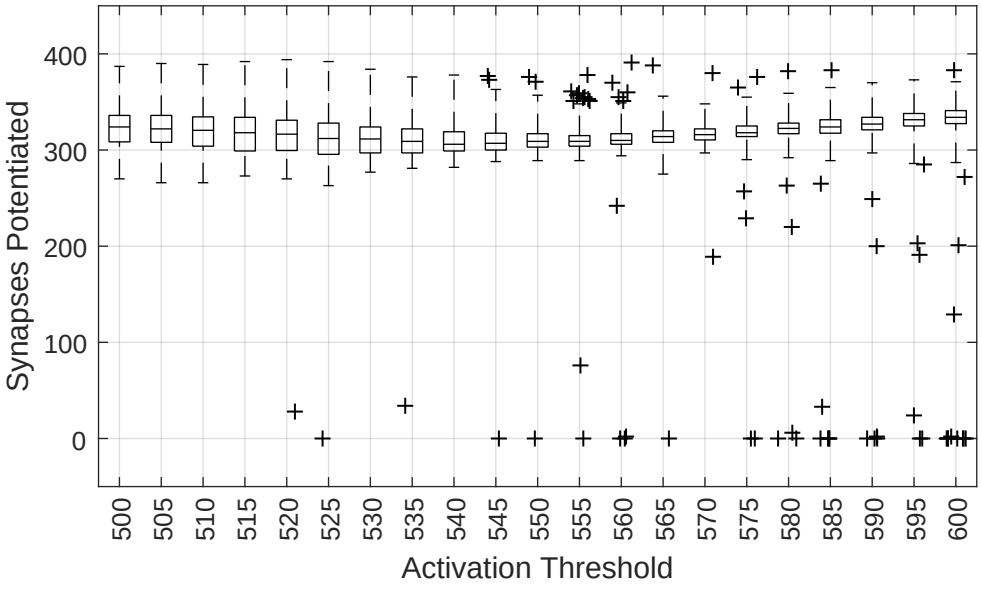


**Figure 6.7:** The final number of potentiated synapses ($w_{i,j} > 0.9$) to each postsynaptic neurons for increasing activation threshold. The box plot uses the same format for statistics as Figure 6.6.

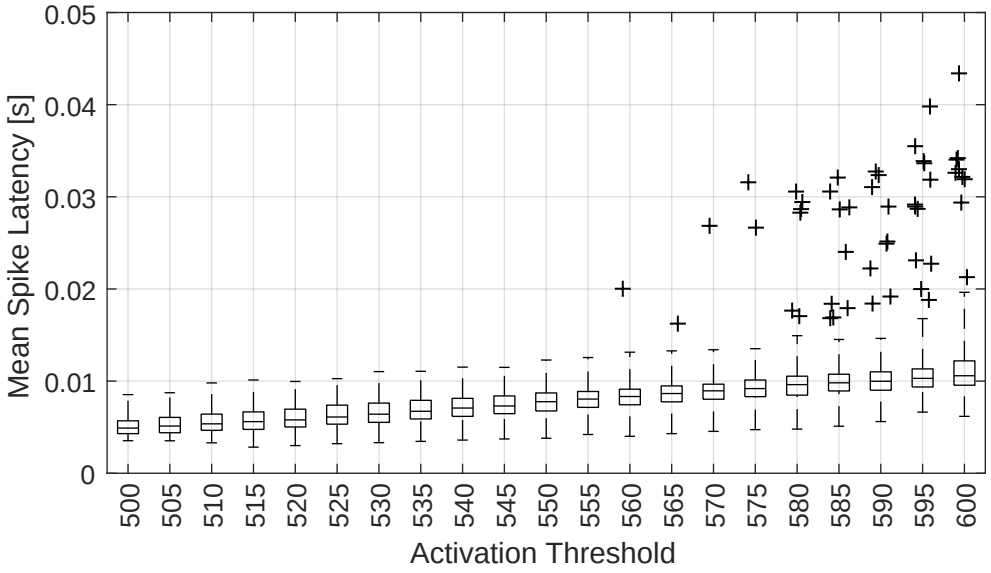


**Figure 6.8:** The average latency of postsynaptic spikes relative to the start of repeating patterns during the test period (last 150 s) for increasing postsynaptic neuron activation threshold. The box plot uses the same format for statistics as Figure 6.6. Postsynaptic neurons with a true positive rate of less than 98% or any false positives have been excluded from the statistics presented in this figure.

into a bimodal distribution. Random fluctuations in the postsynaptic spike times and synapse weights may continue indefinitely; however, the random fluctuations have a low probability of causing the subsequent detuning of a postsynaptic neuron for the timescales simulated.

The total input weights and rates of synapse potentiation recorded show signs of postsynaptic neurons failing to tune to the repeating pattern. High final values of total input weight (Figure 6.6) can result from the postsynaptic neuron ceasing to spike, which causes the depression of synapses to cease prematurely. Zero or low numbers of potentiated synapses (Figure 6.7) can also result from the postsynaptic neuron ceasing to spike, which causes the potentiation of synapses to cease prematurely.

The mean spike latencies (Figure 6.8) show that a significant contributor to the steep reduction in the number of successful neurons is the mean spike latency increasing to more than 10 ms. The median spike latency increases from 4.90 ms at a threshold of 500 to 10.58 ms at a threshold of 600. At an activation threshold of 600 the conditions for true positive rate greater than 98% and zero false positives were satisfied in 156/200 simulations (78%). The addition of the mean spike latency criterion for learning success reduced the success rate to 49/200 simulations (24.5%) for the same activation threshold.

**Table 6.1:** Gaussian probability distribution parameters from additive STDP estimation. Parameters for estimated weights and weights initialised from Gaussian models are taken from $2 \times 10^7$ synapses across 100 simulations with 100 postsynaptic neurons.

| | Estimated Weights | | | Gaussian Parameters | |
|---|---|---|---|---|---|
| STDP Iterations | Mean, $\bar{x}$ | Variance, $\bar{s}$ | Gaussian Model | Mean, $\bar{x}$ | Variance, $\bar{s}$ |
| 1 | 0.4729 | 0.01103 | 1 | 0.4729 | 0.01103 |
| 5 | 0.4647 | 0.05513 | 2 | 0.4646 | 0.05513 |
| 10 | 0.4543 | 0.1103 | 3 | 0.4543 | 0.1103 |
| 15 | 0.4440 | 0.1654 | 4 | 0.4441 | 0.1648 |
| 20 | 0.4337 | 0.2201 | 5 | 0.4354 | 0.2143 |

### 6.6.2 Constructed Synapse Weights

Three methods for calculating synapse weights for neuron construction have been described in Section 6.4. The first method calculates the synapse weights assuming that the observed pattern of presynaptic spike times has repeated a specified number of times and calculates the corresponding additive STDP (see equations in Section 3.2.3.1). The second method, drawing synapse weights from Gaussian distributions, is based on the mean and standard deviation of synapse weights calculated from the first method. The third method assumes that only the presynaptic spikes that produce potentiation were repeated in past synapse updates but that the overall input weight remained constant (implemented with normalisation).

The mean and standard deviation of synapse weights constructed assuming specified iteration numbers of relative spike times and additive STDP were found (Table 6.1). The first neuron is constructed before many presynaptic neurons have spiked and results in anomalous synapse weights; therefore, the first neuron constructed in simulations has been excluded from the synapse weight results.

Histograms of the synapse weights after 10 and 20 postsynaptic spikes in non-constructive STDP simulations are presented with the histograms of the synapse weights constructed for the same number of iterations (Figure 6.9). The histograms of the synapse weights resulting from non-constructive simulations of plasticity have been collected from 100 simulations of 1 postsynaptic neuron with 2000 synapses, giving $2 \times 10^5$ synapses in each histogram. All histograms of synapse weights produced through construction have been collected from 100 simulations with 100 constructed neurons each with 2000 synapses, giving $2 \times 10^7$ synapses in each histogram.

Non-constructive simulations had a single postsynaptic neuron, which would initially spike rapidly in response to the high level of presynaptic neuron activity. The average time between the first 21 postsynaptic neuron spikes was 0.0172 s; the $21^{\text{st}}$ postsynaptic spike occurred at an average simulation time of 0.3540 s. Synapse weights from non-constructive simulations were recorded prior to the STDP update of the following postsynaptic spike. That is, non-constructive simulations have synapse weights

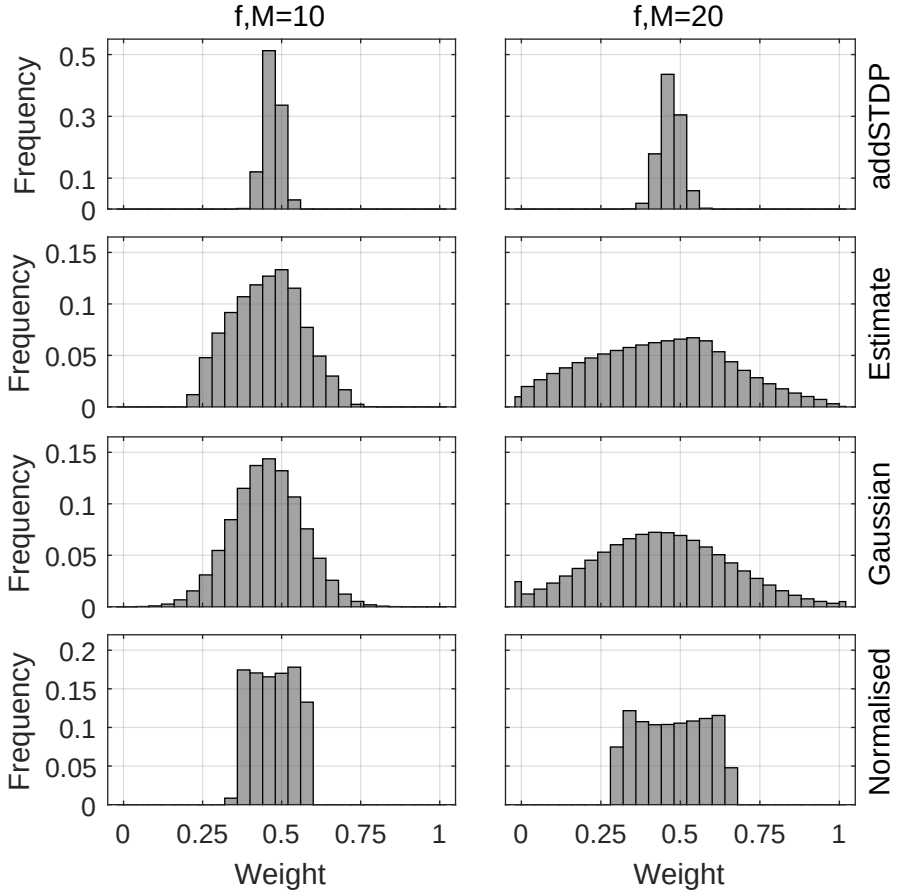


**Figure 6.9:** Histograms of individual synapse weights resulting from non-constructive simulations of STDP and from the construction of synapses. Bins have a width of 0.04 with separate bins for the weight limits 0 and 1; the frequency indicates the decimal fraction of all synapse weights within that bin range. Histograms are organised in columns for 10 and 20 simulated postsynaptic spikes ($f$) and equal past repetitions of spike times ($M$) assumed in constructive calculations. Methods of producing synapse weights are organised in rows: Top, non-constructive simulation of additive STDP; Second, constructed additive STDP estimates for $M$ past iterations; Third, Gaussian distributions based on constructed additive STDP estimates; and Bottom, constructed estimates of positive STDP for $M$ past iterations with normalisation.

recorded just prior to the $11^{\text{th}}$ postsynaptic spike update for $f = 10$ and prior to the $21^{\text{st}}$ postsynaptic spike update for $f = 20$. Early activity of the predefined postsynaptic neurons may occur during a repetition of the spike pattern; however, the postsynaptic neuron spike latency relative to the spike pattern start varies and will be unlikely to have experienced a precise repetition of the relative spike times.

The synapse weights constructed as estimates of repeated additive STDP can be seen to significantly over-estimate the spreading seen in the non-constructive simulations of STDP for equivalent numbers of postsynaptic neuron spikes (Figure 6.9). This is not unexpected: similar results have been shown in Chapters 4 and 5 and are the result of assuming no noise in the repetition of relative spike times. When presynaptic activity is stochastic, relative spike times do not repeat exactly. Instead, plasticity updates can cancel and cause a slow spreading of the weight distribution.

The histograms of synapse weights (Figure 6.9) show the results of synapse weight construction using Gaussian probability functions and synapse weights calculated from the estimation of additive STDP potentiation with normalisation. Parameters for normal distributions are calculated from synapse weight estimates to use in expanding simulations with synapse weights drawn from Gaussian probability distributions as a control (Table 6.1). A number of synapses weights drawn from the normal distributions exceed the minimum and maximum weights. These weights are clipped to the limits and have a small influence on the final synapse weight distributions created. The final distributions of all synapse weights drawn from the normal distributions modelled on constructed estimates of 15 and 20 iterations of additive STDP were $\bar{x}_{15} = 0.4441$ (a difference of $1.8582 \times 10^4$), $\bar{s}_{15} = 0.1648$ (a difference of $-5.8419 \times 10^4$), $\bar{x}_{20} = 0.4354$ (a difference of $1.6519 \times 10^3$), and $\bar{s}_{20} = 0.2143$ (a difference of $-5.7999 \times 10^3$).

### 6.6.3 Neuron Learning Success

The performance of postsynaptic neurons has been evaluated using the criteria for learning success provided in the past study (Masquelier et al., 2008). Constructive simulations were performed for one hundred presynaptic neuron spike trains and compared against the performance of non-constructive simulations of a postsynaptic neuron for the same presynaptic input (Table 6.2). Two observations about the overall learning success rate results are of particular interest: 1) simulations with neuron construction achieve high rates of learning success that are comparable to non-constructive simulations, and 2) increasing iteration numbers in STDP estimates for neuron construction tends to reduce the rate of learning success.

The first observation confirms the primary aim of this study: the simulations of STDP achieve comparable neuron behaviour (learning success) at close to the same rate whether the neurons are constructed or initialised in a non-constructive simulation. This is evidence that constructive algorithms can be compatible with simulations of biological neural network that incorporate STDP.

There are several factors that contribute to the second observation, that is, the reduction in neuron learning success rate of STDP estimates for increasing assumed past iterations of spike times:

**Table 6.2:** Learning success rates of postsynaptic neurons simulated with one-hundred sets of presynaptic neuron spikes trains. Non-constructive simulations were performed with one postsynaptic neurons with all weights initialised to $w_{i,j}(0) = 0.475$. Constructive simulations (STDP Estimates, Gaussian Models, and Normalised STDP Estimates) produced one-hundred postsynaptic neurons in each simulation for a total of $10^5$ postsynaptic neurons.

| Iterations, $M$<br>Spikes, $f$ | Non-constructive | STDP Estimates | Gaussian Models | Normalised STDP Estimates |
|---|---|---|---|---|
| 0 | 95.00% | – | – | – |
| 1 | – | 94.06% | 94.49% | 94.37% |
| 5 | – | 94.40% | 94.33% | 94.02% |
| 10 | – | 93.20% | 94.34% | 93.94% |
| 15 | – | 91.59% | 93.10% | 93.64% |
| 20 | – | 89.32% | 91.50% | 93.20% |

1. The influence of the proxy neuron spike time relative to activity in the repeating pattern on the calculation of synapse weights.

2. The relative timing of the early spikes of constructed postsynaptic neurons and the resulting STDP updates.

3. The overall neuron input weight and the duration of early tuning in response to indiscriminate postsynaptic spiking.

The first two items can be examined by viewing the rates of neuron success for changing proxy neuron spike time relative to the start of a pattern repetition (Figures 6.10, 6.11 and 6.12).

There is a substantial decrease in the learning success rate (down to 68.30% for $M = 20$) for neurons that are constructed using additive STDP estimates from a proxy neuron spike in the first 10 ms of the repeating pattern (Figure 6.10). Presynaptic neuron spikes in the repeating pattern that occur after the proxy neuron spike are in the depressing region of the STDP curve. The result can be a magnified depression of synapses from active pattern neurons. This increases the chance of the neuron failing to maintain sufficient input weight from pattern neurons to remain active as subsequent STDP updates decrease the total input weight.

A similar drop in the learning success rate is observed in neurons constructed using normalised positive STDP estimates (Figure 6.12). In this method of synapse weight calculation, depression of synapses occurs through the normalisation process (Equation 6.1). Synapses that receive a low or zero positive STDP update have synapses weights scaled to lower than the initial value. The maximum silent period for the initial generation of presynaptic spikes is 50 ms (Section 6.3). However, if a neuron is inactive prior to an inserted repeating pattern and also inactive early in the repeating pattern, the combined neuron inactivity can last for longer than 50 ms.

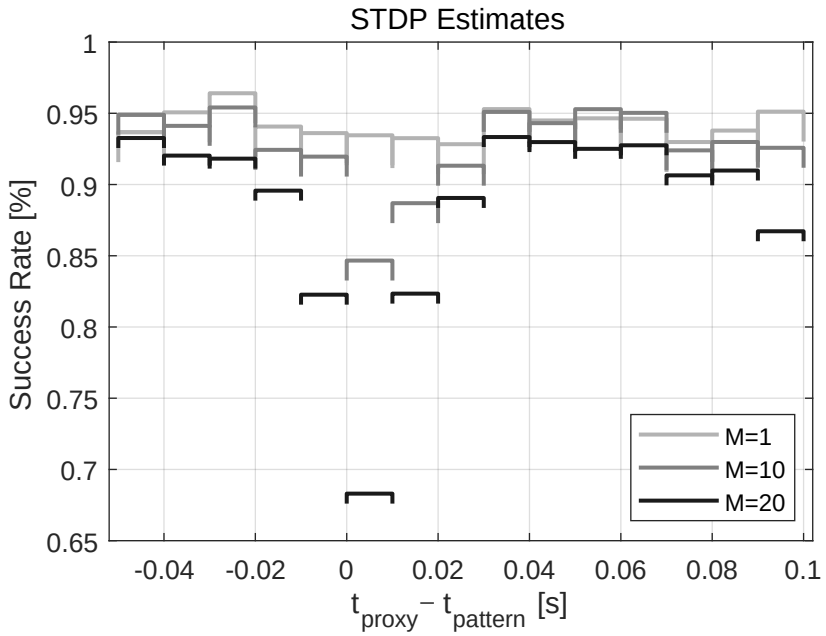


**Figure 6.10:** The rates of learning success of neurons constructed with additive STDP estimates from proxy neuron spikes, $t_{\text{proxy}}$, relative to the start time of a repeating pattern, $t_{\text{pattern}}$. Synapse weights are calculated to estimate additive STDP for $M = 1$, 10 and 20 iterations of activity. The repeating presynaptic spike pattern ends at 0.05 s. The proxy neuron spike time determines the potentiation or depression of constructed synapses using the additive STDP model. Neurons are collected into 10 ms time-bins, for example, [0 ms, 10 ms), with numbers of neurons in bins ranging from 467 to 530.

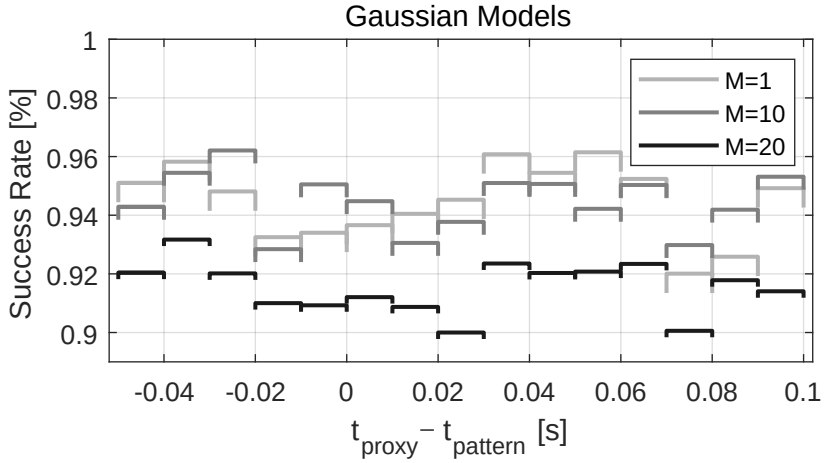


**Figure 6.11:** The rates of learning success of neurons constructed using Gaussian models of synapse weight distributions and proxy neuron spikes, $t_{\text{proxy}}$, relative to the start time of a repeating pattern, $t_{\text{pattern}}$. Synapse weights are drawn from Gaussian models of synapse weights estimating additive STDP for $M = 1$, 10 and 20 iterations of activity.

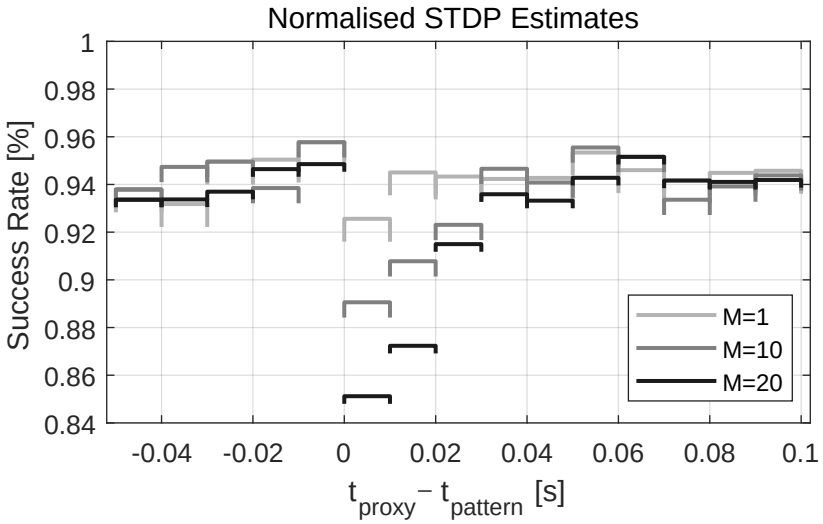


**Figure 6.12:** The rates of learning success of neurons constructed using normalised STDP estimates from proxy neuron spikes, $t_{\text{proxy}}$, relative to the start time of a repeating pattern, $t_{\text{pattern}}$. Synapse weights are calculated to estimate positive additive STDP for $M = 1$, 10 and 20 iterations of activity and then normalised.

Normalised positive STDP estimation depresses the synapses of inactive presynaptic neurons. Therefore, the presence of presynaptic neurons with long silent periods produces a surge in the number of synapses from pattern neurons that remain at the initial weight. The lack of neuron activity early in the pattern, however, does not preclude the neuron from having a high rate of activity late in the pattern segment. The depression of these synapses reduces the likelihood of the neuron maintaining sufficient input weight from neurons with high activity late in the repeating pattern to achieve learning success. A similar surge in synapses that remain at the initial weight (prior to normalisation) appears after the repeating pattern ends; however, this does not produce the same change in the learning success rate as the inactive neurons are those with low activity during the repeating pattern.

The synapse weights drawn from Gaussian distributions are independent of the presynaptic activity during construction. There are no strong trends in the learning success rate relative to the proxy neuron spike time and repeating pattern time that triggers neuron construction (Figure 6.11). Nevertheless, peaks appear in the learning success rates for proxy neuron spike times relative to repeating pattern start times, $t_{\text{proxy}} - t_{\text{pattern}}$, within ranges $-0.05\,\text{s}$ to $-0.02\,\text{s}$ and $0.03\,\text{s}$ to $0.07\,\text{s}$. These peaks may be explained as a result of postsynaptic neurons constructed at these times having an increased probability of spiking during a following pattern repetition.

A visual representation of the relationship between the proxy neuron spike, postsynaptic neuron construction and the first postsynaptic neuron spike relative to repeating patterns is provided in Figure 6.13. The construction and simulation of a postsynaptic neuron is delayed until $t_{\text{c}} = t_{\text{proxy}} + 2 \cdot \tau_{-} = t_{\text{proxy}} + 0.0674\,\text{s}$. The average delay from

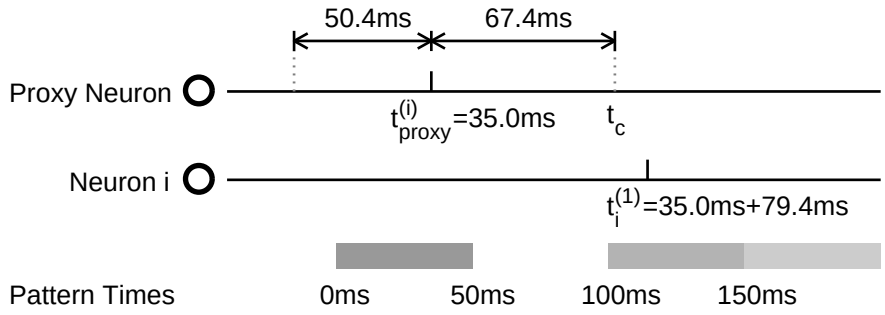


**Figure 6.13:** Example of the sequence of timing of a proxy neuron spike, neuron construction, and first postsynaptic neuron spike relative to a repetition of the pattern in presynaptic activity. The presynaptic activity pattern repeated from 0 ms to 50 ms is indicated by a grey bar. Subsequent grey bars indicate possible later pattern repetitions. The process for presynaptic activity generation does not allow consecutive pattern repetitions. The $i$th proxy neuron spike occurs $t^{(i)}_{\text{proxy}} = 35\,\text{ms}$ relative to the presynaptic pattern starting at $t_{\text{pattern}} = 0\,\text{ms}$. The proxy neuron spike triggers the construction of postsynaptic neuron $i$ that first spikes 79.4 ms after the proxy neuron.

the neuron construction to the first postsynaptic spike is between 0.0108 s (Gaussian model for $M = 1$) and 0.0120 s (Gaussian model for $M = 20$). The average total time from proxy neuron spike to the first spike of the constructed neuron is between 0.0782 s and 0.0794 s. Therefore, proxy neuron spikes in the bins from $-0.05$ s to $-0.02$ s result in constructed postsynaptic neurons that first spike in the range 0.03 s to 0.05 s. This results in the synapses of neurons active in the repeating pattern receiving the first potentiating update and improving the chances that these synapses will have sufficient weight for successful learning.

The peaks in learning success rate for proxy neuron spikes in the bins from 0.03 s to 0.07 s may be the result of postsynaptic neuron spikes first occurring in the range of the next spike pattern repetition in 0.11 s to 0.15 s. Given that the procedure for generating the presynaptic activity does not allow the repeating pattern to occur in consecutive segments, pattern repetitions have an increased probability of being separated by one 0.05 s segment of presynaptic activity. The dip in learning success rate in the bin $[0.07\,\text{s}, 0.08\,\text{s})$ may be due to depression of synapses from pattern neurons when the next pattern repetitions starts two segments later at 0.15 s.

Reductions in learning success rate for additive STDP estimates of increasing iterations and the related Gaussian distributions are also correlated with decreasing mean (and total) input weight (Table 6.1). The past study (Masquelier et al., 2008) indicated that reducing the initial weights (and total weight of synaptic input) in simulations results in a decline in the rate of neuron learning success. The graphical depiction of this decrease indicates that for an initial synapse weight of 0.425 the learning success had reduced to 87/100. This is likely to have contributed to the decline in learning success rates for neurons constructed with STDP estimates and Gaussian-distributed synapse

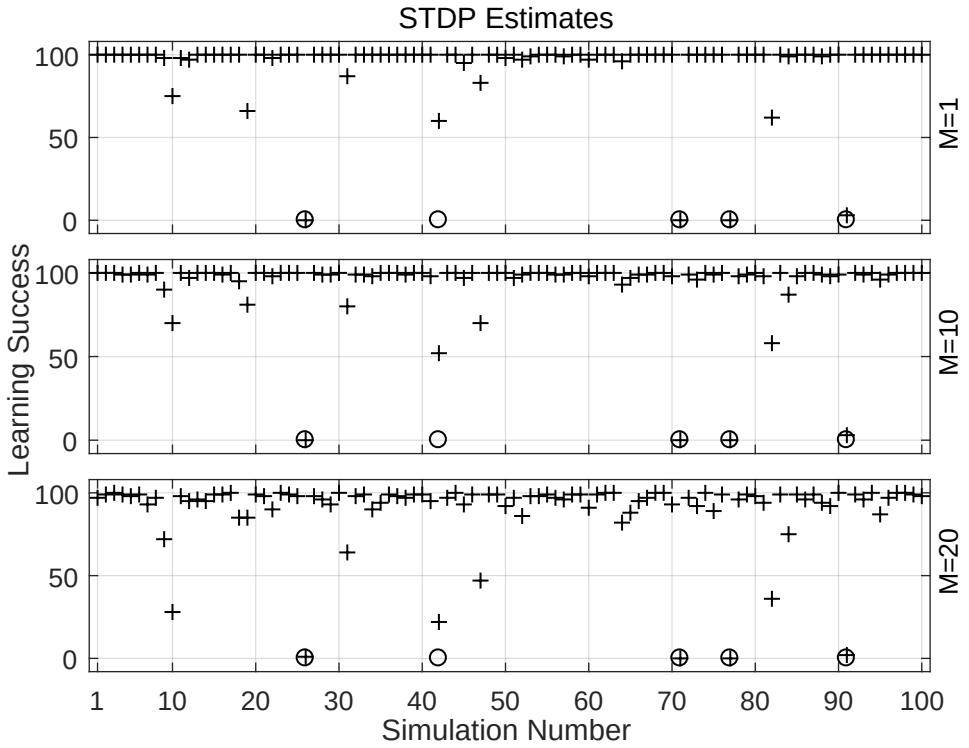


**Figure 6.14:** The number of constructed postsynaptic neurons that achieve learning success for each simulation with synapse weights calculated using additive STDP estimation. Data was collected from 100 simulations with 100 neurons constructed for each set of presynaptic neuron spike trains. The number of constructed neurons that achieve learning success are marked with crosses (+) and non-constructive simulations that did not achieve learning success are marked with circles (∘).

weights. The normalisation of STDP potentation estimates results in a constant total neuron input weight for all iteration numbers; this corresponds with smaller decreases in learning success rate.

Examining the learning success rates for individual sets of generated presynaptic activity indicate that the stochastic initialisation of the repeating pattern has a significant impact (Figures 6.14, 6.15 and 6.16). Each method for calculating synapse weights (STDP estimates, Gaussian models, and normalised positive STDP estimates) achieved learning success for all constructed neurons in 77/100 simulations for $M = 1$. Increasing the number of assumed iterations, $M$, reduced the rate of neuron learning success for all synapse weight calculation methods, but most severely for estimates of additive STDP that include depression. For $M = 20$, all neurons achieve learning success in 14/100 simulations with synapse weights calculated using estimates of additive STDP, 43/100 simulations with synapse weights drawn from Gaussian models, and 71/100 simulations with synapse weights calculated using estimates of positive additive STDP with normalisation.

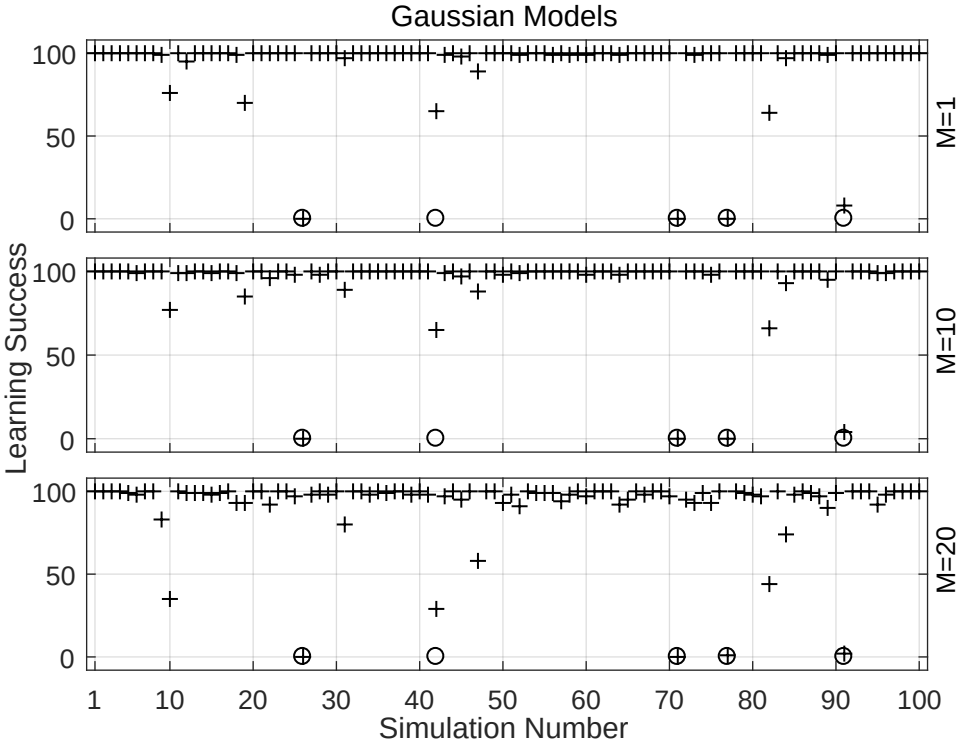


**Figure 6.15:** The number of constructed postsynaptic neurons that achieve learning success for each simulation with synapse weights drawn from Gaussian distributions modelled on additive STDP estimates. Data was collected from 100 simulations with 100 neurons constructed for each set of presynaptic neuron spike trains. The number of constructed neurons that achieve learning success are marked with crosses (+) and non-constructive simulations that did not achieve learning success are marked with circles (∘).

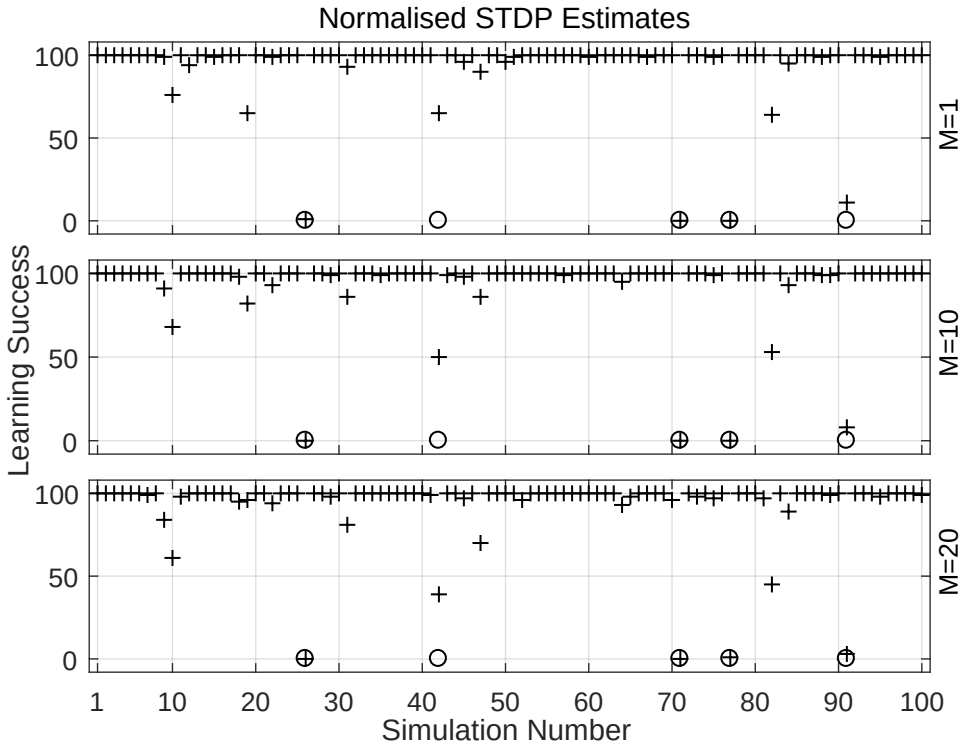


**Figure 6.16:** The number of constructed postsynaptic neurons that achieve learning success for each simulation with synapse weights calculated using positive additive STDP estimates with normalisation. Data was collected from 100 simulations with 100 neurons constructed for each set of presynaptic neuron spike trains. The number of constructed neurons that achieve learning success are marked with crosses (+) and non-constructive simulations that did not achieve learning success are marked with circles (∘).

**Table 6.3:** The number of constructive and non-constructive simulations (out of one-hundred) that produced one or more neurons that achieved learning success. Constructive simulations produced one-hundred postsynaptic neurons with varying synapse weight initialisation methods. Non-constructive simulations were performed with one postsynaptic neuron with all weights initialised to $w_{i,j}(0) = 0.475$.

| Iterations, $M$ Spikes, $f$ | Non-constructive | STDP Estimates | Gaussian Models | Normalised STDP Estimates |
|---|---|---|---|---|
| 0 | 95 | – | – | – |
| 1 | – | 97 | 97 | 98 |
| 5 | – | 98 | 97 | 98 |
| 10 | – | 97 | 97 | 97 |
| 15 | – | 97 | 98 | 98 |
| 20 | – | 98 | 98 | 98 |

The non-constructive simulations that fail to achieve learning success correlate with the highest failure rates in constructive simulations (Figures 6.14, 6.15 and 6.16). Constructive simulations, however, produce one or more successful postsynaptic neuron in cases where non-constructive simulations fail (Table 6.3). This indicates that neuron construction can improve the overall probability that a pattern will be detected.

### 6.6.4 Neuron Learning Speed

In the simulated conditions with consistently high overall presynaptic activity, an important aspect of learning is ceasing response to activity that is not part of the repeating pattern. The activation of a postsynaptic neuron outside the 50 ms pattern of presynaptic neuron activity is recorded as a false positive. Therefore, the speed of learning has been estimated using the total number of false positives generated by postsynaptic neurons. The distributions of the total false positives of postsynaptic neurons is presented in a series of box plots (Figure 6.17).

Constructive simulations with synapse weights calculated using estimates of additive STDP and associated Gaussian distributions show declining total false positives for estimates of increasing past iterations of spike times. The maximum difference in quartile and median statistics from the estimates of STDP and the associated Gaussian distributions is 1, with most statistical values being equal. The normalised STDP potentiation estimates demonstrate a small increasing trend in the total number of false positives, despite the total synapse weight remaining constant.

The spreading of synapse weight distributions increases the postsynaptic sensitivity to a set of presynaptic neurons and can prolong the response to background presynaptic activity. Nevertheless, a reduction in the total input weight is correlated with a decrease in the number of STDP updates that are necessary to depress the input weights below a threshold where background presynaptic activity produces postsynaptic spikes. These results indicate that the faster rejection of non-pattern activity is more strongly linked with lower total synaptic input rather than synapse weight estimates.

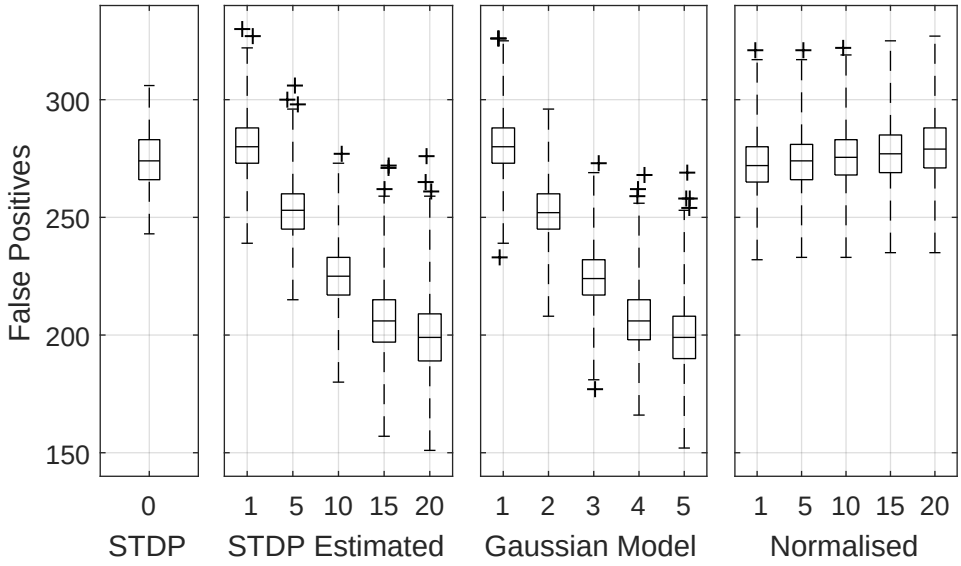


**Figure 6.17:** Box plots of total false positives of postsynaptic neurons. From left to right: non-constructive simulations of additive STDP; simulations with neuron construction using additive STDP estimates; simulations with neuron construction using Gaussian models of additive STDP estimates; simulations with neuron construction using normalised estimates of positive additive STDP. From top to bottom, the horizontal box lines provide the third-quartile ($q_3$), median and first-quartile ($q_1$) values. The whiskers extend to the recorded values up to the limits $[q_1 - 2.5 \cdot (q_3 - q_1), q_3 + 2.5 \cdot (q_3 - q_1)]$. Observations outside this range are indicated as outliers and are represented as crosses (+).

## 6.7 Discussion

The primary aim of the study presented in this chapter was to demonstrate that constructed neurons can produce the behaviour exhibited by predefined neurons in non-constructive simulation of STDP. This aim was met: all the constructive techniques tested produced learning behaviour and performance comparable to the past non-constructive study (Masquelier et al., 2008). This is evidence that neuron construction (based on simulation expansion) will be compatible with a range of computational studies of neuroscience, particularly when implemented in combination with a model of spike-timing-dependent plasticity.

An additional aim of the presented study was the investigation of the effect on learning performance (pattern detection and noise rejection) of synapse weight calculation processes for neuron construction. The results collected suggest that the implemented proxy postsynaptic neuron for triggering construction and synapse weight calculations from STDP estimates do not produce an advantage when compared to a normal distribution that approximates the STDP estimates. The reduced learning success rate of neurons constructed during the repeating spike pattern with STDP estimates and

normalised positive STDP estimates was an unexpected finding (Figures 6.10 and 6.12).

The explanation proposed here is that the synapse weight calculations depress synapses from presynaptic neurons that are inactive early and active late in the repeating spike pattern. This suggests that a successful synapse weight calculation process should avoid depressing these synapses. Alternatively, given that a successful postsynaptic neuron spikes in the first 10 ms of the repeating presynaptic activity pattern, performance may be improved by making postsynaptic neurons much more sensitive to the presynaptic neurons that are active immediately before the proxy neuron spike time.

Given the simulation conditions, lower total synaptic input is considered to be the most significant factor in faster rejection of non-pattern activity. At the end of non-constructive STDP simulations, postsynaptic neurons had median total input weights in the range $[363.7, 387.9]$. This indicates that fast or immediate rejection of background activity may be possible with neurons constructed with lower total input weights.

Even in the event of neuron construction achieving lower success rates, the construction of neurons can improve the overall probability of achieving at least one successful neuron over non-constructive simulations. This is indicative of a core advantage that neuron construction provides in spike pattern detection in challenging simulation conditions: construction allows simulations to make continual attempts at detecting new patterns.

The high levels of presynaptic neuron activity in the repeating pattern and background created unfavourable conditions for pattern detection and spike time prediction by the proxy postsynaptic neuron and for the synapse weight calculation processes. The conditions of the simulation do not allow for untuned or unbiased neurons and proxy neurons to distinguish the repeated pattern from the background and reject background activity. Nevertheless, the proxy neuron spike-triggered construction does ensure that presynaptic activity levels do meet a minimum threshold for activating a constructed neuron. This detection of presynaptic activity above a minimum threshold is important in simulations that have larger variation in the presynaptic activity levels. In other conditions for presynaptic activity, surges in activity may consistently indicate the presence of a significant pattern of presynaptic spikes and be effectively detected with an unbiased proxy neuron.

The conditions of the reproduced simulation include some contrivances to increase the difficulty of the learning task to highlight the learning power of STDP; however, these conditions may reduce the overall correspondence with biological systems and warrant some discussion. Perhaps most notable is that the average stochastic presynaptic neuron activity of 64 Hz is significantly higher than the average activity observed and predicted in many areas of the mammalian cortex (Shoham, O'Connor, & Segev, 2006). The synapse weight calculation processes investigated in this chapter are based upon an STDP model of repeated spike pairings that neglect the effect of high rates of activity (for example, Pfister & Gerstner, 2006). This is also a potential shortcoming of the past study (Masquelier et al., 2008), which did not incorporate rate effects into the plasticity model implemented.

Additionally, given the high rate of presynaptic neuron activity and the existence of any repeating activity patterns it would be surprising if there were only a single repeating pattern. The capabilities of STDP to facilitate the development of neurons to detect multiple patterns was explored in another past study (Masquelier et al., 2009). This study of competitive pattern detection has been reproduced and extended with algorithms for neuron construction in the next chapter of this thesis.

# Chapter 7

# Continual Learning of Spike Patterns

This chapter presents additional constructive algorithm developments and analysis in a task of detecting multiple spike patterns concealed within noise. A study of STDP and lateral inhibition producing competitive detection of multiple hidden spike patterns is reproduced from past literature (Masquelier et al., 2009). This past study is extended to a task of learning new sets of hidden spike patterns introduced during operation. Findings from Chapter 6 are incorporated into a process for synapse weight calculations that approximates the convergence of STDP. Simulation conditions result in a high rate of neuron construction; therefore, methods for controlling proxy neuron spikes are investigated. Methods for cancelling neuron construction and pruning neurons are also investigated for controlling the size of the simulated neural network.

All constructive algorithm processes are developed to be compatible with the interpretation of neuron construction and pruning as simulation expansion and contraction (Chapter 3). The resulting constructive algorithm performs reliable one-shot detection of hidden spike patterns for the given conditions. In simulations with dense repetition of spike patterns, construction is effectively limited by the inhibition of the proxy neuron and the cancellation of neuron construction. The performance of the constructive algorithm assuming synapse weight convergence indicates a potential for further application in computational studies of neuroscience and in machine learning.

## 7.1 Background

A past study (Masquelier et al., 2009) demonstrated the learning capabilities of additive STDP with multiple postsynaptic neurons and lateral inhibition (Figure 7.1) in the competitive detection of multiple spike patterns concealed in background activity (Figure 7.2). This study reproduced and extended the models used in the earlier study of STDP tuning a postsynaptic neuron to detect the start of a single repeating hidden spike pattern (Masquelier et al., 2008). The inclusion of lateral inhibitory connections between postsynaptic neurons produced competition between neurons in tuning to spike

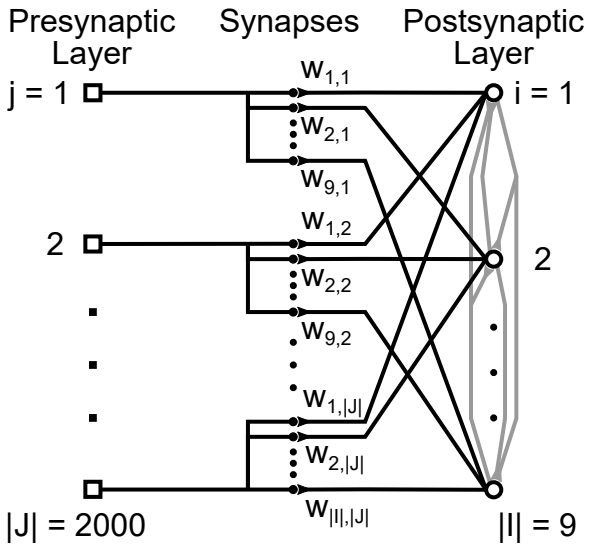


**Figure 7.1:** Neural network structure with lateral inhibition. Simulations of competitive spike pattern detection without construction are performed for a network with $|J| = 2000$ presynaptic neurons and $|I| = 9$ postsynaptic neurons. The postsynaptic layer has lateral inhibitory connections (shown in grey).

patterns, resulting in postsynaptic neurons tuning to selectively respond to one part of one of the repeating patterns. Postsynaptic neurons that tune to the same pattern compete and settle into a spike order with a low variation in spike latency. In 100 simulations of nine neurons and three hidden patterns, 63.4% of the simulated postsynaptic neurons achieved the given success criteria (Masquelier et al., 2009). Neurons that failed to tune to a pattern were reported to become inactive.

Details of the simulation and models are provided in the publications of the past studies (Masquelier et al., 2008, 2009). This thesis has presented reproductions of these models and simulations with details provided in Chapters 5 and 6. To avoid redundancy, the descriptions of models and simulation processes presented in this chapter focus on the differences from those implemented in earlier chapters of this thesis.

## 7.2 Aims

The primary aim of the developments and investigation presented in this chapter is the demonstration of neuron construction and pruning that is compatible with a computer simulation of biological learning (STDP) in a neural network with lateral inhibition. To achieve this aim the constructive algorithm is developed according to the principles of simulation expansion and contraction (Chapter 3), and the behaviour of constructed neurons is examined and compared with the results of non-constructive simulations. Success will be demonstrated in the event that the constructed neurons present similar behaviour to neurons tuned through STDP in non-constructive simulations. The

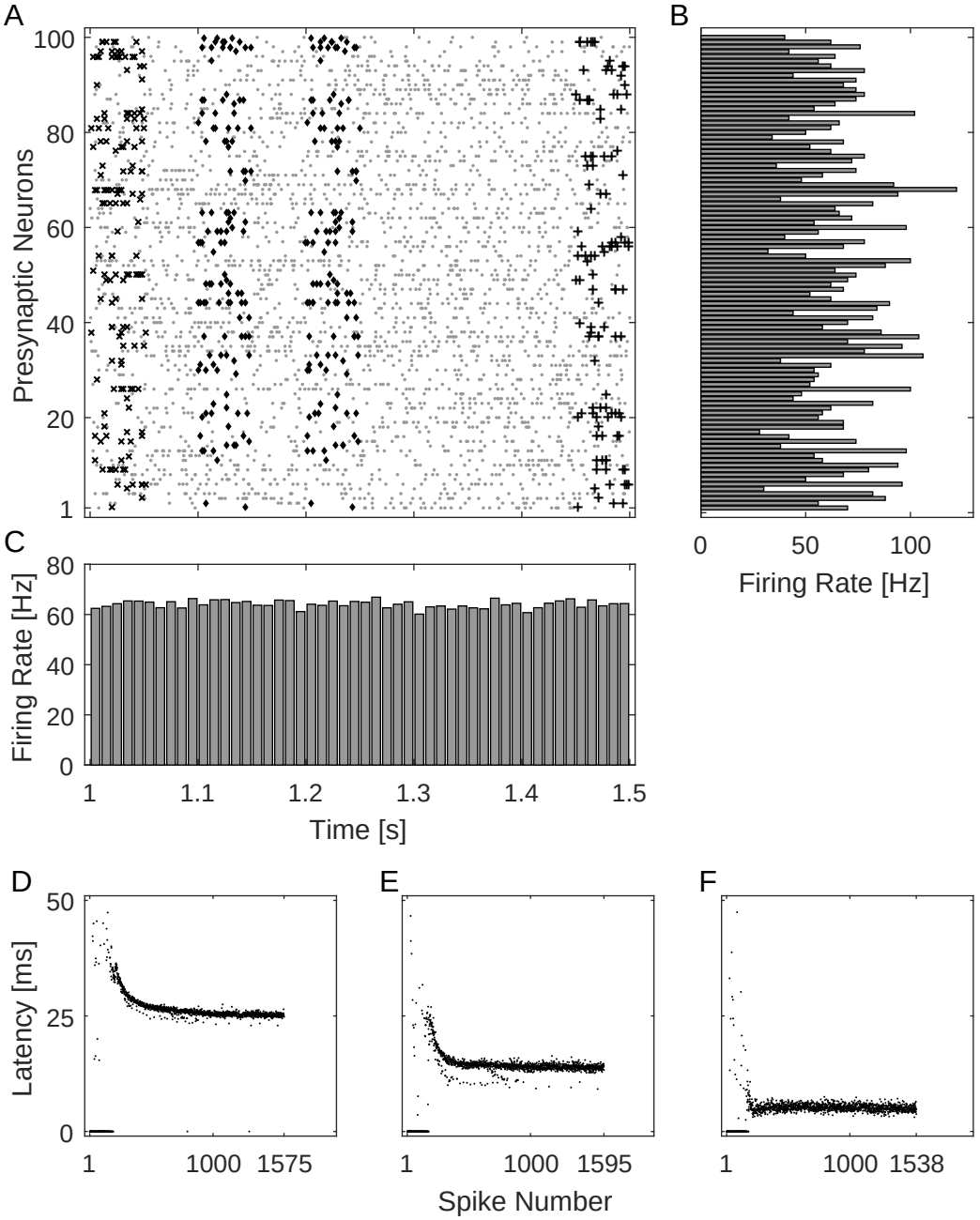


**Figure 7.2:** Example of presynaptic neuron and postsynaptic neuron activity in a reproduced simulation of competitive spike pattern detection (Masquelier et al., 2009). A: The spike times of 100 presynaptic neurons over 0.5 s with spikes the three repeating hidden patterns overlaid with a ×, ♦, or +. B: The spike rate of each of the 100 neurons in this 0.5 s time window. C: The average spike rate of all 2000 presynaptic neurons in 10 ms time-bins. D–F: Spike latencies of three postsynaptic neurons tuned to the same 50 ms spike pattern (0 ms indicates a false positive).

evaluation of behaviour will include some qualitative examination of the activity of constructed neurons; quantitative analyses will focus on criteria for spike pattern detection capabilities provided by the authors of the past studies of STDP (Masquelier et al., 2008, 2009).

Benefits of neuron construction that have been proposed in the course of this thesis include: 1) the automatic selection of the neural network size and 2) the potential to adapt the neural network size during the simulation or operation in response to changing conditions. The study presented in this chapter aims to demonstrate these capabilities in the competitive spike pattern detection task. The conditions of the original simulation are suitable for evaluating the algorithm performance for the automatic selection of the neural network size. A second set of simulations is conducted with conditions extended to periodically introduce new sets of spike patterns. This allows further evaluation of the algorithm performance in the adaptation of the neural network size in response to changing conditions.

Achieving the listed aims requires modification of the proxy neuron spike-triggered constructive algorithm presented in Chapter 5. In this chapter, additional novel constructive algorithm processes are developed and tested: 1) processes for synapse weight calculation; and 2) processes for evaluating the neuron and simulation performance to cancel neuron construction and trigger neuron pruning.

The processes for synapse weight calculation are developed from the observations of synapse weight convergence presented in the previous chapter (STDP Simulation with Neuron Construction):

1. STDP causes the synapse weights of successful neurons converge toward bimodal distributions.

2. STDP lowers the total input weight to reject background activity (950 decreases to the range $[353.5, 378.9]$ for a threshold of 550).

3. Postsynaptic neurons can respond to combinations of active presynaptic neurons in spike patterns in less than 10 ms.

The aim of synapse weight calculations is to create neurons that are immediately selective to an observed spike pattern and meet past criteria for learning success.

Processes for evaluating neuron and simulation performance are developed to control the size of the neural network simulation. Simulation conditions with consistently high average presynaptic activity were intentionally designed to create challenging conditions for the detection of the repeating spike patterns. The constructive algorithm uses proxy neuron spikes to predict postsynaptic spike times and trigger neuron construction. Lateral inhibition mechanisms are proposed and studied to limit proxy neuron spikes and neuron construction. A process of evaluating the activity of new postsynaptic neurons is proposed and studied for removing ineffective neurons. The aim of developing these performance evaluation processes is to minimise the number of inactive and redundant neurons that are constructed and remain in the simulation.

## 7.3 Models and Simulation

Simulation models have been reproduced from the descriptions provided in the past studies of additive STDP tuning neurons to detect hidden spike patterns (Masquelier et al., 2008, 2009). Most features of these models have been implemented in studies presented in earlier chapters of this thesis. To reduce redundancy, references are made to the relevant sections of the previous chapters. This section focuses on new features and differences in the models and simulations presented in this chapter.

The generation of presynaptic neuron activity extends the process for simulations with a single repeating hidden spike pattern, described in Section 6.3. The changes in input neuron activity generation can be summarised as 1) input activity is generated in 225 s batches rather than 150 s batches and 2) multiple patterns are copied and repeated. Each 225 s batch of input activity is divided into 0.05 s segments, giving a total of 4500 segments. Two cases of multiple spike patterns and repetition are examined:

1. Three 0.05 s spike patterns are repeated ($p = 3$), and the total number of repetitions of each pattern is $1/3 \times 1/p \times 4500 = 750$. This results in two thirds of the 0.05 s segments containing stochastic activity without a pattern repetition. A single 225 s batch of activity is generated and then repeated two times for a total simulation time of 675 s.

2. Four 0.05 s spike patterns are repeated ($p = 4$), and the total number of repetitions of each pattern is $1/p \times 4500 = 1125$. This results in all 0.05 s segments containing a pattern repetition. Three different 225 s batches of presynaptic activity are generated and simulated sequentially (totalling 675 s) to introduce new sets of repeating spike patterns.

Time segments are randomly selected to include repetitions of patterns, without allowing the same pattern to occur two times in a row or two different patterns to occur simultaneously.

A set of 1000 presynaptic neurons is selected at random as the pattern neurons for each repeating pattern. Distinct 0.05 s segments are selected at random from the initial stochastic activity to be the bases of the repeating patterns. The spike times of the pattern neurons in the base activity segments are then copied to replace the pattern neuron activity in the selected segments for pattern repetitions. As in the previous chapter, individual presynaptic neuron spike times are copied with zero-mean, 1 ms-standard deviation Gaussian noise.

After the selection and copying of pattern repetitions an additional 10 Hz Poisson activity is generated for each neuron as additional spike noise. An average spike rate of the complete spike trains was reported to be approximately 64 Hz in the original study and this overall spike rate was found in the reproduced presynaptic spike trains.

The majority of the model features for synaptic transmission and postsynaptic neuron activity are unchanged; however, changes have been incorporated to match the past study of STDP producing competitive spike pattern detection (Masquelier et al., 2009). The unchanged features are communicated in the previous chapters (see

Section 5.4.3). The changes to the models can be summarised: 1) an increase in the refractory period after a spike to 5 ms; 2) a postsynaptic activation threshold of $\theta = 550$; 3) the random initialisation of synapse weights; and 4) the introduction of lateral inhibition between postsynaptic neurons.

The non-constructive simulations initialise excitatory synapses from the input neurons with random weights with uniform probability in $[w_{\min}, w_{\max}] = [0, 1]$. In the past study (Masquelier et al., 2009), lateral inhibition was produced by all postsynaptic neuron spikes and transmitted to all other postsynaptic neurons. This lateral inhibition transmission is implemented using the same process as the excitatory spikes (Equation 5.1) with the synapse weights of inhibitory connections given as a constant $w_{\text{inh}} = -0.25 \cdot \theta = -137.5$. The implementation of additive STDP is unchanged from the simulations in the previous chapters (see Section 3.2.1). Lateral connections do not have adaptive synapse weights.

The event-driven simulation update cycle is extended from the previous chapters (see Section 5.4.3) to include lateral inhibition in the event of a postsynaptic neuron spike. The next section develops new constructive algorithm processes and incorporates them into the simulation update cycle. The development and evaluation of constructive algorithm processes is done incrementally using the reproduced simulation conditions with intermittent repetition of multiple patterns.

The simulations have been performed in MATLAB R2014a and R2016a.

## 7.4 Constructive Algorithm Development

This section presents the investigation and development of constructive algorithm processes to create neurons tuned to detect hidden spike patterns. The simulations presented in Chapter 6 had similar presynaptic activity, and the observed results of STDP in tuning neurons to detect the hidden pattern has been used to inform the development of the constructive algorithm here. Principles of simulation expansion and contraction (Section 3.1.4) are considered in the design process to facilitate compatibility with simulations of biology. An extended discussion of the compatibility of the STDP learning model and developed constructive algorithm processes with simulations of biological neural networks is presented at the end of this chapter (Section 7.7).

Postsynaptic neurons tuned to detect the repeating spike patterns have been shown previously (Song et al. (2000) and in Chapter 6) to develop bimodal synapse weight distributions. The chapter develops a synapse weight calculation process for neuron construction that produces bimodal synapse weights and investigates the pattern detection performance of varied total input weights.

The proxy neuron spike-triggered construction demonstrated in Chapter 6 resulted in many neurons that were tuned through STDP to the same repeating spike pattern. The introduction of lateral inhibition may be sufficient to avoid redundant spike latencies, but may also result in neurons that fail or cease to respond to any repeating spike pattern. This chapter develops algorithm processes to prevent the construction of redundant neurons by inhibiting the proxy neuron and examines their performance.

Constructed neurons that become inactive may be pruned to reduce the computational cost of the simulation with little effect on the model. A study of processes for pruning inactive neurons based on the early neuron activity is presented in a later section.

The development of algorithm processes and the evaluation of performance is presented in sections for: 1) synapse weight calculations and 2) neuron construction, cancellation and pruning. The performance of synapse weight calculation processes is evaluated based on recordings of the numbers of neuron spikes and true positives. The evaluations of algorithm conditions for neuron construction, cancellation and pruning use additional recorded values for assessment: the rates of neuron construction and simulation and the final numbers of simulated neurons.

The preliminary evaluations of the constructive algorithm processes have been performed with 225 s of presynaptic neuron activity. The overall aim of algorithm development is to construct neurons that selectively respond to a hidden spike pattern with a one-shot calculation of synapse weights and produce simulations without redundant or inactive neurons. The algorithm development concludes with the selection of algorithm processes and parameter values for the application to full-duration simulations (675 s) for the final investigation.

### 7.4.1 Synapse Weight Calculation

Simulations with neuron construction in Chapter 6 showed a decreased rate of learning success from increases in the number of activity iterations estimated in constructed synapse weight calculations. Although the trend of increasing STDP estimates indicated worsening performance, this does not rule out that synapse weight calculations that assume full convergence of STDP may improve performance. Two relevant findings from the previous simulation results were:

1. Learning success from tuning synapses through additive STDP coincided with a reduction in the total input weight from an initial value of 950 to less than half the initial value (Figure 6.6).

2. The majority of the total neuron input weight was localised in synapses with weights greater than 0.9 (Figure 6.7).

Calculating weights for new synapses assuming that additive STDP had converged was considered in Section 3.2.3.4. Methods of implementation and the spike pattern detection performance of this approach will be examined in this section.

The findings of the previous chapter showed that the additive STDP convergence resulted in a reduction of the total input weight. Selecting presynaptic neurons using an eligibility window can result in variable numbers of neurons being selected for synapse weight calculations and variable initial total input weights to constructed postsynaptic neurons. The total input weight is an important factor in the ability of the neuron to reject background activity; therefore, the initial total input weight of constructed neurons will be controlled. This preliminary investigation examines the performance of

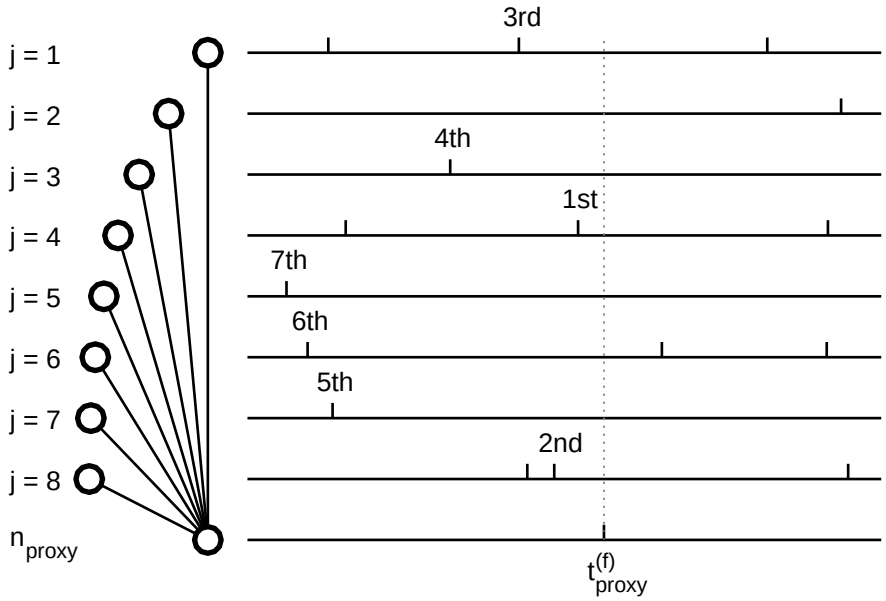


**Figure 7.3:** The most recent presynaptic neurons to spike relative to the proxy neuron spike. The number of presynaptic neurons selected depends on the initial total input weight and the maximum weight of synapses. For example, an initial total weight of 4 and a $w_{\max} = 1$ would result in the four most recent spiking presynaptic neurons being selected: in order from 1st to 4th, $j = 4, 8, 1$ and 3.

postsynaptic neurons constructed with total input weights $\sum_{j \in J} w_{i,j}(t_c) = 300$ to 800 in increments of 50.

The convergence of additive STDP is assumed to result in synapses with either the maximum or minimum synapse weight ($w_{\max} = 1$ and $w_{\min} = 0$). To ensure an initial total input weight, the number of presynaptic neurons selected to have synapses potentiated must be controlled. Instead of using an eligibility time window, which can select a variable number of presynaptic neurons, the presynaptic neurons are selected in order of most recent spike to least recent (Figure 7.3). The initial total input weight and maximum synapse weight determines the number of presynaptic neurons selected, $n_{\text{select}} = 300, 350, 400, \ldots, 800$.

The constructive algorithm is simplified for this preliminary investigation (Algorithm 9): neuron construction occurs at $T_c = 10\,\text{ms}$ time intervals starting with a construction time $t_c = 60\,\text{ms}$ (a proxy neuron is not simulated). At each construction time, a postsynaptic neuron is constructed for each initial total input weight, $\sum_{j \in J} w_{i,j}(t_c)$, by selecting presynaptic neurons to have the maximum synapse weight, $n_{\text{select}} = 300, 350, 400, \ldots, 800$. Construction ceases once $n_{\max} = 150$ neurons have been constructed for each initial total input weight. Lateral inhibition of postsynaptic neurons is excluded in the first of these preliminary simulations.

The performance of postsynaptic neurons constructed with synapse weights calculated using STDP convergence and specified total input weight is examined here. The

**Algorithm 9** Event-driven simulation with periodic neuron construction and specified initial total input weights.

**initialise** neurons constructed, $n_c \leftarrow 0$; maximum neurons, $n_{\max} \leftarrow 150$; construction time, $t_c \leftarrow 60\,\text{ms}$; construction period, $T_c \leftarrow 10\,\text{ms}$; select presynaptic neurons, $n_{\text{select}} \in \{300, 350, \ldots, 800\}$; neural network parameters, $J$, $I$, $p_{m,I}$, $p_{s,I}$, $\theta$, etc
**for** $t \leftarrow t_J^{(1)}, t_J^{(2)}, t_J^{(3)}, \ldots, t_J^{(G-1)}, t_J^{(G)}, \ldots, t_J^{(\Omega)}$ **do**
    **if** $t \geq t_c$ **and** $n_c < n_{\max}$ **then**
        Find $n_{\text{select}}$ most recent spiking presynaptic neurons, $J_{\text{recent}}$ (Figure 7.3)
        Calculate synapse weights: $\mathbf{w}_J \leftarrow w_{\min}$, $\mathbf{w}_{J_{\text{recent}}} \leftarrow w_{\max}$
        Add the new neuron to the network: $n_c \leftarrow n_c + 1$, $\mathbf{w}_{n_c,J} \leftarrow \mathbf{w}_J$, $f_{n_c} \leftarrow 0$
        $t_c \leftarrow t_c + 10\,\text{ms}$
    **end if**
    Find change in simulation time: for $t_J^{(G)} = t$, $\Delta t \leftarrow t_J^{(G)} - t_J^{(G-1)}$,
    Update potential (decay) of $I$ (Equations 5.6a and 5.6b)
    Find the set of spiking postsynaptic neurons, $F$: $i \in I$, $p_i > \theta$ **and** $t > t_{\text{ref},i}$
    **if** any postsynaptic neurons spike, $|F| > 0$ **then**
        **for each** $i \in F$ **do**
            Increment the number of spikes for $i$: $f_i \leftarrow f_i + 1$
            Set potential to spike values: $p_{m,i}(t) \leftarrow p_{m,\text{spike}}$, $p_{s,i}(t) \leftarrow p_{s,\text{spike}}$
            Set refractoriness times: $t_{\text{ref},i} \leftarrow t + 5\,\text{ms}$
            Synapses to $i$ receive a positive weight update (Equation 5.9)
        **end for**
    **end if**
    **for each** presynaptic neuron $j$, $t_j^{(g)} = t$ **do**
        Synapses from $j$ receive a negative weight update (Equation 5.10)
        Add potential to $I$ and proxy neuron from $j$ (Equations 5.11a and 5.11b)
    **end for**
**end for**

total number of postsynaptic spikes and the number of true positives (pattern repetitions with a postsynaptic spike) in the 225 s simulation are of primary interest and presented visually in Figure 7.4. This figure shows that neurons constructed during a repeating pattern with initial total input weights of 400 and 450 immediately detect that hidden spike pattern and reject background noise and other repeating patterns. This is a significant improvement in the detection of repeating hidden spike patterns over STDP estimates with specified iteration numbers and high total input weight (Chapter 6).

At initial total weights of 400 and 450 the neurons constructed during repeating patterns have total numbers of spikes in the range $[456, 500]$ and $[433, 500]$, respectively (Figure 7.4). Neurons constructed outside of any repeating pattern have between 1 and 3 spikes in total, including the postsynaptic spike assumed for synapse weight calculations. Neurons constructed with these initial total weights during a repeating pattern generally had two or fewer false positives in total. Exceptions were observed

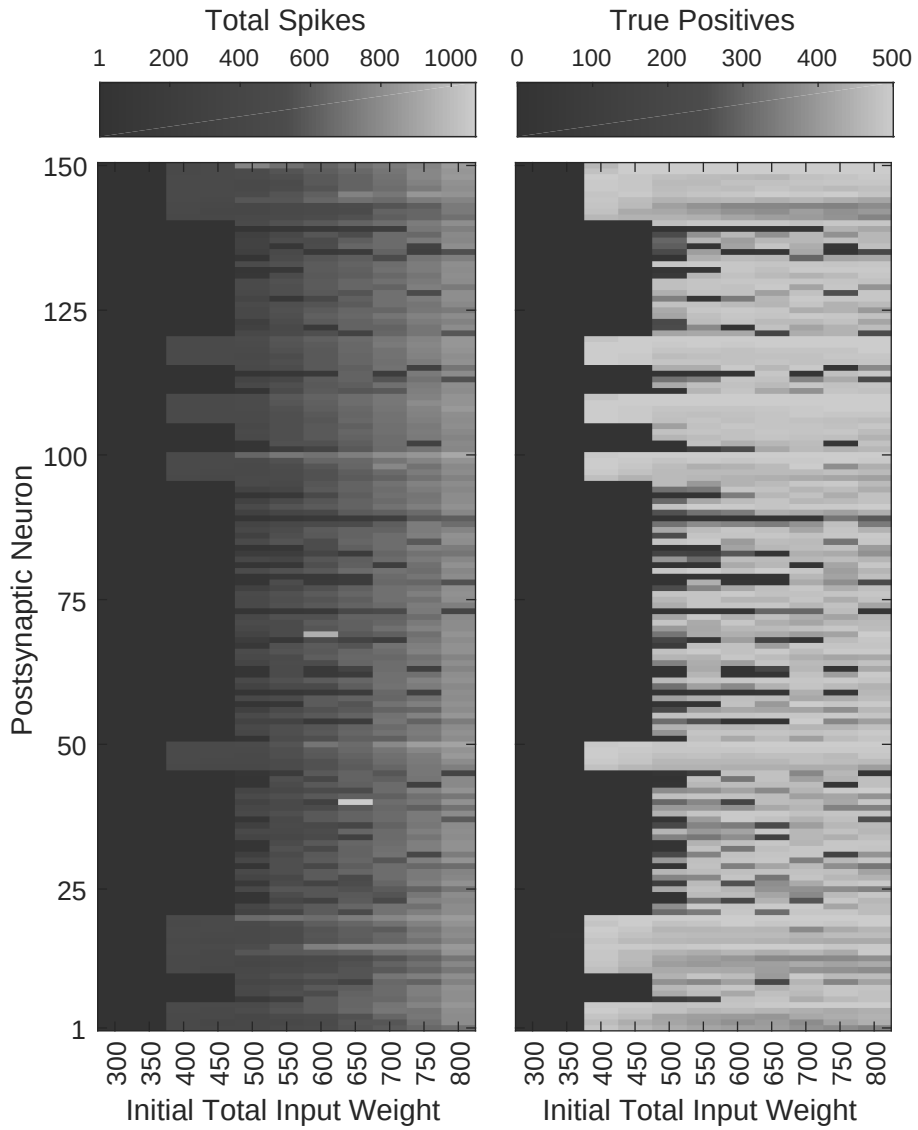


**Figure 7.4:** The total number of postsynaptic neuron spikes (left) and maximum number of true positives (right) for increasing initial total input synapse weight. Different grey-scales are indicated above each plot. New neurons are introduced at 10 ms intervals starting at $t = 0.06$s until 1.55 s ($n_{\max} = 150$). Three 0.05 s spike patterns repeat over 225 s and occur inside the time range 0.06 s to 1.55 s for neuron construction: pattern 1 at 1.1 s, 1.2 s and 1.5 s; pattern 2 at 0.2 s, 0.5 s and 1 s; and pattern 3 at 0.05 , 0.15 and 1.45

for neurons with an initial total weight of 400 introduced at the pattern end, which have as many as 29 false positives due to delayed neuron spikes occurring slightly after the 50 ms pattern.

For total input weight values below 400 the number of postsynaptic spikes tends to remain around 1 (including the postsynaptic spike assumed at the time of construction) with a few exceptions for an initial total weight of 350 producing a maximum of 14 spikes for one neuron. The neurons have insufficient total input weight to activate in response to the repeating spike pattern.

Increasing the initial total weight above 450 caused new neurons to spike an increasing number of times and allows neurons produced outside of pattern times to tune to patterns. The increased weight produces higher numbers of false positives before tuning through STDP reduces the total weight and ends the indiscriminate activation. Two neurons were constructed that tuned to two patterns (Figure 7.4): neuron 69 with initial input weight 600 tuned to patterns 1 (88.6%) and 3 (81.4%); neuron 40 with initial input 650 tuned to patterns 2 (93.6%) and 3 (92.0%).

The next step in the evaluation of synapse weight calculations introduced lateral inhibition between the postsynaptic neurons (Figure 7.1). Neurons with the same initial total input weight have been simulated as separate networks, that is, lateral inhibitory connections were restricted to neurons constructed with the same initial total input weight. All postsynaptic neuron spikes produce lateral inhibition except for the postsynaptic spike assumed at the time of neuron construction.

Lateral inhibition produces competition between neurons that respond to the same pattern, reducing the number of neurons with high activity and reducing the total number of postsynaptic spikes (Figure 7.5). Many neurons constructed with higher initial total input weights had activity reduced by more than 50% from the introduction of lateral inhibition.

Neurons constructed with an initial total input weight of 400 and 450 during a repeating input pattern were immediately selectively responsive to that pattern (perfect or high true positive rates with zero or low false positive rates). The competition introduced from lateral inhibition, however, prevented many neurons constructed during repeating input patterns from consistently spiking during that pattern. Less frequent neuron construction may reduce the overlap and interference between neurons constructed during the same repeating pattern of presynaptic activity.

Neurons constructed outside of a repeating input pattern with a total input weight of less than 500 do not spike after construction. Investigated initial total input weights values of 500 and above resulted in postsynaptic neurons spikes outside of repeating pattern activity that could continue for hundreds of simulation seconds. At the initial total input weight of 800 almost all neurons constructed outside of pattern times spike between 140 and 150 times and continue spiking until the end of the 225 s simulation.

The construction of neurons with lateral inhibition may conflict with the principle of plausible effects for simulation expansion and reduce the compatibility of the constructive algorithm with simulations of biological networks. If a constructed neuron has a rapid spike rate and produces lateral inhibition, the lateral inhibition can cause

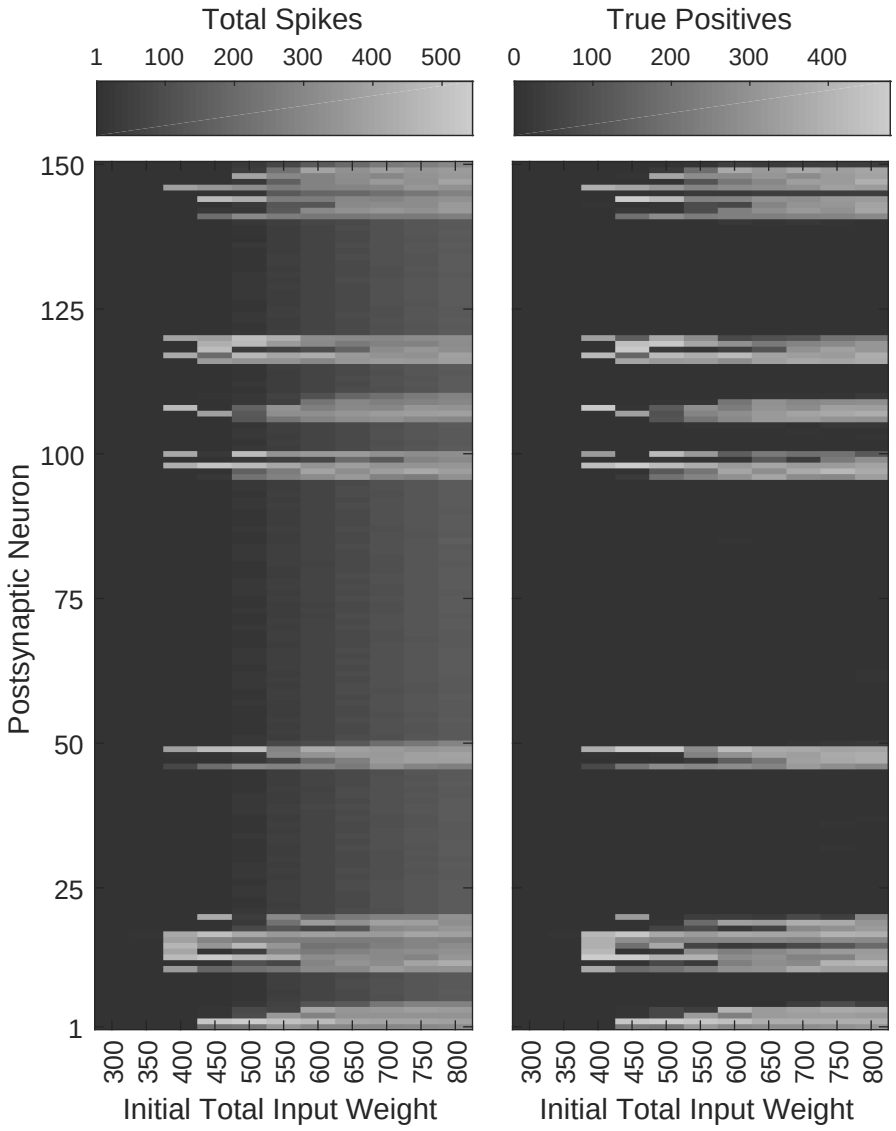


**Figure 7.5:** The total number of postsynaptic neuron spikes (left) and maximum number of true positives (right) with lateral inhibition of postsynaptic neurons for increasing initial total input synapse weight. Different grey-scales are indicated above each plot.

a significant effect on the activity of other simulated neurons. This sudden inhibition of simulated neurons may be biologically implausible. This interference should be minimised by constructing neurons that spike infrequently and do not conflict with the existing postsynaptic neuron activity.

The results of this preliminary investigation indicate that it may be possible to achieve immediate selective pattern detection with neuron construction for the given simulation conditions. The next preliminary investigations study the effects of methods for decreasing the rate of construction and decreasing the number of inactive neurons simulated. The initial total input weight of new neurons was chosen to be $\sum_{j \in J} w_{i,j}(t_c) = 450$ for all following constructive simulations presented in this chapter.

### 7.4.2 Construction, Cancellation and Pruning Processes

An effective proxy neuron will provide spike time predictions that can be used for neuron construction and will neither spike nor trigger neuron construction if too few presynaptic neurons are active to elicit future postsynaptic spikes. Simulations in this chapter and in Chapter 6, however, do not have increases in the overall input activity associated with significant (repeating) presynaptic spike patterns. This prevents unbiased proxy neurons from distinguishing the hidden repeating patterns from background activity. Rapid, indiscriminate proxy neuron spikes resulted in a high rate of neuron construction. Methods for preventing the construction of redundant neurons and pruning ineffective neurons are explored in this section.

The development of processes for neuron construction, cancellation and pruning are extensions of the processes of proxy neuron spike-triggered construction (Chapter 5). The proxy neuron, $n_{\text{proxy}}$ or $k$, is implemented using the same base properties and characteristics as in Chapters 5 and 6. In summary, the proxy neuron is simulated using the postsynaptic neuron model provided previously (Section 5.4.3) and has connections from the full set of simulated presynaptic neurons, $J$, with static synapse weight, $w_{k,J} = 0.5$. Figure 5.2 provides a visual representation of a neural network with a proxy neuron.

In the present network model (Figure 7.1), a spike from a simulated postsynaptic neuron indicates that a pattern in the input activity is detected and lateral inhibition reduces the likelihood of competing neurons spiking. The proxy neuron is a predictor of postsynaptic spikes in the surrounding network, but precedence is given to the current simulated neurons. Lateral inhibition from the proxy neuron to simulated postsynaptic neurons could prevent spikes that indicate the detection of a spike pattern; therefore, the proxy neuron spikes do not produce lateral inhibition of other simulated postsynaptic neurons. Successful inhibition of the proxy neuron, however, can prevent the construction of competing or redundant neurons and reduce the overall rate of neuron construction.

Two methods for modulating the activity of the proxy neuron based on lateral inhibition from simulated neurons are examined in this section:

1. Simulated postsynaptic neuron spikes produce an inhibitory synaptic transmission to the proxy neuron.

2. Simulated postsynaptic neuron spikes produce a period of absolute inhibition of proxy neuron spikes.

These methods of proxy neuron inhibition are examined in combination with processes for pruning neurons.

Section 7.4.1 showed that neurons constructed with converged synapse weights during periods of non-repeating background activity resulted in inactive neurons. Ideally, neuron construction would not occur at these times; however, the high average rate of overall presynaptic activity does not allow the presence of a significant or repeating spike pattern to be ruled out. Furthermore, the constructive algorithm is desired to be capable of accommodating the introduction of new and unknown numbers of repeating patterns in the simulation. Therefore, rather than potentially fail to detect a new presynaptic activity pattern, it has been deemed more desirable to continually construct neurons and prune those that are found to be ineffective.

Pruning methods developed in this section assume that the construction of a successful neuron will immediately and consistently detect a repeating spike pattern. An unsuccessful neuron will cease to respond, but may spike more than once when first constructed. Spike pattern repetitions are at least 0.1 s apart; therefore, a neuron that detects a repeating pattern should produce consecutive spikes at least 0.1 s apart. Two sets of activity conditions for pruning constructed neurons are tested in this section:

1. A constructed neuron is pruned unless it has any consecutive spikes at least 0.1 s apart ($t_i^{(f+1)} - t_i^{(f)} \geq 0.1\,\mathrm{s}$) in the first 10 s of simulation after construction ($t_{\mathrm{prune}} = t_{\mathrm{proxy}}^{(f)} + 10\,\mathrm{s}$).

2. A constructed neuron is pruned unless it spikes at least 5 times ($f \geq 5$, excluding the proxy neuron spike) in the first 5 s of simulation after construction ($t_{\mathrm{prune}} = t_{\mathrm{proxy}}^{(f)} + 5\,\mathrm{s}$).

Pruning (or contraction) of a neuron that produces lateral inhibition may violate the principle of plausible effects (Section 3.1.4) unless the activity of the neuron and associated lateral inhibition is deemed insignificant. The first pruning method does not guarantee the pruned neuron had a low activity level; however, constructed neurons are expected to be inactive unless tuned a repeating spike pattern. The second pruning method does guarantee that pruned neuron had a spike rate of less than 1 Hz and can be assumed to have a negligible effect on the simulation. The biological plausibility of the simulation is expected to be unaffected from these pruning methods.

The aim of introducing proxy neuron inhibition and neuron pruning is to control the rate of neuron construction and prevent a runaway increase in the neural network size. Given that the presence of a repeating pattern during background activity cannot be ruled out, ongoing neuron construction is performed. Nevertheless, construction should not occur at the same time as a simulated postsynaptic neuron spike. Pruning is required to counteract ongoing neuron construction; the network size should reach an equilibrium when the number of repeating patterns remains constant.

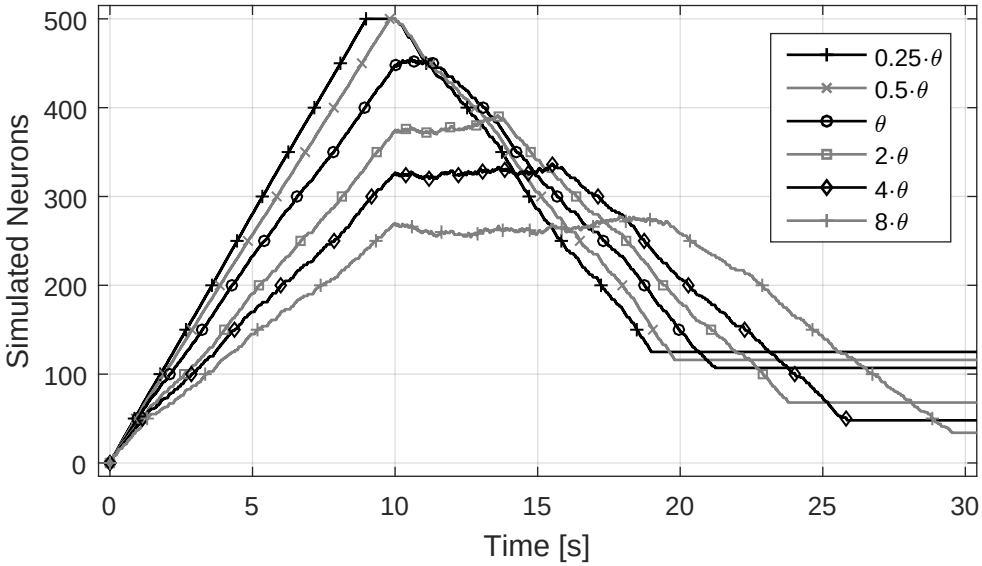


**Figure 7.6:** The number of simulated neurons over time for increasing lateral inhibition of the proxy neuron from simulated neurons. Neurons that do not have consecutive spikes 0.1 s apart in the first 10 s after construction are pruned.

In case pruning fails to prevent a runaway increase in the number of simulated neurons, a hard limit on the number of neurons that are constructed was also implemented. A neuron is only added to the simulation from a proxy neuron spike if fewer than the maximum number of postsynaptic neurons, $n_{\max} = 500$, have been constructed.

The first simulation evaluated the first proxy neuron inhibition method (inhibition as a synaptic transmission from simulated postsynaptic neuron spikes) and the first pruning method (neurons must have consecutive spikes 0.1 s apart in the 10 s after construction). Lateral inhibition of the proxy neuron is tested in multiples of the base inhibition weight, that is, $w_{k,I} = -0.25 \cdot \theta$, $-0.5 \cdot \theta$, $-\theta$, $-2 \cdot \theta$, $-4 \cdot \theta$, and $-8 \cdot \theta$.

The change in the number of simulated neurons for the pruning and proxy neuron inhibition methods are shown in Figure 7.6. The rate of neuron construction decreases for increasing proxy neuron inhibition weight. At the 10 s mark, the pruning process begins to remove neurons and counteract neuron construction. All proxy neuron inhibition weights result in the maximum number of neurons being constructed, after which the number of simulated neurons declines to a final simulation size. Note that low proxy neuron inhibition weights reach the maximum number of constructed neurons before the pruning process begins.

An important observation is that the different proxy neuron inhibition weights resulted in different final simulation sizes. The lowest three inhibition weights produce final simulation sizes with over 100 postsynaptic neurons. Given that the simulation only has three repeating spike patterns, this indicates that a high number of redundant or inactive neurons remain.

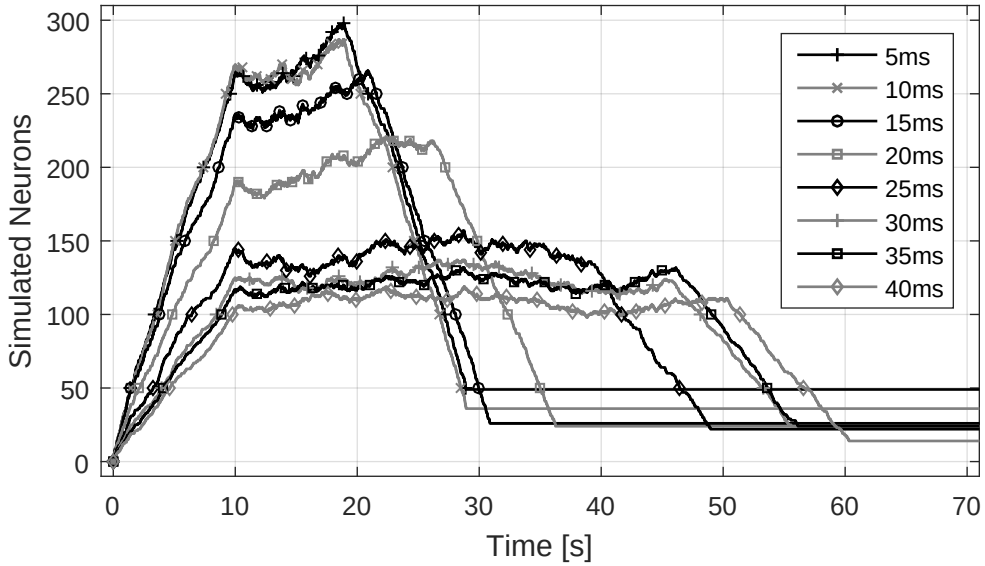


**Figure 7.7:** The number of simulated neurons over time for increasing times of proxy neuron inhibition and construction cancellation. Neurons that do not have consecutive spikes 0.1 s apart in the first 10 s after construction are pruned.

Inhibiting the proxy neuron as a synaptic transmission only prevents proxy neuron spikes after the spikes of other simulated postsynaptic neurons; however, a proxy neuron spike immediately before a spike of simulated postsynaptic neuron can still result in a redundant neuron. Preventing redundant neuron construction also requires that simulated postsynaptic neuron spikes result in the cancellation of any neuron construction that has occurred too recently.

The next set of simulations replaced lateral inhibition of the proxy neuron as a synaptic transmission with time-based absolute inhibition and cancellation of neuron construction. The absolute inhibition of the proxy neuron is implemented as a refractory period: the neuron parameters are updated, but the neuron cannot spike. Absolute inhibition is tested in 5 ms increments from the standard postsynaptic neuron refractory period: $t_{\text{proxy,inh}} \in \{5\,\text{ms},\ 10\,\text{ms},\ 15\,\text{ms},\ 20\,\text{ms},\ 25\,\text{ms},\ 30\,\text{ms},\ 35\,\text{ms}$ and $40\,\text{ms}\}$. The refractory periods of non-proxy postsynaptic neurons are unaffected.

The cancellation of a constructed neuron occurs if any other simulated neuron spikes within a construction cancellation time,

$$t_{\text{cancel}} = t^{(f)}_{\text{proxy}} + t_{\text{proxy,inh}}. \tag{7.1}$$

The absolute inhibition time, $t_{\text{proxy,inh}}$, has also been used as the cancellation time to reduce the initial search-space of values. Absolute inhibition and cancellation significantly reduced the rate of neuron construction and the final numbers of simulated postsynaptic neurons (Figure 7.7). Note that a constructed neuron that is cancelled is

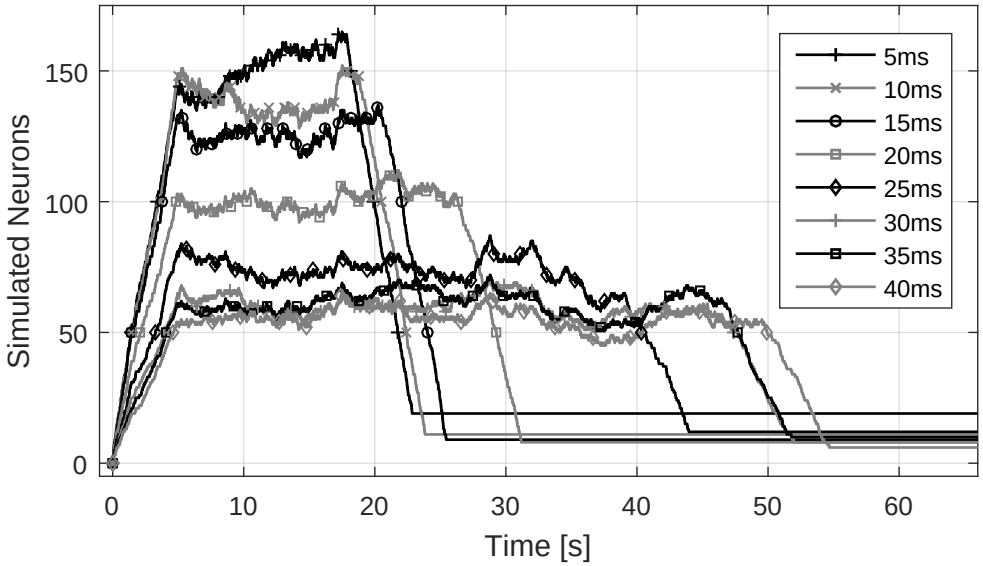


**Figure 7.8:** The number of simulated neurons over time for increasing times of proxy neuron inhibition and construction cancellation. Neurons that do not spike 5 times (in addition to the construction time) in the first 5 s after construction are pruned.

not counted toward the number of simulated and constructed neurons for visualisations or the maximum neuron construction limit ($n_{\max}$).

The past non-constructive study of competitive spike pattern detection (Masquelier et al., 2009) simulation would produce up to three postsynaptic neurons tuned to different parts of a 50 ms repeating spike pattern (see Figure 7.2). Therefore, assuming an aim of three neurons tuned to each 50 ms spike pattern and the first spike 10 ms after the pattern start, the delay between construction of neurons should be less than 20 ms. The preliminary test of absolute inhibition of proxy neurons of 5 ms, 10 ms and 15 ms resulted in 49, 36 and 26 final simulated postsynaptic neurons, respectively. This is still an undesirably high number of simulated neurons given an expectation of three postsynaptic neurons per 50 ms pattern.

The second pruning method (neurons must spike 5 times in the first 5 s) has been tested in combination with the absolute inhibition and construction cancellation (Figure 7.8). This pruning method reduces the peak number of simulated neurons and the final number of simulated neurons. The performance of the neurons that remain after pruning can be examined for the range of proxy neuron inhibition and construction cancellation times. The final number of simulated neurons and the average true positive rate and false positive rate of neurons are provided in Table 7.1.

This preliminary evaluation did not perform sufficient numbers of simulations to make estimates of variance; nevertheless, trends in outcomes were observed. The final number of neurons simulated was highest at the lowest proxy neuron inhibition time, but does not have a regular trend across the other inhibition times. The number of

**Table 7.1:** The final number of neurons simulated, average true positive spikes per neuron, and average false positive spikes per neuron over the course of the 225 s simulation using absolute inhibition of construction and a minimum activity threshold.

| Inhibition Time [ms] | Final Neurons Simulated | True Positives /Neuron | False Positives /Neuron |
|---|---|---|---|
| 5 | 19 | 250.37 | 10.05 |
| 10 | 11 | 403.00 | 0 |
| 15 | 9 | 478.00 | 0 |
| 20 | 8 | 470.63 | 17.50 |
| 25 | 12 | 344.08 | 4.167 |
| 30 | 8 | 368.88 | 45.00 |
| 35 | 10 | 359.20 | 12.10 |
| 40 | 6 | 424.83 | 44.67 |

true positives per neuron peaked at a cancellation period of 15 ms. The false positives per neuron were at zero at 10 ms and 15 ms and fluctuate for other inhibition times.

The sources of fluctuations will be investigated further in multiple-simulation studies presented later in this chapter; however, some preliminary observations can be made. The times of neuron construction during a repeating spike pattern has an element of randomness introduced from the accumulation of delays in proxy neuron spike times and the randomisation of repetition times of spike patterns. Spikes in false positives per neuron may occur when one or more neuron tune to the last few milliseconds of a pattern and then frequently spiking after the 50 ms pattern time.

This constructive algorithm development and search for parameter values has not been exhaustive; however, the preliminary investigation suggests a significant advance on the performance of the constructive algorithms presented in Chapter 6. The constructive algorithm implemented in the remaining simulations presented in this chapter use the proxy neuron inhibition and construction cancellation time of 15 ms and performs pruning of neurons that spike less than 5 times in the first 5 s after construction. Pseudo-code for the neural network simulation with neuron construction, cancellation and pruning is given in Algorithm 10.

A number of the constructive algorithm processes are dependent on the timing of neuron spikes:

- Spike-triggered construction initiates other constructive algorithm processes at the times of proxy neuron spikes.
- The synapses are constructed with maximum or minimum weight depending on the relative times of presynaptic neuron spikes.
- The inhibition and cancellation of neuron construction is dependent on the timing of postsynaptic neuron spikes.

**Algorithm 10** Event-driven simulation with spike-triggered neuron construction, cancellation and pruning.

**initialise** constructive algorithm parameters (Section 7.4.2): $n_\mathrm{c} \leftarrow 0$, $t_\mathrm{cancel} \leftarrow -\infty$, etc; neural network parameters (Section 7.3): $J$, $I \leftarrow \varnothing$, $\theta$, $p_{m,\mathrm{proxy}} \leftarrow 0$, $p_{s,\mathrm{proxy}} \leftarrow 0$, $\mathbf{w}_{\mathrm{proxy},J} \leftarrow 0.5$, etc
**for** $t \leftarrow t_J^{(1)}, t_J^{(2)}, t_J^{(3)}, \ldots, t_J^{(G-1)}, t_J^{(G)}, \ldots, t_J^{(\Omega)}$ **do**
    Find change in simulation time: for $t_J^{(G)} = t$, $\Delta t \leftarrow t_J^{(G)} - t_J^{(G-1)}$
    **for each** postsynaptic neuron, $i \in I$ **do**
        **if** $t_{\mathrm{prune},i} < t$ **then**
            Prune neuron $i$: $I \leftarrow I/i$
        **end if**
    **end for**
    Update potential (decay) of $I$ and proxy neuron (Equations 5.6a and 5.6b)
    Find the set of spiking postsynaptic neurons, $F$: $i \in I$, $p_i > \theta$ **and** $t > t_{\mathrm{ref},i}$
    **if** $|F| > 0$ **and** $F \neq \{n_\mathrm{c}\}$ **and** $t < t_\mathrm{cancel}$ **then**
        Cancel last neuron construction, $n_\mathrm{c}$
        Delete the parameters of that neuron: $\mathbf{w}_{n_\mathrm{c},J}$, $n_\mathrm{c} \leftarrow n_\mathrm{c} - 1$, $t_\mathrm{cancel} \leftarrow -\infty$
        Remove any cancelled neuron spikes from set: $F \leftarrow F/n_\mathrm{c}$
    **end if**
    **if** any postsynaptic neurons spike, $|F| > 0$ **then**
        **for each** $i \in F$ **do**
            Increment the number of spikes for $i$: $f_i \leftarrow f_i + 1$
            **if** activity requirement, $f_i \geq 5$, to avoid pruning is met **then**
                Clear pruning time: $t_{\mathrm{prune},i} \leftarrow \infty$
            **end if**
            Add lateral inhibition to $I$ (Equations 5.11a and 5.11b)
        **end for**
        Inhibit proxy neuron (reusing refractoriness time), $t_{\mathrm{ref,proxy}} \leftarrow t + 15\,\mathrm{ms}$
        **for each** $i \in F$ **do**
            Set potential to spike values: $p_{m,i}(t) \leftarrow p_{m,\mathrm{spike}}$, $p_{s,i}(t) \leftarrow p_{s,\mathrm{spike}}$
            Set refractoriness times: $t_{\mathrm{ref},i} \leftarrow t + 5\,\mathrm{ms}$
            Synapses to $i$ receive a positive weight update (Equation 5.9)
        **end for**
    **end if**
    **if** proxy neuron spikes, $p_\mathrm{proxy} > \theta$ **and** $t > t_{\mathrm{ref,proxy}}$ **and** $n_\mathrm{c} < n_\mathrm{max}$ **then**
        Find 450 most recent spiking presynaptic neurons, $J_\mathrm{recent}$
        Calculate synapse weights: $\mathbf{w}_J \leftarrow w_\mathrm{min}$, $\mathbf{w}_{J_\mathrm{recent}} \leftarrow w_\mathrm{max}$
        Add the new neuron to the network: $n_\mathrm{c} \leftarrow n_\mathrm{c} + 1$, $\mathbf{w}_{n_\mathrm{c},J} \leftarrow \mathbf{w}_J$, $f_{n_\mathrm{c}} \leftarrow 0$
        Set cancellation and pruning times: $t_\mathrm{cancel} \leftarrow t + 15\,\mathrm{ms}$, $t_{\mathrm{prune},n_\mathrm{c}} \leftarrow t + 5\,\mathrm{s}$
    **end if**
    **for each** presynaptic neuron $j$, $t_j^{(g)} = t$ **do**
        Synapses from $j$ receive a negative weight update (Equation 5.10)
        Add potential to $I$ and proxy neuron from $j$ (Equations 5.11a and 5.11b)
    **end for**
**end for**

Therefore, the algorithm meets the criterion for spike-timing-dependent construction (STDC) described in Section 2.1.3. The first pruning process (pruning neurons that do not have consecutive spikes 0.1 s apart in the 10 s after construction) is dependent on spike timings. The second pruning process (pruning neurons that do not spike at least 5 times in the 5 s after construction), however, is more accurately described as dependent on the spike rate rather than directly dependent on the timing of spikes. Nevertheless, the second pruning process showed better performance and has been selected for the simulations presented in the following sections.

## 7.5 Data Collection and Analysis

Data has been collected to examine the pattern detection performance of constructed neurons and neurons competitively tuned through STDP to concealed spike patterns. The aim of the constructive algorithm development has been to produce neurons that immediately selectively respond to a single repeating spike pattern with detection performance approximately equivalent to the results of tuning through STDP. Finding approximately equivalent behaviour of constructed neurons and neurons tuned through STDP simulation will be considered evidence of the compatibility of the constructive algorithm with the simulation of a biological neural network. The constructive simulations are expected to have advantages over simulations with a predefined structure: allowing continual learning of new spike patterns and producing few or zero neurons that are unresponsive.

The pattern detection performance of constructed neurons will be examined (individually and collectively) and compared to the performance of neurons tuned through STDP in equivalent non-constructive simulations. The simulations performed are reproduced and extended from a past study (Masquelier et al., 2009); therefore, the past data collection and analyses are reproduced here and applied to the reproduced non-constructive simulation and the constructive simulation for comparison. Examples of postsynaptic neuron activity in constructive simulations are presented and discussed with comparisons made to the reproduced non-constructive simulation (Figure 7.2).

The final performance of neurons tuned through additive STDP in non-constructive and constructive simulations are compared using the criteria for learning success from the past study (Masquelier et al., 2009). The last 75 s of simulated time is taken as a test period. Each postsynaptic neuron has a true positive percentage and false positive rate calculated for each pattern. A postsynaptic spike during one repeating pattern is counted as a true positive for that pattern but is counted as a false positive for other repeating patterns. The pattern with the highest true positive percentage and the associated false positive rate is selected for each neuron. A neuron is deemed to have achieved learning success if its highest true positive percentage is greater than 90% and the associated false positive rate is less than 1 Hz.

The development of the constructive algorithm has aimed to provide immediate selective detection of repeating spike patterns. Preliminary investigations have shown that neurons constructed with synapse weights estimating the convergence of additive

STDP can produce immediate detection of spike patterns and rejection of background activity. This will be investigated in more detail by examining the true positive percentage and false positive rate in the first 15 s of neuron simulation (simulation start or neuron construction).

An aim of constructive algorithm development is the automatic selection of the neural network size. Preliminary investigations have shown (Section 7.4.2) that the constructive algorithm with cancellation and pruning can automatically select a final neural network simulation size. Nevertheless, neuron construction occurs continuously between pattern repetitions when simulated postsynaptic neurons are inactive. The change in the number of neurons constructed and the number of simulated (unpruned) neurons over the simulation time will be recorded and presented.

Simulations with dense repetition of spike patterns (no periods of pure background activity) are expected to produce consistent postsynaptic neuron activity and provide conditions suited to controlled neuron construction. The change in the number of constructed neurons and the number of simulated (unpruned) neurons over the simulation time will be recorded and presented for simulations with dense spike pattern repetition.

A potential capability of constructive algorithms is the ability to perform continual online learning using neuron construction. Simulations with dense pattern repetition have new repeating spike patterns in each of the three consecutive 225 s batches of activity. Learning success criteria are applied to the last 75 s of each 225 s batch.

The presentation of simulation results is divided into sections for simulations with the intermittent repetition of spike patterns and for simulations with the dense repetition of spike patterns. Data is collected from 100 non-constructive simulations and 100 constructive simulations for both intermittent pattern repetition and dense pattern repetition.

## 7.6 Results

The results are divided into two sections: the first for simulations with intermittent repetition of spike patterns and the second for simulations with dense repetition of spike patterns.

### 7.6.1 Intermittent Repetition of Spike Patterns

An example of the early construction and spike times of postsynaptic neurons given in Figure 7.9 demonstrates that neurons constructed during a repeating spike pattern immediately respond selectively to that pattern. Neurons constructed outside of repeating pattern times cease to respond as the background activity changes and are removed after their pruning time elapses (5 s after construction). The latencies of spikes from neurons that avoid pruning over the full simulation are presented in Figure 7.10. After construction, neurons are tuned through additive STDP and typically settle into a stable sequence of spike latencies relative to the detected repeating pattern (see Pattern 2 in Figure 7.10).

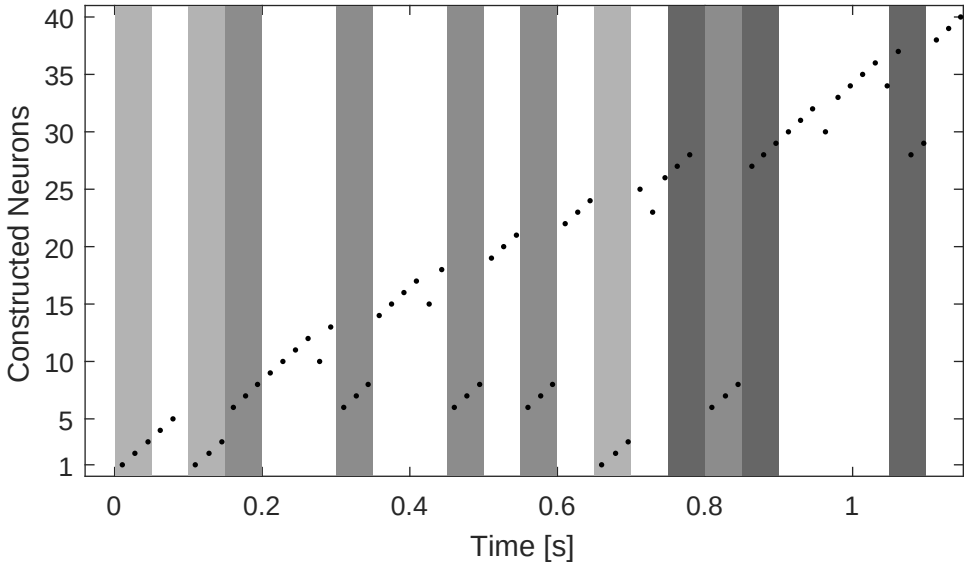


**Figure 7.9:** Times of neuron construction and spikes demonstrating the one-shot learning of hidden spike patterns. Neurons are numbered in order of construction. The time of construction is shown as the first spike time (dot) for that number neuron. The grey bands represent simulation times that contain a hidden spike pattern with the darkness of the tone indicating which of the three patterns is occurring (the lightest band is pattern 1; the darkest band is pattern 3). The spike latencies of the final simulated neurons over the full simulation are given in Figure 7.10.

Tuning of synapse weights through STDP has been observed to produce a range of subtly different outcomes in postsynaptic activity. In Figure 7.10, Neurons 2 and 3 (tuned to Pattern 1) exhibit less frequent activity at a consistent earlier spike latency. This can occur when a postsynaptic neuron that usually has an early spike latency fails to spike. The absence of the earlier postsynaptic spike and associated lateral inhibition results in postsynaptic neurons tuned to later sections of the same pattern spiking earlier.

Neuron 3 also develops an increasing rate of false positives for Pattern 1 as tuning progresses. Neuron 3 spikes late in the 50 ms repeating spike pattern and spikes with a second, less-frequent latency. This increases the number of presynaptic neurons with regular activity in the potentiating region of the STDP curve. The increased number of potentiated synapses increases the total input weight to a point where there is a non-negligible probability of the neuron spiking from background activity.

Another possible result is two neurons competitively tuning to and sharing approximately the same spike latency. For example, Figure 7.10 shows Neurons 28 and 29 tune to Pattern 3 with average spike latencies in the final 75 s test period of 18.8 ms and 23.4 ms, respectively. Neurons that tune to similar spike latencies for the same pattern rarely spike for the same occurrence of that spike pattern. Neurons 28 and

**Figure 7.10:** Spike latencies relative to pattern times for the final simulated postsynaptic neurons after construction and pruning. Spike latencies of the same neuron are organised in columns with the relative spike patterns organised in rows. Within each plot the spike latencies are represented as dots with spike number increasing from the first to last along the axis (the final number of spikes is given by the axis tick value). The spike latency is given a range from 0 ms to 50 ms with spikes occurring outside the 50 ms pattern duration given a latency of 0 ms and recorded as a false positive.

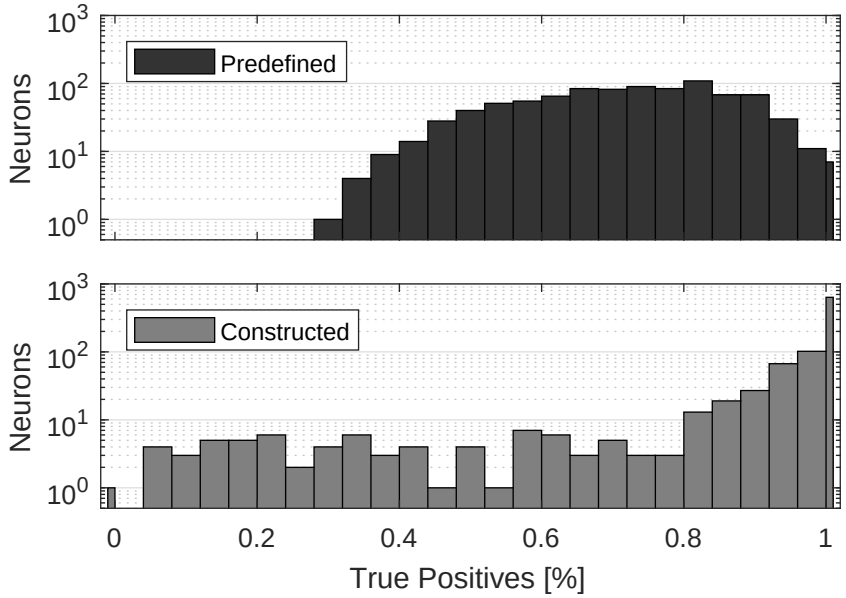


**Figure 7.11:** Histograms of true positive percentages in the first 15 s of neuron simulation in non-constructive simulations (top; 900 neurons) and constructive simulations (bottom; 939 neurons). A true positive is recorded when the neuron spikes in the 50 ms duration of a pattern. Each postsynaptic neuron has a true positive percentage for each pattern with the maximum percentage for each neuron presented. In 15 s of simulation time the expected number of repetitions of each pattern is 33.3.

29 spike in the same occurrence of the pattern in only 5/166 occurrences in the test period. Individually, neurons that have such close spike latencies may have low true positive percentage (Neuron 28: 34.34%; Neuron 29: 68.67%); however, simulation results showed that the combined activity of these neurons often detects all occurrences of the pattern in test periods. When the activity of Neurons 28 and 29 are combined all occurrences of Pattern 3 are detected.

The early performance (true positives percentage and false positive rate) of neurons in the first 15 s of simulation has been recorded and is presented for predefined neurons and constructed neurons in Figures 7.11 and 7.12. The predefined neurons initialised in non-constructive simulations have a high initial spike rate resulting in a broad distribution of true positive percentages and high false positive rates. Constructed neurons that are not pruned have true positive percentages concentrated at perfect pattern detection and an initial false positive rate that is low or zero.

The early simulation of networks with predefined structure (Figure 7.1) produced a true positive percentage greater than 90% in 80/900 (8.89%) of the postsynaptic neurons. Simulations with predefined structure and random weight initialisation show 7/900 neurons achieve 100% true positives in the first 15 s of the simulation. None of these neurons, however, satisfy the learning success criterion of a false positive rate of less than 1 Hz until STDP sufficiently depresses the total input weight of the postsy-

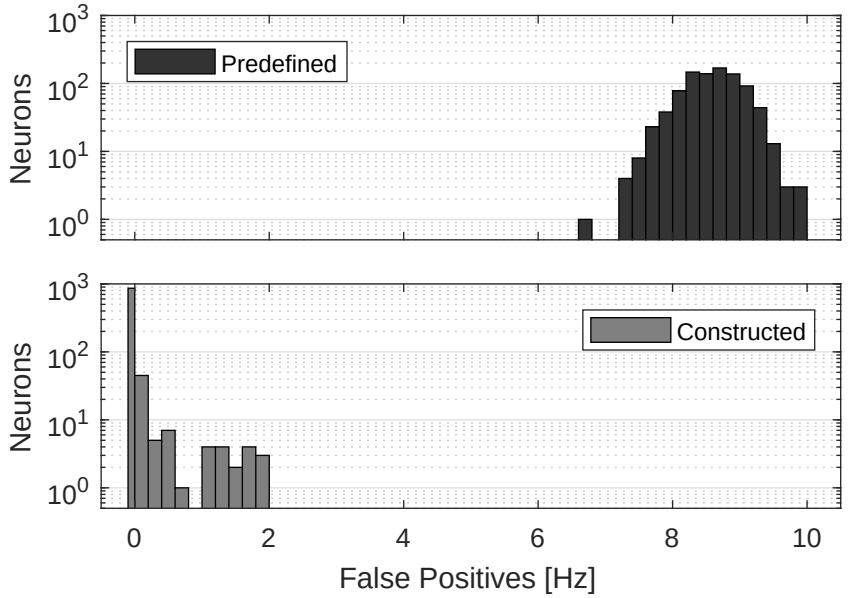


**Figure 7.12:** Histograms of false positive spike rates in the first 15 s of neuron simulation in non-constructive simulations (top; 900 neurons) and constructive simulations (bottom; 939 neurons). A false positive is recorded when the neuron spikes outside the 50 ms duration of a pattern. Each postsynaptic neuron has a false positive rate for each pattern with the minimum rate for each neuron presented.

naptic neurons to prevent indiscriminate spiking.

The construction of neurons with a lower total input weight eliminates the requirement of a period of synaptic depression before selectively responding to spike patterns. For the given simulation conditions (Section 7.3), detection of spike patterns is successfully achieved by potentiating synapses from presynaptic neurons with the most recent spikes. The number of neurons with a true positive percentages greater than 90% in the first 15 s of neuron simulation is 819/939 (87.2%) and all of these neurons satisfy the false positive rate criterion. The majority of the unpruned constructed neurons (635/939) have a true positive rate of 100% in the first 15 s of simulated time after construction (Figure 7.11).

The predefined neurons in all non-constructive simulations have a false positive rate between 6.73 Hz to 9.80 Hz in the first 15 s of the simulation (Figure 7.12). High initial spike rates of the postsynaptic neurons in simulations with a predefined structure are a product of the high initial total input weight (2000 synapses with an initial uniform distribution in $[0, 1]$ gives an expected total input weight of 1000). This causes the neurons to spike indiscriminately in the presence of the high overall rate of presynaptic neuron activity. The spike rate of postsynaptic neurons drops sharply once STDP sufficiently depresses the majority of synapse weights. If the remaining potentiated synapses sufficiently correspond with the active presynaptic neurons in a repeating hidden spike pattern, the postsynaptic neuron will selectively respond to that pattern.

In the first 15 s after construction, 864/939 constructed neurons have a false positive rate of 0 Hz. The maximum false positive rate recorded was 1.80 Hz (Figure 7.12). The constructed neurons with the highest false positive rates are a result of the neuron spiking a few milliseconds after the pattern segment ends. Treating spikes up to 5 ms after the pattern as a true positive increases the number of neurons with zero false positives to 895/939 and reduces the maximum false positive rate to 0.067 Hz (that is, the neuron produces a single false positive spike in the 15 s after construction). Applying this less strict definition of a true positive has a negligible effect on the range of false positive rates of predefined neurons (6.73 Hz to 9.73 Hz).

The past study of spike-timing-dependent plasticity (Masquelier et al., 2009) evaluated the neuron performance in the last 75 s of the simulation time. The past study reported an average of 5.71/9 neurons (63.4%) successfully learning to detect a section of one of three repeating spike patterns. The reproduction of this non-constructive simulation produced a similar result with 100 simulations producing an average of 5.85/9 neurons (65.0%) that achieve the learning success criteria. Constructive simulations performed on the same presynaptic neuron activity produced an average of 7.46 successful neurons with an average of 9.39 final simulated neurons (79.4% success rate).

The rates of neurons that achieve learning success in non-constructive simulations and constructive simulations are presented in a bar chart (Figure 7.13). The figure also presents a chart of the differences in learning success for the same presynaptic neuron activity (successful constructed neurons minus successful predefined neurons).

The processes for constructing and pruning neurons in simulations show an improvement in the final rate of learning success. A comparison of the frequency of neurons with true positive percentage (Figure 7.14) and false positive rates (Figure 7.15) in simulations with construction and with predefined structure had similar trends. The histograms of true positive percentages show that both types of simulations had neuron frequency peak at perfect pattern detection. Both types of simulations resulted in neurons with true positive percentages across the full range of intermediate values, $(0, 1)$. Simulations with construction, however, had no neurons with zero true positives in the 75 s test period, while non-constructive simulations had a total of 148 neurons with zero true positives.

The rates of false positives in simulations with predefined structure and with neuron construction (Figure 7.15) show some differences in trends and possible outcomes. Simulations with predefined neurons produced 5 neurons with false positive rate greater than 1 Hz. This occurred exclusively when neurons tuned to more than one pattern, with one occasion producing greater than 90% true positives for two patterns in the test period. The constructive simulations do not produce any neurons with false positive rate above 1 Hz but do show a higher number of neurons with false positive rates from zero to 0.55 Hz. Neuron construction results in more neurons with late spike latencies, which can have STDP increase the total input weight to a point where background activity causes spikes (seen in Neuron 3 in Figure 7.10).

The past study (Masquelier et al., 2009) reported that neurons would typically become inactive if unable to tune to a pattern. Neurons with long inactive periods

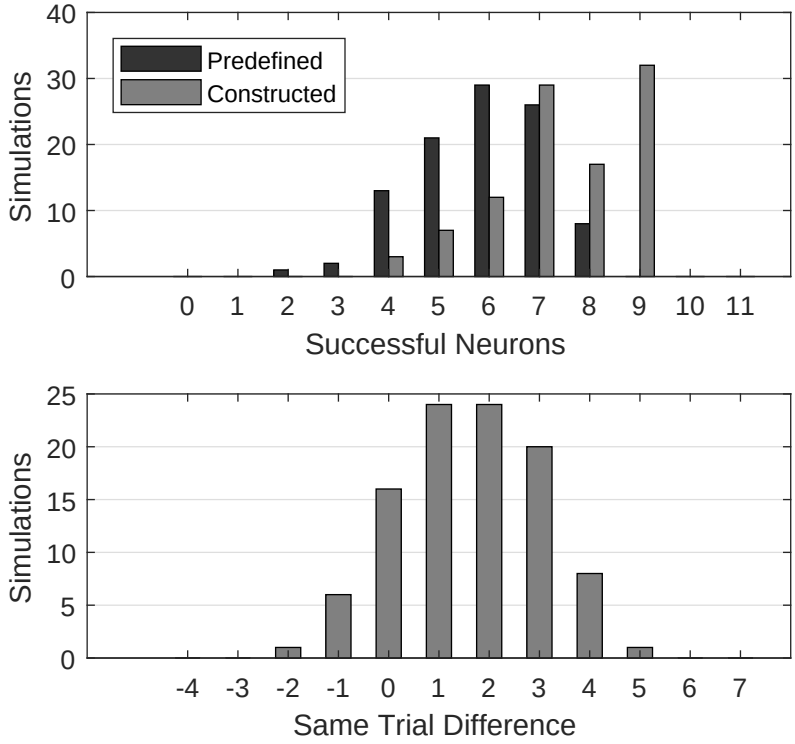


**Figure 7.13:** Charts of the rate of simulations that produce a number of successful neurons (top) and of the difference in neuron success numbers for the same input activity (bottom). Criteria for success are provided in Section 7.5. Constructive and non-constructive simulations have been performed for one hundred different sets of presynaptic activity. The difference is calculated as the number of successful constructed neurons minus the number of successful neurons in the simulation with predefined structure for the same input activity.

including the whole test period (the last 75 s of simulation time) were observed in the reproduced simulations with predefined structure (Figure 7.16). Constructive simulations did have neurons with low rates of activation but all unpruned neurons recorded at least one spike in the test period. The success rates and overall activity of neurons produced in constructive simulations are greater than those observed in the simulations with predefined structure.

The conditions for intermittent pattern generation present a challenge for the developed constructive algorithm in controlling the simulation size. The number of simulated neurons rises rapidly before neuron pruning starts and causes the number of simulated neurons to plateau (Figure 7.17). Construction stops at the maximum number of neurons and then pruning reduces the neural network to its final size. This process occurs in all simulations with intermittent repetition of spike patterns. Neuron construction introduces a significant computational expense with many neurons simulated for 5 s before being pruned.

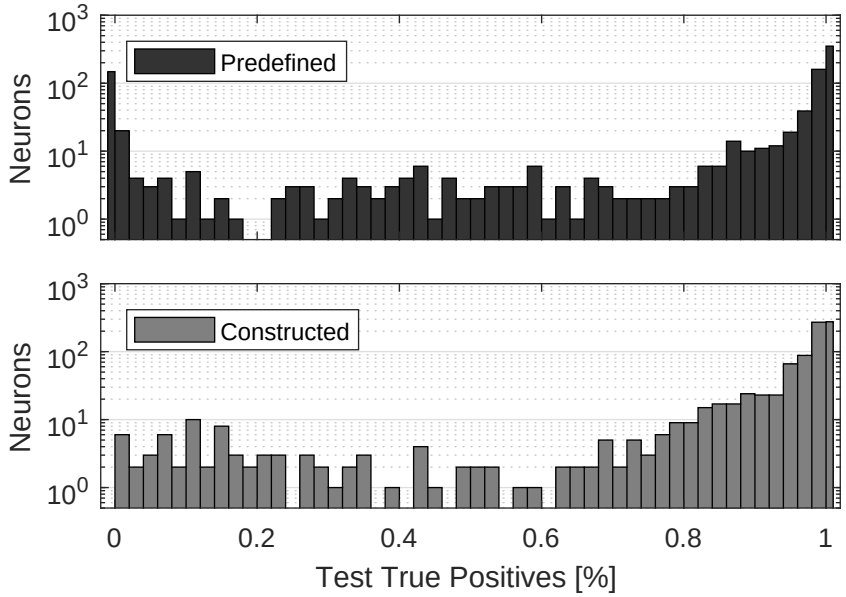


**Figure 7.14:** Histograms of true positives percentages in the final 75 s of simulations with predefined structure (900 neurons) and simulations with constructed neurons (939 neurons).

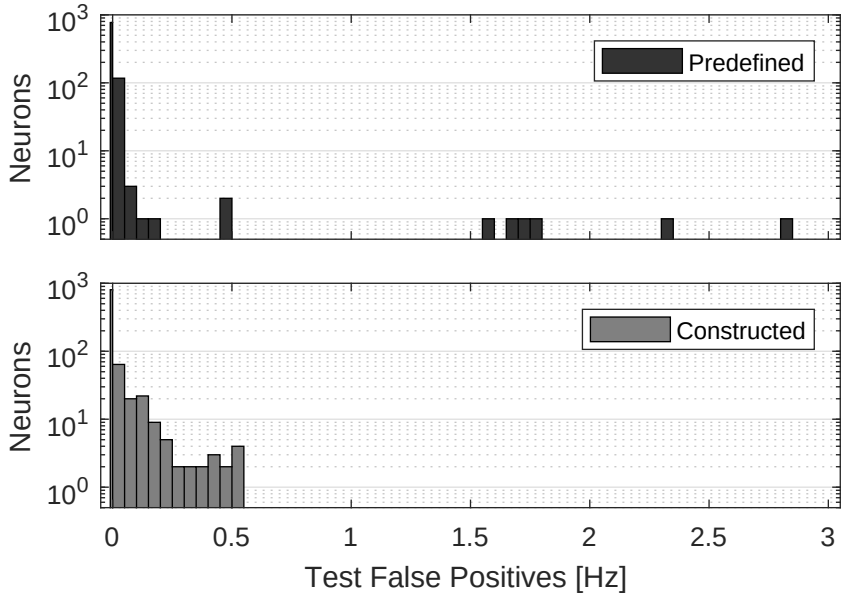


**Figure 7.15:** Histograms of false positives spike rates in the final 75 s of simulations with predefined structure (900 neurons) and simulations with constructed neurons (939 neurons).

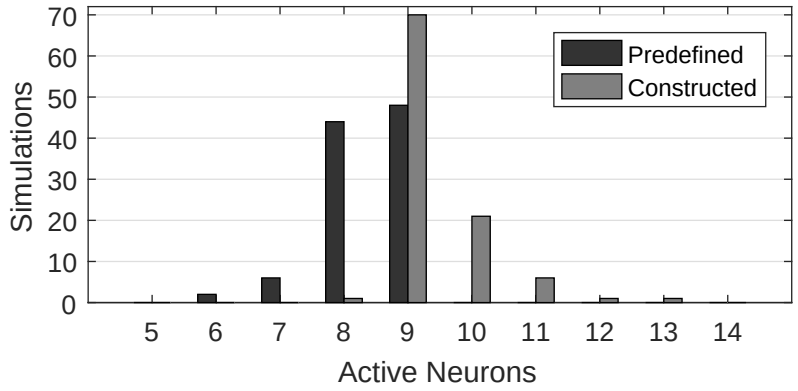


**Figure 7.16:** Chart of the number of active neurons during simulation tests with predefined structure and constructed neurons. A neuron is deemed to be active if it spiked once in the test phase of the simulation (the last 75 s).

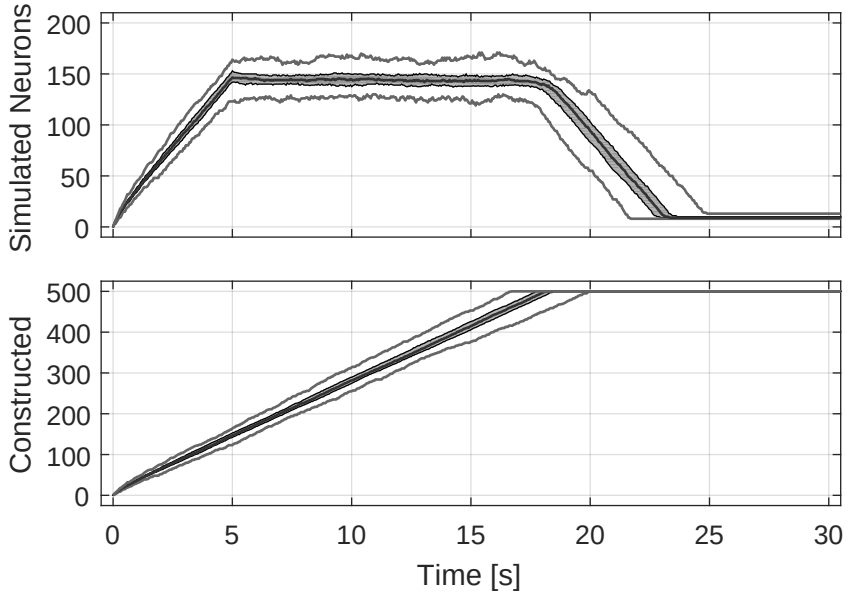


**Figure 7.17:** Plot of the number of simulated neurons (top) and the number of neurons constructed (bottom) over the first 30 s of the simulations. The shaded area represents the range between the first and third quartile values for 100 simulations over time; the solid black line inside the shaded area represents the median. Grey lines above and below the shaded area represent the minimum and maximum number of neurons in the 100 simulations. Values are calculated in 50 ms increments and exclude cancelled neuron construction.

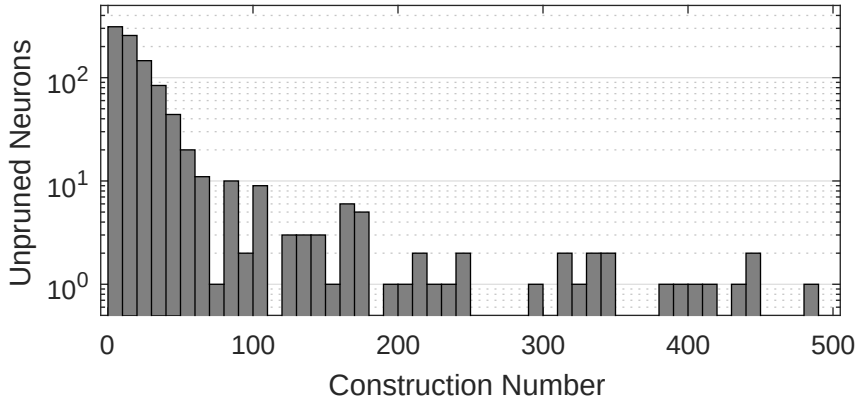


**Figure 7.18:** Histogram of the construction numbers of neurons that remain unpruned at the simulation end.

The total number of neurons constructed in these simulation is controlled by the constructive algorithm limit, $n_{\max} = 500$. The 100 constructive simulations averaged 9.39 neurons after pruning; therefore on average, approximately 490.6 neurons are simulated for 5 s before being pruned. Approximately 89.6% of the neurons (841/939) that remain in the final simulation were in the first 50 neurons constructed (Figure 7.18). In general, it may not be possible to predict if new spike patterns of significance will occur in later activity, and new patterns may not be distinguishable from background activity. Nevertheless, in cases where no new spike patterns are expected, the maximum number of neurons could be lowered to reduce the computational expense.

### 7.6.2 Dense Repetition of Spike Patterns

The second set of simulations were performed with dense repetition of spike patterns, that is, without periods of pure background activity between repetitions of spike patterns. The biological plausibility of dense pattern repetition is discussed in Section 7.7. The constructive algorithm produced a stable number of simulated neurons in these simulations, with neuron construction almost exclusively restricted to the appearance of new spike patterns (Figure 7.19).

The number of neurons simulated and the total number of neurons constructed (Figure 7.19) show a trend of rapidly rising in response to the introduction of new patterns (at times 0 s, 225 s and 450 s) before stabilising. The median number of simulated neurons after each 225 s batch was 15 neurons at 225 s, 33 neurons at 450 s, and 50 neurons at 675 s. The median number of neurons constructed (including those pruned) was 18 neurons at 225 s, 37 neurons at 450 s, and 57 at 675 s. The differences between the first and third quartile values were found to be small for the simulated neurons (a maximum of 4 neurons) and the total neurons constructed (a maximum of 5 neurons).

The dense repetition of spike patterns causes regular simulated postsynaptic neuron spikes, which result in continuous inhibition and cancellation of neuron construction.

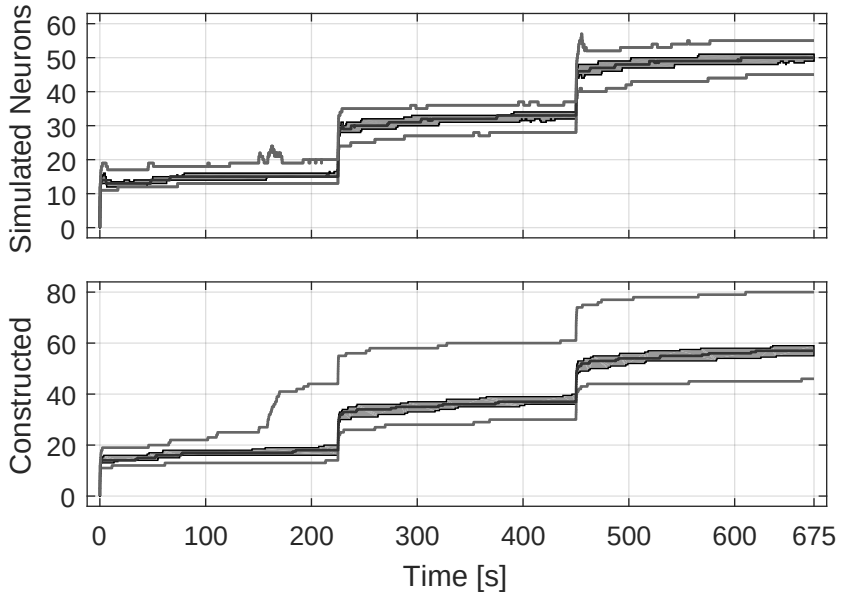


**Figure 7.19:** The number of neurons simulated (top) and the number of neurons constructed (bottom) for simulations with dense repetition of spike patterns. The shaded area represents the range between the first and third quartile values for 100 simulations over time; the solid black line inside the shaded area represents the median. Grey lines above and below the shaded area represent the minimum and maximum number of neurons in the 100 simulations. Values are calculated in 50 ms increments and exclude cancelled neuron construction.

As a result, the neuron construction limit is not reached in any of the simulations with dense pattern repetition. Under these conditions, the absence of postsynaptic activity is indicative of a new repeating pattern in presynaptic activity that can be immediately learned with neuron construction, even after long simulation times.

The numbers of neurons that achieve learning success for each 225 s period that introduces new patterns are presented for constructive and non-constructive simulations in Figure 7.20. Non-constructive simulations with 9 postsynaptic neurons demonstrated success learning patterns in the first 225 s period but failed to detect patterns introduced in later periods of activity. The dense repetition of spike patterns resulted in non-constructive simulations producing a higher rate of postsynaptic neurons responding to more than one pattern: 219/900 neurons had test true positive percentages above 50% for more than one pattern; 131/900 neurons had test true positive percentages above 90% for more than one pattern. A small scale test (results not shown) found that this effect was reduced by increasing the number of postsynaptic neurons, but increasing the number of postsynaptic neurons up to 24 did not improve the detection of spike patterns introduced in later 225 s activity periods.

The constructive simulations responded to the appearance of new repeating hidden patterns immediately through neuron construction and achieved high rates of learning

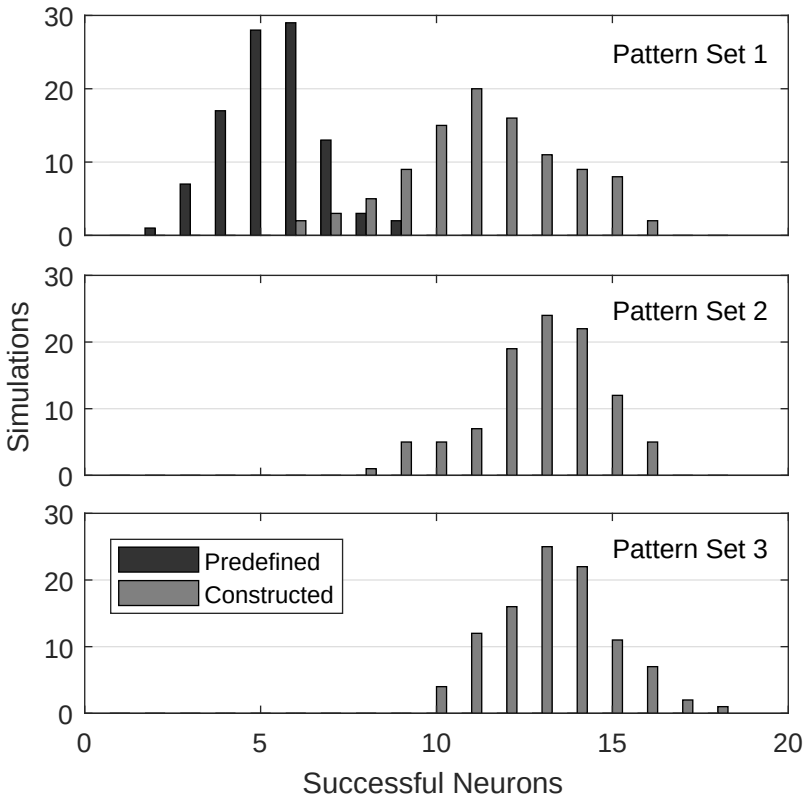


**Figure 7.20:** The distributions of the numbers of successful neurons in simulations (of one hundred) with predefined structure and with constructed neurons. Criteria for success are provided in Section 7.5. Note that the simulations with predefined structure do not produce any successful neurons for Pattern Sets 2 and 3.

success for each set of patterns that was introduced (Figure 7.20). In 100 constructive simulations, neurons constructed during:

- The first pattern set had a 72.51% success rate (1137/1568 simulated neurons; 1843 total neurons constructed);
- The second pattern set had a 74.67% success rate (1288/1725 simulated neurons; 1955 total neurons constructed); and
- The third pattern set had a 77.14% success rate (1326/1719 simulated neurons; 1951 total neurons constructed).

These results clearly demonstrate that simulations with the constructive algorithm have an advantage in automating the network size selection and in performing continual learning.

## 7.7 Discussion

A constructive algorithm has been developed in this chapter to achieve a number of goals: 1) demonstrate compatibility with a simulation of biological learning, 2) demonstrate automatic selection of the neural network simulation size, and 3) demonstrate neuron construction producing continual one-shot learning of hidden spike patterns. This section discusses the constructive algorithm developments and simulated experiment findings in the context of these goals and broader applications.

An earlier study of models of biological models for competitive learning of spike patterns (Masquelier et al., 2009) has been reproduced in this chapter, and constructive and non-constructive simulations obtained comparable quantitative and qualitative results. This is evidence for the compatibility of the constructive algorithm with simulations of these models of biological learning, including STDP. The theoretical biological plausibility of simulations with the constructive algorithm was briefly discussed during development (Section 7.4). This discussion is revisited and extended here.

The principle of plausible effects (Section 3.1.4) states that the construction or pruning of neurons should not produce biologically implausible changes to the behaviour of the simulated neural network. Constructing or pruning neurons with lateral inhibition can result in implausible changes in the activity of simulated neurons from increasing or decreasing inhibition signals. This interference has been minimised by constructing neurons to reproduce the activity of mature competitively tuned neurons: each postsynaptic neuron only responds to a specific portion of a repeating spike pattern. Implausible effects are avoided by cancelling constructed neurons that will compete with and inhibit an existing simulated neuron and by only pruning neurons that have insignificant levels of early activity. This constructive algorithm satisfies the principle of plausible effects for this network model and, therefore, has theoretical support for its compatibility with similar computer simulated models of biological neural networks.

The behaviour of the network of untuned postsynaptic neurons presents additional complications for the application of a constructive algorithm. In the original study of STDP (Masquelier et al., 2009), simulations were initialised with untuned neurons that would spike rapidly and indiscriminately until their overall input weight was sufficiently depressed. Constructing neurons with lateral inhibition and high spike rates would have significant effects on the activity of other simulated neurons and may rule the neuron construction as biologically implausible.

Considering the perspective of simulated and surrounding neurons (Figure 3.1), the original non-constructive simulation does not incorporate lateral inhibition from surrounding neurons. Given the rapid early activity of simulated neurons, the existence of similar surrounding neurons with lateral inhibition and high spike rates has a low probability as these neurons would produce substantial interference. The assumption that the small number of simulated postsynaptic neurons have no lateral interactions with surrounding neurons might not be biologically plausible either. Explicit incorporation of the effects of surrounding neurons may improve the plausibility of the model of biological learning and allow the implementation of a constructive algorithm.

The proposed constructive algorithm and other designs based on the principles of simulation expansion and contraction may be investigated further for compatibility with simulations of biological neural networks. Studies that focus on the function of mature biological neural systems may achieve the desired behaviour and performance through expanding the simulation with mature neurons and synapse weights calculated from estimates of plasticity convergence. Bypassing the simulation of early development and tuning of the network could significantly reduce the computational cost of simulated studies. Simulated studies of early neural development and plasticity could still benefit from the construction of untuned or immature neurons to automate selection of the simulation size.

The results of simulations showed that the performance of the constructive algorithm in the automatic selection of the simulation size was dependent on the frequency of repeating patterns in the input activity. The processes that control neuron construction, cancellation and pruning automatically selected effective simulation sizes; however, simulations with intermittent repetition of spike patterns resulted in the rapid construction of many neurons that were ultimately pruned. Cancellation and inhibition of neuron construction occurs when simulated postsynaptic neurons spike. In simulations with intermittent spike pattern repetition, simulated postsynaptic neuron activity was irregular and not sufficient to cancel or prevent regular neuron construction. The methods for controlling the number of simulated neurons could be further tuned and developed to reduce the computational cost. Nevertheless, this continual construction of neurons may be an unavoidable cost in conditions where new patterns of interest continue to be introduced, are intermittent, and are not adequately separable from the background activity.

In simulations with dense repetition of spike patterns, the processes of neuron construction, cancellation and pruning are successful at automatically selecting the simulation size with few extraneous neurons. Regular postsynaptic neuron spikes cancel and inhibit neuron construction, resulting in the construction of neurons performed almost exclusively in response to the presence of new spike patterns. The total number of neurons and the number of simulated neurons both have a brief but sharp rise in response to new spike patterns with new neurons successfully detecting the new patterns.

The biological plausibility of simulations with dense repetition of patterns may be defended with reference to the efficient coding hypothesis. The efficient coding hypothesis (Blättler & Hahnloser, 2011; Olshausen & Field, 2004; Shoham et al., 2006) states that neurons produce the minimum number of spikes to encode a signal. The input activity was consistently high in all simulations presented in this chapter whether the repetition of spike patterns was intermittent or dense. The simulations with intermittent repetition of spike patterns have high levels of neural activity that do not encode a signal and, therefore, are not consistent with the efficient coding hypothesis. Dense pattern repetition has no periods of pure background noise; therefore, the activity can be said to more efficiently code signals. Based on this criterion, the simulations of dense pattern repetition are more biologically plausible.

Proxy neurons were not effective in predicting when to perform construction in the

high-activity simulations presented in this chapter. Neural systems that have sparse activity and are consistent with the efficient coding hypothesis may more closely resemble the low-noise simulations presented in Chapter 5. Simulations of these biological neural systems will allow proxy neurons to more effectively detect patterns of significance, and perform neuron construction selectively to produce stable network sizes.

Constructive simulations that introduced new sets of repeating spike patterns demonstrated that the constructive algorithm could perform ongoing one-shot learning of new spike patterns. These results provide evidence that constructive algorithms can provide capabilities for continual learning and adaptation over the course of long simulations. This is a significant qualitative improvement in performance that may not be possible to achieve in simulations with predefined structure. The additive STDP model depresses the majority of the synapses to a neuron, preventing activation from background activity but also preventing the detection and tuning to different repeating patterns. The plasticity model may be adapted to allow simulations with predefined structures to tune to detect new patterns; however, this would be a slower iterative process and would increase the risk of forgetting past patterns.

The simulation expansion interpretation views the constructed postsynaptic neurons as having existed in the larger surrounding network. From this perspective, the constructive algorithm is not performing learning but is selecting or recalling past learned patterns. This distinction may only have philosophical interest, but this perspective might also have sufficient significance in the development and application of algorithms that perform simulation expansion and contraction to warrant further investigation. Future directions of research based on the concepts of simulation expansion and contraction are discussed in Chapter 8.

From a machine learning perspective, the developed constructive algorithm demonstrates promising capabilities. The algorithm demonstrates continual online one-shot learning in conditions with high levels of noise. Future directions for research in machine learning applications of developments are discussed in Chapter 8.

# Chapter 8

# Discussion

This chapter summarises the contributions and findings presented in this thesis and discusses their significance. Directions of future development and applications in the fields of machine learning and computational neuroscience are also discussed.

## 8.1 Summary of Contributions and Findings

The contributions of this thesis can be grouped into three types: 1) development of concepts, theory and design methodologies; 2) novel constructive algorithm processes; and 3) demonstrations of neuron construction modelling STDP. The contributions to the concepts, theory, and design methodologies of constructive neural networks can be summarised:

- A design methodology for constructive neural networks has been developed (and applied) from the identification of standard components (Section 2.1):
  - The artificial neural network and models;
  - Processes for performance evaluation;
  - Processes for parameter calculation.
- Spike-timing-dependent construction (STDC) has been proposed to identify and design constructive algorithm processes for spiking neurons (Section 2.1.3).
- Concepts of simulated neurons and surrounding neurons have been developed into principles for the design of algorithms for neural network simulation expansion, contraction, construction and pruning (Section 3.1).

The novel constructive algorithm processes can be summarised:

- Processes for calculating synapse weights based on different approaches to estimating the results of STDP models:
  - Specified iterations of past activity and additive STDP (Section 3.2.3.1);

  - Specified iterations of past activity and multiplicative STDP (Section 3.2.3.2);
  - Continuous-iteration approximation of past activity and multiplicative STDP (Section 3.2.3.3);
  - Convergence of additive STDP for presynaptic activity within a time window (Section 3.2.3.4);
  - Bimodal convergence of additive STDP for recent presynaptic activity up to a total input weight (Section 7.4.1).

- The evaluation of ANN performance using the spike times of a proxy neuron (Chapter 5):
  - As an unsupervised process for triggering neuron construction;
  - As a prediction of the spike time of a surrounding postsynaptic neuron for synapse weight calculations.

- Unsupervised neuron pruning based on the early activity (performance) of constructed neurons (Section 7.4.2).

- Inhibition and cancellation of neuron construction in response to simulated postsynaptic neuron spikes (Section 7.4.2).

Developed constructive algorithm processes have been tested numerically and in simulations and the findings can be summarised:

- Synapse weight calculation equations based on specified iterations of activity with STDP models have been found accurate up to numerical precision for ideal conditions (Section 4.3).

- Synapse weight calculations based on STDP estimates using a single observation of stochastic relative spike timings amplify noise in the synapse weights when compared to the simulation of STDP (Sections 4.4 and 4.5).

- Neurons constructed using proxy neuron spikes and synapse calculations based on additive STDP are found to reproduce the spike times of neurons simulated with additive STDP to within 0.5 ms for the tested noisy conditions (Section 5.4.6).

Constructive algorithms based on proxy neuron spikes and additive STDP estimation have been developed and evaluated in a simulation of STDP tuning neurons to detect a hidden repeating spike pattern (Masquelier et al., 2008). Constructed neurons tuned to detect hidden repeating patterns at high rates, similar to the original study of STDP (Section 6.6.3); however, overall success rates decreased slightly for synapse weight calculations based on increasing numbers of specified iterations of activity in STDP estimates. Sharp decreases in learning success were observed for neurons that were constructed with predicted spike times early in the repeating spike pattern. This decrease in the success rate was correlated to the depression of synapses from presynaptic neurons that were active in the pattern.

A modified constructive algorithm was applied to the conditions of a simulated study of STDP producing the competitive tuning of neurons to detect one of many hidden repeating spike patterns (Masquelier et al., 2009). Synapse weights calculated as the bimodal convergence of additive STDP with a total input weight was found to produce one-shot detection of hidden spike patterns (Section 7.6). The pruning of constructed neurons that did not meet minimum activity requirements controlled the size of the simulated network when the repetition of patterns was intermittent. The inhibition and cancellation of neuron construction prevented unnecessary construction in simulations with dense repetition of patterns. Simulations that introduced new sets of hidden repeating spike patterns demonstrated that the constructive algorithm was capable of ongoing or continual one-shot learning.

## 8.2 Significance of Contributions (Revisited)

Contributions of this thesis and their significance for machine learning and computational neuroscience were outlined in Section 1.2.1. The contributions of this thesis have aimed to address specific limitations and gaps identified in the research literature (Section 2.3):

1. A design methodology for constructive neural networks suitable for the incremental development and analysis of constructive algorithms.

2. Theory for the development of constructive algorithms that are compatible with simulations of biological neural networks.

3. Development of constructive algorithm processes:

   (a) Compatible with continuous spiking neural network simulations.

   (b) Compatible with simulated studies that include STDP models.

   (c) Compatible with highly noisy conditions and periods of pure noise.

   (d) To reduce and remove extraneous constructed neurons.

4. Study of the behaviour, learning performance, and compatibility of constructive algorithms in simulations of STDP.

Developments in constructive neural networks have significance for the use of artificial neural networks. The constructive algorithms can add neurons to neural network models to respond to new or changing conditions, providing additional capacity for learning and to correct instances of poor performance (as demonstrated in this thesis). Objective processes evaluating the neural network performance are used to automate selection of the network size, relieving the need to hand-design aspects of the structure of the neural network model. Given objective processes that construct neurons, the constructed network size can also be used as an estimate of the number of spike patterns or features.

The methodology for the design and analysis of constructive neural networks developed and demonstrated in this thesis (Chapter 2) has significance for the field of machine learning and the development of constructive neural networks. The absence of design methodologies in the constructive neural networks literature is likely to be a significant factor in the research literature being fragmented in its exploration of the design space. The design methodology demonstrated can be used as a guide for the organisation of constructive neural networks research and the incremental development and analysis of constructive algorithm processes.

The concepts of simulation expansion and contraction proposed in this thesis (Chapter 3) were the basis for developing theory and principles for performing construction in simulations without introducing biological implausible effects in the model. This has substantial significance for the field of computational neuroscience. Simulated studies in neuroscience can take advantage of the benefits of constructive algorithms, changing the structure of simulated neural networks, while maintaining biological plausibility. Furthermore, simulation expansion and contraction concepts set the foundation for a new research topic: algorithms for dynamically selecting and simulating neurons in memory (Appendix A). This avenue of research (discussed further in Section 8.3) could potentially impact both the fields of machine learning and computational neuroscience.

Given the general compatibility of the developed constructive algorithm processes with spiking neuron models, continuous simulations, STDP models, and noisy neuron activity, the constructive algorithms developed are expected to be applicable to a wide range of neural network models. This gives the developed constructive algorithm processes and the learning performance demonstrated in this thesis a general significance and applicability in the fields of computational neuroscience and machine learning.

The processes for synapse weight calculation developed in this thesis accommodate different states of network learning or maturity. Neurons assumed to have zero or a low trace of past STDP updates may be constructed and then tuned to patterns activity. This constructive process may be preferred for simulated studies of the behaviour of neuroplasticity models such as STDP. Neurons assumed to have tuned through STDP to the point of synapse weight convergence may be added to a network to immediately detect spike patterns. This constructive process may be preferred for studies of the function and behaviour of mature and functional neural systems.

The developed constructive algorithms are also significant for their learning performance, demonstrating that new neurons immediately detect spike patterns and allow the neural network size and model capacity to adapt automatically in response to the appearance of new spike patterns (Section 7.6.2). Neural networks with a predefined structure demonstrated intrinsic limitations, including a limited capacity for storage and an inability to learn patterns introduced after an initial learning period. Constructed neurons and synapses did not affect existing neurons; therefore, the constructive algorithm did not cause forgetting through the retuning of synapse weights.

The impact that the contributions of this thesis will have depends on future work and research to extend and apply the thesis developments in computational neuroscience and machine learning.

## 8.3 Future Directions of Research

The future directions of research may take the form of applications, extensions and new research topics arising from the work presented in this thesis. The constructive algorithms and processes developed in this thesis may be applicable to a range of models for computational neuroscience and to tasks in machine learning. Given that no prior work has been found that considers the direct application of constructive algorithms to simulations of biological neural networks, and the novelty of the concepts of simulation expansion and contraction, there is fertile ground for new topics of study in these areas.

Neuroscientific studies may use simulations to develop and compare mathematical models with observations from biology, to predict the function and behaviour of neural systems, and to investigate computational interpretations of these neural functions. This thesis has proposed that neural system models may consider surrounding neurons and construct neurons to account for the existence of neurons outside the simulated network. Future research may investigate whether the function and behaviour of specific neural systems can be modelled more comprehensively with the application of the developed constructive algorithms. Neural systems involved with perception, such as the visual cortex (Olshausen, 2013), are well studied and may be good candidates for the application of constructive algorithms.

Neuron construction and pruning aim to ensure that the network model includes all neurons that have a significant function. This process may account for the growth of neurons and synapses, the death or pruning of neurons and synapses, and the gradual increase or decrease in neuron significance resulting from synaptic potentiation and depression. Future theoretical research may attempt to characterise the computational operations and capabilities that result from network expansion and contraction. This theoretical research topic may have similarities to studies of the computational effects of neuron growth (Aimone, Wiles, & Gage, 2009).

Models of other neural systems and neural functions, for example, the basal ganglia involved in action selection (Stewart, Bekolay, & Eliasmith, 2012), may also see benefits from constructive algorithms, but are likely to require the further development of algorithm processes. The plausibility of incorporating neuron construction and pruning in neural systems with distributed simultaneous activity and lateral connectivity is a topic that requires further study. This further study could result in the development of a range of new constructive algorithm processes that are compatible with simulation expansion and contraction principles for different neural network models.

The synapse weight calculation processes for constructive algorithms developed in this thesis are based on STDP models. Studies of biology have found that different types of neurons may demonstrate different relationships between the change in synapse efficacy and the relative spike timing of presynaptic and postsynaptic neurons (Caporale & Dan, 2008). The STDP models used in the constructive algorithms developed in this thesis do not account for the effects of spike rate. Future work may develop and investigate the performance of synapse weight calculations based on other models of synaptic plasticity, such as the incorporation of slower calcium ion concentration

transients (Graupner & Brunel, 2012).

The performance evaluation processes developed in this thesis are based on proxy neuron and simulated postsynaptic neuron activity. The evaluation of the performance of a simulated biological neural network may need calculations and conditions specific to the neural system being modelled. The developed performance evaluation processes are based on low-level neuron function of the neural network simulation; however, models of neural systems may have high-level population or feedback signals that indicate performance. The development of performance evaluation methods for neuron construction for specific models of biological neural systems may be a topic of future research.

Machine learning has not widely adopted spiking neural networks, much less constructive spiking neural networks. Future research may investigate the performance of constructive spiking neural networks in machine learning tasks. Given the intrinsic time-dependence of spiking neural network operation, constructive spiking neural networks may provide performance advantages in pattern detection in real-time data streams such as video, audio, and motion, and in time-dependent or time series data, such as economic and financial analysis and forecasting. Different tasks will likely require modifications to the network model (changing numbers of neurons and layers and the encoding inputs as spike times or spike rates) and constructive algorithm parameters. The application of constructive spiking neural networks to machine learning tasks will benefit from an exploration of methods for selecting parameters and network interfaces.

Few algorithms that perform spike-timing-dependent construction (STDC) were found in prior machine learning literature; therefore, it is likely that there are many possible designs that remain unexplored. Future research in STDC may focus on innovations in any of the base components of constructive neural networks. This topic of research may focus on machine learning tasks or on applications in computational neuroscience.

The developed constructive algorithm achieved continual one-shot learning of hidden repeating spike patterns for a given set of conditions. Performing continual learning through neuron construction avoids the retraining of synapse weights and may reduce the risk of catastrophic forgetting. Future research may investigate the performance of constructive algorithms in reducing forgetting in standard machine learning tasks (Goodfellow et al., 2013).

Recent machine learning research has made significant advances in performance using deep neural networks (Schmidhuber, 2015). Constructive algorithms have been applied to multilayer networks; however, constructive neural networks have received little attention in the deep learning literature. Literature surveyed in this thesis includes constructive algorithms that add neurons to a single layer of a deep multilayer perceptron (for example, Fukushima, 2014). Nevertheless, the design space for constructive algorithms and the performance in deep networks is far from exhaustively explored.

There are many potential approaches to constructing deep neural networks that could be topics of future research. Few constructive algorithms procedurally select the

number of layers or sets of neurons within the neural network structure (Fahlman & Lebiere, 1990). New constructive algorithm processes for adding layers to the network may be developed and investigated. Construction of deep neural networks may be performed at different rates (one neuron at a time or multiple neurons concurrently) and sequences (one layer at a time or concurrently across multiple layers). These constructive processes may aim to produce simple detectors of features in the previous layer, reproducing the observed result of deep neural network training (Krizhevsky et al., 2012).

The implementation of constructive algorithms to produce deep neural networks may have increased susceptibility to over-fitting (Larochelle, Bengio, Louradour, & Lamblin, 2009). Although over-fitting can be a significant problem in deep learning, it can also be desirable to perform fast learning on few new training examples without overwriting past training results (Vinyals, Blundell, Lillicrap, Kavukcuoglu, & Wierstra, 2016). Future research could investigate the extent of over-fitting from different neuron construction processes and the use of neuron construction as a method for one-shot learning with deep neural networks.

Performing simulation expansion and contraction as the selective simulation of neurons in memory (see Appendix A) presents a new topic of study. At present, neural network simulations typically update all neurons in memory at all update steps. The computational resources required for this approach can reduce the feasibility of operating with large neural networks, such as large-scale brain simulations (de Garis, Shuo, Goertzel, & Ruiting, 2010). The selective simulation of stored neurons has conceptual parallels to the hypothesised function of attention mechanisms in the human brain and the selective activity modulation of different brain regions (Corbetta & Shulman, 2002; Petersen & Posner, 2012). Attention mechanisms in the brain include top-down and bottom-up signals which may be modelled to select sets of neurons in large-scale brain models for simulation.

Future research could aim to develop methods for dynamically predicting and selecting the most significant neurons in memory. A successful neuron selection algorithm could allow model function and behaviour to be accurately simulated while updating smaller subsets of the stored network model. Similar to the reduction in metabolic costs from selective activation of brain areas, the development of methods for the selective simulation of neurons could potentially reduce the computational cost of large-scale brain simulations. The reduction in computational expense from selective neuron simulation will depend on the cost of processes for neuron selection and the number of neurons that can be excluded from the simulation at any moment. Processes for dynamic selection of neurons to simulate may also have applications in deep neural networks for machine learning.

The prediction of neurons to simulate may potentially be performed with proxy neurons that represent sets of neurons in memory. A proxy neuron spike would predict that there will be activity in the associated neuron set and that the neuron set should be simulated to find the specific activity. This approach would require the development of methods for grouping neurons in memory into sets and selecting proxy neuron

parameters such that activity in the set of neurons in memory is accurately predicted from the simulation of a smaller set of representative proxy neurons.

Future research topics of constructive algorithms and selective neuron simulation may also be combined. This combination of algorithms may provide the advantages of constructive algorithm processes (continual one-shot learning without overwriting synapse weights) and advantages of selective neuron simulation (avoiding excessive computational costs from operating very large neural networks). If successful, this approach to simulating neural networks and learning may have profound effects on future work in computational neuroscience and machine learning

# Appendix A

# Network Sets and Expansion

This thesis has introduced the concepts of simulation expansion and contraction based on definitions of the sets of neurons and synapses that are simulated, surrounding and in memory (Chapter 3). This appendix provides set notation to describe the relationship between these sets of neurons and synapses and extends the concept of simulation expansion and contraction with a network stored in memory.

## A.1 Neuron Sets: Simulated, Surrounding and Memory

The simulated neurons and synapses, $N_{\text{sim}}$ and $S_{\text{sim}}$, are assumed to be subsets of a larger neural system with neurons $N$ and synapses $S$. The sets of surrounding neurons and synapses, $N_{\text{sur}}$ and $S_{\text{sur}}$, are the neurons in the larger neural system that are not simulated. A graphical representation of the relationships between the simulated and surrounding sets of neurons and synapses is presented in Figure 3.1.

The relationship between these sets can be represented with the relative complement ($\setminus$) of the neural system and the simulated sets,

$$\begin{aligned} N_{\text{sur}} &= N \setminus N_{\text{sim}} = N_{\text{sim}}^{\complement}, \text{ and} \\ S_{\text{sur}} &= S \setminus S_{\text{sim}} = S_{\text{sim}}^{\complement}. \end{aligned} \tag{A.1}$$

Equivalently, the sets of all neurons and all synapses in the neural system are the union ($\cup$) of the set of simulated neurons and surrounding neurons,

$$\begin{aligned} N &= N_{\text{sim}} \cup N_{\text{sur}}, \text{ and} \\ S &= S_{\text{sim}} \cup S_{\text{sur}}. \end{aligned} \tag{A.2}$$

Note that the sets of simulated neurons and simulated synapses and the sets of surrounding neurons and surrounding synapses are by definition disjoint (they have no shared members). In set notation this may be represented as,

$$\begin{aligned} N_{\text{sim}} \cap N_{\text{sur}} &= \varnothing, \text{ and} \\ S_{\text{sim}} \cap S_{\text{sur}} &= \varnothing, \end{aligned} \tag{A.3}$$

where $\varnothing$ denotes the empty set (no members).

## A.2 Expansion, Construction, Contraction and Pruning

Constructive algorithms reported in literature are often not accompanied by a detailed implementation in program code; however, the construction and pruning of neurons and synapses from an ANN can often be assumed to coincide with the creation or deletion of neurons and synapses from the computer memory. The construction of neurons and synapses has been defined as the addition of these components to the computer memory. In set notation this may be represented as the addition of a neuron or synapse to the memory sets,

$$N_{\text{mem}} \leftarrow N_{\text{mem}} \cup n, \tag{A.4}$$

$$S_{\text{mem}} \leftarrow S_{\text{mem}} \cup s. \tag{A.5}$$

Pruning of neurons and synapses has been defined as the removal of these components from the computer memory. In set notation this may be represented as the removal of a neuron or synapse with the memory sets,

$$N_{\text{mem}} \leftarrow N_{\text{mem}} \setminus n, \tag{A.6}$$

$$S_{\text{mem}} \leftarrow S_{\text{mem}} \setminus s. \tag{A.7}$$

Expansion of a neural network simulation has been defined as a transfer of neurons and synapses from the sets of surrounding neurons and synapses, $N_{\text{sur}}$ and $S_{\text{sur}}$, to the sets of simulated neurons and synapses, $N_{\text{sim}}$ and $S_{\text{sim}}$. In set notation, the expansion of the set of simulated neurons $N_{\text{sim}}$ to include neuron $n \in N_{\text{sur}}$ can be represented as two set operations:

$$\begin{aligned} N_{\text{sim}} &\leftarrow N_{\text{sim}} \cup n, \text{ then} \\ N_{\text{sur}} &\leftarrow N_{\text{sur}} \setminus n. \end{aligned} \tag{A.8}$$

Similarly for synapses, the expansion of the set of simulated synapses $S_{\text{sim}}$ to include synapse $s \in S_{\text{sur}}$ can be represented as two set operations:

$$\begin{aligned} S_{\text{sim}} &\leftarrow S_{\text{sim}} \cup s, \text{ then} \\ S_{\text{sur}} &\leftarrow S_{\text{sur}} \setminus s. \end{aligned} \tag{A.9}$$

Similar to expansion, contraction of a neural network simulation has been defined as a transfer of neurons and synapses from the simulated neuron and synapse sets, $N_{\text{sim}}$ and $S_{\text{sim}}$, to the surrounding neuron and synapse sets, $N_{\text{sur}}$ and $S_{\text{sur}}$. In set notation, the contraction of the set of simulated neurons $N_{\text{sim}}$ to remove neuron $n \in N_{\text{sim}}$ can be defined as two set operations:

$$\begin{aligned} N_{\text{sur}} &\leftarrow N_{\text{sur}} \cup n, \text{ then} \\ N_{\text{sim}} &\leftarrow N_{\text{sim}} \setminus n. \end{aligned} \tag{A.10}$$

Similarly for synapses, the contraction of the set of simulated synapses $S_{\text{sim}}$ to remove synapse $s \in S_{\text{sim}}$ can be defined as two set operations:

$$\begin{aligned} S_{\text{sur}} &\leftarrow S_{\text{sur}} \cup n, \text{ then} \\ S_{\text{sim}} &\leftarrow S_{\text{sim}} \setminus s. \end{aligned} \tag{A.11}$$

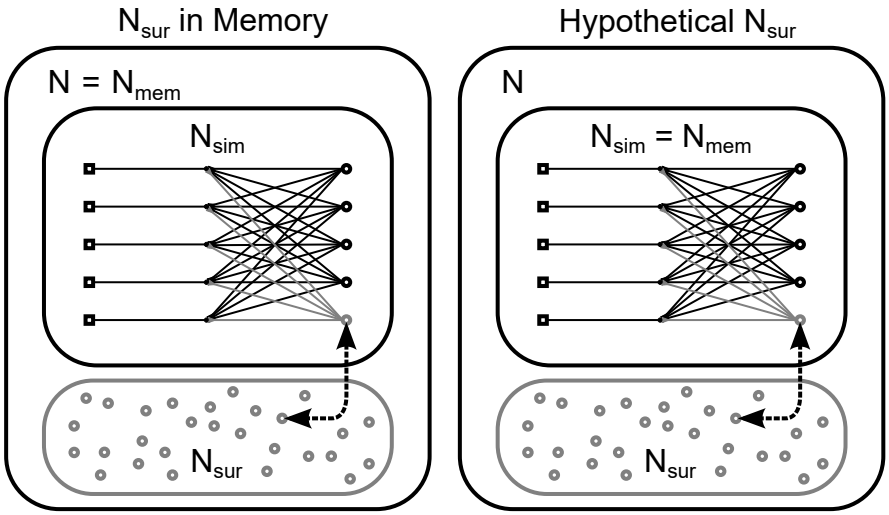


**Figure A.1:** Graphical representations of simulation expansion and contraction as a process of selecting neurons stored in memory (left) and creating or deleting neurons in memory with an assumed surrounding network (right).

## A.3 Hypothetical Neurons and Memory

An algorithm designed to perform simulation expansion and contraction assumes that the neurons and synapses exist before and after the transfer between the simulated and surrounding network. This assumption does not require that surrounding neurons and synapses exist in computer memory; it may be satisfied with a hypothetical surrounding network. The concepts of simulation expansion and neuron construction overlap when the surrounding network is hypothetical and not in the computer memory.

Algorithms that perform simulation expansion and contraction with a hypothetical surrounding network may be implemented to create and delete neurons and synapses from memory. Simulation expansion and contraction can also be performed with a surrounding network that is stored in the computer memory. The transfer of neurons to the simulated network performed by these algorithms may be implemented as a process of selecting neurons and synapses in memory to simulate.

The difference in algorithms that select neurons in memory and those that create neurons in memory can be described in set theory notation and presented graphically in Figure A.1. In summary, an algorithm that performs simulation expansion and construction through the selection of neurons and synapses in memory can have the set of all neurons equivalent to the set of neurons in memory, $N = N_{\text{mem}}$ (and for synapses, $S = S_{\text{mem}}$).

An algorithm that has a hypothetical surrounding network may have the sets of simulated neurons and synapses equivalent to the sets in memory, $N_{\text{sim}} = N_{\text{mem}}$ and $S_{\text{sim}} = S_{\text{mem}}$. Then the surrounding neurons and synapses are those not in memory, $N_{\text{sur}} = N_{\text{mem}}^{\complement}$ and $S_{\text{sur}} = S_{\text{mem}}^{\complement}$.

It is also possible a neural network could operate with simulation expansion algorithms that perform both processes: the selection of neurons in memory to simulate, and the creation and deletion of neurons in memory. The sets of surrounding neurons and synapses remain, by definition, the respective complements of the simulated neurons ($N_{\text{sur}} = N_{\text{sim}}^{\complement}$) and simulated synapses ($S_{\text{sur}} = S_{\text{sim}}^{\complement}$). However, now the surrounding sets include hypothetical neurons and synapses and a selection of the sets of neurons and synapses in memory.

There is potential for future research in the development of algorithms that dynamically select simulated sets of neurons and synapses from those stored in memory. Algorithms can efficiently predict which neurons in the memory are important to update may produce significant reductions in computational cost of large neural networks simulations. This could allow very large neural networks to be implemented on fewer and lower-powered computer processors.

# Appendix B

# STDP Estimate Derivations

Equations for calculating synapse weights for neuron construction were developed from spike-timing-dependent plasticity (STDP) estimates (see Section 3.2.3). Derivations for equations for estimating multiplicative STDP are provided here.

The estimates are derived from the online nearest-neighbour STDP model. The online updating of the synapse weight, $w_{i,j}$, between postsynaptic neuron $i$ and presynaptic neuron $j$ occurs at the times of neuron spikes. The nearest neighbour STDP model only updates synapse weights for the first postsynaptic neuron spike after a presynaptic neuron spike and the first presynaptic neuron spike after a postsynaptic neuron spike. For further details see Section 3.2.1.

The estimation of past STDP assumes that a predicted relative spike timing of neurons was repeated in the past and was responsible for all past synaptic plasticity of significance. Therefore STDP estimates can be represented simply in terms of the discrete number of iterations, $m$, the positive STDP factor, $\Delta w_+$ for presynaptic then postsynaptic spike order, and the negative STDP factor, $-\Delta w_-$ for postsynaptic then presynaptic spike order. The values of $\Delta w_+$ and $\Delta w_-$ are calculated based on the recorded relative spike timing of neurons and the STDP curve (Figure 3.4).

In summary, the equations for positive multiplicative STDP updates used in derivations can be represented,

$$\begin{aligned} w_{i,j}[m] &= w_{i,j}[m-1] + (w_{\max} - w_{i,j}[m-1]) \cdot \Delta w_+, \\ w_{i,j}[m] &= (1 - \Delta w_+) w_{i,j}[m-1] + w_{\max} \Delta w_+. \end{aligned} \tag{B.1}$$

The equations for negative multiplicative STDP updates can be represented,

$$\begin{aligned} w_{i,j}[m] &= w_{i,j}[m-1] + (w_{\min} - w_{i,j}[m-1]) \cdot \Delta w_-, \\ w_{i,j}[m] &= (1 - \Delta w_-) w_{i,j}[m-1] + w_{\min} \Delta w_-. \end{aligned} \tag{B.2}$$

## B.1 Sum of Multiplicative STDP Updates

Derivations for synapse weight calculations of multiplicative STDP updates for $M$ iterations of a spike pair or spike triplet are produced using the sum of a geometric series.

Two cases are provided here: the only eligible presynaptic spike produces depression; and eligible presynaptic spikes occur in a spike triplet with presynaptic spikes before and after a predicted postsynaptic spike time resulting in a depressing and potentiating update.

### B.1.1 Multiplicative STDP Depression

This section presents the derivation of an equation for the direct calculation of a synapse weight based on an estimate of $M$ iterations of a depressing multiplicative STDP update. First, the equation for multiplicative STDP updates for depression can be applied iteratively on the initial synapse weight $w_{i,j}[0]$,

$$\begin{aligned}
w_{i,j}[1] &= (1-\Delta w_-)w_{i,j}[0] + w_{\min}\Delta w_-,\\
w_{i,j}[2] &= (1-\Delta w_-)w_{i,j}[1] + w_{\min}\Delta w_-,\\
w_{i,j}[2] &= (1-\Delta w_-)^2 w_{i,j}[0] + (1-\Delta w_-)w_{\min}\Delta w_- + w_{\min}\Delta w_-,\\
w_{i,j}[3] &= (1-\Delta w_-)^3 w_{i,j}[0] + (1-\Delta w_-)^2 w_{\min}\Delta w_-\\
&\quad + (1-\Delta w_-)w_{\min}\Delta w_- + w_{\min}\Delta w_-,\\
&\vdots\\
w_{i,j}[M] &= (1-\Delta w_-)^M w_{i,j}[0] + (1-\Delta w_-)^{M-1} w_{\min}\Delta w_- + \ldots\\
&\quad + (1-\Delta w_-)^2 w_{\min}\Delta w_- + (1-\Delta w_-)w_{\min}\Delta w_- + w_{\min}\Delta w_-. \qquad \text{(B.3)}
\end{aligned}$$

The sum of a geometric series with a base of $w_{\min}\Delta w_-$ and ratio of $(1-\Delta w_-)$ can be extracted,

$$\begin{aligned}
S_{\mathrm{D},M-1} &= (1-\Delta w_-)^{M-1} w_{\min}\Delta w_- + (1-\Delta w_-)^{M-2} w_{\min}\Delta w_- + \ldots\\
&\quad + (1-\Delta w_-)^2 w_{\min}\Delta w_- + (1-\Delta w_-)w_{\min}\Delta w_- + w_{\min}\Delta w_-.
\end{aligned}$$

The solution to this sum of a geometric series may be simplified through expansion using a factor $[1-(1-\Delta w_-)]$ and then cancelling terms,

$$\begin{aligned}
[1-(1-\Delta w_-)]\cdot S_{\mathrm{D},M-1} &= (1-\Delta w_-)^{M-1} w_{\min}\Delta w_-\\
&\quad + \ldots\\
&\quad + (1-\Delta w_-)w_{\min}\Delta w_-\\
&\quad + w_{\min}\Delta w_-\\
&\quad - (1-\Delta w_-)^M w_{\min}\Delta w_-\\
&\quad - (1-\Delta w_-)^{M-1} w_{\min}\Delta w_-\\
&\quad - \ldots\\
&\quad - (1-\Delta w_-)w_{\min}\Delta w_-,\\
[1-(1-\Delta w_-)]\cdot S_{\mathrm{D},M-1} &= w_{\min}\Delta w_- - (1-\Delta w_-)^M w_{\min}\Delta w_-,
\end{aligned}$$

$$S_{\mathrm{D},M-1} = \frac{w_{\min}\Delta w_- - (1-\Delta w_-)^M w_{\min}\Delta w_-}{1-(1-\Delta w_-)},$$
$$S_{\mathrm{D},M-1} = w_{\min} - (1-\Delta w_-)^M w_{\min}. \tag{B.4}$$

Substituting the sum of the geometric series (Equation B.4) into the weight after the $M^{\mathrm{th}}$ update (Equation B.3) gives a direct solution:

$$\begin{aligned} w_{i,j}[M] &= (1-\Delta w_-)^M w_{i,j}[0] + S_{D,M-1}, \\ w_{i,j}[M] &= (1-\Delta w_-)^M w_{i,j}[0] + w_{\min} - (1-\Delta w_-)^M w_{\min}, \\ w_{i,j}[M] &= w_{\min} + (w_{i,j}[0] - w_{\min})(1-\Delta w_-)^M. \end{aligned} \tag{B.5}$$

This synapse weight resulting from discrete updates can be used as the weight value at the construction time,

$$\hat{w}_{i,j}(t_c) = w_{i,j}[M] = w_{\min} + (w_{i,j}[0] - w_{\min})(1-\Delta w_-)^M. \tag{B.6}$$

This concludes the derivation of the synapse weight calculation based on $M$ iterations of a depressing multiplicative STDP update.

### B.1.2 Multiplicative STDP Spike Triplet

This section presents the derivation of an equation for the direct calculation of a synapse weight based on an estimate of $M$ iterations of a pre-post-pre spike triplet that produces both depressing and potentiating multiplicative STDP updates. The equations for multiplicative STDP updates can be applied iteratively on the initial synapse weight $w_{i,j}[0]$. The first update results from the pre-post spike pair and causes an increase in synapse weight,

$$w^*_{i,j}[1] = (1-\Delta w_+) w_{i,j}[0] + w_{\max}\Delta w_+. \tag{B.7}$$

Here the superscript asterisk, $w^*_{i,j}[m]$, is used to denote the intermediate weight value in the $m^{\mathrm{th}}$ iteration of the spike triplet. Next the post-pre spike pair occurs and decreases the synapse weight,

$$\begin{aligned} w_{i,j}[1] &= (1-\Delta w_-) w^*_{i,j}[1] + w_{\min}\Delta w_- \\ &= (1-\Delta w_-)[(1-\Delta w_+) w_{i,j}[0] + w_{\max}\Delta w_+] + w_{\min}\Delta w_- \\ &= (1-\Delta w_+)(1-\Delta w_-) w_{i,j}[0] + (1-\Delta w_-) w_{\max}\Delta w_+ + w_{\min}\Delta w_-. \end{aligned} \tag{B.8}$$

This can be repeated iteratively up to $M$ iterations,

$$
\begin{aligned}
w^*_{i,j}[2] &= (1-\Delta w_+)w_{i,j}[1] + w_{\max}\Delta w_+, \\
&= (1-\Delta w_+)[(1-\Delta w_+)(1-\Delta w_-)w_{i,j}[0] + (1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-] \\
&\quad + w_{\max}\Delta w_+, \\
&= (1-\Delta w_+)^2(1-\Delta w_-)w_{i,j}[0] + (1-\Delta w_+)(1-\Delta w_-)w_{\max}\Delta w_+ \\
&\quad + (1-\Delta w_+)w_{\min}\Delta w_- + w_{\max}\Delta w_+, \\
w_{i,j}[2] &= (1-\Delta w_-)w^*_{i,j}[2] + w_{\min}\Delta w_-, \\
&= (1-\Delta w_-)[(1-\Delta w_+)^2(1-\Delta w_-)w_{i,j}[0] + (1-\Delta w_+)(1-\Delta w_-)w_{\max}\Delta w_+ \\
&\quad + (1-\Delta w_+)w_{\min}\Delta w_- + w_{\max}\Delta w_+] + w_{\min}\Delta w_-, \\
&= (1-\Delta w_+)^2(1-\Delta w_-)^2 w_{i,j}[0] + (1-\Delta w_+)(1-\Delta w_-)^2 w_{\max}\Delta w_+ \\
&\quad + (1-\Delta w_+)(1-\Delta w_-)w_{\min}\Delta w_- + (1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-, \\
&\vdots \\
w_{i,j}[M] &= (1-\Delta w_+)^M(1-\Delta w_-)^M w_{i,j}[0] \\
&\quad + (1-\Delta w_+)^{M-1}(1-\Delta w_-)^M w_{\max}\Delta w_+ \\
&\quad + (1-\Delta w_+)^{M-1}(1-\Delta w_-)^{M-1} w_{\min}\Delta w_- \\
&\quad + (1-\Delta w_+)^{M-2}(1-\Delta w_-)^{M-1} w_{\max}\Delta w_+ \\
&\quad + \dots \\
&\quad + (1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-. \qquad \text{(B.9)}
\end{aligned}
$$

Excluding the first term, $(1-\Delta w_+)^M(1-\Delta w_-)^M w_{i,j}[0]$, the terms for $w_{i,j}[M]$ can be grouped in pairs to produce a geometric series with a base, $b = (1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-$, and a ratio, $r = (1-\Delta w_+)(1-\Delta w_-)$,

$$
\begin{aligned}
S_{\mathrm{ST},M-1} &= (1-\Delta w_+)^{M-1}(1-\Delta w_-)^M w_{\max}\Delta w_+ \\
&\quad + (1-\Delta w_+)^{M-1}(1-\Delta w_-)^{M-1} w_{\min}\Delta w_- \\
&\quad + (1-\Delta w_+)^{M-2}(1-\Delta w_-)^{M-1} w_{\max}\Delta w_+ \\
&\quad + \dots \\
&\quad + (1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-, \\
S_{\mathrm{ST},M-1} &= [(1-\Delta w_+)(1-\Delta w_-)]^{M-1}[(1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-] \\
&\quad + [(1-\Delta w_+)(1-\Delta w_-)]^{M-2}[(1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-] \\
&\quad + \dots \\
&\quad + [(1-\Delta w_-)w_{\max}\Delta w_+ + w_{\min}\Delta w_-], \\
&= r^{M-1}b + r^{M-2}b + \dots + rb + b.
\end{aligned}
$$

The sum of the geometric series to $M-1$ terms can be simplified through multiplication with a factor $[1-r]$ and then cancelling terms,

$$\begin{aligned}
[1-r]\cdot S_{\mathrm{ST},M-1} &= r^{M-1}b + r^{M-2}b + \cdots + rb + b \\
&\quad - r^{M}b - r^{M-1}b - \cdots - r^{2}b - rb, \\
&= b - r^{M}b, \\
S_{\mathrm{ST},M-1} &= \frac{b - r^{M}b}{1-r}.
\end{aligned} \tag{B.10}$$

Substituting the sum of the geometric series (Equation B.10) into the equation for the weight after $M$ iterations of the spike triplet (Equation B.9) gives a direct solution,

$$\begin{aligned}
w_{i,j}[M] &= (1-\Delta w_+)^M (1-\Delta w_-)^M w_{i,j}[0] + S_{\mathrm{ST},M-1}, \\
w_{i,j}[M] &= r^M w_{i,j}[0] + \frac{b - r^M b}{1-r}, \\
w_{i,j}[M] &= \frac{b}{1-r} + \Big(w_{i,j}[0] - \frac{b}{1-r}\Big) r^M.
\end{aligned} \tag{B.11}$$

This equation has an asymptote value,

$$\begin{aligned}
A_{\mathrm{ST}} &= \frac{b}{1-r}, \\
&= \frac{w_{\max}\Delta w_+ (1-\Delta w_-) + w_{\min}\Delta w_-}{1-(1-\Delta w_+)(1-\Delta w_-)}.
\end{aligned} \tag{B.12}$$

Substituting this asymptote and the value of the ratio, $r$, back into the equation, a direct calculation for the synapse weight from $M$ discrete updates is found. This equation can be used as the weight value at the neuron construction time,

$$\hat{w}_{i,j}(t_c) = w_{i,j}[M] = A_{\mathrm{ST}} + (w_{i,j}[0] - A_{\mathrm{ST}})(1-\Delta w_+)^M (1-\Delta w_-)^M, \tag{B.13}$$

This concludes the derivation of the synapse weight calculation based on $M$ iterations of a spike triplet with multiplicative STDP updates.

## B.2 Multiplicative STDP Continuous Approximation

Methods for calculating new synapse weights using a continuous approximation of multiplicative STDP have been presented in this thesis. The iterative updates of synapse weights from multiplicative STDP can be represented as a continuous function on the number of iterations $m$ of a spike pair or spike triplet. The derivations of these synapse weight calculations for construction are presented in this section.

### B.2.1 Continuous Approximation for Spike Pairs

An incremental change in the synapse weight from a multiplicative STDP update can be approximated as linear increase over the change $\Delta m$ in the number of iterations $m$,

$$\hat{w}_{i,j}(m) = \hat{w}_{i,j}(m - \Delta m) + [w_{\max} - \hat{w}_{i,j}(m - \Delta m)]\Delta w_+ \Delta m, \tag{B.14a}$$

$$\hat{w}_{i,j}(m) = \hat{w}_{i,j}(m - \Delta m) + [w_{\min} - \hat{w}_{i,j}(m - \Delta m)]\Delta w_- \Delta m, \tag{B.14b}$$

for the positive STDP curve value, $\Delta w_+$, for a pre-post spike pair, and for the negative STDP curve value, $\Delta w_-$, for a post-pre spike pair, respectively. The linear approximation of the change in synapse weight can be represented,

$$\Delta\hat{w}_{i,j}(m) = \hat{w}_{i,j}(m) - \hat{w}_{i,j}(m - \Delta m) = [w_{\max} - \hat{w}_{i,j}(m - \Delta m)]\Delta w_+ \Delta m, \tag{B.15a}$$

$$\Delta\hat{w}_{i,j}(m) = \hat{w}_{i,j}(m) - \hat{w}_{i,j}(m - \Delta m) = [w_{\min} - \hat{w}_{i,j}(m - \Delta m)]\Delta w_- \Delta m. \tag{B.15b}$$

Taking the limit of $\Delta m \to 0$ gives a continuous function for the rate of change of the synapse weight. Taking the case of the positive synapse weight update,

$$\begin{aligned}
\dot{\hat{w}}_{i,j}(m) &= \lim_{\Delta m \to 0} \frac{\Delta\hat{w}_{i,j}(m)}{\Delta m}, \\
&= \lim_{\Delta m \to 0} \frac{[w_{\max} - \hat{w}_{i,j}(m - \Delta m)]\Delta w_+ \Delta m}{\Delta m}, \\
&= -\Delta w_+ \hat{w}_{i,j}(m) + w_{\max}\Delta w_+.
\end{aligned} \tag{B.16a}$$

The rate of change for negative updates can be similarly found,

$$\dot{\hat{w}}_{i,j}(m) = -\Delta w_- \hat{w}_{i,j}(m) + w_{\min}\Delta w_-. \tag{B.16b}$$

Choosing an integrating factor $\mu(m) = \exp(\Delta w_+ m)$, $\dot{\mu}(m) = \mu(m)\Delta w_+$, allows a solution to the differential equation to be found,

$$\begin{aligned}
\mu(m)\dot{\hat{w}}_{i,j}(m) + \mu(m)\Delta w_+ \hat{w}_{i,j}(m) &= \mu(m) w_{\max}\Delta w_+, \\
\mu(m)\dot{\hat{w}}_{i,j}(m) + \dot{\mu}(m)\hat{w}_{i,j}(m) &= \mu(m) w_{\max}\Delta w_+, \\
\frac{d}{dm}(\mu(m)\hat{w}_{i,j}(m)) &= \mu(m) w_{\max}\Delta w_+.
\end{aligned}$$

Integrating both sides with respect to $m$,

$$\begin{aligned}
\mu(m)\hat{w}_{i,j}(m) &= \mu(m) w_{\max} + C, \\
\hat{w}_{i,j}(m) &= w_{\max} + C/\mu(m), \\
\hat{w}_{i,j}(m) &= w_{\max} + C \cdot \exp(-\Delta w_+ m).
\end{aligned}$$

Given an initial value $\hat{w}_{i,j}(0) = w_{i,j}[0]$, $C = w_{i,j}[0] - w_{\max}$. The same process gives the solution for the case of a depressing spike pair. The resulting continuous approximation

can be used to calculate the synapse weights of a new neuron constructed at time $t_c$ for $M$ iterations of the spike pair, matching the equations presented in Section 3.2.3.3,

$$\hat{w}_{i,j}(t_c) = \begin{cases} w_{\max} + (w_{i,j}[0] - w_{\max}) \cdot \exp(-M \cdot \Delta w_+) & \text{if } \Delta t_{i,j}^{(f,g)} < 0, \\ w_{\min} + (w_{i,j}[0] - w_{\min}) \cdot \exp(-M \cdot \Delta w_-) & \text{if } \Delta t_{i,j}^{(f,g)} \geq 0. \end{cases} \tag{B.17}$$

Note that when applied to synapse weight calculations for neuron construction, the continuous variable $m$ only takes positive integer values representing the full number of STDP iterations, $M$, assumed to have occurred prior and up to construction.

### B.2.2 Continuous Approximation for Spike Triplets

A function for calculating synapse weights using a continuous approximation of the number of iterations of a spike triplet iterations has been developed in this thesis. The order of spikes in the triplet is not considered; therefore, the synapse weight function for a spike triplet is taken as a linear combination of the STDP update equations for depression and potentiation,

$$\dot{\hat{w}}_{i,j}(m) = -(\Delta w_+ + \Delta w_-)\hat{w}_{i,j}(m) + w_{\max}\Delta w_+ + w_{\min}\Delta w_-. \tag{B.18}$$

Now the integrating factor chosen is

$$\mu(m) = \exp([\Delta w_+ + \Delta w_-]m), \tag{B.19}$$
$$\dot{\mu}(m) = \mu(m)(\Delta w_+ + \Delta w_-), \tag{B.20}$$

and the equation for the synapse weight is found,

$$\begin{aligned} \mu(m)\dot{\hat{w}}_{i,j}(m) + \mu(m)(\Delta w_+ + \Delta w_-)\hat{w}_{i,j}(m) &= \mu(m)(w_{\max}\Delta w_+ + w_{\min}\Delta w_-), \\ \mu(m)\dot{\hat{w}}_{i,j}(m) + \dot{\mu}(m)\hat{w}_{i,j}(m) &= \mu(m)(w_{\max}\Delta w_+ + w_{\min}\Delta w_-), \\ \frac{d}{dm}(\mu(m)\hat{w}_{i,j}(m)) &= \mu(m)(w_{\max}\Delta w_+ + w_{\min}\Delta w_-). \end{aligned}$$

Integrating both sides with respect to $m$,

$$\begin{aligned} \mu(m)\hat{w}_{i,j}(m) &= \mu(m)\frac{w_{\max}\Delta w_+ + w_{\min}\Delta w_-}{\Delta w_+ + \Delta w_-} + C, \\ \hat{w}_{i,j}(m) &= \frac{w_{\max}\Delta w_+ + w_{\min}\Delta w_-}{\Delta w_+ + \Delta w_-} + C/\mu(m), \\ \hat{w}_{i,j}(m) &= \frac{w_{\max}\Delta w_+ + w_{\min}\Delta w_-}{\Delta w_+ + \Delta w_-} + C \cdot \exp(-[\Delta w_+ + \Delta w_-]m). \end{aligned}$$

An asymptote value for the synapse weight approximation can be defined,

$$A_{\mathrm{STa}} = \frac{w_{\max}\Delta w_+ + w_{\min}\Delta w_-}{\Delta w_+ + \Delta w_-}. \tag{B.21}$$

Given an initial value $\hat{w}_{i,j}(0) = w_{i,j}[0]$, $C = w_{i,j}[0] - A_{\text{STa}}$. Substituting these values into the synapse weight function, $\hat{w}_{i,j}(m)$, the calculation of synapse weights at the time of neuron construction, $t_c$, presented in Section 3.2.3.3 is found,

$$\hat{w}_{i,j}(t_c) = A_{\text{STa}} + (w_{i,j}[0] - A_{\text{STa}}) \cdot \exp(-M \cdot [\Delta w_+ + \Delta w_-]). \tag{B.22}$$

This concludes the derivations of synapse weight calculations for neuron construction.

# References


Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity: Taming the beast. *Nature Neuroscience*, *3*, 1178–1183. doi: 10.1038/81453
7

Aimone, J. B., Li, Y., Lee, S. W., Clemenson, G. D., Deng, W., & Gage, F. H. (2014). Regulation and function of adult neurogenesis: From genes to cognition. *Physiological Reviews*, *94*(4), 991–1026. doi: 10.1152/physrev.00004.2014
10

Aimone, J. B., Wiles, J., & Gage, F. H. (2009). Computational influence of adult neurogenesis on memory encoding. *Neuron*, *61*(2), 187–202. doi: 10.1016/j.neuron.2008.11.026
189

Arena, P., Fortuna, L., Frasca, M., & Patané, L. (2009). Learning anticipation via spiking networks: Application to navigation control. *IEEE Transactions on Neural. Networks*, *20*(2), 202–216. doi: 10.1109/TNN.2008.2005134
4, 8

Ash, T. (1989). Dynamic node creation in backpropagation networks. *Connection Science*, *1*(4), 365–375. doi: 10.1080/09540098908915647
8, 9, 17, 18, 20, 21

Atsumi, M. (2005). Saliency-driven scene learning and recognition based on competitively growing neural network using temporal coding. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, *9*(3), 235–243. doi: 10.20965/jaciii.2005.p0235
10

Bi, G.-q., & Poo, M.-m. (1998). Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of Neuroscience*, *18*(24), 10464–10472. Retrieved from `http://www.jneurosci.org/content/18/24/10464`
7, 52

Blättler, F., & Hahnloser, R. H. R. (2011). An efficient coding hypothesis links sparsity and selectivity of neural responses. *PLOS ONE*, *6*(10), 1–17. doi: 10.1371/journal.pone.0025506
182

Blumenfeld, H. (2010). *Neuroanatomy through clinical cases* (2nd ed.). Sunderland,

MA: Sinauer Associates.
4

Bouganis, A., & Shanahan, M. (2010). Training a spiking neural network to control a 4-DoF robotic arm based on spike timing-dependent plasticity. In *The 2010 International Joint Conference on Neural Networks (IJCNN).* doi: 10.1109/IJCNN.2010.5596525
8

Brader, J. M., Senn, W., & Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Computation*, *19*(11), 2881–2912. doi: 10.1162/neco.2007.19.11.2881
33

Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., ... Destexhe, A. (2007). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, *23*(3), 349–398. doi: 10.1007/s10827-007-0038-6
20, 44, 52, 53

Butz, M., Lehmann, K., Dammasch, I. E., & Teuchert-Noodt, G. (2006). A theoretical network model to analyse neurogenesis and synaptogenesis in the dentate gyrus. *Neural Networks*, *19*(10), 1490–1505. doi: 10.1016/j.neunet.2006.07.007
10

Caporale, N., & Dan, Y. (2008). Spike timing-dependent plasticity: A Hebbian learning rule. *Annual Review of Neuroscience*, *31*(1), 25–46. doi: 10.1146/annurev.neuro.31.060407.125639
7, 21, 52, 53, 121, 189

Corbetta, M., & Shulman, G. L. (2002). Control of goal-directed and stimulus-driven attention in the brain. *Nature Reviews Neuroscience*, *3*(3), 201–215. doi: 10.1038/nrn755
191

Dayan, P., & Abbott, L. F. (2001). *Theoretical neuroscience: Computational and mathematical modeling of neural systems.* Cambridge, MA: MIT Press.
3, 21, 78

de Garis, H., Shuo, C., Goertzel, B., & Ruiting, L. (2010). A world survey of artificial brain projects, Part I: Large-scale brain simulations. *Neurocomputing*, *74*(1–3), 3–29. doi: 10.1016/j.neucom.2010.08.004
191

Delorme, A., & Thorpe, S. J. (2001). Face identification using one spike per neuron: Resistance to image degradations. *Neural Networks*, *14*(6–7), 795–803. doi: 10.1016/S0893-6080(01)00049-1
42

Dora, S., Subramanian, K., Suresh, S., & Sundararajan, N. (2016). Development of a self-regulating evolving spiking neural network for classification problem. *Neurocomputing*, *171*, 1216–1229. doi: 10.1016/j.neucom.2015.07.086
22, 30, 31, 34, 35, 37, 38, 39, 43

Dora, S., Sundaram, S., & Sundararajan, N. (2015). A two stage learning algorithm for a Growing-Pruning Spiking Neural Network for pattern classification problems. In *2015 International Joint Conference on Neural Networks (IJCNN)*. doi: 10.1109/IJCNN.2015.7280592
22, 31, 34, 35, 37, 38, 39

Dora, S., Suresh, S., & Sundararajan, N. (2015). A sequential learning algorithm for a spiking neural classifier. *Applied Soft Computing*, *36*, 255–268. doi: 10.1016/j.asoc.2015.06.062
22, 32, 34, 35, 37

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., & Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, *11*, 625–660. Retrieved from `http://www.jmlr.org/papers/v11/erhan10a.html`
6

Fahlman, S. E., & Lebiere, C. (1990). The Cascade-Correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 524–532). Retrieved from `http://papers.nips.cc/paper/207-the-cascade-correlation-learning-architecture`
8, 18, 20, 191

Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in neural information processing systems 7* (pp. 625–632). Retrieved from `http://papers.nips.cc/paper/893-a-growing-neural-gas-network-learns-topologies`
8, 16, 18, 20

Fukushima, K. (2014). One-shot learning with feedback for multi-layered convolutional network. In S. Wermter et al. (Eds.), *Artificial Neural Networks and Machine Learning – ICANN 2014* (Vol. 8681, pp. 291–298). doi: 10.1007/978-3-319-11179-7_37
8, 190

Fusi, S., Annunziato, M., Badoni, D., Salamon, A., & Amit, D. J. (2000). Spike-driven synaptic plasticity: Theory, simulation, VLSI implementation. *Neural Computation*, *12*(10), 2227–2258. doi: 10.1162/089976600300014917
33

Gardner, B., & Grüning, A. (2016). Supervised learning in spiking neural networks for precise temporal encoding. *PLOS ONE*, *11*(8), 1–28. doi: 10.1371/journal.pone.0161335
6

Gerstner, W., & Kistler, W. M. (2002). *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge, UK: Cambridge University Press.
3, 4, 35, 71, 78, 93, 98

Goodfellow, I. J., Mirza, M., Xaio, D., Courville, A., & Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *ArXiv e-prints*. Retrieved from `https://arxiv.org/abs/1312.6211`

6, 190

Goyal, P., Dollár, P., Girshick, R. B., Noordhuis, P., Wesolowski, L., Kyrola, A., ... He, K. (2017). Accurate, large minibatch SGD: Training ImageNet in 1 hour. *ArXiv e-prints*. Retrieved from `https://arxiv.org/abs/1706.02677`

5, 6

Graupner, M., & Brunel, N. (2012, March). Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. *Proceedings of the National Academy of Sciences*, *109*(10), 3991–3996. doi: 10.1073/pnas.1109359109

45, 66, 190

Gutig, R., & Sompolinsky, H. (2006). The tempotron: A neuron that learns spike timing-based decisions. *Nature Neuroscience*, *9*(3), 420–428. doi: 10.1038/nn1643

43

Hagan, M. T., & Menhaj, M. B. (1994, Nov). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, *5*(6), 989–993. doi: 10.1109/72.329697

4

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *ArXiv e-prints*. Retrieved from `https://arxiv.org/abs/1510.00149`

9, 16

Herculano-Houzel, S. (2009). The human brain in numbers: A linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, *3*(31), 1–11. doi: 10.3389/neuro.09.031.2009

47

Huang, G.-B., Saratchandran, P., & Sundararajan, N. (2005). A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Transactions on Neural Networks*, *16*(1), 57–67. doi: 10.1109/TNN.2004.836241

8, 16, 18

Huang, G.-B., Wang, D. H., & Lan, Y. (2011). Extreme learning machines: A survey. *International Journal of Machine Learning and Cybernetics*, *2*(2), 107–122. doi: 10.1007/s13042-011-0019-y

16

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv e-prints*. Retrieved from `http://arxiv.org/abs/1502.03167`

5

Izhikevich, E. M. (2004). Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, *15*(5), 1063–1070. doi: 10.1109/TNN.2004.832719

4, 44

Izhikevich, E. M. (2006). Polychronization: Computation with spikes. *Neural Computation*, *18*(2), 245–282. doi: 10.1162/089976606775093882
40, 53

Kasabov, N., Dhoble, K., Nuntalid, N., & Indiveri, G. (2013). Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition. *Neural Networks*, *41*, 188–201. doi: 10.1016/j.neunet.2012.11.014
22, 29, 32, 33, 34, 43, 45

Kasabov, N., Scott, N. M., Tu, E., Marks, S., Sengupta, N., Capecci, E., ... Yang, J. (2016). Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: Design methodology and selected applications. *Neural Networks*, *78*, 1–14. doi: 10.1016/j.neunet.2015.09.011
43

Kasiński, A., & Ponulak, F. (2006). Comparison of supervised learning methods for spike time coding in spiking neural networks. *International Journal of Applied Mathematics and Computer Science*, *16*(1), 101–113.
6

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., ... Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*. doi: 10.1073/pnas.1611835114
6

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59–69. doi: 10.1007/BF00337288
5, 6

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, & K. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Retrieved from `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks`
6, 191

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, *10*, 1–40. Retrieved from `http://www.jmlr.org/papers/v10/larochelle09a.html`
191

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, *10*, 508. doi: 10.3389/fnins.2016.00508
4, 6

Legenstein, R., Naeger, C., & Maass, W. (2005). What can a neuron learn with spike-timing-dependent plasticity? *Neural Computation*, *17*(11), 2337–2382. doi: 10.1162/0899766054796888
7, 52, 53, 66, 121

Lichman, M. (2013). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

43

Lightheart, T., Grainger, S., & Lu, T.-F. (2010). A constructive spiking neural network for reinforcement learning in autonomous control. In G. Wyeth & B. Upcroft (Eds.), *Proceedings of the 2010 Australasian Conference on Robotics & Automation.* Retrieved from `http://www.araa.asn.au/acra/acra2010/papers/pap155s1-file1.pdf`

13

Lightheart, T., Grainger, S., & Lu, T.-F. (2011). Constructive network reinforcement learning for autonomous mobile robots. In *Proceedings of the 2011 International Conference on Mechatronics Technology [CD].*

13

Lightheart, T., Grainger, S., & Lu, T.-F. (2013). Spike-timing-dependent construction. *Neural Computation*, *25*(10), 2611–2645. doi: 10.1162/NECO_a_00501

13, 94

Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, *7*, 13276. doi: 10.1038/ncomms13276

6

Maass, W. (1997). Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In M. C. Mozer, M. I. Jordan, & T. Petsche (Eds.), *Advances in neural informational processing systems 9* (pp. 211–217). Retrieved from `http://papers.nips.cc/paper/1307-noisy-spiking-neurons-with-temporal-coding-have-more-computational-power-than-sigmoidal-neurons`

4

Masquelier, T. (2012). Relative spike time coding and STDP-based orientation selectivity in the early visual system in natural continuous and saccadic vision: A computational model. *Journal of Computational Neuroscience*, *32*(3), 425–441. doi: 10.1007/s10827-011-0361-9

7, 100, 121

Masquelier, T. (2013). Neural variability, or lack thereof. *Frontiers in Computational Neuroscience*, *7*, 7. doi: 10.3389/fncom.2013.00007

56, 70, 78

Masquelier, T., Guyonneau, R., & Thorpe, S. J. (2008). Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains. *PLOS ONE*, *3*(1), e1377. doi: 10.1371/journal.pone.0001377

7, 11, 12, 43, 71, 97, 98, 100, 103, 104, 105, 108, 119, 121, 122, 124, 125, 126, 129, 131, 132, 137, 141, 146, 147, 149, 150, 152, 153, 186

Masquelier, T., Guyonneau, R., & Thorpe, S. J. (2009). Competitive STDP-based spike pattern learning. *Neural Computation*, *21*(5), 1259–1276. doi: 10.1162/neco.2008.06-08-804

7, 11, 12, 43, 53, 66, 121, 148, 149, 150, 151, 152, 153, 154, 165, 168, 174, 181, 187

Masquelier, T., & Thorpe, S. J. (2007). Unsupervised learning of visual features through spike timing dependent plasticity. *PLOS Computational Biology*, *3*(2), 247–257. doi: 10.1371/journal.pcbi.0030031
8, 121

Mitchell, T., Buchanan, B., DeJong, G., Dietterich, T., Rosenbloom, P., & Waibel, A. (1990). Machine learning. *Annual Review of Computer Science*, *4*(1), 417–433. doi: 10.1146/annurev.cs.04.060190.002221
2

Montavon, G., Orr, G. B., & Müller, K.-R. (Eds.). (2012). *Neural networks: Tricks of the trade* (2nd ed., Vol. 7700). Berlin, Germany: Springer. doi: 10.1007/978-3-642-35289-8
5

Morrison, A., Diesmann, M., & Gerstner, W. (2008). Phenomenological models of synaptic plasticity based on spike timing. *Biological Cybernetics*, *98*(6), 459–478. doi: 10.1007/s00422-008-0233-1
7, 21, 45, 52, 53, 54, 121

Mundt, M., Weis, T., Konda, K., & Ramesh, V. (2017). Building effective deep neural network architectures one feature at a time. *ArXiv e-prints*. Retrieved from `https://arxiv.org/abs/1705.06778`
8, 9

Nicoletti, M. d. C., Bertini, J. a., Elizondo, D., Franco, L., & Jerez, J. (2009). Constructive neural network algorithms for feedforward architectures suitable for classification tasks. In L. Franco, D. Elizondo, & J. Jerez (Eds.), *Constructive neural networks* (Vol. 258, pp. 1–23). doi: 10.1007/978-3-642-04512-7_1
8, 16

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, *15*(3), 267–273. doi: 10.1007/BF00275687
5

Olshausen, B. A. (2013). 20 years of learning about vision: Questions answered, questions unanswered, and questions not yet asked. In J. M. Bower (Ed.), *20 years of computational neuroscience* (pp. 243–270). doi: 10.1007/978-1-4614-1424-7_12
189

Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, *14*(4), 481–487. doi: 10.1016/j.conb.2004.07.007
182

Petersen, S. E., & Posner, M. I. (2012). The attention system of the human brain: 20 years after. *Annual Review of Neuroscience*, *35*(1), 73–89. doi: 10.1146/annurev-neuro-062111-150525
191

Pfister, J.-P., & Gerstner, W. (2006). Triplets of spikes in a model of spike timing-dependent plasticity. *The Journal of Neuroscience*, *26*(38), 9673–9682. doi: 10.1523/JNEUROSCI.1425-06.2006
7, 66, 147

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, *3*(2), 213–225. doi: 10.1162/neco.1991.3.2.213
18, 20

Ponulak, F., & Kasiski, A. (2010). Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting. *Neural Computation*, *22*(2), 467–510. doi: 10.1162/neco.2009.11-08-901
6, 45

Reed, R. (1993). Pruning algorithms–a survey. *IEEE Transactions on Neural Networks*, *4*(5), 740–747. doi: 10.1109/72.248452
16, 18

Roy, S., & Basu, A. (2016). An online structural plasticity rule for generating better reservoirs. *Neural Computation*, *28*(11), 2557–2584. doi: 10.1162/neco_a_00886
22, 29, 30, 41, 43

Roy, S., & Basu, A. (2017). An online unsupervised structural plasticity algorithm for spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(4), 900–910. doi: 10.1109/TNNLS.2016.2582517
16, 21, 22, 26, 27, 30, 36, 41, 43

Roy, S., San, P. P., Hussain, S., Wei, L. W., & Basu, A. (2016). Learning spike time codes through morphological learning with binary synapses. *IEEE Transactions on Neural Networks and Learning Systems*, *27*(7), 1572–1577. doi: 10.1109/TNNLS.2015.2447011
22, 26, 29, 30, 40, 43

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. doi: 10.1038/323533a0
2, 4

Sanderson, C., & Paliwal, K. K. (2004). Identity verification using speech and face information. *Digital Signal Processing*, *14*(5), 449–480. doi: 10.1016/j.dsp.2004.05.001
42

Schliebs, S., Hamed, H. N. A., & Kasabov, N. (2011). Reservoir-based evolving spiking neural network for spatio-temporal pattern recognition. In B.-L. Lu, L. Zhang, & J. Kwok (Eds.), *Neural Information Processing: 18th International Conference, ICONIP 2011, Shanghai, China, November 13-17, 2011, Proceedings, Part II* (Vol. 7063, pp. 160–168). doi: 10.1007/978-3-642-24958-7_19
43

Schliebs, S., & Kasabov, N. (2013). Evolving spiking neural network–a survey. *Evolving Systems*, *4*(2), 87–98. doi: 10.1007/s12530-013-9074-9
8, 16, 22, 25, 30, 32, 35, 42

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. doi: 10.1016/j.neunet.2014.09.003
2, 4, 6, 190

Serre, T., Wolf, L., Bileschi, S., Riesenhuber, M., & Poggio, T. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis*

*and Machine Intelligence*, *29*(3), 411–426. doi: 10.1109/TPAMI.2007.56
2

Shirin, D., Savitha, R., & Suresh, S. (2013). A basis coupled evolving spiking neural network with afferent input neurons. In *The 2013 International Joint Conference on Neural Networks (IJCNN).* doi: 10.1109/IJCNN.2013.6706964
22, 32, 34, 37

Shoham, S., O'Connor, D. H., & Segev, R. (2006). How silent is the brain: Is there a "dark matter" problem in neuroscience? *Journal of Comparative Physiology A*, *192*(8), 777–784. doi: 10.1007/s00359-006-0117-6
147, 182

Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, *3*(9), 919–926. doi: 10.1038/78829
7, 53, 59, 60, 65, 132, 154

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. Retrieved from `http://jmlr.org/papers/v15/srivastava14a.html`
6

Stewart, T., Bekolay, T., & Eliasmith, C. (2012). Learning to select actions with spiking neurons in the basal ganglia. *Frontiers in Neuroscience*, *6*(2), 14. doi: 10.3389/fnins.2012.00002
189

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, 17–19 Jun). On the importance of initialization and momentum in deep learning. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th international conference on machine learning* (Vol. 28, pp. 1139–1147). Atlanta, Georgia, USA: PMLR. Retrieved from `http://proceedings.mlr.press/v28/sutskever13.html`
5

Takita, K., & Hagiwara, M. (2005). A pulse neural network reinforcement learning algorithm for partially observable Markov decision processes. *Systems and Computers in Japan*, *36*, 42–52. doi: 10.1002/scj.10645
10, 22, 23, 24, 29, 30, 31, 36, 42, 45

Takizawa, H., Hiroi, N., & Funahashi, A. (2012). Mathematical modeling of sustainable synaptogenesis by repetitive stimuli suggests signaling mechanisms in vivo. *PLOS ONE*, *7*(12), e51000. doi: 10.1371/journal.pone.0051000
10

Thorpe, S., Delorme, A., & Rullen, R. V. (2001). Spike-based strategies for rapid processing. *Neural Networks*, *14*(6–7), 715–725. doi: 10.1016/S0893-6080(01)00083-1
31

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., & Wierstra, D. (2016). Matching Networks for One Shot Learning. *ArXiv e-prints*. Retrieved from

https://arxiv.org/abs/1606.04080
191

Wang, J., Belatreche, A., Maguire, L., & McGinnity, T. (2015a). Dynamically evolving spiking neural network for pattern recognition. In *2015 International Joint Conference on Neural Networks (IJCNN)*. doi: 10.1109/IJCNN.2015.7280649
22, 32, 34, 35, 37, 40

Wang, J., Belatreche, A., Maguire, L., & McGinnity, T. M. (2014). An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing*, *144*, 526–536. doi: 10.1016/j.neucom.2014.04.017
22, 29, 32, 33, 35, 37, 38, 39, 45

Wang, J., Belatreche, A., Maguire, L. P., & McGinnity, T. M. (2015b). SpikeComp: An evolving spiking neural network with adaptive compact structure for pattern classification. In S. Arik, T. Huang, W. K. Lai, & Q. Liu (Eds.), *Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part II* (pp. 259–267). doi: 10.1007/978-3-319-26535-3_30
22, 32, 34, 38, 39

Wang, J., Belatreche, A., Maguire, L. P., & McGinnity, T. M. (2017). SpikeTemp: An enhanced rank-order-based learning approach for spiking neural networks with adaptive structure. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(1), 30–43. doi: 10.1109/TNNLS.2015.2501322
16, 22, 30, 32, 34, 37, 43

Watt, A., & Desai, N. (2010). Homeostatic plasticity and STDP: Keeping a neuron's cool in a fluctuating world. *Frontiers in Synaptic Neuroscience*, *2*, 5. doi: 10.3389/fnsyn.2010.00005
129

Watts, M. J. (2009). A decade of Kasabov's evolving connectionist systems: A review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, *39*(3), 253–269. doi: 10.1109/TSMCC.2008.2012254
8, 16

Wenisch, O. G., Noll, J., & Hemmen, J. L. v. (2005). Spontaneously emerging direction selectivity maps in visual cortex through STDP. *Biological Cybernetics*, *93*(4), 239–247. doi: 10.1007/s00422-005-0006-z
7

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2006). On-line learning with structural adaptation in a network of spiking neurons for visual pattern recognition. In S. D. Kollias, A. Stafylopatis, W. Duch, & E. Oja (Eds.), *Artificial Neural Networks – ICANN 2006* (Vol. 4131, pp. 61–70). doi: 10.1007/11840817_7
10, 42

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2008). Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. *Neurocomputing*, *71*(13–15), 2563–2575. doi: 10.1016/j.neucom.2007.12.038
42

Wysoski, S. G., Benuskova, L., & Kasabov, N. (2010). Evolving spiking neural networks for audiovisual information processing. *Neural Networks*, *23*(7), 819–835. doi: 10.1016/j.neunet.2010.04.009
16, 17, 18, 19, 20, 21, 22, 25, 26, 29, 30, 31, 34, 36, 37, 42, 44

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 27* (pp. 3320–3328). Retrieved from `http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf`
6