Doktors der Naturwissenschaften (Dr. rer. nat.)

PhD Thesis: Computer Science

Event-Driven Strategies for Biologically Plausible Learning

Jesús Andrés Espinoza-Valverde

Faculty of Mathematics and Natural Sciences
University of Wuppertal
Gaußstraße 20
Wuppertal - 42119, Germany

Event-Driven Strategies for Biologically Plausible Learning

A thesis submitted in fulfillment of the requirements for the degree of

Doktors der Naturwissenschaften (Dr. rer. nat.)

in

Computer Science

by

Jesús Andrés Espinoza-Valverde

to

Faculty of Mathematics and Natural Sciences

BERGISCHE
UNIVERSITÄT
WUPPERTAL

University of Wuppertal
Gaußstraße 20
Wuppertal - 42119, Germany

Supervised by: Prof. Dr. Matthias Bolten

Abstract

Spiking neural networks (SNNs) promise energy-efficient, scalable computation inspired by biological brains. However, translating biologically plausible learning algorithms into effective artificial systems presents significant computational and conceptual challenges. This thesis addresses two important issues: scalable event-driven learning and biologically inspired continual learning, through extensions and adaptations of the Eligibility Propagation (e-prop) algorithm, a biologically inspired alternative to backpropagation through time (BPTT).

In the first project, we introduce an event-based adaptation of e-prop, transitioning from continuous, time-driven computations to asynchronous, event-driven updates. Our approach respects biological constraints, such as synaptic delays and spatial-temporal locality, thereby increasing biological realism. By decoupling time constants and introducing flexible update mechanisms triggered by neuronal spikes, we maintain computational efficiency and demonstrate scalable performance on benchmarks like neuromorphic MNIST.

As a natural continuation that builds on these foundations, the second project addresses catastrophic forgetting, an essential challenge in continual learning. Inspired by synaptic consolidation and neuromodulated plasticity observed in biological systems, we propose a framework utilizing per-synapse Online Saliency Traces, continuously updated via locally available gradient information without relying on explicit task boundaries or multiple data passes. Task transitions are autonomously identified using Bayesian Online Changepoint Detection (BOCD), triggering neuromodulatory signals to consolidate knowledge effectively.

Together, these projects demonstrate that biologically inspired mechanisms, implemented with respect to biological constraints, enable both scalable event-based learning and robust continual learning.

Research Contributions

This dissertation is presented in the monographic format. The following research outcomes and academic activities were carried out during the doctoral research period from 2021 to 2025.

Publications

- Korcsak-Gorzo, A., Stapmanns, J., Espinoza Valverde, J. A., Plesser, H. E., Dahmen, D., Bolten, M., van Albada, S. J., Diesmann, M. *Event-driven eligibility propagation in large sparse networks: reconciling efficiency with biological realism.* **In preparation**.

- Espinoza Valverde, J. A., Bolten, M. *Biologically Inspired Continual Learning through Online Saliency Traces.* **In preparation**.

Contributions to Research Software

- Espinoza Valverde, J. A., Müller, E., Haug, N., Schöfmann, C. M., Linssen, C., Senk, J., Spreizer, S., Trensch, T., Lober, M., Jiang, H., Kurth, A., Acimovic, J., Korcsak-Gorzo, A., Welle Skaar, J.-E., Terhorst, D., Stapmanns, J., Graber, S., de Schepper, R., Eppler, J. M., . . . Plesser, H. E. (2024).
  NEST 3.7 (Version 3.7). Zenodo. https://doi.org/10.5281/zenodo.10834751

Conference Presentations

- **NEST Conference 2024**
  Event-Based Eligibility Propagation with Additional Biologically Inspired Features
  Authors: Agnes Korcsak-Gorzo, Jonas Stapmanns, Jesús Andrés Espinoza Valverde, et al.

- **SNUFA 2024**
  Event-Based Eligibility Propagation with Additional Biologically Inspired Features
  Authors: Agnes Korcsak-Gorzo, Jonas Stapmanns, Jesús Andrés Espinoza Valverde, et al.

- **SIAM CSE25**
  Event-Based Eligibility Propagation with Additional Biologically Inspired Features
  Authors: Agnes Korcsak-Gorzo, Jonas Stapmanns, Jesús Andrés Espinoza Valverde, et al.

- **NEST Conference 2025**
  Mitigating Catastrophic Forgetting in Biologically Plausible Learning
  Authors: Jesús Andrés Espinoza Valverde, Matthias Bolten

Acknowledgments

I extend my deepest gratitude to everyone who supported me throughout this journey. My family and friends deserve heartfelt thanks for their unwavering encouragement and belief, which were a constant source of strength. I'm also grateful to my colleagues and mentors for their invaluable insights and support, which significantly shaped this work.

A very special thank you to my supervisor, Prof. Dr. Matthias Bolten, whose guidance and encouragement were absolutely vital every step of the way. I'm also grateful to Prof. Dr. Hans Ekkehard Plesser for his time, involvement, and technical contributions, and to Prof. Dr. Markus Diesmann for his insightful feedback and all the thought-provoking ideas. And to Agnes Korcsak-Gorzo, I'm especially appreciative of your dedication, thoughtful feedback, and the generous time you invested in this project.

I also want to acknowledge Ing. María Fernanda Granados Arias, whose invaluable advice and help in designing many of the schematics and figures greatly enhanced the visual presentation of this thesis.

Finally, I'm thankful to the University of Wuppertal and the Faculty of Mathematics and Natural Sciences for providing the excellent infrastructure and academic environment that made this work possible.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE **Page**

Note: the figures 1.1, 2.1-2.9, 3.1, 3.2, 3.3, 4.1-4.4 were created in https://BioRender.com

LIST OF TABLES

TABLE **Page**

Chapter 1

Introduction

Spiking Neural Networks (SNNs) are increasingly recognized as a compelling alternative to conventional Artificial Neural Networks (ANNs), primarily due to their closer resemblance to brain function and their potential technological advantages over standard ANNs [1, 2, 3, 4]. In SNNs, information flows through asynchronous, event-driven spike communications, enabling sparse computations and substantially reduced energy consumption compared to traditional approaches [5, 2]. These properties make SNNs particularly effective for deployment on specialized spike-based substrates such as neuromorphic hardware, which facilitates low-power and real-time processing in edge computing systems [6, 7, 8, 9].

A further advantage of SNNs lies in their close alignment with the computational principles of the brain. Unlike traditional computing architectures based on the von Neumann model (Figure 1.1b), biological computation employs asynchronous neuronal units, sparse synaptic connections, and discrete spikes that carry information in a temporally and spatially sparse manner (Figure 1.1a). In the brain, learning emerges largely through local and online plasticity mechanisms modulated by global feedback signals, enabling continuous and adaptive learning at very low power consumption [10, 9] (Figure 1.1c).

Motivated by these biological insights, computational neuroscience and machine learning researchers are developing biologically plausible learning algorithms and techniques inspired by brain-like computation [11, 12, 13, 14, 15, 16]. The goal of this emerging research field is not merely biological imitation but the creation of algorithms that harness the most efficient biological learning principles, producing robust, scalable, and effective intelligent systems.

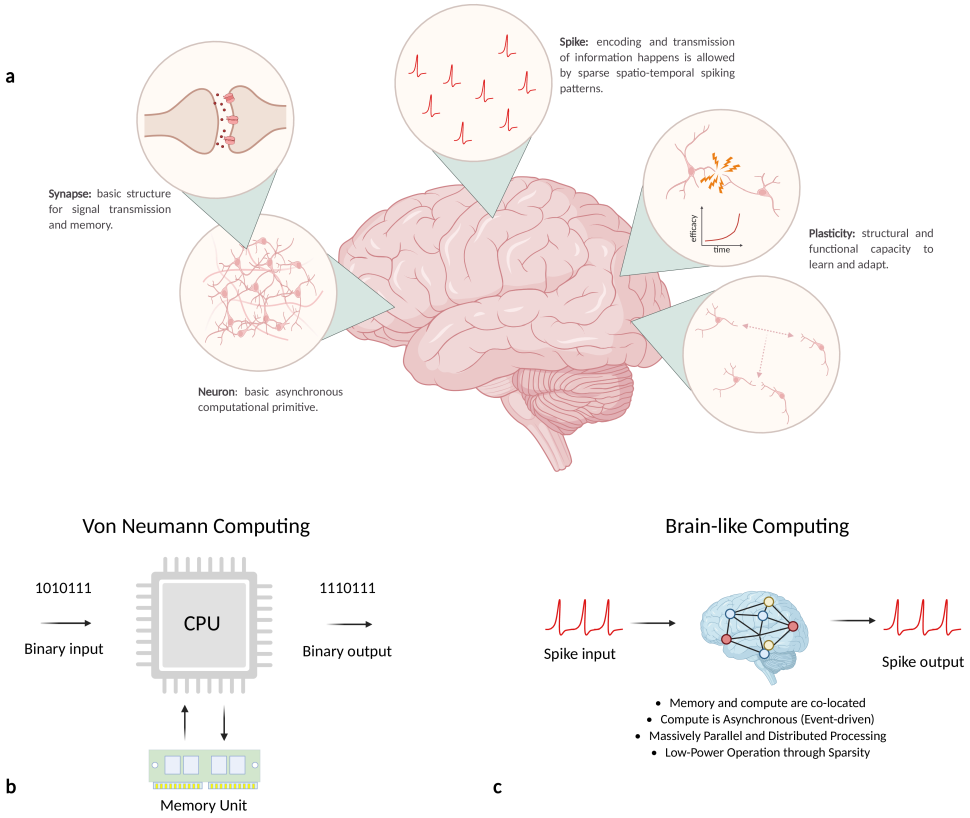

a
Spike: encoding and transmission of information happens is allowed by sparse spatio-temporal spiking patterns.
Synapse: basic structure for signal transmission and memory.
Plasticity: structural and functional capacity to learn and adapt.
efficacy
time
Neuron: basic asynchronous computational primitive.
Von Neumann Computing
1010111
Binary input
CPU
1110111
Binary output
Memory Unit
b
Brain-like Computing
Spike input
Spike output
Memory and compute are co-located
Compute is Asynchronous (Event-driven)
Massively Parallel and Distributed Processing
Low-Power Operation through Sparsity
c

Figure 1.1: **Core components of brain-like computation** (a) Building blocks of biological information processing. Neurons serve as the basic asynchronous computational units. Synapses serve as the substrate for memory and mediate signal transmission. Spikes are discrete, temporally sparse electrical events that are used to encode and transmit information. Learning is supported by plasticity mechanisms, which enable synaptic connections to change over time as a result of experience. These components work together to allow for effective, distributed, and flexible brain computation, which is a major source of inspiration for models of spiking neural networks. (b) Von Neumann computing. The traditional computing architecture, where a central processing unit (CPU) executes instructions sequentially, with memory and storage units that are separate from the processing unit. This architecture is not well-suited for the parallel and event-driven nature of biological computation. (c) Brain-like computation. A more distributed and parallel approach to computation, where processing and memory is co-located, and updates are asynchronous. Figure concept inspired by [17, 18].

1.1 Research Questions

Within this broader research landscape, the present thesis seeks to contribute by addressing two core questions that remain open in the current literature.

1.1.1 Enhancing Biological Plausibility in Learning with Spiking Neural Networks

Prominent learning algorithms for Spiking Neural Networks (SNNs) often emphasize mathematical precision and strict optimization criteria, such as exact gradients [19], synchronized updates [16], and instantaneous signal transmission [20]. Among these, *eligibility propagation* (e-prop) [16] stands out as one of the most influential and biologically motivated approaches, and it serves as the primary learning method investigated in this thesis.

Despite its neuroscientific inspiration, e-prop and related algorithms are typically implemented in time-driven frameworks that don't fully exploit the inherently event-based, sparse, and asynchronous nature of SNN computation. They often rely on machine learning conventions that lack strong biological justification, including: synchronized, equidistant synaptic updates; global neural and synaptic resets; batch learning; normalization of probabilities; and instantaneous communication between spatially separated network components. While these methods have demonstrated strong performance on benchmark tasks, they tend to underutilize core principles of biological neural systems, such as delayed, spike-driven plasticity shaped by asynchronous interactions.

In light of this, the first research question addressed in this thesis is:

> **Question 1: To what extent can mathematical precision and structural constraints be relaxed to enhance biological plausibility in spiking neural network models, without significantly compromising training performance?**

This question seeks to explore the trade-off between computational performance and biological realism. It aims to investigate how far we can depart from rigid, mathematically idealized structures while still preserving effective learning capabilities. Concretely, the focus lies on relaxing several common assumptions in machine learning models when applied to e-prop. For example, the global synchronization required for batch updates can be replaced by more biologically plausible, spike-triggered weight updates using partial gradient contributions applied asynchronously during the presentation of a training sample. Artificial resets of neurons and synapses between samples can be avoided to maintain temporal continuity. Moreover, parameters that are mathematically linked but physically distant in the network, such as time constants or filters, can be decoupled to better reflect biological constraints.

Simultaneously, this work explores the integration of biologically inspired constraints into the learning framework. For instance, synaptic and signal communication is modeled as non-instantaneous, respecting the delays and propagation times observed in biological systems. Dale's law is enforced to ensure that each neuron is strictly excitatory or inhibitory, in accordance with biological evidence. Furthermore, the framework moves away from biologically implausible components, such as probabilistic output layers based on softmax and cross-entropy loss, opting instead for classification approaches more aligned with spike-based signal processing. The overarching goal is to evaluate whether such biologically inspired modifications can preserve, or even enhance, learning performance on standard tasks such as regression, classification, while offering better scalability and interpretability.

1.1.2 Bridging Biologically Inspired Learning with Continual Learning

Another key aspect of biological learning that remains relatively underexplored in the context of biologically plausible learning with SNNs is continual learning. In the brain, learning unfolds as a lifelong process, allowing organisms to acquire new skills and adapt to changing environments without overwriting previously learned information [21]. This stands in stark contrast to many current machine learning systems and methods, including eligibility propagation (e-prop) [16], which tend to suffer from catastrophic forgetting when trained sequentially on different tasks.

This observation motivates the second research question addressed in this thesis:

> **Question 2: How can biologically plausible learning mechanisms be adapted to enable spiking neural networks to acquire new tasks continually without forgetting previously learned knowledge?**

Established and powerful Continual learning (CL) methods such as Elastic Weight Consolidation (EWC) [22] and Synaptic Intelligence (SI) [23] aim to mitigate catastrophic forgetting by identifying and preserving task-relevant parameters. These approaches typically constrain changes to important weights during sequential learning, thereby retaining knowledge from prior tasks. However, they rely on biologically implausible mechanisms, such as computing parameter importance via a second pass over the training data or assuming access to explicit task boundaries, both of which are unrealistic in natural learning scenarios.

Biological systems, by contrast, rely on local mechanisms such as synaptic traces and modulatory signals to regulate plasticity. These processes operate online, without the need for revisiting past data or external supervision to signal task transitions. Neuromodulators such as dopamine play a critical role in gating plasticity in response to salient events, rewards, or novelty, supporting flexible and context-sensitive learning [24, 25, 26].

The goal of this research question is to explore whether conceptual and mechanistic bridges can be built between biologically inspired plasticity rules and continual learning strategies. In particular, it investigates whether parameter importance can be reinterpreted in terms of locally computed, synapse-specific quantities such as saliency traces. It also considers the possibility of replacing externally defined task boundaries with internally generated signals, such as surprise or uncertainty, that trigger consolidation through neuromodulated pathways. By framing memory retention and adaptive plasticity as manifestations of neuromodulator-driven synaptic processes, this thesis looks for ways to bring continual learning in SNNs closer to biological plausibility.

1.2 Thesis Structure

The present thesis is organized as follows:

Chapter 2 provides the necessary background and foundational concepts relevant to the work presented in this thesis. It begins with an overview of biological neurons, synapses, and neural networks, followed by a discussion of the mechanisms of neural plasticity, including Hebbian learning, spike-timing dependent plasticity (STDP), and three-factor learning rules. The chapter then transitions into the computational modeling of these biological components, introducing various spiking neuron models such as the leaky integrate-and-fire neuron and its variants, as well as artificial synapses and the architecture of spiking neural networks (SNNs). Gradient-based learning methods are reviewed, with a particular emphasis on their biological plausibility. Finally, the chapter concludes with an overview of simulation tools for SNNs, with a focus on the NEST simulator used throughout this work.

Chapter 3 deals with **Question 1** (see subsection 1.1.1) and presents the first main contribution of this thesis: the development of an event-driven implementation of the eligibility propagation (e-prop) algorithm, extended with additional biologically inspired features.

The chapter begins by revisiting the original formulation of e-prop and its relationship to backpropagation through time (BPTT). It then describes the design and implementation of an event-driven version of e-prop that eliminates time-driven constraints. Several biological enhancements are introduced, including synaptic delays, continuous dynamics, asynchronous weight updates, smooth surrogate gradients, and structural constraints such as Dale's law. The implementation details are discussed, followed by benchmark results that demonstrate both the faithfulness of the event-driven model and the impact of each biological feature on learning performance. The chapter concludes with a discussion of the findings and their implications for biologically plausible SNN training.

Chapter 4 deals with **Question 2** (see subsection 1.1.2) and introduces the second major contribution of this thesis: a biologically inspired framework for continual learning in SNNs based on online saliency traces. The chapter begins by outlining the biological hallmarks of continual learning and contrasting them with established machine learning approaches such as Elastic Weight Consolidation (EWC) and Synaptic Intelligence (SI). Their limitations in terms of biological plausibility are discussed. A new method is proposed, which replaces global mechanisms with synapse-local computations and uses a novelty neuron to detect task transitions via Bayesian Online Changepoint Detection. This modulatory signal gates synaptic consolidation without requiring access to explicit task boundaries or data replaying. The effectiveness of the method is evaluated on Permuted Neuromorphic MNIST (PN-MNIST) continual learning benchmark, followed by a discussion of the results and their relevance in both biological and computational contexts.

Chapter 5 concludes the thesis by summarizing the main contributions and findings in relation to the two research questions. It reflects on how the proposed methods contribute to enhancing biological plausibility in SNN learning algorithms and enabling continual learning without catastrophic forgetting. The chapter also outlines potential directions for future work, including broader applications of the proposed methods and further refinements to the underlying algorithms.

Chapter 2

Background

Central to the present thesis is the topic of biologically-inspired learning mechanisms in spiking neural networks (SNNs), which aim to bridge the gap between neuroscience and artificial intelligence by harnessing principles observed in the brain. To set the stage for the subsequent chapters, this chapter provides an overview of both biological and artificial neural systems, with a focus on neural plasticity, spiking neuron models, and learning paradigms. It begins with a review of the structural and functional properties of biological neurons and synapses, followed by a discussion of key plasticity mechanisms, including Hebbian learning, spike-timing-dependent plasticity (STDP), and three-factor rules that incorporate modulatory signals.

We then shift to artificial neuron models commonly used in computational neuroscience and machine learning, such as the leaky integrate-and-fire (LIF) and adaptive leaky integrate-and-fire (ALIF) models. The modeling of artificial synapses is also addressed, including their role in learning and memory formation. Next, we examine how networks of spiking neurons can be trained efficiently, outlining the strengths and limitations of gradient-based learning methods, particularly in the context of their biological plausibility.

The chapter concludes with practical considerations for simulating spiking neural networks, with an emphasis on the NEural Simulation Tool (NEST), a high-performance simulator widely used in the field for building and analyzing large-scale SNN models.

2.1 Biological Neurons

The nervous system comprises specialized cells called neurons, which transmit information using electrical and chemical signals. The cell membrane of each neuron is composed of a lipid bilayer that acts as a capacitor, keeping ions separated inside and outside the cell. Ion pumps, such as those for sodium (Na^+) and potassium (K^+), maintain ionic balance, leading to the creation of the resting membrane potential [27]. Figure 2.1a shows the fundamental structure of a neuron, with emphasis on its essential parts [28].

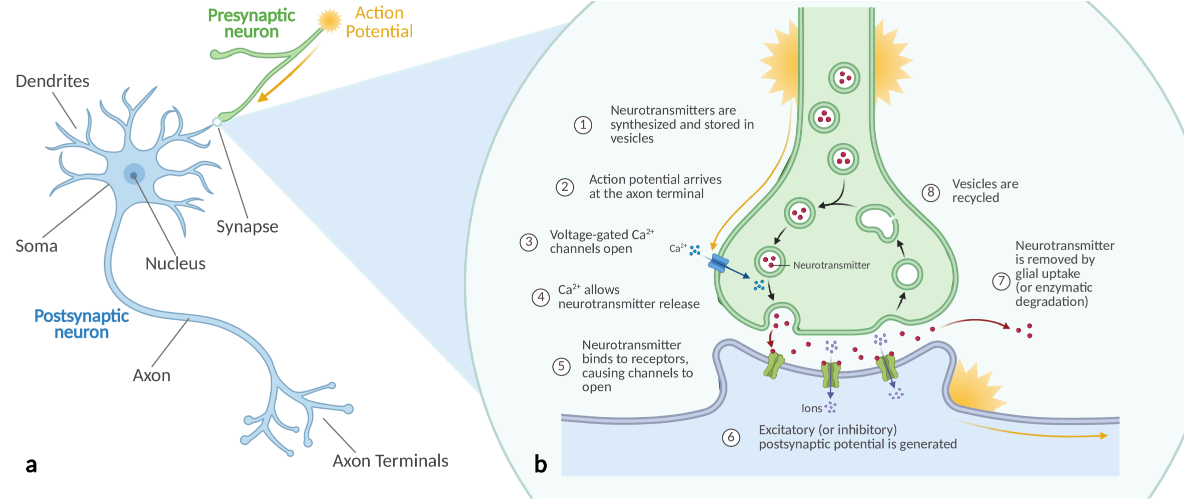

Presynaptic neuron
Action Potential
Dendrites
Soma
Nucleus
Synapse
Postsynaptic neuron
Axon
Axon Terminals
a
b
1 Neurotransmitters are synthesized and stored in vesicles
2 Action potential arrives at the axon terminal
3 Voltage-gated Ca^{2+} channels open
Ca^{2+}
4 Ca^{2+} allows neurotransmitter release
Neurotransmitter
5 Neurotransmitter binds to receptors, causing channels to open
Ions
6 Excitatory (or inhibitory) postsynaptic potential is generated
7 Neurotransmitter is removed by glial uptake (or enzymatic degradation)
8 Vesicles are recycled

Figure 2.1: **Biological Neuron and Synapse.** (a) Key structural features of a neuron. (b) Schematic representation of synaptic communication, showing the conversion of a presynaptic electrical signal into a postsynaptic potential.

Neurons communicate primarily by generating action potentials (APs), which are commonly known as spikes. APs represent rapid, transient variations in membrane potential that operate on an all-or-none principle, requiring them to be either fully activated or not activated at all [28, 29]. Ion channels open and close sequentially, allowing action potentials to travel as electrical signals along the axon. After producing an action potential, neurons go through a refractory period during which they cannot generate another action potential [29].

2.2 Biological Synapses

Neurons transmit signals through specialized connections known as synapses, which function by chemical or electrical means [28]. Chemical synapses are by far the most common and are the ones whose modeling is considered in this thesis. For this type of synapse, the release of neurotransmitters into the synaptic cleft enables communication between neurons. As a result of the emission of an AP by the presynaptic neuron, calcium ions (Ca^{2+}) are allowed to enter the presynaptic terminal [30]. This process, in turn, triggers the release of neurotransmitters. Neurotransmitters function as chemical signals that bind to specific receptor proteins located on the postsynaptic membrane. The interaction between neurotransmitters and their receptors produces postsynaptic potentials, which involve a change in the electrical state of the postsynaptic neuron. The nature of postsynaptic potentials depends on both the specific neurotransmitter and the receptor, resulting in either excitatory postsynaptic potentials (EPSPs), usually involving glutamate, or inhibitory postsynaptic potentials (IPSPs), frequently mediated by gamma-aminobutyric acid (GABA) [31].

Figure 2.1b presents an illustration of the process with detailed synaptic mechanisms.

2.3 Biological Neural Networks

Although neurons serve as the essential elements of brain structure they function together as part of complex networks. Neurons develop complex interconnected systems which support various brain functions that range from simple reflex actions to sophisticated cognition [28]. It is estimated the human brain contains 86 billion neurons, each involved in thousands of synaptic connections [32, 33]. The estimated number of synapses that emerge from these neuronal connections amounts to 100 trillion. Each neuron accepts signals through about 1,000 to 10,000 synapses which enables it to process and propagate intricate neural information patterns [34].

This intricate synaptic architecture gives rise to biological neuronal networks. The stability of these networks demands a precise balance between excitatory and inhibitory activity [35].

The brain displays hierarchical and modular organization of neural networks primarily within its cortical regions. Within the cortex, neurons are organized systematically into layers and columns to perform specialized functions. Cortical areas specialized in sensory input process external stimulus inputs before transmitting this information to association areas which further process and synthesize these information to facilitate complex cognitive functions. The frontal cortex has a crucial role in executive tasks including planning actions, making decisions and controlling impulses. The operation of these functions requires ongoing cooperation between different cortical regions. The networks function efficiently and in a coordinated manner across different cognitive areas because of their hierarchical and modular structural organization, allowing the brain to integrate information from diverse sources and adapt flexibly to changing behavioral demands.

2.4 Neural Plasticity and Learning in Biological Systems

Plasticity is the capacity of the nervous system to structurally and functionally adapt in response to experience, sensory inputs, or injury. Plasticity involves modifications in the structure and strength and structure of synaptic connections between neurons, which influence the flow and processing of information, and thus shapes behavioral response. Plasticity can be categorized broadly into structural and synaptic plasticity [36].

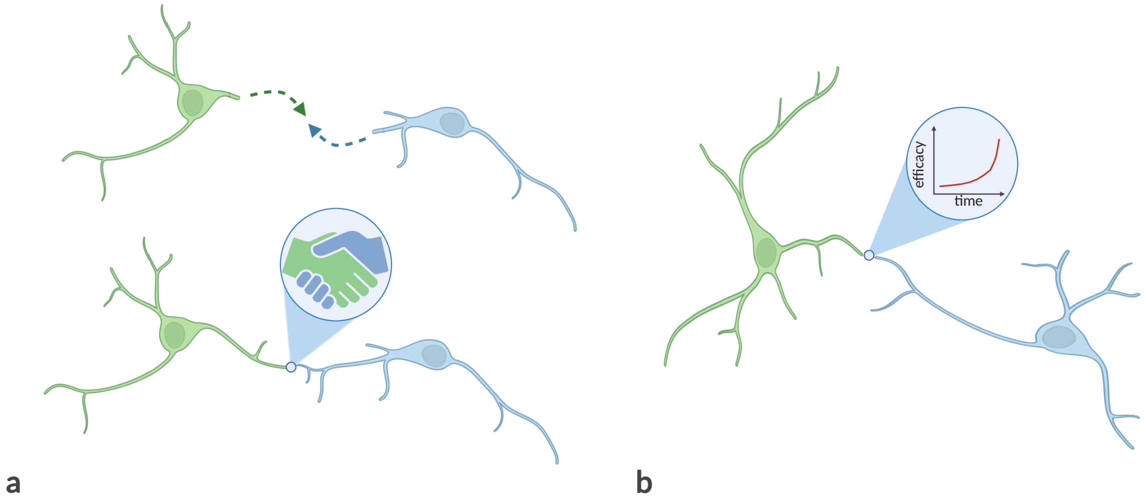


Figure 2.2: **Illustrations of structural and synaptic plasticity.** (a) Structural plasticity: formation of new neuronal connections through processes such as dendritic branching and synaptogenesis. (b) Synaptic plasticity: changes in synaptic strength, leading to either the potentiation or depression of synaptic connections.

Structural plasticity refers to changes in the physical architecture of neurons and their connections. This involves phenomena such as dendritic branching, axonal sprouting, and the formation of new synapses. **Synaptic plasticity** manifests as adjustments in synaptic efficacy, which is a phenomenon driven by patterns of correlation of neuronal activity [37, 38].

Plasticity occurs over diverse temporal scales, which ranges from minutes to months. Plasticity plays a crucial role in important neurological functions, including learning, memory consolidation, and recovery from brain injury [36, 39].

2.4.1 Hebbian Plasticity

The foundational notion of Hebbian plasticity was introduced by Donald O. Hebb [40] and plays a crucial role in both neuroscience and computational learning theory. The principle states that synaptic connections between two neurons become stronger when they show persistent and simultaneous activity. This idea is often summarized by the famous phrase:

> “Cells that fire together, wire together.”

Hebbian plasticity captures the idea that the correlation of activity between a presynaptic neuron and a postsynaptic neuron leads to synaptic strengthening. A generic mathematical expression of this principle is:

$$\Delta w_{ji} = \eta f(x_i(t), y_j(t)). \tag{2.1}$$

Here, Δw_{ji} denotes the change in synaptic weight from presynaptic neuron i to postsynaptic neuron j, $x_i(t)$ and $y_j(t)$ are their respective activity levels at time t, η is the learning rate, and $f(\cdot)$ is a function that captures how activity correlations drive plasticity. The precise form of f can vary depending on the specific Hebbian model employed—ranging from linear correlations to nonlinear or normalized formulations (see e.g. [41, 42]).

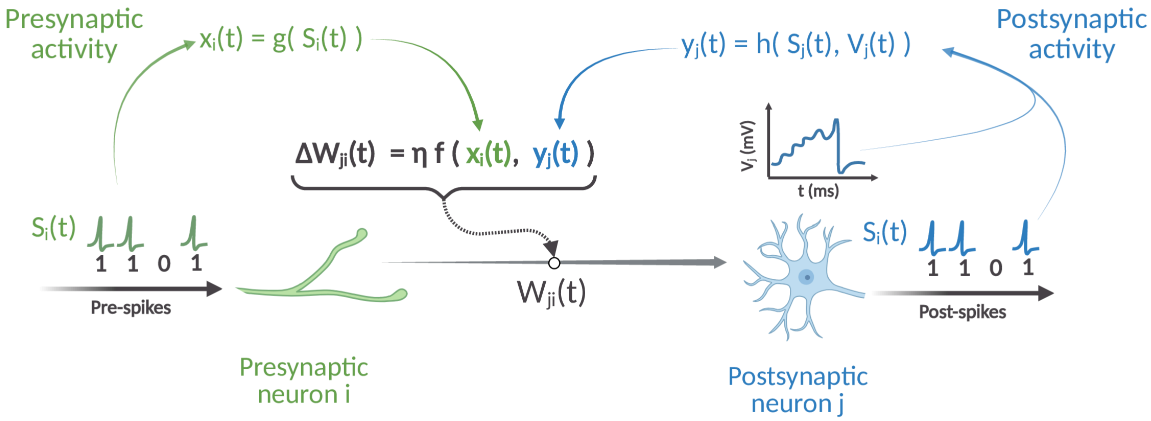


Figure 2.3: **Illustration of the core components of Hebbian plasticity.** The co-activity of pre- and postsynaptic neurons leads to synaptic modification.

This rule serves as the conceptual backbone and starting point for other biologically realistic models, including spike-timing-dependent plasticity (STDP) and three-factor learning rules [43].

2.4.1.1 Long-Term Potentiation (LTP) and Long-Term Depression (LTD)

The persistent synaptic plasticity forms known as Long-Term Potentiation (LTP) and Long-Term Depression (LTD) play an essential role in Hebbian learning.

- **LTP** establishes a lasting enhancement in synaptic strength which occurs due to continuous and synchronized activity between neurons. The biological basis of LTP includes both an increase in postsynaptic receptor numbers and greater amounts of neurotransmitter release.
- **LTD** refers to the stable decrease in synaptic strength that results from particular neuronal activity patterns including low-frequency stimulation or uncorrelated firing sequences. LTD leads to synapse weakening through decreased receptor density alongside diminished neurotransmitter release.

LTP and LTD work together to maintain necessary balance for effective synaptic changes which support learning abilities and dynamic neuronal activities [44].

2.4.2 Spike-Timing Dependent Plasticity (STDP)

Spike-Timing Dependent Plasticity (STDP) extends the classical Hebbian learning rule by incorporating the precise timing of neuronal spikes. This captures the idea that the direction and magnitude of synaptic change strongly depends on the temporal order of presynaptic and postsynaptic action potentials [44]. Specifically, the synaptic weight change is described by the time difference $\Delta t = t_{\text{post}} - t_{\text{pre}}$, where t_{pre} and t_{post} denote the firing times of the presynaptic and postsynaptic neurons, respectively.

- If the presynaptic neuron fires shortly before the postsynaptic neuron ($\Delta t > 0$), the synapse is potentiated (i.e., strengthened).
- If the presynaptic neuron fires shortly after the postsynaptic neuron ($\Delta t < 0$), the synapse undergoes depression (i.e., weakened).

This temporal asymmetry forms the basis of the STDP learning window, which is commonly described by the following mathematical expression:

$$\Delta w_{ij}(\Delta t) = \begin{cases} A_+ \exp(-\Delta t/\tau_+), & \Delta t > 0 \\ -A_- \exp(\Delta t/\tau_-), & \Delta t < 0 \end{cases} \tag{2.2}$$

Here, A_+ and A_- determine the maximum change in synaptic strength for potentiation and depression, while τ_+ and τ_- control the temporal decay of the learning window on either side of $\Delta t = 0$.

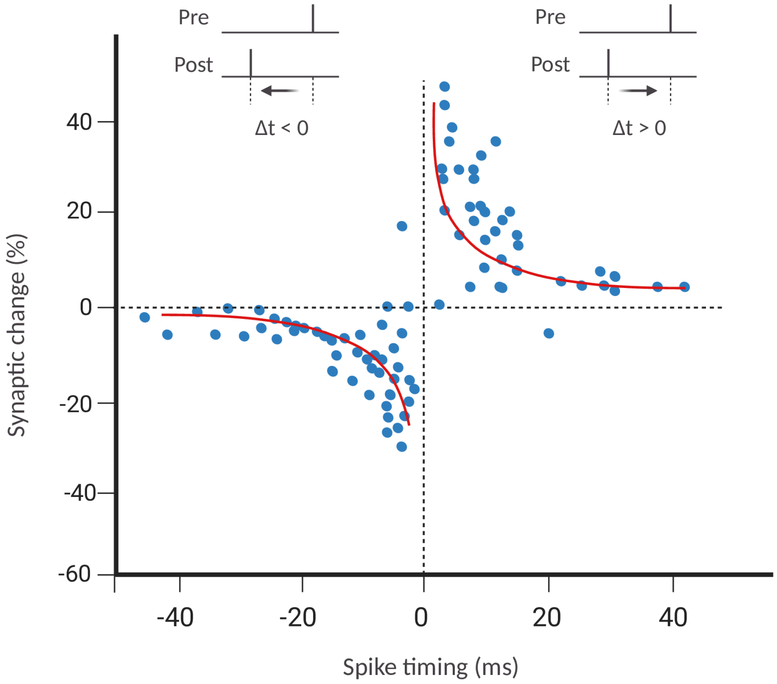


Figure 2.4: **Illustration of spike-timing-dependent plasticity (STDP).** The graph shows how the change in synaptic weight depends on the time difference between presynaptic and postsynaptic spikes. Positive Δt (presynaptic spike precedes postsynaptic spike) leads to long-term potentiation (LTP), whereas negative Δt (presynaptic spike follows postsynaptic spike) results in long-term depression (LTD). Figure concept inspired by [45].

2.4.3 Three-Factor Plasticity

Hebbian plasticity and spike-timing-dependent plasticity (STDP) are often described as two-factor learning rules because synaptic changes arise from pre- and postsynaptic activity alone. Many forms of plasticity in the brain, however, require an additional third component to modulate, or gate, synaptic weight changes. Three-factor learning rules build upon the Hebbian framework by adding a third signal such as a neuromodulator or global feedback to represent behavioral outcomes and context or environmental conditions [46].

According to these rules, synaptic change feasibility arises from pre- and postsynaptic coordination which forms an eligibility trace but permanent synaptic modification requires an appropriate third factor signal at the correct moment [43]. A third signal delivers information about reward, punishment, surprise, or other behavioral feedback which enables synaptic changes to depend on the success or saliency of recent neural activity [46, 43].

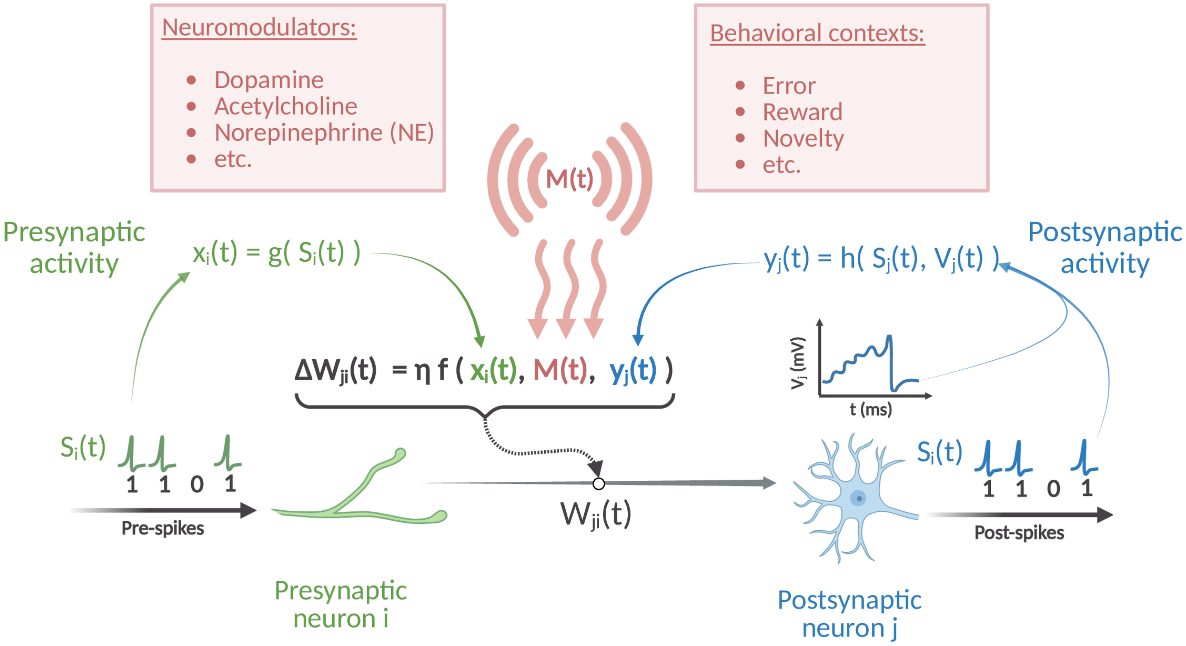


Figure 2.5: **Illustration of the core components of three-factor plasticity.** These models extend Hebbian plasticity by introducing a third modulatory factor. While the co-activity of pre- and postsynaptic neurons establishes an eligibility trace (in line with Hebbian learning), long-term synaptic modification occurs only if a suitable third factor, such as a neuromodulatory signal, arrives within a specific temporal window.

Three-factor rules allow credit assignment by merging global outcome signals with local co-activity. In mathematical form, a generic three-factor learning rule can be written as an interaction between a Hebbian eligibility trace and a modulatory third-factor signal:

$$\Delta w_{ji}(t) = \eta f(e_{ji}(t), M(t)). \tag{2.3}$$

Here w_{ji} is the synaptic weight from presynaptic neuron i to postsynaptic neuron j, η is a learning rate, $e_{ji}(t)$ is an "eligibility" variable capturing the coincidence of presynaptic activity $x_i(t)$ and postsynaptic activity $y_j(t)$ (the Hebbian two-factor product), and $M(t)$ is the third-factor signal (a neuromodulator or other instructive signal) available at time t. The eligibility trace $e_{ji}(t)$ marks the synapse as potentially plastic based on recent pre-post firing (often e_{ji} is transient, decaying within a second or so), and the factor $M(t)$ determines whether and how that potential change is actually realized [43]. If the third factor is absent ($M(t) = 0$), no lasting change occurs despite pre/post co-activation [43]. Conversely, when the third factor is present and coincident with the eligibility trace, it "unlocks" synaptic plasticity: Δw_{ji} becomes non-zero, with the direction and magnitude of change possibly influenced by the value of M (e.g. a positive M signaling reward might cause potentiation, whereas a negative M could cause depression) [43].

Experimental evidence from diverse species and brain areas confirms that three-factor learning processes exist. A recurring finding across these studies demonstrates how synapses incorporate a third signaling molecule such as dopamine or acetylcholine. This signal represents behavioral context elements such as reward, arousal and novelty which produce synaptic plasticity only during simultaneous pre-post neural activity. Such results across insects, rodents, and multiple brain areas provide strong support for the necessity of three-factor rules in biological learning [47, 48, 49].

2.5 Artificial Spiking Neurons

Spiking neurons represent computational models that better simulate biological neuronal temporal dynamics compared to standard real-valued models used in deep learning (refer to Figure 2.6 for a comparison). Spiking neuron models differ from traditional models because they simulate the discrete properties and exact timing of neuronal spikes that are fundamental to neural communication and brain information processing [50].

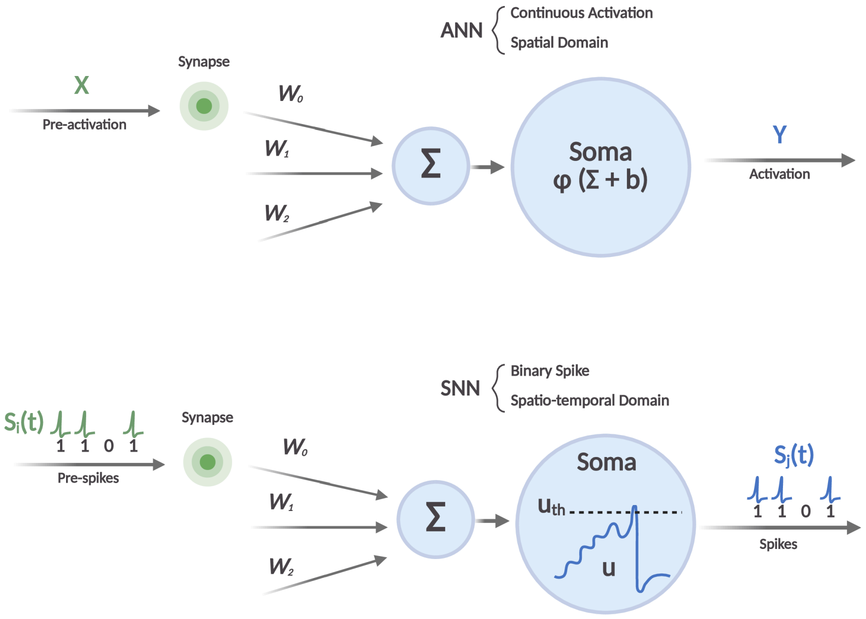


Figure 2.6: **Comparison between traditional real-valued neurons and spiking neurons.** Traditional neurons represent activity as continuous values, while spiking neurons communicate through discrete spikes. Figure concept inspired by [51].

These models represent how neurons accumulate inputs from synapses over time which results in a slow modification of their membrane potential. A neuron produces a spike when its membrane potential reaches a specific threshold value which leads to a reset of its potential followed by a refractory period that prevents further spikes. Downstream neurons receive transmitted spikes which then alter their membrane potentials through excitatory or inhibitory synapses.

The degree of biological abstraction in spiking neuron models differs across various models. Some models focus on spike timing patterns while others integrate complex biological processes including ion channel dynamics. The leaky integrate-and-fire (LIF) model stands out as one of the most widely used models because it encapsulates key neuronal behavioral features using a straightforward differential equation for its membrane potential. More detailed computational models like Hodgkin-Huxley [52] and Izhikevich [53] incorporate ionic conductances and nonlinear dynamics which enable them to simulate a vast array of neuronal behaviors including bursting and spike frequency adaptation. A different modeling methodology uses multi-compartment models to depict neurons as linked spatial compartments [54, 55].

Researchers select models depending on their research goal and application needs such as simulating large-scale neural networks or analyzing individual neuron computational properties.

2.5.1 Leaky Integrate-and-Fire Neuron

The leaky integrate-and-fire (LIF) neuron is one of the most fundamental models in computational neuroscience, balancing biological plausibility with computational simplicity. It models a neuron as an electrical RC circuit, where the membrane capacitance C_m represents the ability to store charge, and the membrane resistance R_m accounts for the passive leakage of ions through membrane channels (see Figure 2.7). The subthreshold membrane potential $V_j(t)$ for a neuron j evolves according to:

$$C_m \frac{dV_j(t)}{dt} = -\frac{V_j(t)}{R_m} + I_j(t), \tag{2.4}$$

which can be rewritten in terms of the membrane time constant $\tau_m = R_m C_m$ as:

$$\tau_m \frac{dV_j(t)}{dt} = -V_j(t) + R_m I_j(t), \tag{2.5}$$

where $I_j(t)$ is the input current flowing towards j [56].

The LIF model assumes a point-like neuron with no spatial extent or dendritic structure. This abstraction enables efficient implementation in both software simulations and

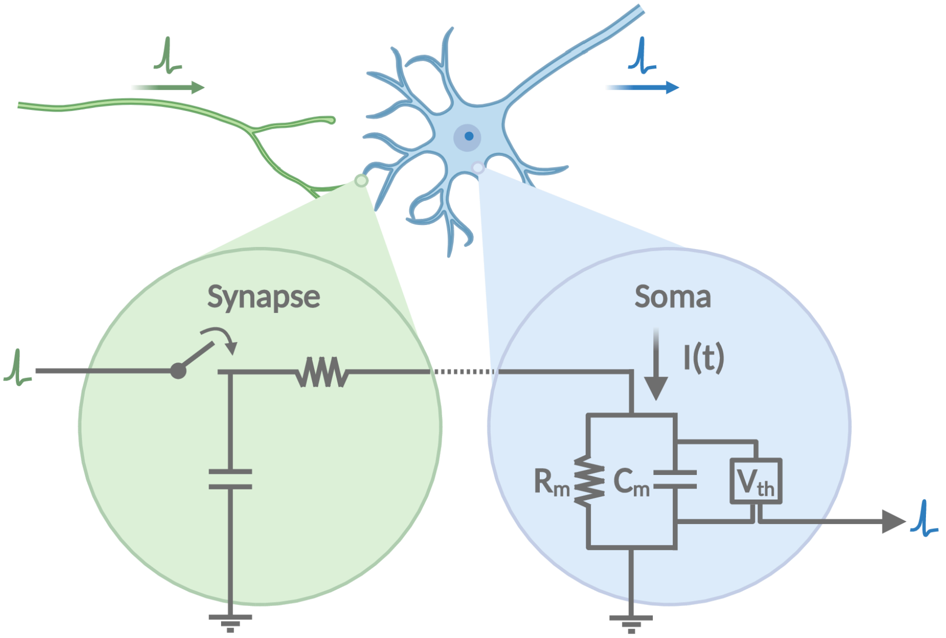


Figure 2.7: **Illustration of a leaky integrate-and-fire (LIF) neuron modeled as an RC circuit.** The membrane capacitance (C_m) stores incoming charge, while the membrane resistance (R_m) allows charge to leak over time, emulating passive ion channels. When the membrane potential reaches a threshold, a spike is emitted and the potential is reset. This abstraction captures key aspects of neuronal excitability with minimal computational complexity. Figure concept inspired by [50].

neuromorphic hardware. These features make the LIF model a popular choice for large-scale spiking neural networks [50].

Synaptic Input and Spike Generation

Synaptic input is typically modeled as a spike train. For a presynaptic neuron i, the spike train is represented as:

$$z_i(t) = \sum_s \delta(t - t^{\text{spk}}), \tag{2.6}$$

the total input current to a postsynaptic neuron j is given by:

$$I_j(t) = \sum_i \theta_{ji}\, z_i(t - d_{ji}), \tag{2.7}$$

where θ_{ji} is the synaptic weight from neuron i to neuron j, and the term d_{ji} represents the synaptic delay for the $j \leftarrow i$ connection. These delay accounts for the time it takes for a spike from neuron i to influence the membrane potential of neuron j.

In some models, the spike train $z_i(t)$ is further convolved with a synaptic kernel to capture the temporal dynamics of synaptic transmission more accurately, like alpha functions or exponential decays [57, 50]. This convolution introduces a gradual rise and decay of postsynaptic currents, reflecting the biological time course of synaptic conductances rather than assuming instantaneous transmission.

A postsynaptic neuron j emits a spike when its membrane potential exceeds a fixed threshold V_{th}:

$$z_j(t) \;=\; \Theta\big(V_j(t) - V_{\text{th}}\big), \qquad \Theta(x) \;=\; \begin{cases} 1, & x > 0, \\ 0, & x \le 0. \end{cases} \tag{2.8}$$

Following spike emission, the membrane potential is reset either to a fixed reset value V_r, or by subtracting a constant, as in some variants such as the model proposed by Bellec et al. [16].

$$V_j(t^+) \;=\; \begin{cases} V_r, & \text{(hard reset)}, \\ V_j(t) - V_{\text{th}}, & \text{(soft reset, e.g. Bellec \textit{et al.})} \end{cases} \qquad \text{whenever } z_j(t) = 1, \tag{2.9}$$

following this reset, the neuron immediately enters a refractory period t_{ref} during which it is temporarily unable to spike, regardless of incoming input.

Numerical Implementation of the LIF Neuron

To simulate leaky integrate-and-fire (LIF) neurons in discrete time, the continuous membrane potential dynamics are reformulated using a time step δt. A common and efficient approach is to use a forward Euler update with a precomputed decay constant:

$$\alpha = \exp\left(-\frac{\delta t}{\tau_m}\right), \tag{2.10}$$

which captures the exponential decay of the membrane potential over time. The discretized update rule for the membrane potential v_j^t of neuron j at time step $t + \delta t$ is given by:

$$v_j^{t+\delta t} = \alpha\, v_j^t + (1-\alpha)\, R_m\, I_j^t \tag{2.11}$$

where θ_{ji} is the synaptic weight from presynaptic neuron i to postsynaptic neuron j, z_i^t is the binary spike output from neuron i at time t, and R_m is the membrane resistance.

For simplicity, the factor $(1-\alpha)R_m$ can be absorbed into the synaptic weights, yielding a more compact form:

$$v_j^{t+\delta t} = \alpha\, v_j^t + \sum_i \theta_{ji}\, z_i^t, \tag{2.12}$$

Here, the total input current is expressed directly as a weighted sum of presynaptic spike trains, without the need to explicitly model the input current.

A spike is emitted by neuron j at time t if its membrane potential exceeds the firing threshold V_{th}, which can be written using the Heaviside step function:

$$z_j^t = \Theta\left(v_j^t - V_{\text{th}}\right), \qquad \Theta(x) = \begin{cases} 1, & x > 0, \\ 0, & x \leq 0. \end{cases} \tag{2.13}$$

After a spike is emitted, the membrane potential $v_j^{t+\delta t}$ is reset to a fixed value V_r in the case of a standard hard reset. In the case of a soft reset, the membrane potential is reduced by the threshold value V_{th}. This can be incorporated into the update rule as follows [56]:

$$v_j^{t+\delta t} = \alpha\, v_j^t + \sum_i \theta_{ji}\, z_i^t - V_{\text{th}}\, z_j^t, \tag{2.14}$$

The neuron then enters a refractory period, during which it is prevented from spiking—even if its membrane potential exceeds the threshold. This numerically stable and efficient update scheme makes the LIF model particularly suitable for large-scale simulations and deployment on neuromorphic hardware [58].

2.5.2 Adaptive Leaky Integrate-and-Fire Neuron

The Adaptive Leaky Integrate-and-Fire (ALIF) neuron extends the standard LIF model by incorporating an adaptive firing threshold A_j^t, which evolves dynamically in response to the neuron's spiking history. This adaptive mechanism enables the neuron to exhibit spike-frequency adaptation, which is a biologically observed phenomenon where the firing rate decreases over time under sustained stimulation.

Following [56], the dynamics of the adaptive threshold in discrete time are defined as:

$$A_j^t = V_{\text{th}} + \beta a_j^t, \tag{2.15}$$

$$a_j^{t+\delta t} = \rho a_j^t + (1-\rho) z_j^t, \tag{2.16}$$

where a_j^t is the internal adaptation variable, β is a scaling factor, and z_j^t is the binary spike output of neuron j at time t. The decay factor ρ governs the temporal evolution of the adaptation state and is given by:

$$\rho = \exp\left(-\frac{\delta t}{\tau_a}\right), \tag{2.17}$$

with τ_a denoting the adaptation time constant and δt the simulation time step.

The spiking condition is then modified to reflect the adaptive threshold:

$$z_j^t = \Theta\left(V_j^t - A_j^t\right), \qquad \Theta(x) = \begin{cases} 1, & x > 0, \\ 0, & x \leq 0. \end{cases} \tag{2.18}$$

This adaptive threshold mechanism allows the ALIF neuron to incorporate history dependent dynamics, thereby enabling richer temporal processing capabilities. As demonstrated in Bellec et al. [16], ALIF neurons are particularly effective in tasks requiring memory and the integration of temporal information over extended durations.

2.5.3 Leaky Integrator Neuron

The Leaky Integrator (LI) neuron is a simplified model of neuronal dynamics that captures the gradual accumulation and decay of input signals over time. In contrast to spiking neuron models, the LI neuron does not emit discrete spikes and is instead characterized by maintaining a continuous-valued membrane potential. The membrane potential $V(t)$ evolves similarly to that of a LIF neuron (they leak and integrate), but without thresholding or reset mechanisms (they do not fire).

Due to its continuous output, the LI neuron is particularly well-suited for producing real-valued responses, such as those required in regression tasks or for interpreting network activity. In this thesis, we use LI neurons in the final layer of our networks to generate continuous outputs and refer to them as **readout neurons**.

2.6 Artificial Synapses

The modeling of synapses between spiking neurons utilizes various abstraction levels to achieve an optimal trade-off between biological realism and computational efficiency. Synaptic interactions are modulated by synaptic weights which control postsynaptic responses when spikes arrive [44]. Moderately detailed models include temporal dynamics of synaptic transmission through mechanisms such as synaptic delays [59] or synaptic plasticity processes [60]. Advanced biophysical models can incorporate neurotransmitter release processes along with receptor dynamics [61, 62, 63]. The selection of synapse models is determined by research applications which vary from simulated neural circuitry on a large scale to detailed studies of synapses during learning and memory processes.

2.7 Spiking Neural Networks

Spiking Neural Networks (SNNs) are artificial neural network models that utilize spiking neurons to emulate both the temporal behaviors and event-based processing found in biological neural circuits. SNNs transmit information via discrete spike events unlike traditional neural networks which function based on continuous activation values. The temporal encoding mechanism supports the natural representation of time-dependent phenomena, such as sensory processing and motor control [50].

SNNs can be conceptualized as directed graphs where nodes symbolize neurons and edges constitute synaptic connections. In this framework, a spike generated by a neuron propagates along its outgoing synapses, each modulating the signal according to its synaptic weight. For a given synapse ($i \to j$) connecting neuron i to neuron j, neuron i is designated as the **presynaptic neuron** and neuron j as the **postsynaptic neuron**. From the perspective of an individual neuron, synapses are classified as **incoming** if they terminate at the neuron, and **outgoing** if they originate from it.

This thesis emphasizes recurrent architectures within the SNN paradigm, particularly Spiking Recurrent Neural Networks (SRNNs) and networks comprised of Leaky Integrate-and-Fire (LIF) neurons.

Spiking Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are distinguished by their ability to maintain internal states, which enable them to process sequences by retaining contextual information from previous time steps. Unlike feedforward networks, RNNs incorporate feedback loops that allow information to circulate, making them well-suited for tasks with temporal dependencies, such as speech recognition and time series prediction [64].

Spiking Recurrent Neural Network (SRNN) build upon this idea through the use of spiking neurons that better replicate the event-driven dynamics found in biological neural circuits. The recurrent design of SRNNs functions as a basic model of the brain's complex feedback networks that support essential cognitive abilities including memory processing, attention, and decision-making [65, 66, 67, 68].

To understand how information flows through the network, we will examine a simple example of a spiking recurrent neural network (SRNN) with three layers: an input layer, a hidden recurrent layer, and a readout layer.

Input Layer The input layer receives external stimuli, represented by a sequence of input data elements, $\mathscr{X}_i$, consisting of T vectors:

$$\mathscr{X}_i = \left(\boldsymbol{x}_i^t\right)_{t=1\ldots T}, \tag{2.19}$$

where each input vector $\boldsymbol{x}_i^t \in \{0,1\}^I$ and I is the feature dimension. For simplicity, we assume this layer consists of I input nodes, each corresponding to a feature. These input nodes transmit binary information to the hidden recurrent layer, which contains H recurrent nodes.

Hidden Recurrent Layer The hidden recurrent layer processes the input over time and maintains a hidden state that captures the temporal dynamics of the input sequence. The hidden binary state at time t is denoted as $\boldsymbol{z}^t \in \mathbb{R}^H$. This state is updated based on the current input $\boldsymbol{x}_i^t$, the previous hidden state $\boldsymbol{z}^{t-1}$ and the previous pre-activation $\boldsymbol{v}^t$.

For a single recurrent node with a leaky integrate-and-fire (LIF) neuron model, the update rule is given by:

$$\boldsymbol{v}^t = \alpha\,\boldsymbol{v}^{t-1} + \theta^{\text{rec}}\,\boldsymbol{z}^{t-1} + \theta^{\text{in}}\,\boldsymbol{x}_i^t, \tag{2.20}$$

$$\boldsymbol{z}^t = \Theta\left(\boldsymbol{v}^t - V_{\text{th}}\mathbb{1}\right), \tag{2.21}$$

where $\boldsymbol{v}^t \in \mathbb{R}^H$ represents the pre-activation (membrane potential) at time t, $\theta^{\text{rec}} \in \mathbb{R}^{H\times H}$ is the recurrent weight matrix, $\theta^{\text{in}} \in \mathbb{R}^{H\times I}$ is the input weight matrix, and α is the decay factor of the membrane potential. The term $V_{\text{th}}\mathbb{1}$ represents the spiking threshold, where $\mathbb{1}$ is a vector of ones with the same dimension as the hidden state. The hidden state $\boldsymbol{z}^t$ is determined by applying a thresholding function Θ to the pre-activation $\boldsymbol{v}^t$.

Readout Layer The readout layer, consisting of O readout nodes, generates the final output based on the processed information. The readout at time t is denoted as $\boldsymbol{y}^t \in \mathbb{R}^O$. The readout layer takes the hidden state $\boldsymbol{z}^t$ as input and produces the final output. For readout neurons modeled as leaky integrators (LI neurons), the update rule is:

$$\boldsymbol{y}^t = \kappa\,\boldsymbol{y}^{t-1} + \theta^{\text{out}}\,\boldsymbol{z}^t, \tag{2.22}$$

where $\boldsymbol{y}^{t-1} \in \mathbb{R}^O$ is the readout at time $t-1$, $\boldsymbol{y}^t$ is the readout at time t, $\theta^{\text{out}} \in \mathbb{R}^{O\times H}$ is the readout weight matrix, and κ is the decay factor.

For each input element $\mathscr{X}_i$, the readout layer generates a sequence of outputs, $\mathscr{Y}_i$, given by:

$$\mathscr{Y}_i = \left(\boldsymbol{y}_i^t\right)_{t=1\ldots T}, \tag{2.23}$$

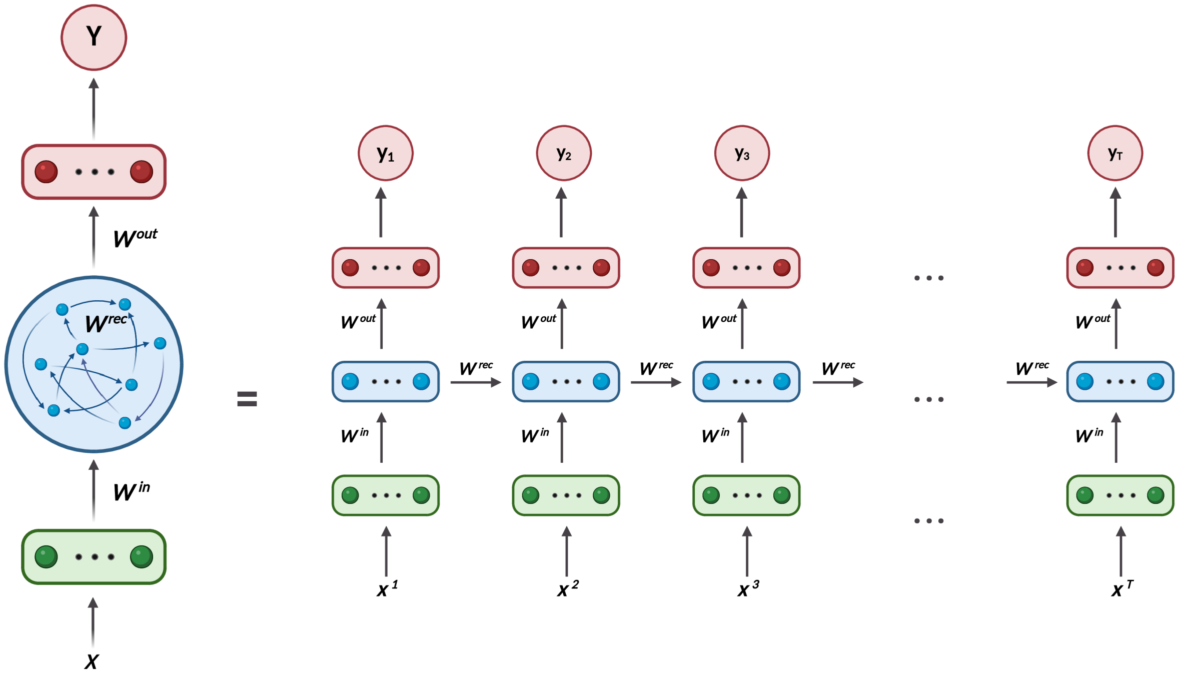


Figure 2.8: **Spiking Recurrent Neural Network (SRNN) Forward Pass.** The figure illustrates an unrolled SRNN over time, where each time step is represented as a distinct layer in a deep feedforward architecture. Input neurons (green), recurrent hidden neurons (blue), and output neurons (red) are shown, with forward connections indicated by black arrows. Figure concept inspired by [69].

where $\boldsymbol{y}_i^t \in \mathbb{R}^O$ is the readout at time t for the input element $\mathscr{X}_i$.

Figure 2.8 illustrates the architecture of the SRNN described above. The arrows represent the flow of information between the input, hidden, and output layers, highlighting the recurrent connections within the hidden layer. The figure also demonstrates the unrolled representation of a SRNN over time, where each time step is visualized as a separate layer in a deep feedforward network.

2.8 Gradient-Based Learning Methods and Their Biological Plausibility

In general terms, learning in artificial neural networks refers to the process of adjusting synaptic weights to improve the network's performance on a given task. Performance is typically measured using a loss function, which quantifies how much the network's output deviates from its desired goal. Broadly speaking, training can be understood as the process of minimizing this loss.

Among the many techniques developed to achieve this, which include perturbation methods [70, 71, 72, 73], and evolutionary algorithms [74, 75], gradient-based methods are by far the most widespread, successful, and efficient [76, 77, 78, 79]. These methods use an optimization process to update network parameters by moving in the steepest descent direction to minimize loss. Backpropagation (BP) [80] together with its recurrent network variant Backpropagation Through Time (BPTT) [81] serve as the main algorithms that powers this process.

The successful application of gradient-based methods combined with the absence of viable alternatives has motivated researchers to hypothesize that the human brain may employ basic processes similar to gradient-based methods [82, 76, 83]. The concept of gradient-based optimization as a possible mechanism for biological learning has generated an increasing amount of interest and research in neuroscience [84].

2.8.1 Supervised Learning

The supervised learning paradigm involves training machine learning models with labeled data consisting of input-target pairs. Supervised learning concerns the construction of a function that connects input data with output results and performs accurately on new, previously unseen data [85]. The supervised learning methodology finds extensive application in several domains such as image classification and natural language processing as well as time series prediction [86, 64].

In this thesis, we investigate the supervised learning of SRNNs, aiming to minimize a loss function that quantifies the discrepancy between the network's output sequence $\mathscr{Y}$ and the corresponding target sequence $\mathscr{Y}^*$:

$$\mathscr{L}(\mathscr{Y},\mathscr{Y}^*). \tag{2.24}$$

Since $\mathscr{Y}$ is a sequence of outputs over time, the total loss is typically defined as the sum of losses at each individual time step:

$$\mathscr{L}(\mathscr{Y},\mathscr{Y}^*) = \sum_{t=1}^{T} \mathscr{L}^t(\boldsymbol{y}^t, \boldsymbol{y}^t_{\text{target}}), \tag{2.25}$$

where $\boldsymbol{y}^t$ is the network output at time t, and $\boldsymbol{y}^t_{\text{target}}$ is the corresponding target output. The function $\mathscr{L}^t$ can take various forms, such as mean squared error (MSE) or cross-entropy loss, depending on the task and the nature of the output.

The supervised training process of a SRNN comprises the following steps:

1. **Forward Pass:** A sequence of input data $\mathscr{X}_i$ is fed into the network. At each time step t, the hidden state is updated according to the network's recurrent dynamics, and the readout layer generates an output $\hat{\boldsymbol{y}}^t$ based on the current hidden state $\boldsymbol{z}^t$.

2. **Loss Computation:** The predicted output $\boldsymbol{y}^t$ is compared with the target output $\boldsymbol{y}^t_{\text{target}}$ to compute the loss $\mathscr{L}^t$, which reflects the network's performance at time t.

3. **Backward Pass:** The gradients of the total loss with respect to the network parameters are computed using Backpropagation Through Time (BPTT, see Section 2.8.2). This involves recursively applying the chain rule through time, propagating error signals backward from the output layer through the recurrent structure.

4. **Parameter Update:** The network parameters are updated using an optimization algorithm (e.g., stochastic gradient descent) based on the computed gradients to minimize the total loss.

This process is repeated for multiple epochs, iterating over the training data until the network converges to a set of parameters that minimize the loss function.

2.8.2 Backpropagation Through Time (BPTT)

A fundamental component of the training procedure described in Section 2.8.1 is the backward pass, which is executed using the Backpropagation Through Time (BPTT) algorithm [81]. BPTT extends the standard backpropagation algorithm to accommodate the temporal dynamics of recurrent neural networks (RNNs) [81], and by extension, spiking recurrent neural networks (SRNNs) [16]. It achieves this by incorporating the recurrent structure of these networks into the gradient computation.

The core idea of BPTT is to unroll the recurrent network across time steps and apply the chain rule recursively. Let $\boldsymbol{h}^t$ denote the hidden state at time step t, $\boldsymbol{\theta}$ the model parameters (e.g., weights), and $\mathscr{L}$ the loss function. The gradient of the loss with respect to the parameters is given by [87]:

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}}{\partial \boldsymbol{h}^t} \frac{\partial \boldsymbol{h}^t}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \left(\frac{\partial \mathscr{L}}{\partial \boldsymbol{h}^{t+1}} \frac{\partial \boldsymbol{h}^{t+1}}{\partial \boldsymbol{h}^t} + \frac{\partial \mathscr{L}^t}{\partial \boldsymbol{h}^t} \right) \frac{\partial \boldsymbol{h}^t}{\partial \boldsymbol{\theta}}. \tag{2.26}$$

This unrolled formulation allows each time step to be interpreted as a layer in a deep feedforward network, enabling the use of conventional backpropagation to compute gradients.

To facilitate this, the sensitivity of the loss with respect to the hidden state is defined recursively as:

$$\boldsymbol{\delta}_h^t = \frac{\partial \mathscr{L}}{\partial \boldsymbol{h}^t} = \frac{\partial \mathscr{L}}{\partial \boldsymbol{h}^{t+1}} \frac{\partial \boldsymbol{h}^{t+1}}{\partial \boldsymbol{h}^t} + \frac{\partial \mathscr{L}^t}{\partial \boldsymbol{h}^t}. \tag{2.27}$$

The recursion starts from the final time step, where the sensitivity is initialized as:

$$\boldsymbol{\delta}_h^{T+1} = \mathbf{0}. \tag{2.28}$$

The backward recursion is performed after the forward pass, iterating from $t = T$ down to $t = 1$, to accumulate the gradient as described in Equation 2.26. This temporal unrolling incurs a memory complexity of $\mathscr{O}(Tn^2)$ and a computational complexity of $\mathscr{O}(Tn^2)$, assuming dense weight matrices and n hidden units (see Table 2.1).

For the SRNN described in Section 2.7, the forward pass at time step t is given by:

$$\begin{aligned}
\boldsymbol{v}^t &= \alpha\, \boldsymbol{v}^{t-1} + \theta^{\text{rec}}\, \boldsymbol{z}^{t-1} + \theta^{\text{in}}\, \boldsymbol{x}_i^t, \\
\boldsymbol{z}^t &= \Theta\left(\boldsymbol{v}^t - V_{\text{th}}\mathbb{1}\right), \\
\boldsymbol{y}^t &= \kappa\, \boldsymbol{y}^{t-1} + \theta^{\text{out}}\, \boldsymbol{z}^t, \\
\mathscr{L}^t &= \mathscr{L}^t\left(\boldsymbol{y}^t, \boldsymbol{y}^t_{\text{target}}\right),
\end{aligned} \tag{2.29}$$

where $\boldsymbol{v}^t$ is the membrane potential (pre-activation), $\boldsymbol{z}^t$ is the spiking output (hidden state), and $\boldsymbol{y}^t$ is the readout at time t. The scalars α and κ represent the decay factors of the membrane and readout dynamics, respectively. The trainable parameters include the synaptic weight matrices θ^{rec}, θ^{in}, and θ^{out}. The total loss over a sequence of length T is defined as:

$$\mathscr{L} = \sum_{t=1}^{T} \mathscr{L}^t.$$

To compute the gradients via BPTT, we introduce sensitivities for the readout, hidden state, and membrane potential at each time step. We denote the sensitivities as $\boldsymbol{\delta}_y^t$, $\boldsymbol{\delta}_z^t$, and $\boldsymbol{\delta}_v^t$ respectively. These represent the gradients of the total loss $\mathscr{L}$ with respect to the intermediate variables of the network at time step t (see Figure 2.9). Specifically:

- $\boldsymbol{\delta}_y^t = \frac{\partial \mathscr{L}}{\partial \boldsymbol{y}^t}$ quantifies how changes in the readout $\boldsymbol{y}^t$ affect the overall loss.
- $\boldsymbol{\delta}_z^t = \frac{\partial \mathscr{L}}{\partial \boldsymbol{z}^t}$ captures the effect of perturbations in the spiking output (hidden state) on the loss.
- $\boldsymbol{\delta}_v^t = \frac{\partial \mathscr{L}}{\partial \boldsymbol{v}^t}$ measures the influence of changes in the membrane potential on the loss.

The BPTT algorithm computes these sensitivities recursively, starting from the output layer and moving backward through the hidden layer to the input layer. The process is as follows:

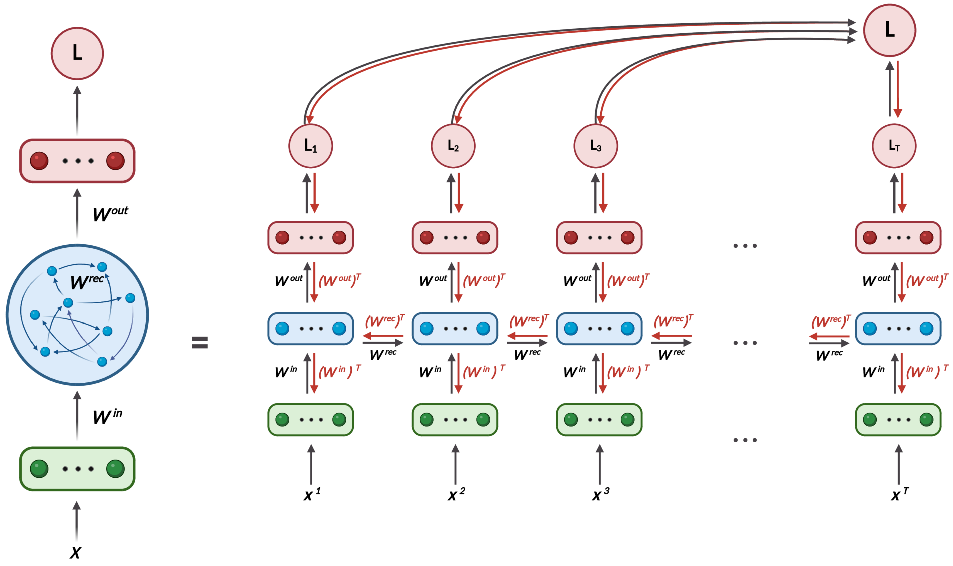


Figure 2.9: **Forward and Backward Passes in a Spiking Recurrent Neural Network (SRNN).** Forward connections (black arrows) propagate neural activity over time to compute hidden states and outputs. Backward connections (red arrows) represent the reverse-time flow of error signals used to compute gradients during training. The backward pass calculates the sensitivity of the loss with respect to internal states, including membrane potentials and hidden activities, and proceeds from the final time step ($t = T$) back to the beginning ($t = 1$) to accumulate gradients for parameter updates. Figure concept inspired by [69].

Readout Layer. Starting by defining the per-step error signal:

$$\boldsymbol{E}^t := \frac{\partial \mathscr{L}^t\left(\boldsymbol{y}^t, \boldsymbol{y}^t_{\text{target}}\right)}{\partial \boldsymbol{y}^t}, \qquad t = 1, \ldots, T. \tag{2.30}$$

we express the sensitivity at the readout layer at time step t as:

$$\boldsymbol{\delta}^t_y = \frac{\partial \mathscr{L}}{\partial \boldsymbol{y}^t} = \boldsymbol{E}^t + \kappa\, \boldsymbol{\delta}^{t+1}_y, \tag{2.31}$$

with the terminal condition $\boldsymbol{\delta}^{T+1}_y = \mathbf{0}$.

Hidden Layer. The sensitivity at the hidden layer (spiking output) at time step t is computed recursively as:

$$\boldsymbol{\delta}^t_z = \left(\frac{\partial \boldsymbol{y}^t}{\partial \boldsymbol{z}^t}\right)^{\top} \boldsymbol{\delta}^t_y + \left(\frac{\partial \boldsymbol{v}^{t+1}}{\partial \boldsymbol{z}^t}\right)^{\top} \boldsymbol{\delta}^{t+1}_v, \tag{2.32}$$

where the membrane potential sensitivity $\boldsymbol{\delta}^t_v$ is given by:

$$\boldsymbol{\delta}^t_v = \left(\frac{\partial \boldsymbol{z}^t}{\partial \boldsymbol{v}^t}\right)^{\top} \boldsymbol{\delta}^t_z + \alpha\, \boldsymbol{\delta}^{t+1}_v, \tag{2.33}$$

with the initial condition $\boldsymbol{\delta}_v^{T+1} = \mathbf{0}$.

At this stage, we temporarily ignore the complications arising from the non-differentiability of the spike function $z = \Theta(\cdot)$. Instead, we denote its surrogate derivative as $\Theta'(\boldsymbol{v}^t)$, which allows gradient-based training to proceed using approximate methods. The remaining partial derivatives are straightforward:

$$\frac{\partial \boldsymbol{y}^t}{\partial \boldsymbol{z}^t} = \theta^{\text{out}}, \quad \frac{\partial \boldsymbol{v}^{t+1}}{\partial \boldsymbol{z}^t} = \theta^{\text{rec}}. \tag{2.34}$$

Substituting these into Equations (2.32) and (2.33), we obtain:

$$\boldsymbol{\delta}_z^t = (\theta^{\text{out}})^\top \boldsymbol{\delta}_y^t + (\theta^{\text{rec}})^\top \boldsymbol{\delta}_v^{t+1}, \tag{2.35}$$

$$\boldsymbol{\delta}_v^t = \left(\Theta'(\boldsymbol{v}^t)\right)^\top \boldsymbol{\delta}_z^t + \alpha \boldsymbol{\delta}_v^{t+1}. \tag{2.36}$$

Parameter Gradients via BPTT

After computing the error-sensitivities $\boldsymbol{\delta}_y^t$, $\boldsymbol{\delta}_z^t$ and $\boldsymbol{\delta}_v^t$ for all $t = 1, \ldots, T$ using the backward recursion, the gradients of the loss $\mathscr{L}$ with respect to the network parameters are obtained by accumulating contributions over time:

$$\frac{\partial \mathscr{L}}{\partial \theta^{\text{out}}} = \sum_{t=1}^{T} \boldsymbol{\delta}_y^t \left(\boldsymbol{z}^t\right)^\top, \quad \frac{\partial \mathscr{L}}{\partial \theta^{\text{rec}}} = \sum_{t=1}^{T} \boldsymbol{\delta}_v^t \left(\boldsymbol{z}^{t-1}\right)^\top, \quad \frac{\partial \mathscr{L}}{\partial \theta^{\text{in}}} = \sum_{t=1}^{T} \boldsymbol{\delta}_v^t \left(\boldsymbol{x}^t\right)^\top. \tag{2.37}$$

Because these terms depend on all intermediate activations and internal states, standard BPTT requires storing $\{\boldsymbol{z}^t\}_{t=1}^T$ and $\{\boldsymbol{v}^t\}_{t=1}^T$ during the forward pass, this requirement is called backward-in-time credit assignment, and is one of the main limitations of BPTT in terms of biological plausibility (see subsubsection 2.8.2.1).

2.8.2.1 Biological Plausibility of BPTT

Despite BPTT being the standard algorithm for training recurrent neural networks, including spiking recurrent neural networks, there are strong theoretical and biological arguments against the idea that the brain directly implements this algorithm. Several critical limitations challenge its feasibility as a biologically realistic learning mechanism:

Backward-in-Time Credit Assignment. BPTT explicitly requires the propagation of error signals back in time, which means that the brain would have to store precise sequences of internal states and then compute error signals in reverse chronological order. This lack of temporal locality is incompatible with the causal and online nature of biological processing [82, 16].

Weight Transport Problem. Backpropagation and BPTT both requires that the transpose of the forward weights must be used during the backward pass. This is known as the *weight transport problem* [88, 89]. In biological systems, there is no known mechanism that ensures such precise symmetry between feedforward and feedback synapses.

Non-differentiable Activation Functions. Spiking neurons generate discrete all-or-none events, which are inherently non-differentiable. BPTT relies on gradient descent, which requires differentiable functions [90, 56].

Non-locality of Computation. Gradient-based algorithms, such as BPTT, require each synapse to access global information, such as subsequent error signals and weights in distant parts of the network. In contrast, biological plasticity is thought to be driven by local information and modulatory signals [46, 16].

Taken together, these issues strongly suggest that while BPTT may be effective in artificial systems, it is not directly implementable in biological networks, which prompts the neccessity of developing thechniques that can bridge the gap between biologically plausible mechanisms and the learning power of BPTT.

2.8.3 Real-Time Recurrent Learning (RTRL)

Real-Time Recurrent Learning (RTRL) [91] is an algorithm designed to compute gradients for recurrent neural networks in an online fashion, allowing immediate updates to synaptic weights based on the current state of the network. The gradient computation can be written as [87]:

$$\frac{\partial \mathscr{L}}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}^t}{\partial \boldsymbol{h}^t} \frac{\partial \boldsymbol{h}^t}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}^t}{\partial \boldsymbol{h}^t} \left(\frac{\partial \boldsymbol{h}^t}{\partial \boldsymbol{\theta}} + \frac{\partial \boldsymbol{h}^t}{\partial \boldsymbol{h}^{t-1}} \frac{\partial \boldsymbol{h}^{t-1}}{\partial \boldsymbol{\theta}} \right), \tag{2.38}$$

which mirrors the chain rule form for BPTT in Equation 2.26. RTRL is mathematically equivalent to BPTT: both compute identical gradients under exact arithmetic. To achieve this, RTRL maintains a running “influence” Jacobian

$$J^t := \frac{d\boldsymbol{h}^t}{d\boldsymbol{\theta}}$$

during the forward pass, enabling online gradient computation. Following the notation of [87], define

$$I^t := \frac{\partial \boldsymbol{h}^t}{\partial \boldsymbol{\theta}}, \quad D^t := \frac{\partial \boldsymbol{h}^t}{\partial \boldsymbol{h}^{t-1}}.$$

Then the recursive update for the influence Jacobian is

$$J^t = I^t + D^t J^{t-1}, \tag{2.39}$$

With the initialization $J^0 = 0$, each influence Jacobian

$$J^t \;=\; \frac{d\boldsymbol{h}^t}{d\boldsymbol{\theta}}$$

summarizes how parameter variations affect the hidden state at time t. We then accumulate the gradient contribution forward in time by defining

$$G^t \;:=\; G^{t-1} \;+\; \frac{\partial \mathscr{L}^t}{\partial \boldsymbol{h}^t}\, J^t,$$

with $G^0 = 0$. At the end of the sequence,

$$G^T \;=\; \frac{d\mathscr{L}}{d\boldsymbol{\theta}}.$$

This forward accumulation, and associated temporal locality, stands in contrast to BPTT's backward pass through time, and offers a solution to the backward-in-time credit assignment problem (see subsubsection 2.8.2.1). However, maintaining and updating the full Jacobian $\boldsymbol{J}_\theta^t$ at each step incurs a memory complexity of $\mathcal{O}(n^3)$ and a computational complexity of $\mathcal{O}(Tn^4)$, which grow poorly with network size and limit RTRL's practicality for large-scale models [91, 82] (see Table 2.1).

Table 2.1: Computational and memory complexities of BPTT and RTRL for a network with n hidden units, T time steps, and dense weight matrices.

Algorithm	**Computational Complexity**	**Memory Complexity**
BPTT	$\mathcal{O}(Tn^2)$	$\mathcal{O}(Tn^2)$
RTRL	$\mathcal{O}(Tn^4)$	$\mathcal{O}(n^3)$

2.8.4 Biologically Plausible Learning Rules

Real-Time Recurrent Learning (RTRL), despite its high computational and memory costs, has inspired the development of several biologically motivated algorithms. This interest stems from RTRL's key advantage: its ability to compute online gradients is a feature that aligns with the temporal locality observed in biological credit assignment mechanisms [84]. However, RTRL's reliance on exact gradients and non-local information significantly limits its viability for real-time learning in spiking neural networks (SNNs).

To overcome these limitations, numerous biologically plausible approximations to RTRL have been proposed, aiming to preserve its benefits while adhering to the constraints of local computation and synaptic plasticity. Superspike [92] introduces surrogate gradients combined with a three-factor learning rule to enable gradient-based learning in spiking systems. Multidigraph Learning (MDGL) [20] approximates credit assignment using structured, biologically inspired feedback. RFLO (Random Feedback Local Online learning) [93] replaces exact gradients with random feedback and local activity. DECOLLE (Deep Continuous Local Learning) [94] employs layerwise error signals and surrogate gradients to support local learning in deep spiking networks.

Other approaches, such as ETLP (Event-based Three-factor Local Plasticity) [95] and OSTL (Online Spatio-Temporal Learning) [96], explore learning mechanisms rooted in spike timing and eligibility traces, further aligning with biological observations. Among these methods, e-prop (eligibility propagation) [16] stands out for its strong balance between biological plausibility and computational effectiveness (see chapter 3). It preserves the online learning capability of RTRL while replacing non-local gradient computations with local eligibility traces modulated by top-down learning signals.

The present thesis focuses on e-prop and investigates algorithmic developments built on top of this framework, aiming to improve its learning efficiency, generalization, and applicability.

2.9 Simulation Technology for Spiking Neural Networks

Simulating spiking neural networks (SNNs) requires dedicated technologies that capture their event-based nature and complex dynamics across time and space.

General-purpose SNN simulators provide flexible, software-only environments for developing and testing spiking models of varying scale and complexity. Tools such as Brian2, NEST, NEURON, BindsNET and Nengo offer high level Python interfaces (often with C++ backends) that enable rapid prototyping, detailed biophysical modelling, and integration with machine learning libraries. Brian2 [97] prioritizes user friendliness and concise model definitions, while NEST [98] excels at scalable, large-scale network simulations. NEURON [99] supports compartmental modelling of single neurons and small networks with fine temporal resolution. BindsNET [100] uses PyTorch for supervised and unsupervised learning in spiking networks, and Nengo [101] provides a cognitive architecture framework that unifies rate-based and spike-based approaches. These general-purpose platforms are useful for exploring algorithmic principles and benchmarking novel neuron and synapse models.

Neuromorphic hardware-oriented SNN simulators help bridge the gap between software models and energy-efficient, event-driven hardware implementations. Frameworks such as Lava [102] for Intel's Loihi chip [103], sPyNNaker [104] for the SpiNNaker [105] platform, and the BrainScaleS [106] system offer software APIs that map high-level network descriptions onto neuromorphic chips, exploiting parallelism and analog/digital co-processing for low-latency execution. Each platform provides specialized toolchains for compilation, placement, and routing of spikes across the hardware's mesh.

2.9.1 The NEST Simulator

NEST (NEural Simulation Tool) is a high-performance, open-source simulator designed for the large-scale simulation of spiking neural networks [98]. It emphasizes biological realism by supporting event-driven synaptic updates, precise spike timing, and efficient communication between neurons. These features make NEST particularly well-suited for modeling brain-like network dynamics and investigating biologically plausible learning rules.

In NEST, networks are built from three fundamental components: *nodes*, *connections*, and *events* [98]. Nodes represent neurons or devices (e.g., spike generators or recorders), connections model synapses with optional plasticity mechanisms, and events represent spike transmissions or parameter updates. NEST employs parallelism via MPI and OpenMP to distribute computations across multiple threads and processes, allowing simulations to scale up to millions of neurons and billions of synapses.

The simulator supports modular extensibility, enabling researchers to define custom neuron and synapse models, learning rules, and network architectures [98]. Recent versions of NEST have introduced support for three-factor learning rules [107], including the eligibility propagation (e-prop) framework [108], alongside comprehensive documentation and tutorials.

This thesis focuses on the implementation of e-prop in NEST, with the goal of enabling scalable and biologically grounded online learning in large spiking networks. In addition to detailing the integration of e-prop into the NEST simulation engine, this work explores both biological and algorithmic improvements to the learning rule. Furthermore, it investigates the application of e-prop to continual learning scenarios, where networks must adapt incrementally to new information while retaining previously acquired knowledge.

Chapter 3

Event-driven Eligibility Propagation with Additional Biologically Inspired Features

Biological neural systems exhibit exceptional learning capabilities through local, sparse, and energy-efficient mechanisms. Emulating these principles in artificial models offers a compelling route toward scalable and adaptive machine learning. However, standard training methods for recurrent networks, such as Backpropagation Through Time (BPTT), remain fundamentally misaligned with biological reality due to their reliance on global signals, dense updates, and offline computation.

Eligibility propagation (e-prop), introduced by Bellec et al. [16], addresses some of these issues by approximating gradient-based learning using local synaptic traces and neuron-specific feedback. Yet, typical implementations of e-prop still rely on time-driven simulation frameworks, which enforce synchronous updates and instantaneous communication. These assumptions that depart from the event-driven and asynchronous nature of biological computation.

In this chapter, we present an event-driven reformulation of the e-prop learning rule tailored for spiking neural networks (SNNs). In our approach, synaptic updates are triggered solely by spike events, closely mirroring the timing and locality of biological learning. This formulation enables more efficient computation in sparse networks and aligns with the structural and temporal constraints of real neural systems.

The method is implemented in the NEST simulator, a high-performance platform for large-scale brain modeling. Our implementation supports biologically realistic components such as delayed signal transmission, continuous membrane dynamics, sparse recurrent connectivity, and strictly local plasticity. We evaluate the approach on multiple tasks, demonstrating that relaxing certain mathematical constraints in favor of biological plausibility can yield effective and efficient learning.

3.1 Eligibility Propagation (e-prop)

In this section the core motivations and methodologies that establish Eligibility Propagation (e-prop) [16] as a biologically inspired alternative to Backpropagation Through Time (BPTT) [81] are discussed. In the first step of this process, the structure of a generic recurrent neural system is formalized. Next, the process of computing gradients through Backpropagation Through Time method (BPTT) for this network is presented. Finally, using this information, the computational bottlenecks and biologically implausible features of BPTT are identified, and e-prop's approach to address them is outlined.

3.1.1 Backpropagation Through Time (BPTT)

Consider a discrete-time recurrent system with input $\boldsymbol{x}^t \in \mathbb{R}^m$, parameters $\boldsymbol{\theta}$, and a scalar loss $\mathscr{L}$ accumulated over time. Each neuron j maintains an internal hidden state $\boldsymbol{h}_j^t \in \mathbb{R}^d$ capturing internal neuron variables such as membrane voltage or adaptation variables, and emits an observable state $z_j^t \in \{0,1\}$, typically representing a spike.

The dynamics of the system are governed by two functions: a hidden-state transition function M, and an observable-state function f:

$$\boldsymbol{h}_j^t = M\left(\boldsymbol{h}_j^{t-1}, z_j^t, \boldsymbol{x}^t; \boldsymbol{\theta}\right), \tag{3.1}$$

$$z_j^t = f\left(z_j^{t-1}, \boldsymbol{x}^t; \boldsymbol{\theta}\right), \tag{3.2}$$

The total loss is defined as the sum of instantaneous losses at each time step:

$$\mathscr{L} = \sum_{t=1}^{T} \mathscr{L}^t\left(\boldsymbol{z}^t\right), \tag{3.3}$$

where $\mathscr{L}^t$ denotes the loss at time t, depending on the vector of observable outputs $\boldsymbol{z}^t$ across all neurons.

Gradient of the Total Loss via BPTT. The goal of learning is to compute the gradient of the total loss $\mathscr{L}$ with respect to the parameters $\boldsymbol{\theta}$, which correspond to the synaptic weights.

Let θ_{ji} denote the weight from presynaptic neuron i to postsynaptic neuron j. By the chain rule, the total derivative of the loss with respect to θ_{ji} is:

$$\frac{d\mathscr{L}}{d\theta_{ji}} = \sum_{t=1}^{T} \frac{d\mathscr{L}}{dz_j^t} \cdot \frac{dz_j^t}{d\theta_{ji}}, \tag{3.4}$$

where each term accounts for the influence of the parameter θ_{ji} on the hidden state z_j^t, and in turn on the instantaneous loss $\mathscr{L}^t$. Importantly, the total derivative $dz_j^t/d\theta_{ji}$ accounts for the indirect influence of θ_{ji} through the entire history of network dynamics.

Expansion of the Loss Gradient. The total derivative $d\mathscr{L}/dz_j^t$ can be expanded using the temporal dependencies of the system. Specifically, each hidden state z_j^t may affect not only the immediate loss $\mathscr{L}^t$, but also all future losses $\mathscr{L}^\tau$ for $\tau > t$:

$$\frac{d\mathscr{L}}{dz_j^t} = \frac{\partial \mathscr{L}}{\partial z_j^t} + \sum_{\tau > t} \frac{\partial \mathscr{L}}{\partial z_j^\tau} \cdot \frac{dz_j^\tau}{dz_j^t}. \tag{3.5}$$

This expression reflects the influence of a state variable on both immediate and future outcomes.

Expansion of the State Derivative. Similarly, the total derivative of a hidden state with respect to a parameter is itself can be expanded, since z_j^t depends on previous states:

$$\frac{dz_j^t}{d\theta_{ji}} = \sum_{t' < t} \frac{\partial z_j^t}{\partial z_j^{t-1}} \cdots \frac{\partial z_j^{t'+1}}{\partial z_j^{t'}} \frac{\partial z_j^{t'}}{\partial \theta_{ji}}. \tag{3.6}$$

This expression captures the recursive nature of the hidden state dynamics, where each state depends on its predecessor.

For a more detailed description of the BPTT algorithm using matrix notation, see subsection 2.8.2.

Biological Plausibility. Equations (3.4) through (3.6) define the complete BPTT procedure for our recurrent system. It requires maintaining and propagating gradients backward through the entire unrolled sequence of network states. While widely used and effective in machine learning applications, this approach is biologically implausible, mainly because it assumes non-local, temporally global backward computations, which demands storage of all intermediate variables across time, a process unfeasible in biological systems. For a more detailed discussion on the biological plausibility of BPTT, see subsubsection 2.8.2.1

3.1.2 Eligibility Propagation (e-prop)

These limitations motivate the search for an alternative, online-compatible approximation that aligns more closely with known mechanisms in neuroscience. This section introduces the key steps that lead to the e-prop algorithm.

Step 1: Defining the Eligibility Trace $\boldsymbol{e_{ji}^t}$. To approximate gradients in a biologically plausible manner, e-prop introduces the notion of an *eligibility trace* e_{ji}^t, which quantifies how the synaptic parameter θ_{ji} influences the output spike z_j^t. This trace is supported by an intermediate quantity, the *eligibility vector* $\boldsymbol{\epsilon}_{ji}^t$, which tracks the influence of θ_{ji} on the internal state $\boldsymbol{h}_j^t$.

For each synapse $(i \to j)$, these quantities are defined as:

$$e_{ji}^t := \frac{dz_j^t}{d\theta_{ji}} = \frac{\partial z_j^t}{\partial \boldsymbol{h}_j^t} \cdot \boldsymbol{\epsilon}_{ji}^t, \tag{3.7}$$

$$\boldsymbol{\epsilon}_{ji}^t := \sum_{t'<t} \frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}} \cdots \frac{\partial \boldsymbol{h}_j^{t'+1}}{\partial \boldsymbol{h}_j^{t'}} \cdot \frac{\partial \boldsymbol{h}_j^{t'}}{\partial \theta_{ji}}. \tag{3.8}$$

To compute $\boldsymbol{\epsilon}_{ji}^t$ efficiently, the chain rule is unrolled into a recursive update:

$$\boldsymbol{\epsilon}_{ji}^t = \frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}} \cdot \boldsymbol{\epsilon}_{ji}^{t-1} + \frac{\partial \boldsymbol{h}_j^t}{\partial \theta_{ji}}, \tag{3.9}$$

with the base condition:

$$\boldsymbol{\epsilon}_{ji}^0 = \mathbf{0}.$$

This recursive formulation allows the eligibility trace e_{ji}^t to be computed *online*, i.e., incrementally at each time step using only local information available at that moment. From this incremental view, the eligibility trace can be interpreted as a memory mechanism that accumulates the influence of past synaptic activity on the current output.

Step 2: Introducing the Learning Signal. The gradient of the total loss with respect to a synaptic parameter θ_{ji} can be expressed in terms of the eligibility trace. From the BPTT formulation, it follows that:

$$\frac{d\mathscr{L}}{d\theta_{ji}} = \sum_{t=1}^{T} \frac{d\mathscr{L}}{dz_j^t} \cdot e_{ji}^t. \tag{3.10}$$

To simplify notation, the *learning signal* is defined as:

$$\tilde{L}_j^t := \frac{d\mathcal{L}}{dz_j^t}, \tag{3.11}$$

which quantifies the sensitivity of the loss to changes in the postsynaptic activity z_j^t. This term captures both the immediate effect of z_j^t on the loss at time t and its indirect influence on future losses via the recurrent dynamics.

Substituting this into Equation (3.10) yields the compact form:

$$\frac{d\mathcal{L}}{d\theta_{ji}} = \sum_{t=1}^{T} \tilde{L}_j^t \cdot e_{ji}^t.$$

Although exact, computing $\tilde{L}_j^t$ generally requires access to future states $z_j^{\tau>t}$, making the expression biologically implausible.

Step 3: The e-prop Approximation. To obtain a biologically plausible learning rule, e-prop makes a key simplification. Instead of computing the full derivative $d\mathcal{L}/dz_j^t$, the result is approximated using only the partial derivative:

$$\tilde{L}_j^t = \frac{d\mathcal{L}}{dz_j^t} \quad \longrightarrow \quad L_j^t := \frac{\partial\mathcal{L}}{\partial z_j^t}. \tag{3.12}$$

That is, the influence of z_j^t on future losses is ignored, and consideration is given to its direct, instantaneous contribution only. This leads to a purely local learning signal:

$$L_j^t = \frac{\partial\mathcal{L}}{\partial z_j^t}.$$

Under this approximation, the gradient of the total loss is estimated as:

$$\frac{d\mathcal{L}}{d\theta_{ji}} \approx \sum_{t=1}^{T} \left(\frac{\partial\mathcal{L}}{\partial z_j^t}\right) \cdot e_{ji}^t = \sum_{t=1}^{T} L_j^t \cdot e_{ji}^t \tag{3.13}$$

This is the core of the e-prop idea: the integration of two crucial components, namely eligibility traces and instantaneous learning signals, results in a fully online, memory-efficient, and neuron-local update mechanism.

- The eligibility trace e_{ji}^t is updated recursively in real time using only local Jacobians, specifically $\partial z_j^t/\partial\theta_{ji}$ and $\partial z_j^t/\partial z_j^{t-1}$, without requiring access to the full history of network states.
- The learning signal $L_j^t = \partial\mathcal{L}/\partial z_j^t$ is computed locally at each time step based solely on the current state z_j^t, avoiding the need to backpropagate information from future states.

3.1.2.1 Surrogate Gradients

A central challenge in applying gradient-based learning to spiking neural networks (SNNs) is the non-differentiability of the spiking function. Since the spike output z_j^t is a binary function of the membrane potential v_j^t, its derivative is undefined in the conventional sense. This prevents the direct use of gradient-based methods, including e-prop.

To address this, surrogate gradient methods introduce a smooth approximation to the non-differentiable step function [90]. Specifically, the *surrogate gradient* (called pseudo-derivative in Bellec et al. [16]) ψ_j^t serves as a replacement for $\partial z_j^t / \partial v_j^t$ during gradient computation. It captures the sensitivity of the spike output to changes in the membrane voltage and enables backpropagation of error signals through the spiking nonlinearity.

In the original formulation of e-prop [16], a piecewise-linear surrogate function is used:

$$\psi_j^t(v_j^t) = \gamma \max\left(0, 1 - \beta \left| v_j^t - v_{\text{th}}^t \right|\right), \tag{3.14}$$

where γ controls the overall scale of the surrogate gradient and β defines its sharpness around the firing threshold v_{th}^t. This function peaks at the threshold and decays linearly to zero on either side, giving in this way a non-zero gradient in the vicinity of spike emission.

3.1.2.2 Direct Feedback Alignment

Another key challenge in implementing biologically plausible learning is the *weight transport problem* [88, 89], which arises from the requirement in standard backpropagation to propagate gradients backward through the network using the exact transposes of the forward weights. This symmetry condition is not biologically realistic, as it implies precise alignment between forward and backward synaptic pathways.

Direct Feedback Alignment (DFA) [89] addresses this limitation by replacing the backward weights with fixed, random feedback connections. Instead of propagating error signals backward layer by layer, DFA directly transmits the output error to each hidden neuron through a random projection.

In the context of e-prop, error signals E_k^t, indexed by output neuron k, are projected back to the hidden layer using the forward output weights:

$$L_j^t = \sum_k \theta_{kj}^{\text{out}} E_k^t. \tag{3.15}$$

To avoid the weight transport problem, the concept of DFA replaces the trainable weights θ_{kj}^{out} with a fixed random feedback matrix B_{jk}, yielding the learning signal:

$$L_j^t = \sum_k B_{jk} E_k^t. \tag{3.16}$$

This modification aligns with the goals of the e-prop framework: the elimination of symmetric forward and backward connections enhances biological plausibility.

3.1.2.3 Network Architecture Overview

This study focuses on recurrent spiking neural networks (RSNNs), which are composed of three principal layers: an input layer, a recurrent hidden layer, and an output layer. The input layer receives external stimuli encoded as spike trains and projects them to the hidden layer. This hidden layer is central to the network's temporal processing capabilities. It consists of neurons with recurrent connections that allow the network to retain information over time, enabling it to detect temporal patterns, maintain short-term memory, and respond appropriately to dynamic inputs.

The recurrent layer's outputs are fed into a readout layer, which consists of leaky integrator units that convert spike trains into continuous-valued signals. This readout layer also receives target signals during training. The difference between the network's predictions and the targets produces local error signals. These are combined with fixed, randomly initialized feedback weights to generate the learning signals that guide synaptic updates in the recurrent layer (see Figure 3.1).

Neuron Models. The recurrent neurons follow leaky integrate-and-fire (LIF) dynamics, optionally enhanced with spike-frequency adaptation, forming adaptive LIF (ALIF) units. The membrane potential v_j^t of neuron j evolves according to:

$$v_j^t = \alpha v_j^{t-1} + \sum_{i \neq j} \theta_{ji}^{\text{rec}} z_i^{t-1} + \sum_i \theta_{ji}^{\text{in}} x_i^t - z_j^{t-1} v_{\text{th},j}^t, \tag{3.17}$$

where $\alpha = \exp(-\Delta t/\tau_{\text{m}})$ represents the membrane decay factor. A spike is emitted when the membrane potential exceeds the adaptive threshold:

$$z_j^t = H(v_j^t - v_{\text{th},j}^t), \tag{3.18}$$

and the threshold itself adapts dynamically via:

$$v_{\text{th},j}^t = v_{\text{th}} + \beta_{\text{a}} a_j^t, \qquad a_j^t = \rho a_j^{t-1} + z_j^{t-1}, \tag{3.19}$$

with $\rho = \exp(-\Delta t/\tau_{\text{a}})$ and β_{a} controlling the strength of adaptation. When $\beta_{\text{a}} = 0$, the neuron reduces to a standard LIF model (see subsection 2.5.2 for more details).

The output neurons integrate incoming spikes using leaky summation:

$$y_k^t = \kappa y_k^{t-1} + \sum_j \theta_{kj}^{\text{out}} z_j^t, \tag{3.20}$$

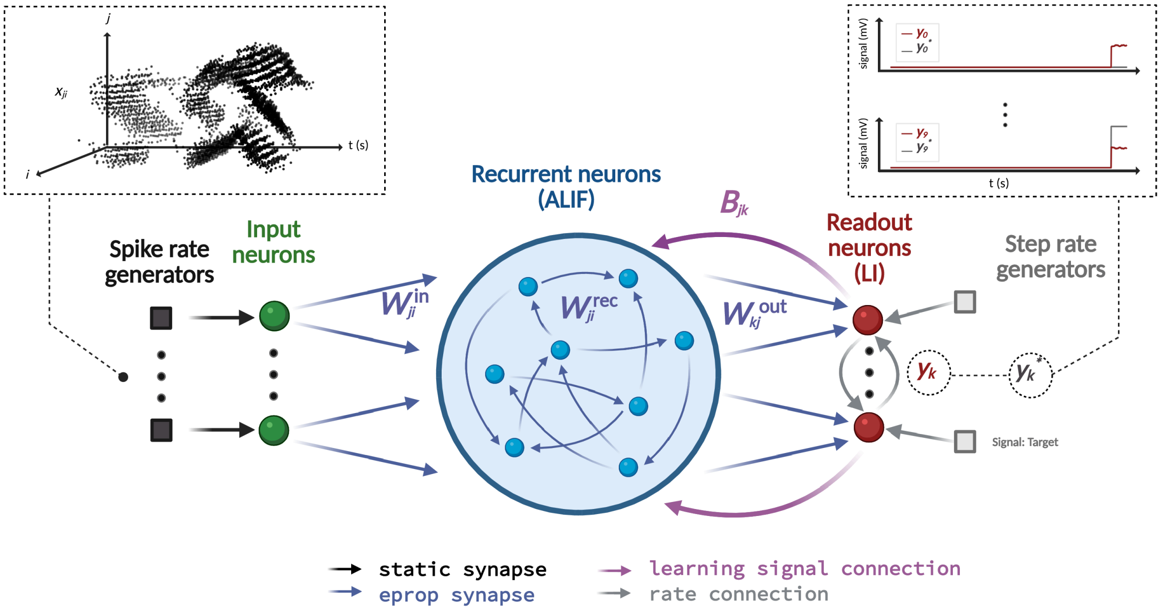


Figure 3.1: **Schematic of the e-prop training architecture.** The input layer encodes spike-based training data and projects to a recurrent layer of leaky integrate-and-fire (LIF) and adaptive leaky integrate-and-fire (ALIF) neurons. The recurrent layer connects to a readout layer of leaky integrators, which produce real-valued outputs. During training, target signals are compared with outputs to compute error signals, which are broadcast back to the recurrent neurons via fixed random feedback paths to form learning signals.

where $\kappa = \exp\left(-\Delta t/\tau_{\mathrm{m,out}}\right)$ defines the output decay. For a more comprehensive description, refer to subsection 2.5.3.

The error signal driving learning is defined as the derivative of the loss with respect to the output:

$$E_k^t = \frac{\partial \mathscr{L}}{\partial y_k^t}, \tag{3.21}$$

quantifying each output neuron's contribution to the total network error.

3.1.2.4 Loss Function

The loss function depends on the nature of the task (regression or classification) and determines the error signal used during learning.

Regression. For regression tasks, the network output y_k^t is trained to match a continuous target signal $y_k^{*,t}$. The loss is defined as the mean-squared error (MSE):

$$\mathcal{L} = \frac{1}{2}\sum_t\sum_k \left(y_k^t - y_k^{*,t}\right)^2. \tag{3.22}$$

The corresponding error signal at each output neuron is:

$$E_k^t = y_k^t - y_k^{*,t}. \tag{3.23}$$

Classification. In classification tasks, the output is interpreted as a probability distribution over classes via the softmax function:

$$\pi_k^t = \mathrm{softmax}\left(y_k^t\right) = \frac{\exp\left(y_k^t\right)}{\sum_{k'}\exp\left(y_{k'}^t\right)}. \tag{3.24}$$

The target label distribution $\pi_k^{*,t}$ is typically one-hot encoded. The associated cross-entropy loss is:

$$\mathcal{L} = -\sum_t\sum_k \pi_k^{*,t}\,\log\pi_k^t. \tag{3.25}$$

To compute the error signal, the chain rule is applied:

$$E_k^t = \frac{\partial\mathcal{L}}{\partial y_k^t} = \sum_{k'}\frac{\partial\mathcal{L}}{\partial\pi_{k'}^t}\frac{\partial\pi_{k'}^t}{\partial y_k^t}. \tag{3.26}$$

Using the identities:

$$\frac{\partial\mathcal{L}}{\partial\pi_k^t} = -\frac{\pi_k^{*,t}}{\pi_k^t}, \tag{3.27}$$

$$\frac{\partial\pi_k^t}{\partial y_{k'}^t} = \begin{cases}\pi_k^t(1-\pi_k^t), & \text{if } k=k',\\ -\pi_k^t\pi_{k'}^t, & \text{if } k\neq k',\end{cases} \tag{3.28}$$

the following simplified form is obtained:

$$E_k^t = \pi_k^t - \pi_k^{*,t}. \tag{3.29}$$

3.1.2.5 Gradient Computation

Given the definition of the output layer dynamics in Equation (3.20), the learning signal L_j^t for the hidden layer can be computed as:

$$L_j^t = \frac{\partial\mathcal{L}}{\partial z_j^t} = \sum_k\theta_{kj}^{\mathrm{out}}\sum_{t'\geq t}E_k^{t'}\,\kappa^{t'-t}, \tag{3.30}$$

where $E_k^t = \partial\mathcal{L}/\partial y_k^t$ is the output error signal and κ is the decay factor for the output neuron.

Recurrrent synapses. For recurrent synapses between hidden ALIF neurons, the fist step is to find the eligibility trace expression. The eligibility trace e_{ji}^t is defined as:

$$e_{ji}^t = \frac{\partial z_j^t}{\partial \boldsymbol{h}_j^t} \cdot \boldsymbol{\epsilon}_{ji}^t, \tag{3.31}$$

where the first factor captures the postsynaptic influence and is approximated using the surrogate gradient (cf. Equation (3.14)):

$$\frac{\partial z_j^t}{\partial \boldsymbol{h}_j^t} = \begin{pmatrix} \frac{\partial z_j^t}{\partial v_j^t} & \frac{\partial z_j^t}{\partial a_j^t} \end{pmatrix} \approx \begin{pmatrix} \psi_j^t & -\beta_{\mathrm{a}} \psi_j^t \end{pmatrix}. \tag{3.32}$$

The second factor, $\boldsymbol{\epsilon}_{ji}^t$, is the eligibility vector, which encodes presynaptic contributions and is derived from the gradient of the internal state variables:

$$\boldsymbol{h}_j^t = \begin{pmatrix} v_j^t \\ a_j^t \end{pmatrix}. \tag{3.33}$$

It evolves recursively according to:

$$\boldsymbol{\epsilon}_{ji}^t = \frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}} \cdot \boldsymbol{\epsilon}_{ji}^{t-1} + \frac{\partial \boldsymbol{h}_j^t}{\partial \theta_{ji}}, \tag{3.34}$$

with gradients given by:

$$\frac{\partial \boldsymbol{h}_j^t}{\partial \boldsymbol{h}_j^{t-1}} = \begin{pmatrix} \frac{\partial v_j^t}{\partial v_j^{t-1}} & \frac{\partial v_j^t}{\partial a_j^{t-1}} \\ \frac{\partial a_j^t}{\partial v_j^{t-1}} & \frac{\partial a_j^t}{\partial a_j^{t-1}} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ \psi_j^{t-1} & \rho - \beta_{\mathrm{a}} \psi_j^{t-1} \end{pmatrix}, \tag{3.35}$$

$$\frac{\partial \boldsymbol{h}_j^t}{\partial \theta_{ji}} = \begin{pmatrix} \frac{\partial v_j^t}{\partial \theta_{ji}} \\ \frac{\partial a_j^t}{\partial \theta_{ji}} \end{pmatrix} = \begin{pmatrix} z_i^{t-1} \\ 0 \end{pmatrix}. \tag{3.36}$$

Substituting into Equation (3.34), the recursive update becomes:

$$\boldsymbol{\epsilon}_{ji}^t = \begin{pmatrix} \mathscr{F}_\alpha\left(z_i^{t-1}\right) \\ \psi_j^{t-1} \epsilon_{ji,v}^{t-1} + (\rho - \beta_{\mathrm{a}} \psi_j^{t-1}) \epsilon_{ji,a}^{t-1} \end{pmatrix}, \tag{3.37}$$

where the low-pass filter is defined as:

$$\mathscr{F}_\alpha\left(z_i^t\right) := \sum_{t' \le t} \alpha^{t-t'} z_i^{t'} = \alpha \mathscr{F}_\alpha\left(z_i^{t-1}\right) + z_i^t. \tag{3.38}$$

Using this formulation, the total gradient with respect to the synaptic weight θ_{ji} can be approximated as:

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\theta_{ji}} \approx \sum_t \sum_k B_{jk} \sum_{t' \geq t} E_k^t \kappa^{t'-t} \psi_j^t \left(\mathscr{F}_\alpha\left(z_i^{t-1}\right) - \beta_{\mathrm{a}} \epsilon_{ji,a}^t\right), \tag{3.39}$$

To express this in a forward view, the summation indices can be exchanged:

$$\sum_t \sum_{t' \geq t} \to \sum_{t'} \sum_{t \leq t'}, \tag{3.40}$$

which yields:

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\theta_{ji}} \approx \sum_k B_{jk} \sum_{t'} E_k^{t'} \sum_{t \leq t'} \kappa^{t'-t} \psi_j^t \left(\mathscr{F}_\alpha\left(z_i^{t-1}\right) - \beta_{\mathrm{a}} \epsilon_{ji,a}^t\right). \tag{3.41}$$

Applying the definition of exponential filtering (see Equation (3.38)), the expression simplifies to:

$$\frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\theta_{ji}} \approx \sum_t \underbrace{\sum_k B_{kj} E_k^t}_{:=L_j^t} \cdot \mathscr{F}_\kappa \left(\underbrace{\psi_j^t \left(\mathscr{F}_\alpha\left(z_i^{t-1}\right) - \beta_{\mathrm{a}} \epsilon_{ji,a}^t\right)}_{:=e_{ji}^t} \right) = \sum_t L_j^t \cdot \mathscr{F}_\kappa\left(e_{ji}^t\right), \tag{3.42}$$

where L_j^t is the learning signal associated with neuron j, and e_{ji}^t is the eligibility trace of synapse $(i \to j)$. This expression highlights the three-factor structure of the update rule.

In some cases, the adaptation term can be neglected (e.g., when applied to postsynaptic LIF neurons), yielding the simplified recurrent contribution:

$$g_{ji}^{\mathrm{rec}} = \sum_t L_j^t \cdot \mathscr{F}_\kappa\left(\psi_j^t \cdot \mathscr{F}_\alpha\left(z_i^{t-1}\right)\right). \tag{3.43}$$

Input to recurrent synapses. For input synapses, the gradient expression mirrors that of recurrent synapses, but with the filtered input activity x_i^t replacing the presynaptic spike train z_i^{t-1}:

$$g_{ji}^{\mathrm{in}} = \sum_t L_j^t \cdot \mathscr{F}_\kappa\left(\psi_j^t \cdot \mathscr{F}_\alpha\left(x_i^t\right)\right). \tag{3.44}$$

Recurrence to output synapses. For output synapses projecting to leaky integrator neurons, the gradient is computed similarly, but without the surrogate gradient since these neurons do not spike:

$$g_{kj}^{\mathrm{out}} = \sum_t E_k^t \cdot \mathscr{F}_\kappa\left(z_j^t\right) \tag{3.45}$$

where $E_k^t = \partial\mathscr{L}/\partial y_k^t$ is the derivative of the loss with respect to the output neuron's membrane potential.

3.1.2.6 Firing Rate Regularization

Firing rate regularization penalizes deviations of a neuron's firing rate from a specified target, promoting stable and biologically realistic activity levels in recurrent spiking networks [16]. The regularization loss is defined as:

$$\mathscr{L}_{\text{reg}} = c_{\text{reg}} \cdot \frac{1}{2} \sum_j \left(\bar{f}_j - f^{\text{target}}\right)^2, \tag{3.46}$$

where $\bar{f}_j = \frac{1}{T} \sum_t z_j^t$ is the average firing rate of neuron j over an interval of T time steps, and f^{target} is the desired firing rate. The coefficient c_{reg} controls the strength of the regularization.

This mechanism encourages each neuron to maintain a firing rate close to f^{target}, despite ongoing weight updates. The gradient contribution from the regularization term is computed as:

$$g_{ji}^{\text{reg}} = c_{\text{reg}} \sum_t \frac{1}{T n_{\text{trial}}} \left(f^{\text{target}} - \bar{f}_j\right) e_{ji}^t, \tag{3.47}$$

$$\bar{f}_j = \frac{1}{T n_{\text{trial}}} \sum_t z_j^t, \tag{3.48}$$

where n_{trial} denotes the number of training trials over which the regularization is averaged.

This term adjusts weights based on how each synapse contributes to the deviation of the postsynaptic neuron's firing rate from the target. Specifically, if $\bar{f}_j > f^{\text{target}}$, the update is negative, reducing the weight; if $\bar{f}_j < f^{\text{target}}$, the update is positive, increasing the weight. The deviation is neuron-specific, but the update at each synapse is modulated by its eligibility trace e_{ji}^t.

Biological motivation. Firing rate regularization is motivated by homeostatic plasticity mechanisms observed in biological neurons, which maintain stable levels of excitability over time despite ongoing synaptic modifications [109, 16]. In vivo, neurons adjust their intrinsic excitability and synaptic strengths to prevent hypoactivity or hyperactivity, ensuring robust network function and avoiding pathological states such as epileptic bursting or neural silence [110].

3.1.2.7 Weight Updates

Weight updates in e-prop are performed using the computed gradients g_{ji} for each synapse type. These updates can be applied either online, that is, after processing a single input sequence, or at the end of a mini-batch of sequences. Both standard and regularization gradients contribute to the total update.

The synapse-type-specific gradients are defined as:

$$g_{ji} = \begin{cases} \sum_t L_j^t \cdot \mathscr{F}_\kappa\Big(\psi_j^t \cdot \mathscr{F}_\alpha\big(z_i^{t-1}\big)\Big), & \text{if } (i \rightarrow j) \text{ is recurrent} \\ \sum_t L_j^t \cdot \mathscr{F}_\kappa\Big(\psi_j^t \cdot \mathscr{F}_\alpha\big(x_i^t\big)\Big), & \text{if } (i \rightarrow j) \text{ is input to recurrent} \\ \sum_t E_k^t \cdot \mathscr{F}_\kappa\Big(z_j^t\Big), & \text{if } (i \rightarrow j) \text{ is recurrent to output} \end{cases} \tag{3.49}$$

The firing rate regularization gradient is applied only to synapses projecting to recurrent neurons:

$$g_{ji}^{\text{reg}} = \begin{cases} 0, & \text{if } (i \rightarrow j) \text{ is recurrent to output} \\ c_{\text{reg}} \sum_t \frac{1}{T n_{\text{trial}}} \left(\bar{f}_j - f^{\text{target}}\right) e_{ji}^t, & \text{otherwise} \end{cases} \tag{3.50}$$

The total e-prop gradient is then given by the sum of the standard and regularization terms:

$$g_{ji}^{\text{e-prop}} = g_{ji} + g_{ji}^{\text{reg}}. \tag{3.51}$$

For gradient descent [78], the weight update rule is:

$$\theta_{ji} \leftarrow \theta_{ji} - \eta \cdot g_{ji}^{\text{e-prop}}, \tag{3.52}$$

where η is the learning rate. More sophisticated optimization algorithms such as Adam [111] can also be used to adaptively adjust learning rates based on the moment statistics of past gradients.

3.2 Time-Driven Implementation of e-prop

The original e-prop algorithm [16] is implemented in a time-driven fashion, where all neuronal and synaptic updates are computed at every discrete time step. This update scheme aligns well with dense tensor operations and is naturally suited to machine learning frameworks such as TensorFlow [112] and PyTorch [113], which are optimized for synchronous, batched computation.

In the time-driven implementation, synaptic gradients for each recurrent connection $(i \rightarrow j)$ are computed by iterating sequentially over the simulation time horizon T, for each training sample. At the beginning of the iteration, gradient accumulators are initialized, including one for the primary e-prop learning rule and one for firing-rate regularization:

$$g_{ji} \leftarrow 0, \qquad g_{ji}^{\text{reg}} \leftarrow 0.$$

At each time step t, an eligibility trace e_{ji}^t is computed based on presynaptic activity and the postsynaptic voltage derivative:

$$e_{ji}^t = \psi_j^t \cdot \mathscr{F}_\alpha(z_i^{t-1}),$$

where ψ_j^t is the local derivative of the postsynaptic membrane potential with respect to its input, and $\mathscr{F}_\alpha$ denotes an exponential low-pass filter with time constant α. Simultaneously, a learning signal L_j^t is computed for each postsynaptic neuron as a weighted sum of output errors:

$$L_j^t = \sum_k B_{jk} E_k^t,$$

where E_k^t is the gradient of the loss with respect to the output potential of neuron k, and B_{jk} are fixed, random feedback weights projecting from output to hidden units. The gradient contribution for the synapse is then accumulated as:

$$g_{ji} += L_j^t \cdot \mathscr{F}_\kappa(e_{ji}^t),$$

where $\mathscr{F}_\kappa$ is another low-pass filter applied to the eligibility trace with time constant κ.
A separate gradient term accounts for firing rate regularization, encouraging neurons to maintain a target firing rate f^{target}. This term is updated as:

$$g_{ji}^{\text{reg}} += c_{\text{reg}} \cdot \frac{1}{T n_{\text{trial}}} \cdot \left(f^{\text{target}} - \bar{f}_j\right) \cdot e_{ji}^t,$$

where $\bar{f}_j$ is the average firing rate of neuron j, and c_{reg} controls the strength of the regularization. After completing all time steps, the two gradient contributions are combined:

$$g_{ji}^{\text{e-prop}} = g_{ji} + g_{ji}^{\text{reg}},$$

and used to update the synaptic weight according to the chosen optimization rule:

$$\theta_{ji}^{\text{rec}} \leftarrow \theta_{ji}^{\text{rec}} + f(g_{ji}^{\text{e-prop}}),$$

where $f(\cdot)$ represents a gradient descent update or other optimizer.

Although this procedure is described at the level of a single synapse for clarity, practical implementations apply these computations in parallel across all neurons and synapses using efficient tensor operations. The original implementation by Bellec et al. [16] employs TensorFlow.

3.3 Event-driven Implementation of e-prop

In this section we present the methodology to implement e-prop's synaptic updates in an event-driven manner, in such a way that the results of the original time-driven e-prop algorithm are preserved. We demostrate this by reproducing the results of the original e-prop paper [16].

3.3.1 The Case for Event-Driven Synaptic Updates

In artificial neural networks (ANNs), all neurons and synapses are updated synchronously at every computational step. In contrast, spiking neural networks (SNNs) exhibit sparse, event-driven dynamics that demand fundamentally different computational strategies [114, 115].

Spikes in SNNs carry no magnitude and encode information through their timing [50]. Unlike ANNs, where every connection is used at every time step, most synapses in SNNs remain inactive for extended periods [114]. While neurons integrate continuous inputs over time and may spike frequently, any individual synapse is activated only occasionally. This temporal sparsity is accompanied by spatial sparsity: even within densely connected brain regions, the probability that two neurons are connected via a specific synapse is low [34]. Nevertheless, neurons still receive input from thousands of synapses, ensuring continuous input at the neuronal level even if individual synaptic events are rare [34].

These properties motivate separating the simulation of neurons and synapses. Many efficient SNN simulation frameworks adopt a *hybrid scheme*: neuron dynamics are updated at fixed time steps (time-driven), while spike transmission and synaptic updates are handled asynchronously (event-driven) [116]. In this model, when a spike (or other punctual event) reaches a postsynaptic neuron, it is multiplied by the synaptic weight and applied instantly, without requiring intervention from the presynaptic neuron [116]. This approach enables scalable simulation of large, sparse networks and has been refined over decades of research [116, 60, 117], proving especially effective for parallel and distributed computing architectures [118, 119].

3.3.2 Main Challenge for Event-Driven e-prop: Instantaneous Transmissions

In the original derivation of the e-prop learning rule [16], recurrent connections are modeled with a fixed delay of one time step, while input connections are assumed to be delay-

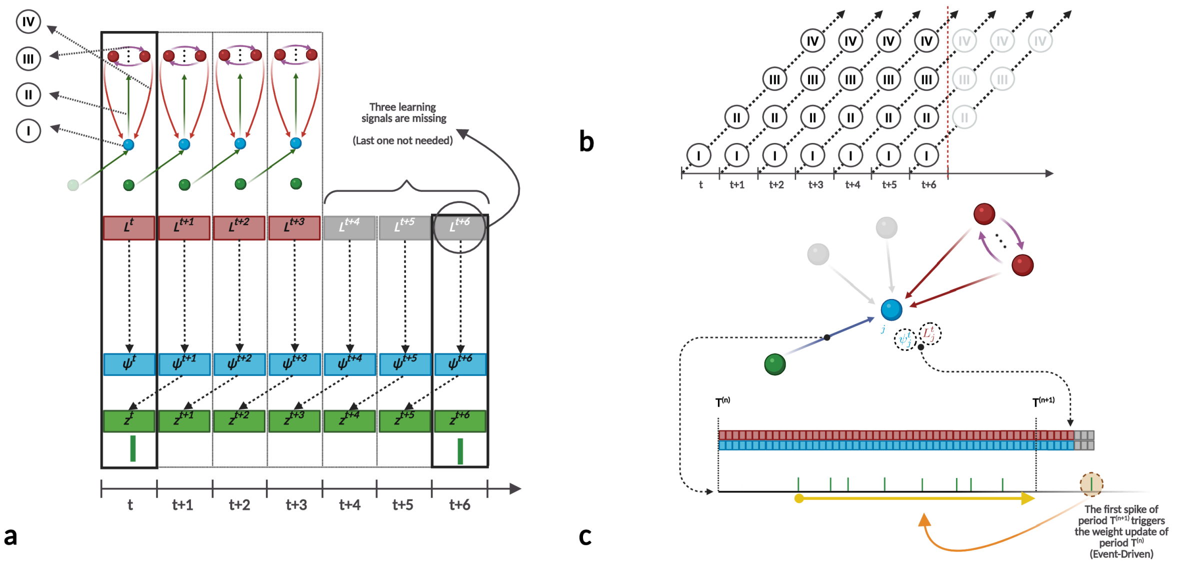


Figure 3.2: **Pipeline execution in event-driven e-prop Learning.** (a) Illustrates forward and backward passes in a neural simulation between time steps t and $t+6$, corresponding to events such as two spikes stimulating a synapse. The sequence involves (I) updating the recurrent layer, (II) transmitting activity to the readout layer and updating it, (III) normalizing the outputs, and (IV) transmitting the learning signal back. Vertical arrows in the backward pass indicate instantaneous dependencies. Continuous integration of the learning rule is infeasible between two points using the method of timeline shifting + pipelining. At the upper limit, in the figure at $t+6$, there is a shortfall of three instantaneous gradients due to the timeline shift, the normalization step, and the minimum resolution time step required for signal travel. The final missing signal can be omitted by adopting a semi-open integration range: open at the start, closed at the end. (b) Demonstrates the pipelining of operations (I) through (IV). In a pipeline, the number of incomplete operations (shown in grey) at a boundary (depicted by a red dashed line) increases with pipeline depth. Here, three learning signals, and therefore three gradient contributions (step IV), are missing, corresponding to the pipeline's depth minus one. (c) Following the archiving framework [107], time-driven neurons archive the histories of quantities required for weight updates, including surrogate gradients and learning signals. At the first spike after each interval T, corresponding to the sample length, the algorithm calculates and accumulates a gradient.

free. Although the supplemental material generalizes the framework to arbitrary input and recurrent delays, all other connections, namely those from recurrent to output neurons, output to recurrent neurons, and within the output layer, are modeled as *instantaneous* in the implementation and experiments. This modeling choice is reflected in the update equation for the output layer:

$$y^t = h\left(y^{t-1}, z^t\right), \tag{3.53}$$

where the output state $\boldsymbol{y}^t$ is computed from the output neuron's previous state and the current recurrent spike vector $\boldsymbol{z}^t$, both indexed at the same time t, indicating zero transmission delay. A similar assumption appears in the softmax function (Equation (3.24)), where numerator and denominator terms share the same time index, implying instantaneous interactions between output neurons.

Moreover, the learning signal L_j^t in the recurrent weight update rule (Equation (3.43)) is indexed at the same time as the surrogate gradient ψ_j^t, further indicating that feedback from the output layer is assumed to reach the recurrent neurons without delay (see Figure 3.2a, backward arrows).

In a hybrid simulation scheme, neuron dynamics are updated in a time-driven manner, while synaptic transmissions are processed event-by-event. This separation introduces a constraint: event-driven mechanisms cannot natively support truly instantaneous transmissions, as they require a discrete propagation delay between source and target. Therefore, interactions assumed to be instantaneous in the original e-prop formulation, such as those between recurrent and output neurons, or within the output layer, must be distributed across multiple time steps in an event-driven framework.

This discrepancy presents a core challenge when reproducing the original e-prop results under an event-driven implementation of synapses. Instantaneous dependencies, including the implied by matching time indices in L_j^t and ψ_j^t, cannot be directly realized without violating the event-driven causality model. Thus, to maintain correctness and reproducibility, these transmissions must be reformulated: they can be conceptualized as a *pipeline* of sequential operations, so that the intended interactions are preserved despite the inherent temporal structure of the event-based framework.

3.3.3 Pipelining

Following the archiving framework of Stapmanns et al. [107], the event-driven e-prop implementation stores short histories of postsynaptic variables: surrogate gradients, learning signals, firing-rate terms, and error signals. When a presynaptic spike arrives, the update uses only the relevant slice of this archive, bounded by the previous update time and the present time and corrected for dendritic delay.

Instantaneous gradient contributions are decomposed into four serial operations: (I) update recurrent neurons, (II) transmit spikes to the readout layer, (III) normalise outputs (soft-max), and (IV) broadcast learning signals (Figure 3.2b). A time-driven simulator executes all four in the same step; an event-driven synapse cannot. We therefore *pipeline* the chain across four successive steps: at any moment each step finalises one stage of four different gradients. Every time step still delivers a complete gradient, but with a constant latency of three steps.

Near the end of an update interval this pipeline is incomplete, so some gradient pieces are still missing. To avoid dropping them, we read the archived postsynaptic history at the *first* spike after the interval length T (Figure 3.2c). If that spike occurs at least three steps into the next interval, all pending contributions are available; otherwise a negligible truncation error is introduced.

The trade-off for correctness is memory: each synapse must buffer a small window of past activity. Although this increases the footprint in large networks, it is required for reproducing the results of Bellec et al. [16]. More aggressive update schedules (e.g., updating on every spike) would accumulate unrecoverable errors because incomplete pipeline stages would be discarded.

3.3.4 Design and Implementation

To implement event-driven e-prop, we propose neuron models that support the storage and retrieval of e-prop signals, connection models capable of delivering learning signals, and synapse models that manage the plasticity process. This approach benefits from the modular design of modern spiking neural network simulators, which segregate neuron and synapse models. Such separation ensures that new neuron models remain compatible with existing synapse models, thereby enhancing integration and extensibility [120]. This scheme also benefits from hybrid parallelization: OpenMP threads handle intra-node parallelization, while the Message Passing Interface (MPI) manages inter-node communication. In parallel simulations, synapses reside on the same MPI process as their postsynaptic neurons [116]. As a result, no inter-process communication is required to retrieve postsynaptically available information for weight updates, and each source neuron communicates only a single spike for all its target neurons within the same MPI process, improving computational efficiency. Our approach can be readily integrated into established simulators like NEST [121] by subclassing existing parent node and connection classes and incorporating the necessary methods for managing e-prop signals and plasticity computation. Additionally, we provide a reference implementation within the NEST simulator to validate the feasibility of our approach [108].

A key precedent for our work is the `ArchivingNode` class [116]. This class serves as a parent for neuron models supporting spike-timing-dependent plasticity (STDP) in event-driven simulations by providing member functions to store and retrieve postsynaptic spike histories. Incoming synapses can then access this archived information to perform event-driven weight updates. Building on this concept, the challenge of implementing voltage-based plasticity rules (where synaptic weight changes depend continuously on postsynaptic membrane potentials) was tackled previously within an event-driven framework [107]. They implemented the voltage based plasticity rules by Clopath et. al. [122], and Urbanczik et. al. [123], introducing the `ClopathArchivingNode` and `UrbanczikArchivingNode`, respectively. Their approach extends the parent classes with a vector-based archive that stores membrane potential-derived information, accessible to incoming plastic synapses. When an incoming spike occurs, synapses use this stored data to trigger plasticity computations, enabling an event-driven implementation of these rules.

Following this line of work, we introduce the `EpropArchivingNode`, a new class that maintains a vector-based archive of e-prop signals. As recurrent and output neurons should archive different types of data to allow plasticity in synapses targeting them, we further define two specialized subclasses: `EpropArchivingNodeReadout` for readout neurons and `EpropArchivingNodeRecurrent` for recurrent neurons, with the idea of efficiently manage different signal types. These classes implement the necessary methods to store and retrieve surrogate gradients, learning signals, and error signals. The `EpropArchivingNode` class provides a common interface for managing e-prop signals, while the specialized subclasses ensure that the correct data is archived and retrieved for each neuron type.

Neuron models that support e-prop inherit from the corresponding subclass and employ its methods to record new data. We append the suffix `_bsshslm_2020` to the neuron and synapse models that implement the e-prop framework and that are able to reproduce the results in Bellec et al. [16]. This suffix is derived from the initials of the authors of the original publication [16] and the year of the publication. In `_bsshslm_2020` models, synaptic updates occur only after processing an entire training sample, with the `update_period` set to the sample length. Update times are defined as

$$\texttt{t\_update} = \texttt{shift} + \texttt{update\_period} \cdot i,$$

where $i = 0, 1, \ldots$. The `shift` parameter aligns weight updates to reproduce the time-driven results by Bellec et al. [16]. Specifically, the `shift` value represents the minimum number of time steps required for an input spike to reach the recurrent layer (for recurrent neurons) or the output layer (for output neurons). A difference of 1 in the `shift` value between recurrent and output neurons accounts for an extra time step needed for spikes to reach the readout layer. In the time-driven implementation, the output layer integrates recurrent spikes without delay; thus, the additional shift for output neurons ensures alignment with those results.

Recurrent neurons call a protected method,

append_new_eprop_history_entry(),

at each timestep to insert an empty history entry. This entry is then populated with surrogate gradients and learning signals using the methods

write_surrogate_gradient_to_history()

and

write_learning_signal_to_history(),

respectively. Readout neurons follow a similar procedure to record their error signals. Since the exact plasticity rule depends on the dynamics of the postsynaptic neuron, we implement the gradient computation as a member function of each neuron model named

compute_gradient().

This function is invoked by incoming synapses when an incoming spike triggers a weight update, with the synapse passing the spike times since the last update.

The e-prop-enabled synapse models can be simply derived from a standard **Connection** class following the same naming convention (adding the suffix **bsshslm_2020** to the base version name).

3.3.5 Cleaning of e-prop History

To optimize memory usage and prevent the unnecessary growth of **eprop_history** for each neuron, a cleaning mechanism is implemented to remove entries that have already been utilized by the weight updates of all incoming synapses. This mechanism is based on the introduction of a second vector-based archive, **update_history**, which is maintained locally for each postsynaptic neuron. The **update_history** is a vector of pairs

$$\texttt{update\_history} = \{(\texttt{t\_update}, \quad \texttt{access\_counter})\},$$

where **t_update** indicates the time point at which a particular synapse's weight is updated, and **access_counter** records the number of synapses whose most recent update occurred at that time. When an incoming synapse triggers a weight update at **t_update** $= t_1$ and had a previous update at **t_update** $= t_0$, the **access_counter** for t_0 is decreased by 1, and if it reaches 0, the entry corresponding to t_0 is removed from **update_history**. Simultaneously, if an entry for t_1 already exists, its **access_counter** is incremented by 1; if not, a new entry is created with an `access_counter` set to 1. Consequently, the growth of `update_history` is inherently bounded by the number of synapses targeting the neuron. This auxiliary history, which monitors the progress of weight updates for all incoming synapses, is then used to identify sections of the `eprop_history` that are no longer required.

In the `bsshslm_2020` version, each e-prop neuron calls the function

```
erase_used_eprop_history()
```

at the beginning of every update period. This function removes both outdated and redundant `eprop_history` entries. Outdated entries are eliminated by deleting all history records that precede the `t_update` entry at the front of the `update_history`, ensuring that `eprop_history` entries are retained until they have been used by all incoming synapses. However, when neurons do not emit spikes over successive update periods or remain silent for extended durations, the corresponding gradients for their outgoing connections remain zero, making the computation and retention of these history entries unnecessary. To mitigate this, all `eprop_history` entries corresponding to update periods that lack a matching `update_history` entry (i.e., those with an `access_counter` of zero) are safely removed.

3.4 Event-Driven e-prop with Additional Biological Features

This section presents biologically inspired refinements to the event-driven e-prop model by removing several components inherited from conventional machine learning frameworks. Most notably, we eliminate *instantaneous transmissions* by rederiving the learning rule to respect causal, delay-based signal propagation. In addition, we replace fixed-length update intervals and global resets with continuous, locally driven updates that more accurately reflect biological learning dynamics, among other changes. These modifications allow asynchronous event-driven synaptic computations and **remove the need for pipelining**

3.4.1 Generalized Delays in e-prop

In the original model [16], certain signal transmissions are assumed to be instantaneous, while the remaining delays are coupled to the simulation's temporal resolution, as discussed in section 3.3. Here, we introduce a more explicit and biologically motivated approach: we decouple these delays from the simulation's resolution and include explicit transmission times for all connections that were previously considered instantaneous.

In particular, the original model does not incorporate a delay between the occurrence of synaptic activities and the arrival of the learning signals that convey feedback about the network's output error. Accounting for this delay (specifically, the time it takes for signals to travel from the recurrent layer to the output layer and back as error-related feedback) is essential for addressing what is known as the **distal reward problem** [124]. Although this problem is usually framed in the context of reinforcement learning, it is also relevant to supervised learning scenarios. In these contexts, any feedback mechanism that modulates synaptic updates might reasonably be expected to exhibit its own temporal lag.

Such delays may arise from several biological considerations. For instance, the propagation of learning signals may involve slower neuromodulatory processes, chemical intermediaries, or other non-instantaneous mechanisms that unfold over longer timescales [125]. Additionally, neural circuits are spatially distributed [126], and the physical distance between neural populations can introduce further signal travel time. By systematically incorporating these realistic transmission delays, we seek to more faithfully model the temporal structure of learning signals and thus improve the biological plausibility of the network's learning dynamics.

3.4.1.1 Delay from the recurrent to the output layer

To incorporate a delay d from the recurrent to the output layer, we modify the update rule for the output layer as shown in Equation 3.53:

$$\boldsymbol{y}^t = h\Big(\boldsymbol{y}^{t-1}, \boldsymbol{z}^{t-d}\Big). \tag{3.54}$$

The loss function for each training sample, denoted as $\mathscr{L}\big(\boldsymbol{y}^0, \ldots, \boldsymbol{y}^T\big)$, is computed where $\boldsymbol{y}^t$ represents the network output at time t, and T is the number of time steps in a training sample. The network activity at time t influences the loss $\mathscr{L}$ at times $t + d$ or later.

To mathematically model this delay, we modify the equations for temporal credit assignment. The influence of z_j^t on $\mathscr{L}$, with $\partial y_k^\tau / \partial z_j^t = 0$ for any $\tau < t + d$ is given by:

$$\frac{d\mathscr{L}}{dz_j^{t'}} = \sum_k \left(\frac{\partial \mathscr{L}}{\partial y^{t'+d}} + \frac{d\mathscr{L}}{dy_k^{t'+d+1}} \frac{\partial y_k^{t'+d+1}}{\partial y_k^{t'+d}} \right) \frac{\partial y_k^{t'+d}}{\partial z_j^{t'}}, \tag{3.55}$$

Unrolling the recursion

$$
\begin{aligned}
\frac{d\mathscr{L}}{dz_j^{t'}} &= \sum_k \left(\frac{\partial \mathscr{L}}{\partial y^{t'+d}} + \frac{d\mathscr{L}}{dy_k^{t'+d+1}} \frac{\partial y_k^{t'+d+1}}{\partial y_k^{t'+d}} \right) \frac{\partial y_k^{t'+d}}{\partial z_j^t} \\
&= \sum_k \left(\frac{\partial \mathscr{L}}{\partial y^{t'+d}} + \left(\frac{\partial \mathscr{L}}{\partial y^{t'+d+1}} + (\dots) \frac{\partial y_k^{t'+d+2}}{\partial y_k^{t'+d+1}} \right) \frac{\partial y_k^{t'+d+1}}{\partial y_k^{t'+d}} \right) \frac{\partial y_k^{t'+d}}{\partial z_j^t} \\
&= \sum_k \sum_{t \geq t'+d}^{T} \frac{\partial \mathscr{L}}{\partial y_k^t} \frac{\partial y_k^t}{\partial y_k^{t-1}} \cdots \frac{\partial y_k^{t'+d+1}}{\partial y_k^{t'+d}} \frac{\partial y_k^{t'+d}}{\partial z_j^t} \\
&= \sum_k \sum_{t \geq t'+d}^{T} \frac{\partial \mathscr{L}}{\partial y_k^t} \left(\prod_{\tau=t'+d}^{t-1} \frac{\partial y_k^{\tau+1}}{\partial y_k^\tau} \right) \frac{\partial y_k^{t'+d}}{\partial z_j^t}.
\end{aligned}
\tag{3.56}
$$

Next, we compute the gradient of the loss function with respect to the recurrent weights:

$$
\frac{d\mathscr{L}}{d\theta_{ji}^{\mathrm{rec}}} \approx \sum_{t=1}^{T} \frac{d\mathscr{L}}{dz_j^t} e_{ji}^t.
\tag{3.57}
$$

Substituting Equation 3.7 and Equation 3.56 into Equation 3.57 we obtain:

$$
\frac{d\mathscr{L}}{d\theta_{ji}^{\mathrm{rec}}} \approx \sum_k \sum_{t'=1}^{T-d_{\mathrm{out}}} \sum_{t \geq t'+d_{\mathrm{out}}}^{T} \frac{\partial \mathscr{L}}{\partial y_k^t} \left(\prod_{\tau=t'+d_{\mathrm{out}}}^{t-1} \frac{\partial y_k^{\tau+1}}{\partial y_k^\tau} \right) \frac{\partial y_k^{t'+d_{\mathrm{out}}}}{\partial z_j^{t'}} \frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} \cdot \epsilon_{ji}^{t'},
\tag{3.58}
$$

Similarly, for the output weights, we derive:

$$
\begin{aligned}
\frac{d\mathscr{L}}{dW_{kj}^{\mathrm{out}}} &= \sum_{t=1}^{T} \frac{\partial \mathscr{L}}{\partial y_k^t} \frac{dy_k^t}{dW_k^{\mathrm{out}}} \\
&= \sum_{t'=1}^{T} \sum_{t \geq t'}^{T} \frac{\partial \mathscr{L}}{\partial y_k^t} \frac{\partial y_k^t}{\partial y_k^{t-1}} \cdots \frac{\partial y_k^{t'+1}}{\partial y_k^{t'}} \frac{\partial y_k^{t'}}{\partial W_{kj}^{\mathrm{out}}} \\
&= \sum_{t'=1}^{T} \sum_{t \geq t'}^{T} \frac{\partial \mathscr{L}}{\partial y_k^t} \left(\prod_{\tau=t'}^{t-1} \frac{\partial y_k^{\tau+1}}{\partial y_k^\tau} \right) \frac{\partial y_k^{t'}}{\partial W_{kj}^{\mathrm{out}}}.
\end{aligned}
\tag{3.59}
$$

In order to maintain a causal temporal directionality in gradient computations, we invert the indices t' and t. To implement this index inversion, consider a generic matrix $A \in \mathbb{R}^{T \times T}$, where each element $A_{t',t}$ represents the influence of an event at time t' on a subsequent event at time t. Under the principle of causality, effects can only propagate forward in time, which constrains the matrix to an upper triangular form, including the diagonal. Entries below the diagonal ($t' > t$) would imply backward-in-time influences and thus violate causality.

Now consider the total contribution of all causal effects represented by this matrix, expressed as a double sum:

$$
\sum_{t'=1}^{T} \sum_{t=t'}^{T} A_{t',t}.
$$

This sum iterates first over all source times t', and then over all future (or equal) times $t \geq t'$, consistent with causal ordering. Since the matrix is triangular, we can equivalently change the order of summation, i.e., first summing over t, then over all $t' \leq t$:

$$\left.\overbrace{\begin{matrix} A_{1,1} & A_{1,2} & \cdots & A_{1,T} \\ & A_{2,2} & \cdots & A_{2,T} \\ & & \ddots & \vdots \\ & & & A_{T,T} \end{matrix}}^{\sum_{t'=1}^{T}\sum_{t\geq t'}^{T} A_{t',t}}\right\} \sum_{t=1}^{T}\sum_{t'\leq t} A_{t',t} \tag{3.60}$$

and since both summations yield the same result, we arrive at the identity:

$$\sum_{t'=1}^{T}\sum_{t\geq t'}^{T} \longleftrightarrow \sum_{t=1}^{T}\sum_{t'\leq t} . \tag{3.61}$$

The causality matrix for a scenario where the causal effect of one event on another occurs at least d time steps later is zero in the first d sub-diagonals, as any pair of events with $t' - t < d$ cannot be causally linked:

$$\left.\overbrace{\begin{matrix} A_{1,1+d} & A_{1,2+d} & \cdots & A_{1,T} \\ & A_{2,2+d} & \cdots & A_{2,T} \\ & & \ddots & \vdots \\ & & & A_{T-d,T} \end{matrix}}^{\sum_{t'=1}^{T-d}\sum_{t\geq t'+d}^{T} A_{t',t}}\right\} \sum_{t=1+d}^{T}\sum_{t'\leq t-d} A_{t',t}, \tag{3.62}$$

and thus the identity incorporating the delay is:

$$\sum_{t'=1}^{T-d}\sum_{t\geq t'+d}^{T} \longleftrightarrow \sum_{t=1+d}^{T}\sum_{t'\leq t-d} . \tag{3.63}$$

On the left side, the outer sum ensures that the cause time t' does not exceed $T - d$ so that the effect time does not surpass T, while the inner sum ensures a minimum delay d between cause t' and effect t. The right side reverses the roles of t and t', with the outer sum ensuring sufficient time for an effect to follow a cause and the inner sum ensuring that causes precede their effects by at least d time steps. This identity allows us to reformulate our time-dependent learning rules while preserving causality, which helps us transition between forward and backward views. Substituting this identity into Equation 3.58, we get:

$$\frac{d\mathscr{L}}{d\theta_{ji}^{\mathrm{rec}}} \approx \sum_{k} \sum_{t=1+d_{\mathrm{out}}}^{T} \frac{\partial \mathscr{L}}{\partial y_k^t} \sum_{t'\leq t-d_{\mathrm{out}}} \left(\prod_{\tau=t'+d_{\mathrm{out}}}^{t-1} \frac{\partial y_k^{\tau+1}}{\partial y_k^{\tau}} \right) \frac{\partial y_k^{t'+d_{\mathrm{out}}}}{\partial z_j^{t'}} \frac{\partial z_j^{t'}}{\partial \mathbf{h}_j^{t'}} \cdot \epsilon_{ji}^{t'}, \tag{3.64}$$

and similarly into Equation 3.59:

$$\frac{d\mathscr{L}}{d\theta^{\mathrm{out}}_{kj}} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}}{\partial y^{t}_{k}} \sum_{t' \leq t}^{T} \left(\prod_{\tau=t'}^{t-1} \frac{\partial y^{\tau+1}_{k}}{\partial y^{\tau}_{k}} \right) \frac{\partial y^{t'}_{k}}{\partial W^{\mathrm{out}}_{kj}}. \tag{3.65}$$

These backward-looking expressions allow for cumulative computation, avoiding the need to defer calculations until all future errors are known.

By incorporating the output delay into the dynamics of the output neurons in Equation 3.20:

$$y^{t}_{k} = \kappa y^{t-1}_{k} + \sum_{j} \theta^{\mathrm{out}}_{kj} z^{t-d}_{j}, \tag{3.66}$$

we can calculate the derivatives:

$$\frac{\partial y^{t+d}_{k}}{\partial z^{t}_{j}} = \theta^{\mathrm{out}}_{kj}, \tag{3.67}$$

$$\frac{\partial y^{t}_{k}}{\partial \theta^{\mathrm{out}}_{kj}} = z^{t-d}_{j}, \tag{3.68}$$

$$\frac{\partial y^{\tau+1}_{k}}{\partial y^{\tau}_{k}} = \kappa, \tag{3.69}$$

which allows us to simplify the product in Equation 3.64:

$$\prod_{\tau=t'+d}^{t-1} \frac{\partial y^{\tau+1}_{k}}{\partial y^{\tau}_{k}} = \kappa^{t-d-t'}. \tag{3.70}$$

Substituting this expression into Equation 3.64, we get:

$$\begin{aligned} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\theta^{\mathrm{rec}}_{ji}} &\approx \sum_{k} \sum_{t=1+d}^{T} \theta^{\mathrm{out}}_{kj} E^{t}_{k} \sum_{t'=1}^{t-d} \kappa^{(t-d-t')} \psi^{t'}_{j} \mathscr{F}_{\alpha}\left(z^{t'-1}_{i}\right) \\ &= \sum_{t=1+d}^{T} L^{t}_{j} \mathscr{F}_{\kappa}\left(\psi^{t-d}_{j} \mathscr{F}_{\alpha}\left(z^{t-d-1}_{i}\right)\right), \end{aligned} \tag{3.71}$$

and into Equation 3.65:

$$\begin{aligned} \frac{\mathrm{d}\mathscr{L}}{\mathrm{d}\theta^{\mathrm{out}}_{kj}} &= \sum_{t=1+d}^{T} E^{t}_{k} \sum_{t'=1}^{t} \kappa^{(t-t')} z^{t'-d}_{j} \\ &= \sum_{t=1+d}^{T} E^{t}_{k} \mathscr{F}_{\kappa}\left(z^{t-d}_{j}\right). \end{aligned} \tag{3.72}$$

The learning signal generated at time t must be paired with the eligibility trace value at $t-d$, as the network activity prior to the output delay caused the corresponding error.

3.4.1.2 Delay from the output to the recurrent layer

Now we consider the case where there is a non-zero delay d_{ls} in the transmission of the learning signal from the output layer to the recurrent layer. This means that at time step t, the most recent learning signal available at a recurrent neuron j is $L_j^{t-d_{\mathrm{ls}}}$ and the most recent instantaneous gradient that can be applied is $g_{ji}^{t-d_{\mathrm{ls}}}$. If the transmission delay from the recurrent to the output layer is d, we can write the learning rule for the recurrent synapses as:

$$\begin{aligned}\frac{\partial \mathscr{L}}{\partial \theta_{ji}^{\mathrm{rec}}} &= \sum_{t=1+d+d_{\mathrm{ls}}}^{T} g_{ji}^{t-d_{\mathrm{ls}}} \\ &= \sum_{t=1+d+d_{\mathrm{ls}}}^{T} L_j^{t-d_{\mathrm{ls}}} \mathscr{F}_\kappa\Big(\psi_j^{t-d-d_{\mathrm{ls}}} \mathscr{F}_\alpha\Big(z_i^{t-d-d_{\mathrm{ls}}-1}\Big)\Big).\end{aligned} \tag{3.73}$$

This can be identified as a delayed optimization process, where updates to the model parameters are based on gradients computed at a delayed state rather than their current state. When gradient descent is used during the optimization process, the resulting algorithm is known as delayed gradient descent (DGD), which is widely used in distributed or asynchronous optimization scenarios [127]. A detailed analysis of the convergence properties of DGD can be found in [128]. Since the learning signal is generated at the output layer, the output layer weights do not depend on the learning signal delay. Therefore, the learning rule for the output synapses remains unchanged.

In all, when both delays are present (from the recurrent to the output layer and from the output to the recurrent layer), the learning rule for the recurrent synapses is modified in a principled way that highlights the causal connections between its three main ingredients. Specifically, the presynaptic input at time $t-d-d_{\mathrm{ls}}-1$ influences the postsynaptic neuron and its surrogate gradient at time $t-d-d_{\mathrm{ls}}$. This activity then affects the output layer at time $t-d_{\mathrm{ls}}$, which generates a learning signal that arrives at the recurrent layer at time t. This delayed chain of events is illustrated in Figure 3.3, enforcing a biologically plausible and temporally consistent learning process.

3.4.2 Continuous Dynamics

In the original implementation of e-prop [16], the internal states of neurons as well as the filtered eligibility traces are reset to zero after each update interval. This design aims to mitigate sample interference by ensuring that residual activity from one sample does not affect the subsequent one. While effective for isolating learning signals across samples, this strategy introduces a biologically implausible discontinuity in neural and synaptic dynamics.

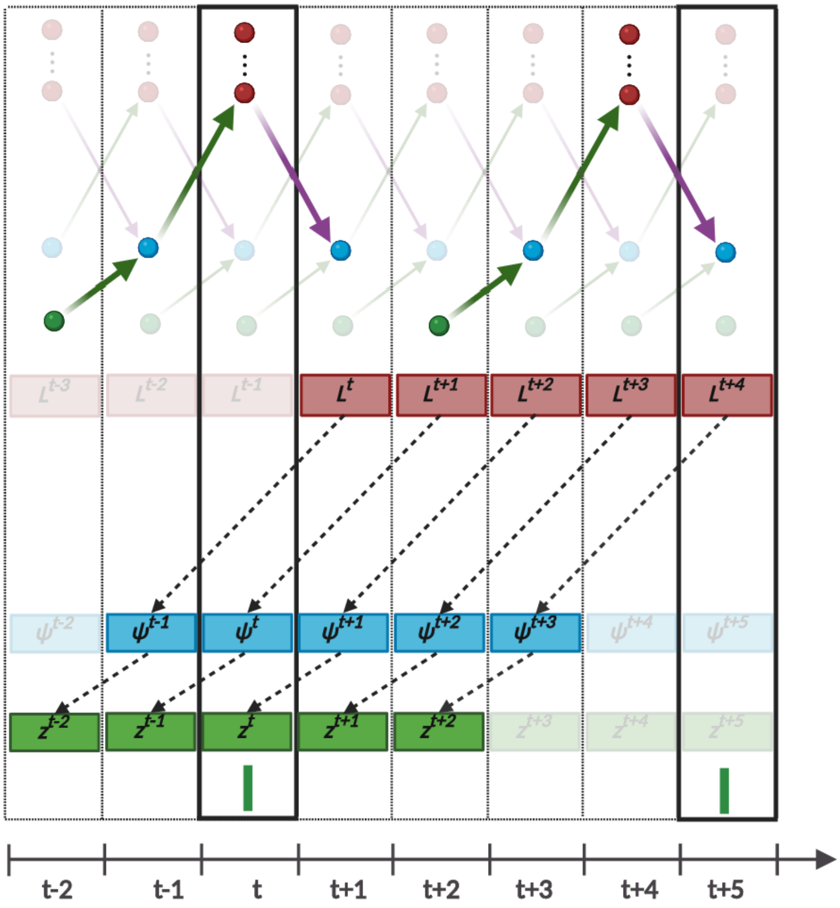


Figure 3.3: **Delayed Signal Transmission in e-prop Learning.** Schematic of forward and backward signal propagation in a neural simulation over eight time steps, highlighting two spikes that activate a synapse. The forward pass computes the output activity, while the backward pass propagates gradients for learning. The diagram demonstrates the effect of introducing explicit delays into the e-prop learning rule: spikes from the recurrent to the output layer are transmitted with a delay of, e.g., $d = 1$ time step, and the learning signal returning from the output to the recurrent layer is delayed by, e.g., $d_{\text{ls}} = 1$ time step. This setup enforces causal, biologically motivated signal transmission in both directions.

In our model, we eliminate these artificial resets, allowing neuronal and synaptic states to evolve continuously across time. The primary concern with this approach is that eligibility traces might retain information from previous samples, potentially affecting the gradients for the current one. However, this influence decays exponentially and vanishes asymptotically, as the traces are filtered with a decay kernel. Based on this observation, we hypothesize that such lingering terms exert minimal influence on learning and can be safely tolerated.

From a biological perspective, this change aligns better with how real neural circuits operate. In the brain, neurons and synapses do not reset in synchrony at the end of each perceptual or behavioral episode. Instead, their dynamics are continuous, shaped by a history of activity that decays naturally over time.

3.4.3 Dynamic Firing Rate Regularization

In both the original time-driven implementation [16] and our event-driven variant, standard firing rate regularization depends on fixed-length update intervals (Equation 3.48). This specific coupling imposes global timing constraints and is incompatible with fully asynchronous learning. To overcome this, we adopt a dynamic regularization strategy that decouples firing rate estimation from fixed intervals and enables synaptic updates to occur immediately after each presynaptic spike.

The corresponding regularization loss is defined as

$$\mathscr{L}_{\text{reg}} = c_{\text{reg}} \cdot \frac{1}{2} \sum_t \sum_j \left(\bar{f}_j^t - f^{\text{target}} \right)^2, \tag{3.74}$$

where $\bar{f}_j^t$ denotes a dynamically estimated firing rate, computed using an exponential moving average over spike trains. This formulation offers two key advantages: (i) it eliminates dependence on an explicit time origin or update interval, and (ii) it naturally prioritizes recent activity, making the regularization more adaptive to changes in neural dynamics. In this exponential formulation, $\bar{f}_j^t$ is given by:

$$\begin{aligned} \bar{f}_j^t &= \mathscr{F}_\beta\left(z_j^t\right) \\ &= \beta \bar{f}_j^{t-1} + (1-\beta) z_j^t \\ &= (1-\beta) \sum_{t'=0}^{t} \beta^{(t-t')} z_j^{t'}, \end{aligned} \tag{3.75}$$

where $\mathscr{F}_\beta(\cdot)$ denotes the filtering operator. The gradient of the regularization loss with respect to the synaptic weight θ_{ji} is:

$$\frac{d\mathscr{L}_{\text{reg}}^t}{d\theta_{ji}} = c_{\text{reg}} \cdot \left(\bar{f}_j^t - f^{\text{target}} \right) \cdot \frac{d\bar{f}_j^t}{d\theta_{ji}}. \tag{3.76}$$

Applying the locality principle of e-prop [16], we express the derivative in terms of the eligibility trace:

$$\begin{aligned}\frac{d\bar{f}_j^t}{d\theta_{ji}} &= \frac{d\mathscr{F}_\beta\left(z_j^t\right)}{d\theta_{ji}} \\ &= (1-\beta)\sum_{t'=0}^{t}\beta^{(t-t')}\frac{dz_j^{t'}}{d\theta_{ji}} \\ &= \mathscr{F}_\beta\left(\frac{dz_j^t}{d\theta_{ji}}\right) \\ &= \mathscr{F}_\beta\left(e_{ji}^t\right).\end{aligned} \tag{3.77}$$

Substituting into Equation 3.76 yields:

$$\frac{d\mathscr{L}_{\text{reg}}^t}{d\theta_{ji}} \approx c_{\text{reg}}\cdot\left(\bar{f}_j^t - f^{\text{target}}\right)\cdot\mathscr{F}_\beta\left(e_{ji}^t\right). \tag{3.78}$$

3.4.4 Classification Without Normalization

The standard approach to classification in artificial neural networks relies on the combination of a softmax output layer with a cross-entropy loss. While effective in practice, this combination presents serious limitations for biologically inspired models. The softmax function requires global normalization over all output neurons at each time step, introducing all-to-all communication and a synchronization step that is neither biologically plausible nor compatible with fully event-driven processing.

To circumvent this issue, we adopt an alternative that is both a more local and more biologically grounded. Applying findings from Hui and Belkin [129], which show that mean-squared error (MSE) performs comparably to cross-entropy on many classification benchmarks, we replace the combination of softmax and cross-entropy with a temporally extended mean-squared error loss:

$$\mathscr{L} = \frac{1}{K}\sum_{t=1}^{T}\sum_{k=1}^{K}\left(y_k^t - y_k^{*,t}\right)^2, \tag{3.79}$$

where K is the number of output classes, y_k^t is the activity of the k-th output neuron at time t, and $y_k^{*,t}$ is a one-hot encoded target signal that is active only within a designated learning window.

This modification brings two major benefits. First, it eliminates the need for global operations among output neurons, thereby preserving the model's compatibility with asynchronous and localized event-driven updates. Second, it avoids probabilistic output interpretations, instead allowing neurons to contribute to classification independently through continuous activation level.

In practice, this change also reduces the number of missing instantaneous gradients caused by delayed or incomplete transmissions, leading to more stable gradient accumulation in event-driven training where weight updates are triggered by every spike (see subsection 3.4.9).

3.4.5 Eligibility Trace Filter Decoupled from Output Time Constant

In the original time-driven formulation [16], the time constant of the eligibility trace filter (Equation 3.42) is tied to the time constant of the output neuron's dynamics (Equation 3.20). This coupling implies that synapses must have knowledge of the output neuron's internal properties properties in order to compute weight updates, which is a requirement that poses a subtle challenge to the locality of the learning rule. To address this, we decouple the eligibility trace filter from the output neuron's time constant.

3.4.6 Surrogate Gradients: Smoothness and Flexibility

Since spiking neurons emit discrete outputs, their activation functions are inherently non-differentiable. In the original time-driven e-prop model, a piecewise linear function is used to replace this derivative (Equation 3.14).

While this choice is computationally efficient, it introduces sharp transitions near the threshold, which are difficult to explain biologically. To address this, we investigate a range of smooth surrogate functions proposed in the literature [90]. Among these, we adopt the exponential surrogate gradient [130] which offers a continuous, bell-shaped profile centered around the threshold. This smoothness reduces abrupt changes in the gradient and reflects the idea that biological neurons exhibit a gradual increase in firing probability near their threshold potential.

To further evaluate the flexibility of surrogate gradient design, we compare the exponential form with other smooth alternatives, such as fast sigmoid derivatives and arctangent functions. We find that, after proper tuning of their height (γ) and width (β) parameters, all functions can be aligned to similar shapes, as shown in Figure 3.9a. This flexibility allows us to choose surrogate gradients that balance computational efficiency with biological realism, while still maintaining effective learning dynamics.

3.4.7 Full Membrane Voltage Reset

In the original model, spike-triggered resets of the membrane potential are implemented as partial decrements, that is, subtracting a fixed value following each spike. In contrast, we implement a full reset scheme, where the membrane voltage is immediately set to a fixed baseline after spiking.

This approach is more consistent with classical formulations of the leaky integrate-and-fire neuron used in theoretical neuroscience, and simplifies the dynamics following spike emission.

3.4.8 Biologically Inspired Connectivity and Weight Constraints

Biological neural circuits are governed by structural and functional constraints that are rarely enforced in artificial neural networks. In particular, synaptic organization in the brain is shaped by two important principles: connectivity sparsity and neuron-type consistency. First, cortical networks are not densely connected, instead, neurons form synapses with only a small subset of their neighbors, being connection probabilities around 10 % [131, 132]. Second, neurons generally respect a rule known as Dale's law, which states that a given neuron releases a single type of neurotransmitter, causing all of its outgoing connections to be exclusively excitatory or inhibitory [133, 134, 135].

In order to explore the functional implications of these constraints, we trained spiking networks on the N-MNIST task while incorporating biologically motivated design elements systematically. We first evaluated the effect of connectivity sparsity, reducing the probability of connections to as low as 1 % in the recurrent layer, while keeping the output layer fully connected.

Next, we turned to weight constraints. Differently from standard artificial networks, where weights can freely switch signs, we imposed fixed sign at the synapse level, ensuring that once a synapse is excitatory or inhibitory, it remains so throughout training. We applied this constraint separately and jointly to input, recurrent, and output weights. In the most biologically faithful configuration, we additionally enforced Dale's law by assigning each neuron a fixed excitatory or inhibitory identity, respecting the typical cortical ratio of 4:1 [136, 137].

3.4.9 Spike-Triggered Asynchronous Weight Updates

In many time-driven learning models, synaptic weights are updated synchronously at fixed intervals, typically at the end of a training sample or batch. This approach assumes that all synapses can coordinate their updates simultaneously based on a shared temporal reference, a requirement that lacks biological justification.

In contrast, biological synapses operate without a central clock. Plasticity is driven by local events, namely, the timing of spikes and the interaction between pre- and postsynaptic activity [43, 44]. To better reflect this decentralized structure, we propose an event-driven scheme in which each presynaptic spike triggers an immediate weight update. The update is computed using information accumulated since the previous spike at that synapse, ensuring that the current spike is processed with the most up-to-date weight.

This approach eliminates the need for synchronized or batched updates and aligns closely with biologically grounded mechanisms like STDP [60]. Specifically, given two consecutive presynaptic spike times, $t_{\text{prev spike}}$ and t_{spike}, the update is calculated over the interval $[t_{\text{prev spike}} + \Delta t, t_{\text{spike}}]$ using the most recent learning signal L^t and the relevant history of eligibility traces. Differently from the time-driven approach, in which weights remain fixed within an update interval, this additional feature of the event-driven formulation allows each new spike to trigger a local and asynchronous modification of synaptic efficacy using the most recent available information.

Enabling such a spike-wise update mechanism requires several biologically inspired modifications to the original model: continuous dynamics without resets (subsection 3.4.2), dynamic firing rate regularization (subsection 3.4.3), a classification objective that avoids the need for global normalization (subsection 3.4.4) and crucially, a strictly causal learning rule obtained by considering and introducing transmission delays between recurrent and output layers (subsubsection 3.4.1.1) as well as from output to recurrent layers (subsubsection 3.4.1.2).

3.4.10 Design and Implementation

The implementation of the event-driven e-prop algorithm with with additional biological features largely follows the principles described in subsection 3.3.4, with some few important modifications. Unlike the base implementation, this version does not aim to replicate the time-driven results exactly, nor does it require introducing a relative temporal shift between recurrent and output neurons, since transmission delays are now explicitly incorporated within the learning rule itself.

A key operational difference is that each presynaptic spike immediately triggers a weight update, satisfying

$$\texttt{t\_update} = \texttt{t\_spike}.$$

This spike-triggered update mechanism significantly reduces the volume of spiking data that must be stored to compute the update rule. However, because synaptic traces are maintained continuously without resets between updates, the synapse must also transmit the current values of relevant running traces alongside spike timing information. The postsynaptic neuron uses these inputs (spike times and synaptic trace values), to compute the gradient needed for the weight update and then returns this gradient to the synapse for immediate application.

3.4.11 Cleaning of e-prop History

As in the base event-driven implementation, the event-driven e-prop algorithm with with additional biological model maintains an `update_history` data structure to track the timing of synaptic updates. However, in this variant, weight updates occur continuously between spikes, and the function `erase_used_eprop_history` is invoked on every incoming spike, rather than only once at fixed update intervals as in the `bsshslm_2020` implementation.

Outdated entries in `eprop_history` are eliminated by removing all records older than the earliest update time `t_update` found at the front of the `update_history` queue. Unlike the fixed-interval model, silent neurons can cause unbounded growth of `eprop_history`, since the usual mechanisms for history cleanup do not apply without synchronized update windows.

To address this, we introduce a cutoff threshold limiting how far back eligibility traces are integrated between spikes. This cutoff is based on the rapid exponential decay characteristic of eligibility traces over long inter-spike intervals, providing a principled and practical approximation to restrict memory usage. Consequently, any `eprop_history` entries lying entirely within this cutoff interval for all incoming synapses can be safely deleted.

Because `update_history` entries are sorted by their `t_update` timestamps, identifying removable entries is straightforward: for two consecutive update times t_0 and t_1, all `eprop_history` entries between $t_0 +$ `eprop_isi_trace_cutoff` and t_1 can be removed, as these will no longer contribute to any forthcoming weight updates.

3.5 Results

This section evaluates the performance of our event-driven e-prop implementation across three benchmark tasks. We compare three variants: the original time-driven model [16], our base event-driven version, and an extended model incorporating additional biologically inspired features. The comparison focuses on learning performance and computational efficiency.

3.5.1 Benchmark Tasks

To validate the proposed models, we examine three tasks of increasing complexity. The first is a regression task in which the network learns to generate a continuous target signal in response to a structured input. The second task models evidence accumulation for decision-making, requiring the network to integrate inputs over time and produce a discrete classification. The final task uses the N-MNIST dataset, a standard benchmark for neuromorphic vision, to test the model's performance on real-world, event-based sensory input.

3.5.1.1 Pattern Generation Task

The pattern generation task is framed as a regression problem in which the network learns to autonomously reproduce a continuous target signal from noisy spike-based input. The target signal is constructed as the sum of four sinusoids with randomly assigned amplitudes and phases [16].

The input consists of 100 Poisson spike trains, each projecting to a recurrently connected population of 100 leaky integrate-and-fire (LIF) neurons. All connections, including input, recurrent, and output synapses, are plastic. Each neuron in the recurrent spiking neural network (RSNN) projects to a single linear readout neuron, which integrates spikes over time to produce the network's output.

Each training iteration corresponds to a 1-second simulation at a temporal resolution of 1 ms, during which the network is expected to generate the target signal in the absence of direct supervision. The task tests the model's ability to form internal temporal representations and generate structured output, key components of biologically inspired motor systems [138, 139].

3.5.1.2 Evidence Accumulation Task

The evidence accumulation task challenges the network with a decision-making scenario inspired by animal behavior studies [16, 140, 141]. In the experimental setup, a mouse runs along a tunnel while receiving brief, lateral cues, from the left and the right. At the tunnel's end, the mouse must choose the direction with the greater number of cues. The network, however, is not given this rule explicitly; it must infer the correct decision based on experience.

To simulate this process, the model receives input from four distinct spiking populations: two deliver Poisson spike trains representing the left and right cues, one provides constant background activity throughout the trial, and the last signals the decision phase by activating only at the end. Synaptic plasticity is restricted to this final phase, creating a clearly defined learning window.

A central difficulty of this task is the temporal separation between cue presentation and decision-making. Maintaining cue information across this gap requires a form of working memory. To address this, the network employs adaptive leaky integrate-and-fire (ALIF) neurons in its recurrent layer (Equation 3.17). These neurons feature slow, activity dependent threshold dynamics, which provide the long time constants necessary to bridge the temporal delay.

The output layer contains two neurons corresponding to the left and right decisions. Their membrane potentials are converted to probability estimates using the softmax function (Equation 3.24), and training is guided by a cross-entropy loss (Equation 3.25). This setup imposes an all-to-all communication requirement among output neurons, as they must share their membrane voltages to compute the softmax probabilities, and adds an extra computational time step to the learning process.

3.5.1.3 Neuromorphic MNIST

The N-MNIST dataset [142] transforms the classical MNIST handwritten digit images into a spiking neuromorphic vision dataset by using dynamic vision sensors (DVS). These type of sensors respond to changes in the visual scene, such as brightness variations or motion, emitting asynchronous spike-like events instead of frame-based images.

To generate N-MNIST, static 2D MNIST digits were displayed sequentially on a monitor while a DVS mounted on a platform scanned the images along a plane parallel to the screen. In the scanning approach, the camera executed three 100 ms saccades tracing a triangle pattern, introducing temporal variations similar to natural eye movements [142].

Each dataset sample consists of a 34 × 34 pixel grid represented as a list of binary events. Each event encodes a timestamp, pixel coordinates (x, y), and a polarity indicating either an increase (ON event) or decrease (OFF event) in pixel intensity. Pixels without intensity changes do not produce events, resulting in sparse, temporally precise data well-suited for spiking neural network (SNN) models.

To improve computational efficiency, pixels with no or negligible event activity are excluded. For the remaining pixels, each is modeled by a spike generator that emits a spike for every detected ON event. These spike generators feed into corresponding input neurons, which project onto a recurrent spiking neural network. The recurrent network connects to 10 output neurons, each representing one of the digit classes. During training, the network output is compared against a teacher signal generated by a rate-based neuron representing the correct digit class.

3.5.2 Proof of Concept: Event-Driven e-prop Reproduces Time-Driven Results

In this section, we demonstrate that our event-driven implementation of the e-prop learning algorithm faithfully replicates the key results originally obtained with the time-driven approach reported by Bellec et al. [16].

We first replicate the supervised regression task from Bellec et al. [16] (see subsubsection 3.5.1.1) using the event-driven e-prop scheme. The network configuration, simulation parameters, and hyperparameters are provided in the appendix Tables 7.1 and 7.2. Figure 3.4a contrasts key dynamic traces, including target signal, network output, membrane potentials, and spike raster recorded *before* and *after* learning. Panel b shows weight distributions at the same two stages. The loss curve obtained with the event-driven code is indistinguishable from that of the reference time-driven implementation (see Figure 3.7a,b). Optimization used standard gradient descent [78] on the mean-squared error loss (Equation 3.22).

Next, we replicate the classification task from Bellec et al [16] (see subsubsection 3.5.1.2). The network configuration, simulation parameters, and hyperparameters are provided in the appendix Tables 7.3 and 7.4. Unlike the previous task, solving this classification problem requires batch learning, and the gradients are optimized using the Adam optimizer [111]. In the original time-driven setup, batch learning is implemented by running multiple parallel instances of the network, one per training example in the batch, with shared initialization. After each iteration, the weight updates are averaged and applied uniformly to all instances. To maintain biological plausibility in our event-driven formulation, we instead process each training sample sequentially and apply the averaged update only after completing a full batch cycle.

Figure 3.5a displays network activity before and after learning. Yellow and red spikes mark the activity of two stimulus-sensitive populations encoding cue presentations on the

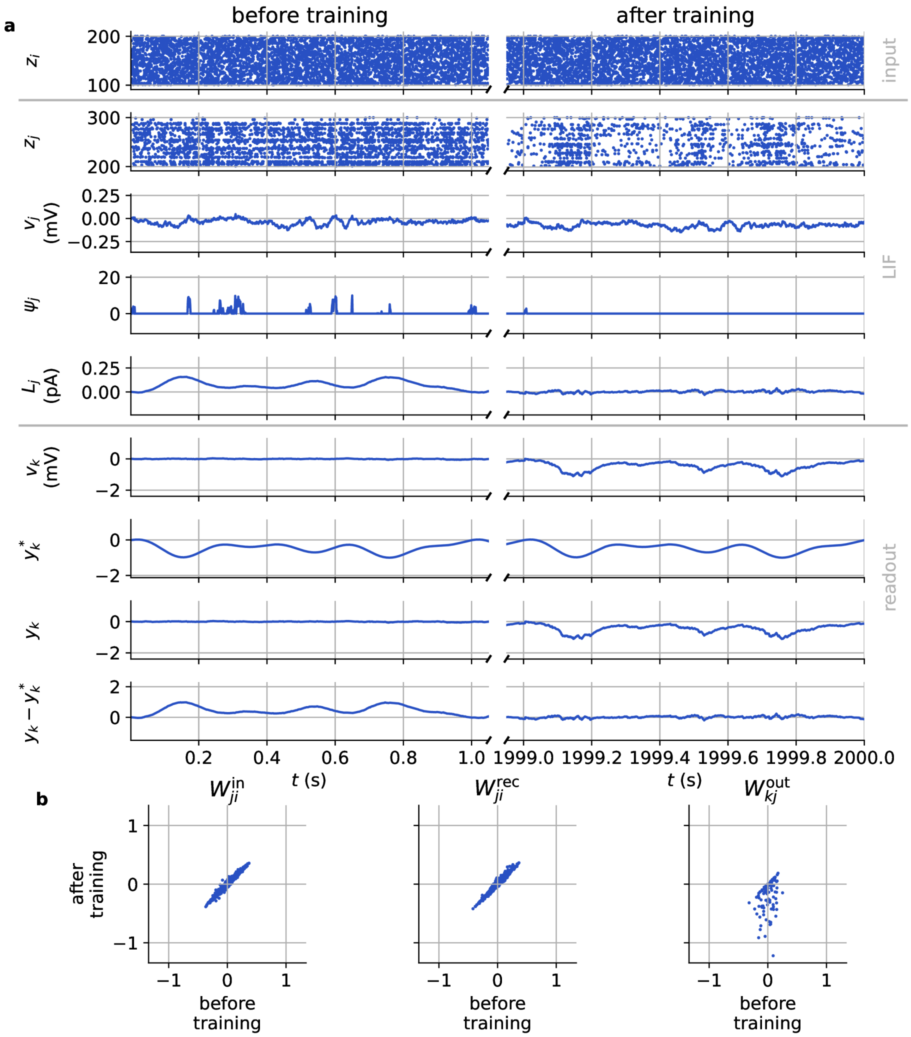


Figure 3.4: **Time courses and weight distributions for the pattern-generation task using event-driven e-prop. (a)** Representative time courses of key dynamic variables before and after training, including spike raster excerpts of input and recurrent neurons, membrane potential of a sample neuron, surrogate gradient, learning signal, network output, target signal, and error signal. **(b)** Distributions of input, recurrent, and output weights before and after training.

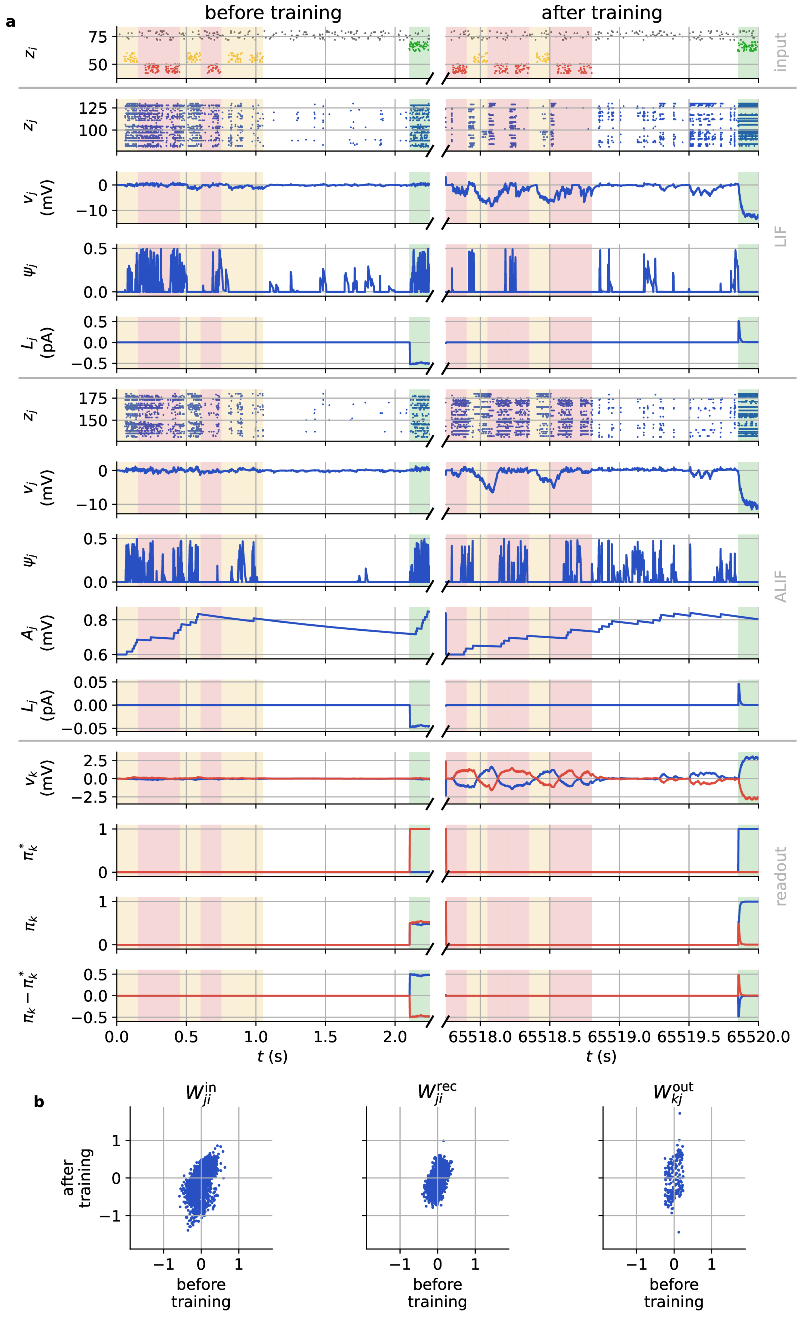


Figure 3.5: **Time courses and weight distributions for the evidence accumulation task using event-driven e-prop. (a)** Representative time courses of key dynamic variables before and after training, including spike raster excerpts of input and recurrent neurons, membrane potential of a sample neuron, surrogate gradient, adaptive threshold, learning signal, network output, target signal, and error signal. Yellow and red spikes correspond to cue-sensitive neuron populations responding to left and right sensory stimuli, respectively; green spikes indicate the decision phase after a latency period. **(b)** Distributions of input, recurrent, and output weights before and after training.

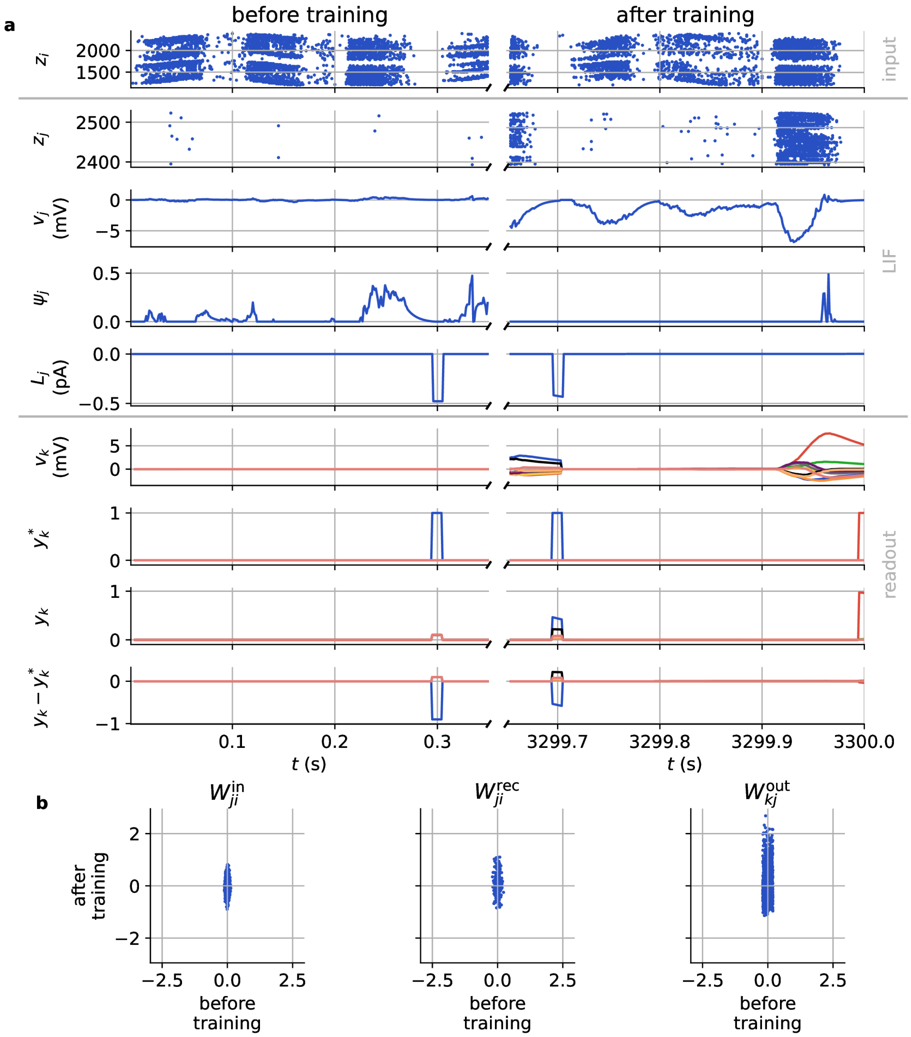


Figure 3.6: **Time courses and weight distributions for N-MNIST using event-driven e-prop. (a)** Representative time courses of key dynamic variables before and after training, including spike raster excerpts of input and recurrent neurons, membrane potential of a sample neuron, surrogate gradient, learning signal, network output, target signal, and error signal. **(b)** Distributions of input, recurrent, and output weights before and after training.

left and right sides, respectively, mirroring a rodent's sensory experience while running on a T-maze. Green spikes indicate decision-related activity following a delay period. Panel b shows the evolution of input, recurrent, and output weight distributions after training.

As shown in Figure 3.7c and d, the event-driven model closely matches the time-driven implementation in terms of classification accuracy and learning dynamics. Minor deviations can be attributed to floating-point precision differences.

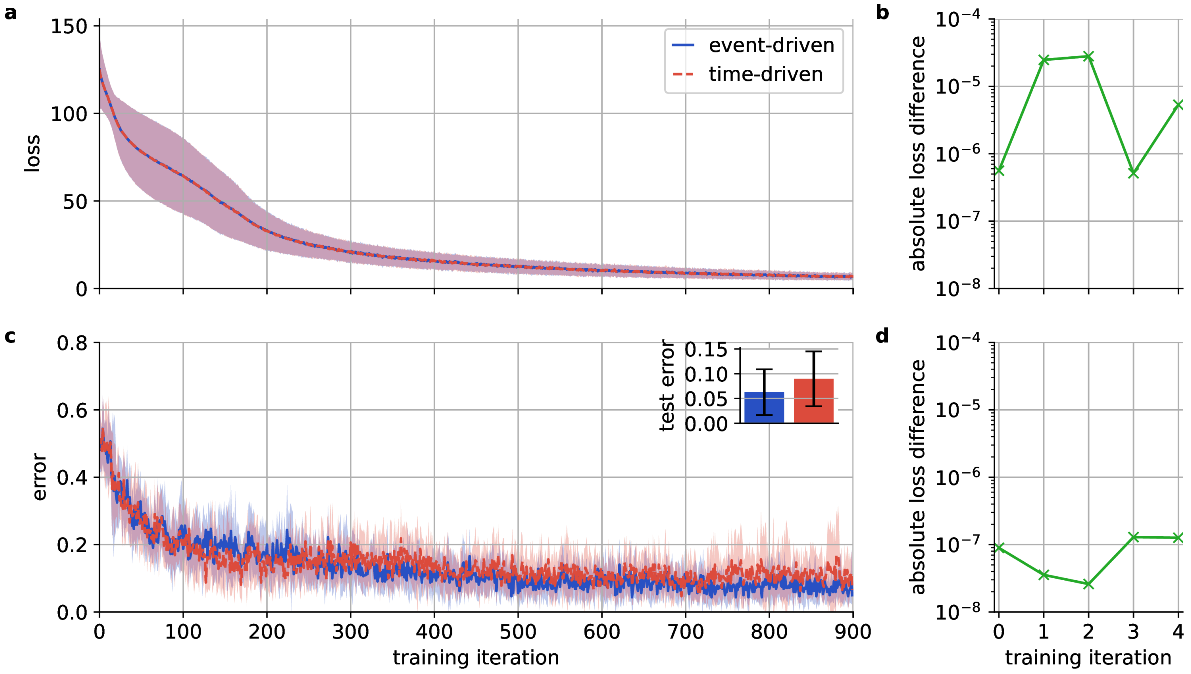


Figure 3.7: **Side-by-side learning performance of time-driven and event-driven e-prop.** (a) Training loss across 900 iterations for the pattern generation task. Solid lines represent the mean, and shaded regions show standard deviation across 10 trials with different random seeds. (b) Absolute loss difference between time- and event-driven implementations during the first five training iterations of a single trial (pattern generation task). (c) Classification error over 900 training iterations for the evidence accumulation task with a batch size of 32. The inset displays the mean and standard deviation of test errors post-training, averaged across 10 independent runs. (d) Absolute loss difference between models for the first five iterations of the evidence accumulation task with a batch size of 1.

Additionally, we implement a classification task using Neuromorphic MNIST (see subsubsection 3.5.1.3), which was not reported in Bellec et al. [16]. The network configuration, simulation parameters, and hyperparameters are provided in the appendix Tables 7.5 and 7.6. Here, training employs cross-entropy loss and gradient descent with a batch size of 1. Figure 3.6a shows the time courses of dynamic variables before and after training, while Figure 3.6b presents the corresponding weight distributions. The prediction error over training is shown in Figure 3.11.

3.5.3 Performance of Event-Driven e-prop with Biological Features

Here, we investigate how incorporating biologically inspired features influences the learning capabilities and efficiency of our event-driven e-prop implementation, as outlined in subsection 3.4.10. Our analysis contrasts this enhanced model against both the baseline event-driven e-prop and the original time-driven approach from Bellec et al. [16].

Focusing on the N-MNIST classification task, we systematically introduce each biological modification individually and measure its effect over 300 training iterations. Transmission delays between recurrent and output layers, as well as feedback delays from output to recurrent neurons, are uniformly set to 1 ms across all experiments. Further information about the network configuration, as well as simulation parameters and hyperparameters, is provided in the appendix Tables 7.7 and 7.8. Learning success is quantified by the classification error averaged across 10 test runs, where a decrease in error indicates improved performance. Detailed outcomes, including absolute errors and relative improvements over the baseline event-driven model, are presented in Table 3.1.

Removing neuron and eligibility trace resets resulted in only 0 % and 3 % reductions in learning performance, respectively, indicating that such resets are not critical and suggesting minimal interference between samples. Replacing the softmax-cross-entropy combination with a mean-squared error objective improved classification performance by 15 %, enhancing both accuracy and biological plausibility. Decoupling the filter time constant from the output neuron's time constant yielded only a 6 % decrease in performance, while removing the filter entirely in a regression task led to improved learning, supporting the independence of synaptic filtering from output dynamics (see Figure 3.8).

Adjusting the scale parameters of the surrogate gradient led to a 3 % performance boost, and switching to a smoother exponential surrogate function provided an additional 6 % gain. Surrogate gradient functions with similar shapes produced nearly identical performance, consistent with prior findings that learning is robust to variations in surrogate gradient shape [143]. These results support replacing the commonly used piecewise-linear function with smoother, more biologically realistic alternatives without performance loss (see Figure 3.9).

When all biological features were integrated, the model achieved learning performance within 1 % of the baseline, and the addition of a full membrane voltage reset improved accuracy by 5 %. The time evolution of dynamic variables and weight distributions before and after training are shown in Figure 3.10, and the classification error time course compared to the time-driven model is shown in Figure 3.11.

Finally, constraining synaptic weights to maintain their sign reduced learning performance by 20 % for input weights, 11 % for output weights, and 47 % when applied globally. Enforcing Dale's law with an excitatory-to-inhibitory ratio of 4:1 led to even greater declines: 44 %, 15 %, and 73 % for input, output, and all weights, respectively. Interestingly, when applied only to recurrent weights, both fixed sign constraints and Dale's law resulted in a 4 % improvement. These findings indicate that biological constraints may benefit learning when applied selectively to recurrent synapses, but tend to hinder performance when extended to input or output connections.

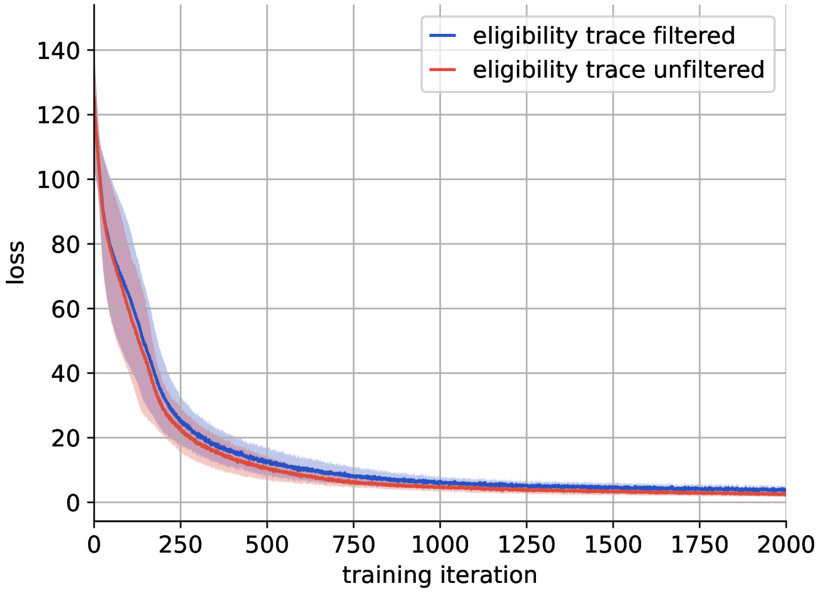


Figure 3.8: **Effect of eligibility trace filtering on learning performance.** Comparison of loss trajectories for the pattern generation task over 2000 training iterations, averaged across 10 different random seeds, highlighting the impact of using filtered versus unfiltered eligibility traces.

Beyond learning performance, we also assessed the computational efficiency of individual biological features by measuring the runtime required to complete 300 training and 10 test iterations on the N-MNIST task, equivalent to a simulated biological time of 2 h and 35 min. The results are reported in Table 3.2 both as absolute runtimes and as deviations relative to the base event-driven model.

Several biologically motivated modifications, including continuous neuron dynamics,

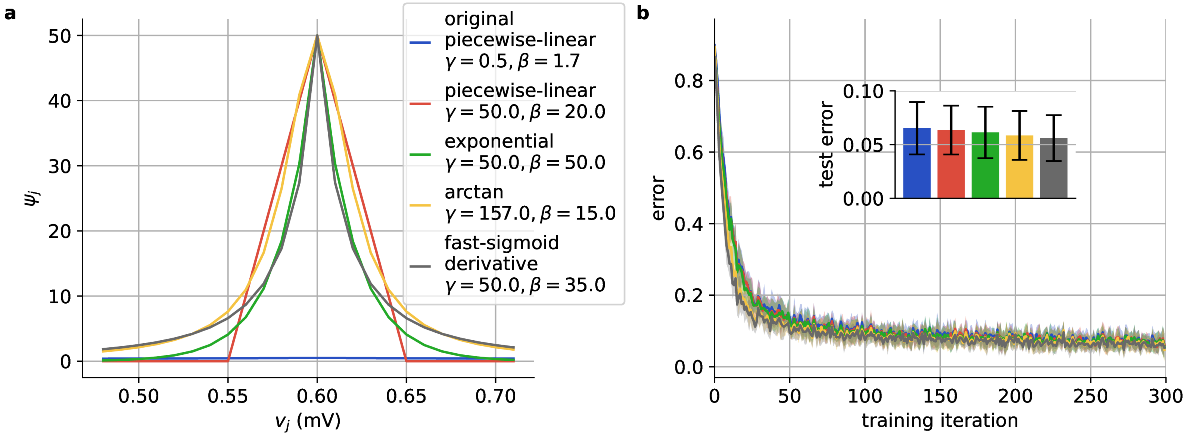


Figure 3.9: **Comparison of surrogate gradient functions and their learning performance.** (a) Surrogate gradient shapes plotted against membrane potential relative to a fixed threshold of 0.6 mV, including piecewise-linear, exponential, fast sigmoid derivative [144], and arctangent-based [145] functions. (b) Training error over time on the N-MNIST classification task using each surrogate gradient. The inset shows the mean test error with standard deviation over 10 trials, demonstrating comparable learning efficacy across surrogate types.

mean-squared error classification, and eligibility traces decoupled from the readout time constant, were found to incur no additional runtime cost. Moderate overheads arose from using continuous e-prop trace dynamics (4 %) and smoothed surrogate gradients, such as the scaled piecewise-linear (16 %) and exponential (14 %) functions. When all biological features were enabled, the total runtime increased by 9 %; incorporating a full membrane voltage reset added a further 6 %.

Weight constraints introduced only minor runtime penalties. Fixing weight signs increased simulation times by up to 16 %, while enforcing Dale's law with a 4:1 excitatory-to-inhibitory ratio added between 7 % and 12 %, depending on the synapse type. Overall, these increases remain well within practical limits, confirming that biologically realistic features can be integrated without prohibitive computational cost.

3.5.3.1 Effect of Larger Transmission Delays on Learning Performance

To evaluate how increased transmission delays affect learning, we conducted additional experiments on the regression task described in subsubsection 3.5.1.1. Specifically, we varied the delays between recurrent and output neurons (d) and from output to recurrent neurons (d_{ls}) across a wide range: from 1 ms to 2048 ms. Further details on the network configuration, simulation parameters, and hyperparameters are provided in the appendix Tables 7.9

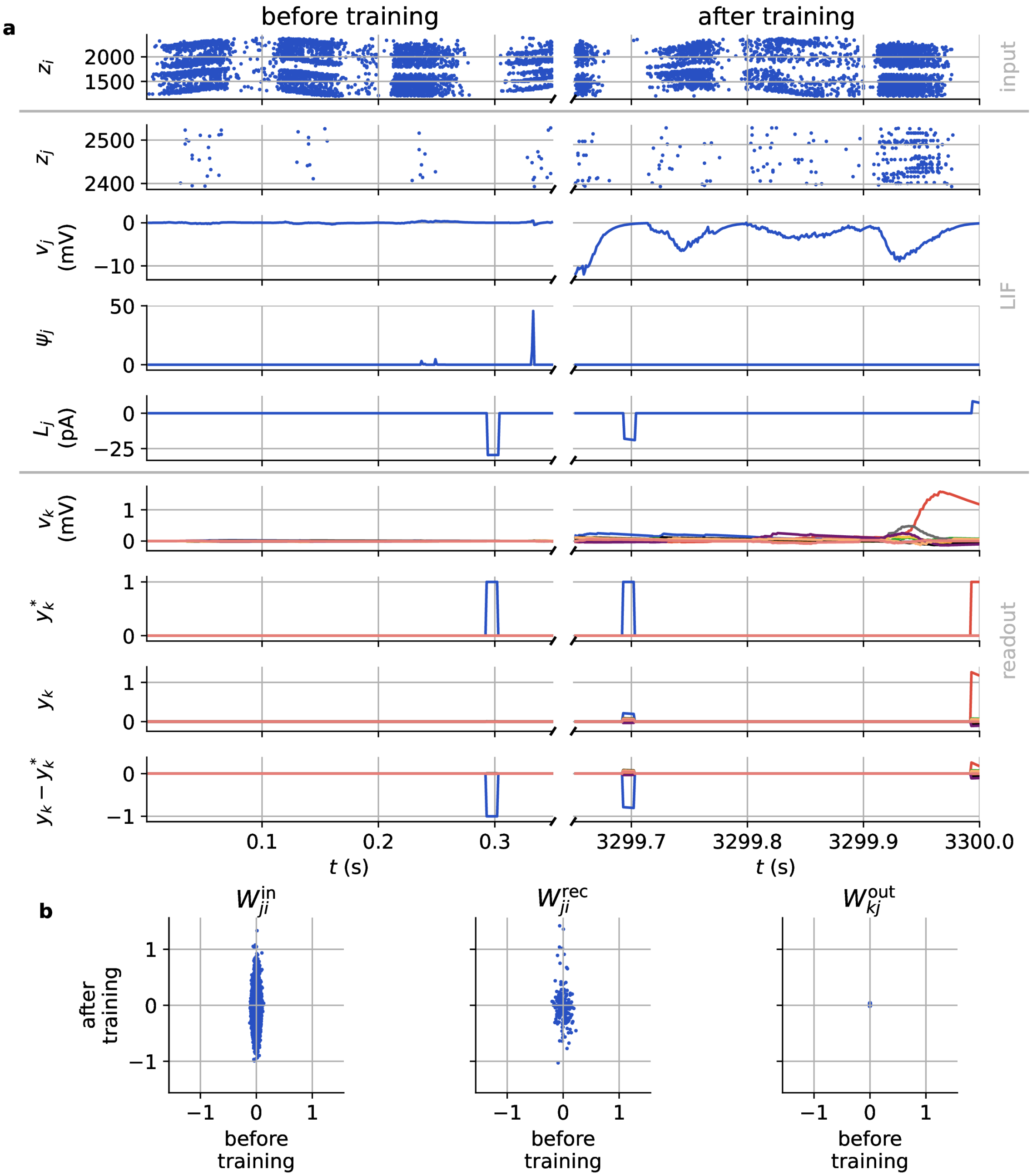


Figure 3.10: **Time courses and weight distributions for N-MNIST using biologically enhanced e-prop. (a)** Representative time courses of key dynamic variables before and after training, including spike raster excerpts, membrane potential, surrogate gradient, learning signal, network output, target signal, and error signal. **(b)** Distributions of input, recurrent, and output weights before and after training.

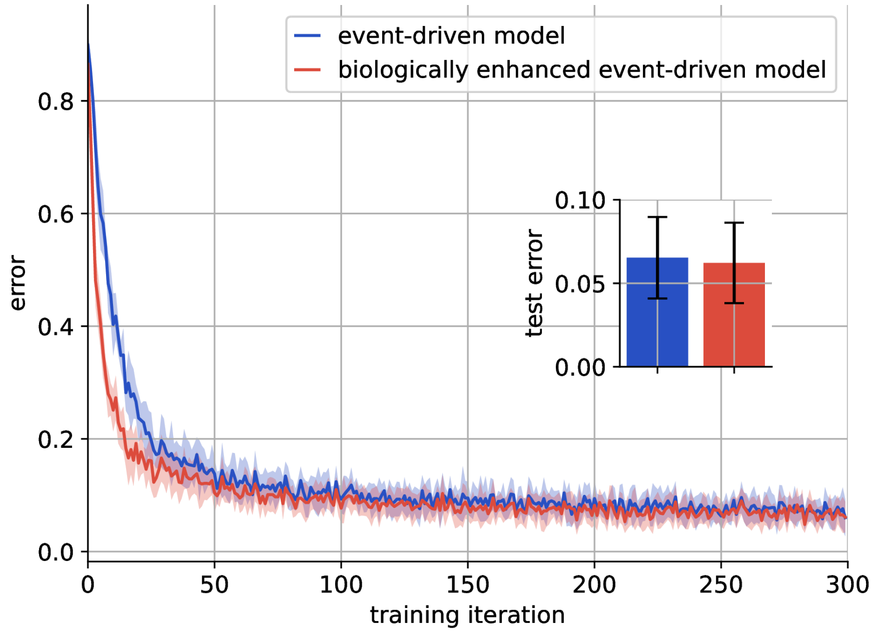


Figure 3.11: **Comparison of learning accuracy for event-driven e-prop with and without biological features.** Prediction error curves on the N-MNIST task contrasting models incorporating additional biological features versus the baseline event-driven e-prop. The inset displays mean test errors with standard deviations over 10 test runs. All error metrics are averaged over 10 random seeds, and test errors additionally include statistics over 10 samples.

and 7.10.

Figure 3.12 illustrates the loss trajectories for fixed output-to-recurrent delay $d_{\text{ls}} = 1$ ms while varying d, and vice versa. As expected, larger delays tend to degrade learning performance, reflected in higher mean-squared errors (MSE). However, the model remains remarkably robust: when either d or d_{ls} is kept small (e.g., 1 ms) while the other increases to 256 ms, the loss curves remain close to the baseline case where both delays are minimal.

Figure 3.13 provides a matrix of final MSE values averaged across five random seeds for all combinations of d and d_{ls}. While the overall trend confirms that larger combined delays ($d + d_{\text{ls}} \geq 512$ ms) degrade learning, the network still performs well with moderately increased delays, underscoring the robustness of the event-driven implementation.

3.6 Discussion

We present an event-driven extension of the e-prop learning rule for spiking neural networks, implemented in the NEST simulator, a platform designed for large-scale and bio-

Table 3.1: **Test error of e-prop with additional biological features.** The results were obtained using the N-MNIST task, simulated for 300 training iterations and 10 test iterations, averaged over 10 random seeds. Absolute test errors were rounded to four decimal places. Relative values were calculated as deviations from the baseline model using unrounded values.

biological feature	absolute	relative %
without features	0.0653(244)	+0
continuous neuron dynamics	0.0654(241)	+0
continuous e-prop trace dynamics	0.0674(251)	+3
mean squared error classification	0.0557(211)	**-15**
eligibility trace filter decoupled from output	0.0692(248)	+6
scaled linear surrogate gradient	0.0635(227)	**-3**
scaled exponential surrogate gradient	0.0613(239)	**-6**
all features combined	0.0659(261)	+1
all features + full reset	0.0622(241)	**-5**
fixed sign input weights	0.0784(249)	+20
fixed sign recurrent weights	0.0630(202)	**-4**
fixed sign output weights	0.0726(251)	+11
fixed sign all weights	0.0957(301)	+47
Dale's law input weights	0.0941(271)	+44
Dale's law recurrent weights	0.0626(239)	**-4**
Dale's law output weights	0.0752(283)	+15
Dale's law all weights	0.1129(337)	+73

logically realistic brain simulations. This implementation faithfully reproduces the original time-driven results by Bellec et al. [16], generalizes effectively to additional tasks such as N-MNIST, and is already available in NEST version 3.7 [108].

Porting the algorithm from TensorFlow to NEST required key adaptations, including explicit transmission delays and strict enforcement of locality and causality. These changes ensure that the model maintaining the core principles of e-prop while allowing the event-driven implementation.

To improve biological plausibility, we replace machine-learning-driven components with mechanisms grounded in neuroscience and assess their impact on learning and runtime using the N-MNIST task. All test classification errors remain below 0.12 , indicating consistently effective learning. Features like continuous neuron and e-prop dynamics have negligible to minor performance effects, and using smoother surrogate gradients or mean-squared error loss enhances learning. Combining all biological features, including spike-triggered updates and dynamic firing rate regularization, results in learning performance

Table 3.2: **Runtime of e-prop with additional biological features.** The results were obtained using the N-MNIST task, simulated for 300 training iterations and 10 test iterations, averaged over 10 random seeds. Absolute runtimes are given in core-hours and rounded to three decimal places. Relative values were calculated as deviations from the baseline model using unrounded values.

biological feature	absolute core · h	relative %
without features	368.532(2986)	+0
continuous neuron dynamics	368.373(1307)	+0
continuous e-prop trace dynamics	383.024(2156)	+4
mean-squared error classification	367.437(2874)	+0
eligibility trace filter decoupled from output	370.026(4478)	+0
scaled linear surrogate gradient	426.177(8718)	+16
scaled exponential surrogate gradient	419.797(8904)	+14
all features combined	402.528(3871)	+9
all features + full reset	391.628(2644)	+6
fixed sign input weights	425.519(3564)	+15
fixed sign recurrent weights	406.037(1184)	+10
fixed sign output weights	394.436(3936)	+7
fixed sign all weights	427.141(3311)	+16
Dale’s law input weights	412.022(3422)	+12
Dale’s law recurrent weights	405.430(3023)	+10
Dale’s law output weights	394.125(2922)	+7
Dale’s law all weights	412.774(3558)	+12

comparable to the baseline.

Simulating 300 training and 10 test iterations requires about 3.2 core · h for the baseline model. Adding biological features increases runtime by up to 15 %, primarily due to higher firing rates from modified dynamics, which constitutes an effect that is mitigable by parameter tuning.

While results focus on N-MNIST, future work should test broader architectures, neuron models, and hyperparameter settings. Comparing performance across simulation frameworks would require fully optimized implementations, but spike-based models still promise significant energy savings and could inspire machine learning advances.

Our work builds on prior efforts to integrate three-factor learning rules into NEST [146, 107] and is part of a broader initiative to advance neural simulation technology [98]. The resulting model offers a flexible foundation for future developments, such as reward-based learning and deployment via NESTML [147].

In parallel with ongoing efforts to port e-prop to diverse platforms, including mlGeNN

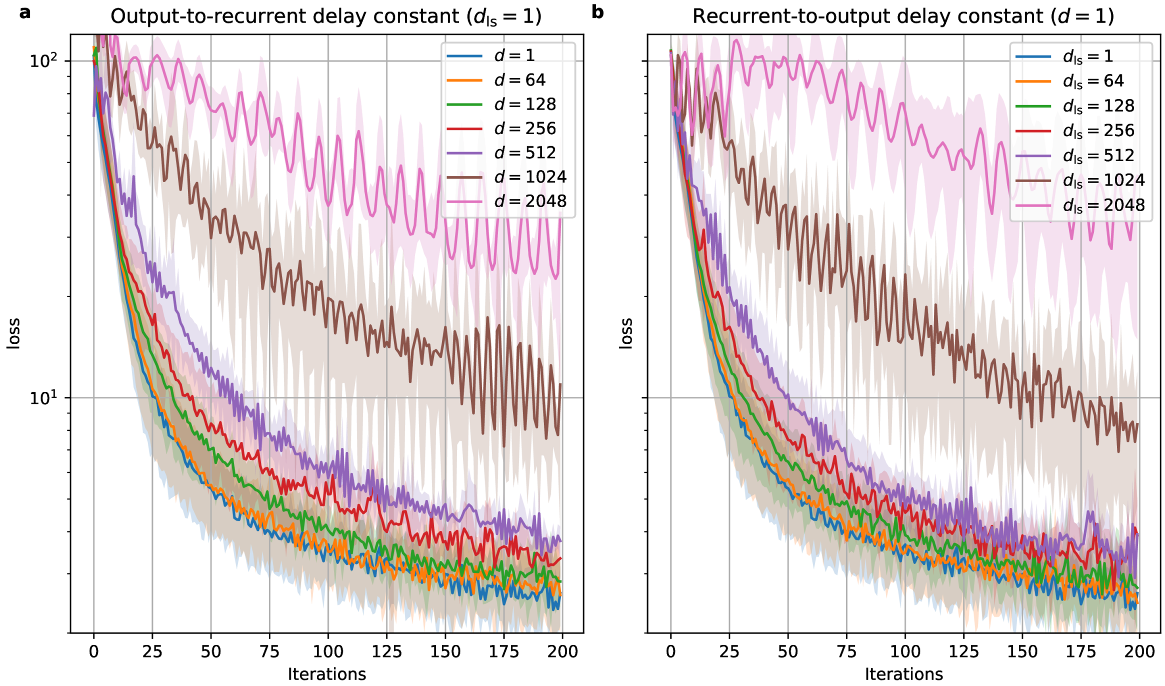
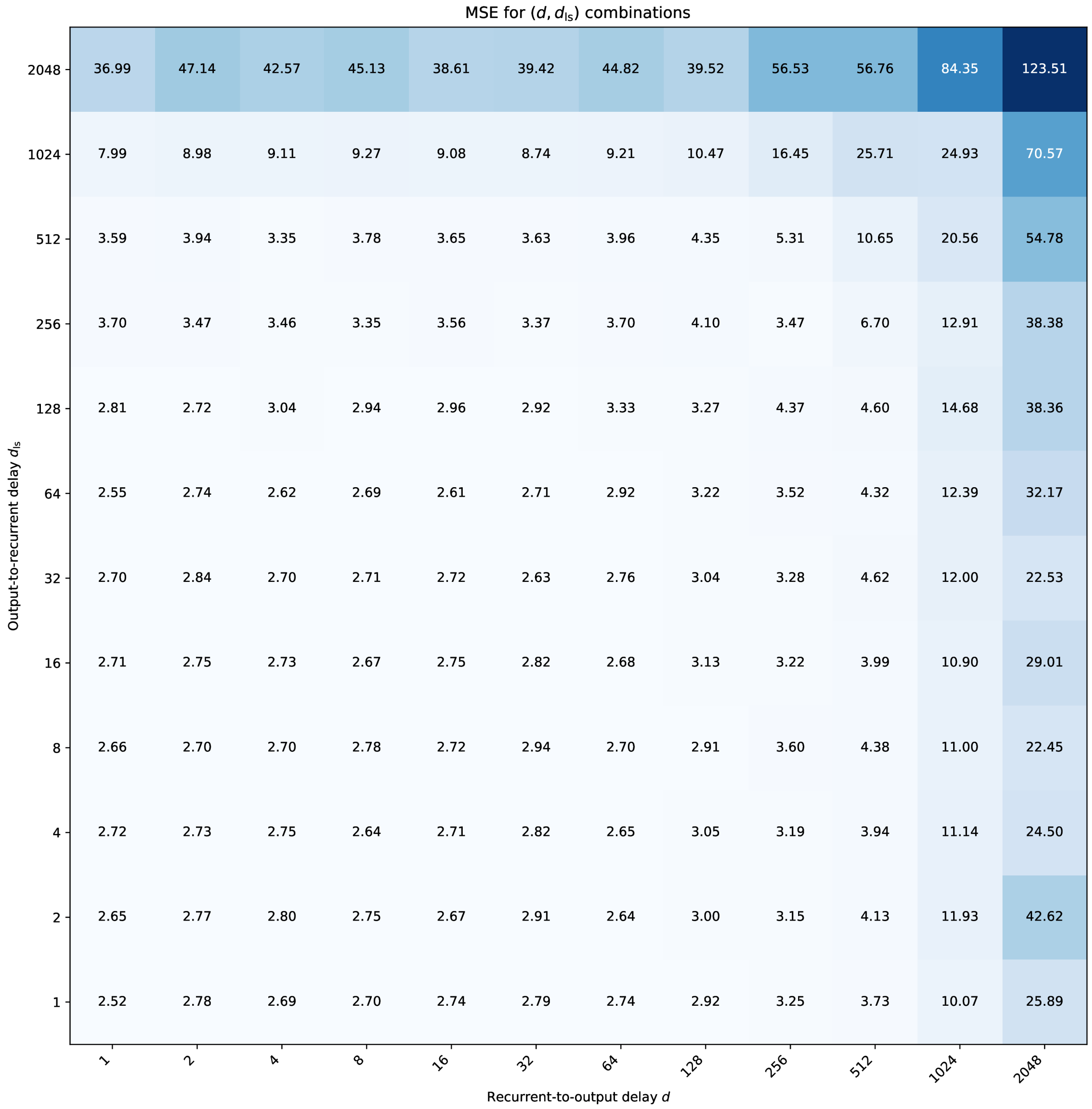


Figure 3.12: **Impact of increased delays on regression performance.** Mean-squared error (MSE) loss over 200 training iterations for the pattern generation task, averaged over 5 random seeds. (a) Output-to-recurrent delay fixed at 1 ms, while recurrent-to-output delay varies from 1 ms to 2048 ms. (b) Recurrent-to-output delay fixed at 1 ms, with output-to-recurrent delay varied similarly. Solid lines denote the mean; shaded areas indicate the standard deviation. Larger delays lead to higher losses, but the model maintains performance up to delays of 256 ms.

[148], SpiNNaker [149, 150], and ReckOn [151], our NEST-based implementation emphasizes generality, extensibility, and biological plausibility over hardware-specific optimization. As such, it may serve as a useful reference for studying and refining three-factor learning rules in a simulator-agnostic context.

This implementation is now integrated into a widely adopted simulator for large-scale spiking networks, and enables neuroscientifically grounded research, supports behavioral experiment modeling, and offers insights into biological learning processes and their applications in machine learning.



d_{ls} \ d	1	2	4	8	16	32	64	128	256	512	1024	2048
2048	36.99	47.14	42.57	45.13	38.61	39.42	44.82	39.52	56.53	56.76	84.35	123.51
1024	7.99	8.98	9.11	9.27	9.08	8.74	9.21	10.47	16.45	25.71	24.93	70.57
512	3.59	3.94	3.35	3.78	3.65	3.63	3.96	4.35	5.31	10.65	20.56	54.78
256	3.70	3.47	3.46	3.35	3.56	3.37	3.70	4.10	3.47	6.70	12.91	38.38
128	2.81	2.72	3.04	2.94	2.96	2.92	3.33	3.27	4.37	4.60	14.68	38.36
64	2.55	2.74	2.62	2.69	2.61	2.71	2.92	3.22	3.52	4.32	12.39	32.17
32	2.70	2.84	2.70	2.71	2.72	2.63	2.76	3.04	3.28	4.62	12.00	22.53
16	2.71	2.75	2.73	2.67	2.75	2.82	2.68	3.13	3.22	3.99	10.90	29.01
8	2.66	2.70	2.70	2.78	2.72	2.94	2.70	2.91	3.60	4.38	11.00	22.45
4	2.72	2.73	2.75	2.64	2.71	2.82	2.65	3.05	3.19	3.94	11.14	24.50
2	2.65	2.77	2.80	2.75	2.67	2.91	2.64	3.00	3.15	4.13	11.93	42.62
1	2.52	2.78	2.69	2.70	2.74	2.79	2.74	2.92	3.25	3.73	10.07	25.89

Figure 3.13: **Final MSE across combinations of recurrent-to-output and output-to-recurrent delays.** Mean-squared error (MSE) after training, averaged across 5 random seeds, for different combinations of recurrent-to-output delays (d) and output-to-recurrent delays (d_{ls}). Performance gradually degrades as total transmission delay increases, though the model remains robust up to a combined delay of approximately 512 ms.

CHAPTER 4

Biologically Inspired Continual Learning through Online Saliency Traces

In the previous chapter, we introduced *Eligibility Propagation* (e-prop), a biologically inspired learning rule that offers a plausible alternative to backpropagation through time (BPTT). While e-prop improves the biological plausibility of training recurrent spiking neural networks, it inherits a key limitation of BPTT: vulnerability to **catastrophic forgetting** (CF). CF arises when a neural network, trained sequentially on multiple tasks, fails to retain earlier knowledge while adapting to new tasks.

Biological organisms, by contrast, exhibit a remarkable ability to learn continuously in changing environments without suffering from CF [152]. This inspires the field of **continual learning** (CL), which aims to equip artificial systems with the same capability [152]. The central goal is to develop algorithms that preserve past knowledge while remaining sufficiently plastic to incorporate new information [153].

Two influential approaches to mitigating CF are **Elastic Weight Consolidation** (EWC) [22] and **Synaptic Intelligence** (SI) [23], which take inspiration from biological mechanisms like synaptic consolidation and long-term potentiation. However, despite their biological motivation, these methods rely on assumptions that limit their plausibility. For example, EWC requires a second pass over training data to estimate parameter importance, and both EWC and SI assume access to explicit task boundaries, which constitutes a condition rarely met in natural settings.

In this chapter, we propose a biologically inspired continual learning framework that draws on ideas from EWC, SI, e-prop, and neuromodulated plasticity. Central to our method is the concept of per-synapse **Online Saliency Traces**, which are updated using an exponential moving average of squared local gradients. Unlike previous approaches, our method requires no second data pass and no externally provided task boundaries.

To detect task transitions, we employ a lightweight **Bayesian Online Changepoint Detection (BOCD)** mechanism [154], implemented in a dedicated neuron. When a change is detected, this neuron broadcasts a modulatory signal that triggers synaptic consolidation by transferring the current saliency trace into long-term memory. This process is reminiscent of neuromodulator-driven plasticity observed in biological systems.

We evaluate our method on the Permuted Neuromorphic MNIST (PN-MNIST) dataset, demonstrating effective knowledge retention through sequences of tasks without explicit task boundary supervision. This chapter presents the theoretical foundations, algorithmic design, and experimental results of our approach, along with its place in the broader landscape of biologically inspired continual learning.

4.1 Biological Mechanisms of Continual Learning

Throughout their lives, animals exhibit the ability to learn continuously, adapting to new experiences and acquiring new skills [21]. This lifelong learning capacity is believed to be supported by various biological processes, including:

Neurogenesis. Neurogenesis refers to the formation of new neurons within the central nervous system. While mainly active during early development, it persists into adulthood. This ongoing generation of neurons contributes to the brain's ability to reorganize itself structurally, thus supporting the incorporation of new information and capabilities [155, 155].

Episodic Replay. Episodic replay is the repetition of neural activity patterns originally produced during waking experiences, which are later reactivated during rest or sleep. This mechanism is widely considered as essential for memory consolidation and illustrates how sleep facilitates the stabilization and integration of learned content [156, 157].

Metaplasticity. Metaplasticity refers to the brain's ability to regulate its own plasticity based on previous activity. This adaptive tuning of learning response plays a role in protecting existing memories and minimizing interference from new learning, thereby contributing to knowledge and memory consolidation [158, 159].

Neuromodulation. Neuromodulation involves chemical signals that modulate neural behaviors and learning dynamics. Several key neuromodulators are believed to contribute to continual learning. **Acetylcholine (ACh)** is associated with attentional control, particularly in conditions involving uncertainty or novel outcomes [160, 161]. **Noradrenaline (NA)** is released in response to surprising events and helps redirect attention toward salient stimuli [26]. **Dopamine** signals discrepancies between expected and actual outcomes, reinforcing behavior based on rewards or novelty. It also supports the transition from transient to lasting synaptic changes through long-term potentiation (LTP) [24, 25, 26].

These biological processes work together to provide flexible learning and memory retention, enabling the brain to maintain important prior experiences while at the same acquiring new knowledge [152].

Given that continual learning (CL) is a hallmark of biological intelligence [152], understanding and mitigating catastrophic forgetting (CF) is essential for developing biologically plausible artificial learning systems. Therefore, CL should be given special focus in the design of biologically inspired learning algorithms.

4.2 Continual Learning in Machine Learning

In machine learning, continual learning (CL) refers to a paradigm in which models learn new tasks or data distributions over time while preserving old knowledge. This capability is essential in scenarios where data is non-stationary and tasks may evolve or shift.

More formally, the continual learning problem can be defined as follows: given a sequence of M tasks $\mathbb{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_M\}$, where each task $\mathcal{T}_m$ is associated with a dataset of N_m input-output vector pairs $(\boldsymbol{x}, \boldsymbol{y})$, with $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$ and $\boldsymbol{y} \in \mathbb{R}^{d_{\text{out}}}$, denoted as

$$\mathscr{D}_m = \{(\boldsymbol{x}_{m,n}, \boldsymbol{y}_{m,n})\}_{n=1}^{N_m}, \tag{4.1}$$

and drawn from a task-specific distribution $P_m(\boldsymbol{x}, \boldsymbol{y})$, the objective is to find model parameters $\boldsymbol{\theta}^\star$ that minimize the cumulative expected loss across all tasks:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \sum_{m=1}^{M} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim P_m(\boldsymbol{x},\boldsymbol{y})} \left[\mathscr{L}_m(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})\right], \tag{4.2}$$

where $\mathscr{L}_m(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta})$ denotes the loss function for task $\mathcal{T}_m$. This formulation is adapted from [162, 163].

A central challenge in continual learning is that data from previous tasks is typically unavailable when learning new ones. As a result, the objective in Eq. (4.2) cannot be optimized directly. This limitation gives rise to the problem of **catastrophic forgetting (CF)**, where the model's performance on earlier tasks degrades significantly after training on new ones [153, 22]. CF occurs because parameter updates that reduce the loss on a new task may overwrite information relevant to prior tasks, rendering the updated parameters suboptimal for those earlier tasks [164].

The phenomenon of CF has led to research focused on developing strategies to mitigate it and facilitate CL in machine learning systems.

Remark on notation regarding time indices. In this chapter, we consider continual learning in the context of training spiking neural networks using eligibility propagation (e-prop). Since this framework operates on data with temporal structure, its notation relies on temporal indices. However, e-prop is not originally designed for multi-task scenarios or training schemes that separate learning into distinct phases. Given that we will address multiple tasks and phase-specific computations, we introduce here a precise convention regarding how to systematically locate any e-prop operation in time.

We denote the global training timeline using the index $t \in \mathbb{N}$. Task-specific boundaries are defined as follows:

- τ_m^{train}: start of training on task m,
- τ_m^{est}: start of the estimation phase on task m, required by methods such as Elastic Weight Consolidation (see Section 4.3).

Within each task m, we denote the beginning of the n-th sequence in each phase as:

- $\tau_{m,n}^{\text{train}}$: start of the n-th training sequence for task m,
- $\tau_{m,n}^{\text{est}}$: start of the n-th estimation sequence for task m.

These reference points are sufficient to locate any operation in time and will be used consistently throughout the chapter.

4.2.1 Mitigation strategies for CF in Machine Learning

There are three main families of approaches typically employed to mitigate catastrophic forgetting (CF) in machine learning: **replay-based methods**, **dynamic architectures**, and **regularization-based methods**. The best option frequently depends on the particular limitations, constraints and objectives of the artificial learning system, as each technique has unique benefits and trade-offs.

Replay-based methods mirror the process of *episodic replay* in the brain (see 4.1). These methods maintain access to data from previous tasks, either by storing real samples or generating synthetic ones, to allow the model to rehearse past knowledge while learning new information. Examples include *Experience Replay* [165, 162] and *Generative Replay* [166, 167], where the latter uses generative models to simulate data from earlier tasks.

Dynamic architectures are conceptually linked to *neurogenesis* (see 4.1). The key idea is to modify the model's structure as new tasks arrive, typically by expanding the network to accommodate new knowledge without interfering with previously learned representations. Notable examples include *Progressive Neural Networks* [168] and *Dynamic Expandable Networks* [169].

Regularization-based methods draw inspiration from *metaplasticity* (see 4.1). The key idea is to constrain parameter updates to preserve important knowledge from previous tasks. This is typically achieved by augmenting the loss function with a regularization term that penalizes changes to parameters considered critical. Representative approaches include *Elastic Weight Consolidation (EWC)* [22], *Synaptic Intelligence (SI)* [23], and *Memory Aware Synapses (MAS)* [170].

In this chapter, we focus on regularization-based methods, with particular attention to *Elastic Weight Consolidation (EWC)* and *Synaptic Intelligence (SI)*, due to their strong relevance to our proposed approach.

4.3 Elastic Weight Consolidation (EWC)

Elastic Weight Consolidation (EWC) [22] is a regularization-based approach to continual learning. The method is grounded in Bayesian reasoning and builds upon the observation that learning a new task should not significantly alter network parameters that are critical for solving previously learned tasks.

The central idea behind EWC is to treat learning sequential tasks as a Bayesian update. Consider a network that has been trained on task $\mathscr{A}$ by maximizing the likelihood $P(\mathscr{A} \mid \boldsymbol{\theta})$, or equivalently, by minimizing the negative log-likelihood loss $\mathscr{L}_{\mathscr{A}}(\boldsymbol{\theta}) = -\log P(\mathscr{A} \mid \boldsymbol{\theta})$, resulting in a posterior distribution $P(\boldsymbol{\theta} \mid \mathscr{A})$ over parameters. When a new task $\mathscr{B}$ arises, naively maxmizing $P(\mathscr{B} \mid \boldsymbol{\theta})$, would risk overwriting prior knowledge about $\mathscr{A}$.

Instead, EWC employs Bayes' rule to guide the learning of task $\mathscr{B}$ while incorporating knowledge from task $\mathscr{A}$:

$$P(\boldsymbol{\theta} \mid \mathscr{B}) = \frac{P(\mathscr{B} \mid \boldsymbol{\theta})\, P(\boldsymbol{\theta} \mid \mathscr{A})}{P(\mathscr{B} \mid \mathscr{A})}. \tag{4.3}$$

In this expression, the posterior distribution $P(\boldsymbol{\theta} \mid \mathscr{A})$ from task $\mathscr{A}$ serves as a prior for the new task $\mathscr{B}$, while the normalization factor $P(\mathscr{B} \mid \mathscr{A})$ (often termed the evidence) ensures a valid posterior distribution. The resulting posterior $P(\boldsymbol{\theta} \mid \mathscr{B})$ can then be optimized to effectively balance new learning with preservation of previous knowledge.

Taking the negative logarithm of this expression leads to a new loss function for task $\mathscr{B}$ that includes a regularization term derived from the posterior of task $\mathscr{A}$:

$$-\log P(\boldsymbol{\theta} \mid \mathscr{B}) = -\log P(\mathscr{B} \mid \boldsymbol{\theta}) - \log P(\boldsymbol{\theta} \mid \mathscr{A}) + \text{const.} \tag{4.4}$$

By dropping the constant term and introducing a scaling hyperparameter λ, EWC defines the following loss function:

$$\mathscr{L}_{\text{EWC}}(\boldsymbol{\theta}) = \mathscr{L}_{\mathscr{B}}(\boldsymbol{\theta}) + \lambda \mathscr{R}(\boldsymbol{\theta}), \tag{4.5}$$

where $\mathscr{L}_{\mathscr{B}}$ is the standard loss on the new task and $\mathscr{R}(\boldsymbol{\theta}) = -\log P(\boldsymbol{\theta} \mid \mathscr{A})$ acts as a regularizer that discourages deviation from parameters important to the previous task.

The posterior $P(\boldsymbol{\theta} \mid \mathscr{A})$ is generally intractable for deep neural networks. To make the computation feasible, EWC employs a *Laplace approximation* [171], which approximates the posterior as a Gaussian centered at the maximum-a-posteriori (MAP) estimate.

The posterior is expressed in exponential form as:

$$P(\boldsymbol{\theta} \mid \mathscr{A}) \propto \exp\left(-\xi(\boldsymbol{\theta})\right), \tag{4.6}$$

where $\xi(\boldsymbol{\theta}) = -\log P(\boldsymbol{\theta}, \mathscr{A})$ is the negative log-joint probability. A second-order Taylor expansion of $\xi(\boldsymbol{\theta})$ around the MAP estimate $\boldsymbol{\theta}^{\star}_{\mathscr{A}}$ gives:

$$\xi(\boldsymbol{\theta}) \approx \xi(\boldsymbol{\theta}^{\star}_{\mathscr{A}}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{\star}_{\mathscr{A}})^{\top} \mathscr{H}_{\mathscr{A}}(\boldsymbol{\theta}^{\star}_{\mathscr{A}})(\boldsymbol{\theta} - \boldsymbol{\theta}^{\star}_{\mathscr{A}}), \tag{4.7}$$

where $\mathscr{H}_{\mathscr{A}}(\boldsymbol{\theta}^{\star}_{\mathscr{A}})$ is the Hessian matrix of second derivatives of ξ, evaluated at the MAP estimate. Substituting this approximation into the posterior expression yields:

$$P(\boldsymbol{\theta} \mid \mathscr{A}) \approx C \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^{\star}_{\mathscr{A}})^{\top} \mathscr{H}_{\mathscr{A}}(\boldsymbol{\theta}^{\star}_{\mathscr{A}})(\boldsymbol{\theta} - \boldsymbol{\theta}^{\star}_{\mathscr{A}})\right), \tag{4.8}$$

where C is a normalization constant. This leads to the Gaussian approximation:

$$P(\boldsymbol{\theta} \mid \mathscr{A}) \approx \mathscr{N}\left(\boldsymbol{\theta}^{\star}_{\mathscr{A}}, \mathscr{H}^{-1}_{\mathscr{A}}(\boldsymbol{\theta}^{\star}_{\mathscr{A}})\right). \tag{4.9}$$

In practice, EWC replaces the Hessian with the Fisher Information Matrix (FIM), which is equivalent to the expected second derivative of the loss near a minimum. The FIM is preferred because it can be estimated using only first-order derivatives and is guaranteed to be positive semi-definite:

$$P(\boldsymbol{\theta} \mid \mathscr{A}) \approx \mathscr{N}\left(\boldsymbol{\theta}^{\star}_{\mathscr{A}}, \mathscr{F}^{-1}_{\mathscr{A}}(\boldsymbol{\theta}^{\star}_{\mathscr{A}})\right). \tag{4.10}$$

The Fisher Information Matrix captures the local curvature of the log-likelihood surface and serves as a measure of parameter sensitivity. It is defined as:

$$\mathscr{F}_{\mathscr{A}}(\boldsymbol{\theta}^{\star}_{\mathscr{A}}) := \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim P_{\mathscr{A}}(\boldsymbol{x},\boldsymbol{y})}\left[\left(\nabla_{\boldsymbol{\theta}} \log P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})\right)\left(\nabla_{\boldsymbol{\theta}} \log P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})\right)^{\top}\right]\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{\star}_{\mathscr{A}}}. \tag{4.11}$$

High curvature in a given direction indicates high sensitivity of the model's predictions to changes in the corresponding parameter, making such parameters crucial to retain. Accordingly, EWC penalizes deviations from $\boldsymbol{\theta}^{\star}_{\mathscr{A}}$ in proportion to their estimated importance, as encoded by the diagonal of the FIM.

To make Elastic Weight Consolidation practically applicable, the Fisher Information Matrix (FIM) used in the Laplace approximation must be estimated. Since the true data-generating distribution $P_{\mathscr{A}}(\boldsymbol{x},\boldsymbol{y})$ is unknown, it is typically approximated empirically using the training data from task $\mathscr{A}$. This yields the following empirical estimate:

$$P_{\mathscr{A}}(\boldsymbol{x},\boldsymbol{y}) \approx \frac{1}{N_{\mathscr{A}}}\sum_{n=1}^{N_{\mathscr{A}}} \delta(\boldsymbol{x}-\boldsymbol{x}_{\mathscr{A},n})\,\delta(\boldsymbol{y}-\boldsymbol{y}_{\mathscr{A},n}), \tag{4.12}$$

where δ denotes the Dirac delta function, and $N_{\mathscr{A}}$ is the number of training samples for task $\mathscr{A}$. Substituting this into the definition of the FIM results in the following approximation:

$$\mathscr{F}_{\mathscr{A}}(\boldsymbol{\theta}^{\star}_{\mathscr{A}}) \approx \frac{1}{N_{\mathscr{A}}}\sum_{n=1}^{N_{\mathscr{A}}} S(\boldsymbol{x}_{\mathscr{A},n}, \boldsymbol{y}_{\mathscr{A},n}, \boldsymbol{\theta}^{\star}_{\mathscr{A}})\, S(\boldsymbol{x}_{\mathscr{A},n}, \boldsymbol{y}_{\mathscr{A},n}, \boldsymbol{\theta}^{\star}_{\mathscr{A}})^{\top}, \tag{4.13}$$

here, $S(\boldsymbol{x}_{\mathscr{A},n}, \boldsymbol{y}_{\mathscr{A},n}, \boldsymbol{\theta}^{\star}_{\mathscr{A}})$ is called score function and is defined as:

$$S(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\theta}^{\star}_{\mathscr{A}}) := \nabla_{\boldsymbol{\theta}} \log P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})\big|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{\star}_{\mathscr{A}}}. \tag{4.14}$$

For scalability, EWC further approximates the FIM by retaining only its diagonal elements, yielding a per-parameter importance estimate:

$$\mathscr{F}_{\mathscr{A},i} \approx \frac{1}{N_{\mathscr{A}}}\sum_{n=1}^{N_{\mathscr{A}}} \left[S_i(\boldsymbol{x}_{\mathscr{A},n}, \boldsymbol{y}_{\mathscr{A},n}, \boldsymbol{\theta}^{\star}_{\mathscr{A}})\right]^2, \tag{4.15}$$

where $S_i(\boldsymbol{x}_{\mathscr{A},n}, \boldsymbol{y}_{\mathscr{A},n}, \boldsymbol{\theta}^{\star}_{\mathscr{A}})$ is the i-th component of the score function. This diagonal approximation assumes that the importance of each parameter can be treated independently, which is often a reasonable assumption in practice, especially for large neural networks where full FIM computation is computationally prohibitive.

As an additional practical trick used in in real-world applications to further improve efficiency, the Fisher estimate is computed over a mini-batch of N_{mini} training samples rather than over the full dataset:

$$\mathscr{F}_{\mathscr{A},i} \approx \frac{1}{N_{\text{mini}}}\sum_{n=1}^{N_{\text{mini}}} \left[S_i(\boldsymbol{x}_{\mathscr{A},n}, \boldsymbol{y}_{\mathscr{A},n}, \boldsymbol{\theta}^{\star}_{\mathscr{A}})\right]^2, \tag{4.16}$$

Following these approximations, the final EWC loss for training on a new task $\mathscr{B}$, while preserving performance on task $\mathscr{A}$, takes the form:

$$\mathscr{L}_{\text{EWC}}(\boldsymbol{\theta}) = \mathscr{L}_{\mathscr{B}}(\boldsymbol{\theta}) + \frac{\lambda}{2}\sum_i \mathscr{F}_{\mathscr{A},i}\left(\theta_i - \theta^{\star}_{\mathscr{A},i}\right)^2, \tag{4.17}$$

where $\mathscr{L}_{\mathscr{B}}$ is the standard loss for task $\mathscr{B}$, $\mathscr{F}_{\mathscr{A},i}$ is the estimated importance of parameter θ_i for task $\mathscr{A}$, $\theta^{\star}_{\mathscr{A},i}$ is the value of parameter θ_i after training on task $\mathscr{A}$, and λ is a hyperparameter controlling the strength of the regularization.

The gradient of the EWC loss with respect to a single parameter θ_i illustrates the mechanism at play:

$$\frac{\partial \mathscr{L}_{\text{EWC}}}{\partial \theta_i} = \frac{\partial \mathscr{L}_{\mathscr{B}}}{\partial \theta_i} + \lambda \mathscr{F}_{\mathscr{A},i}\left(\theta_i - \theta^{\star}_{\mathscr{A},i}\right). \tag{4.18}$$

This expression reveals that each parameter is individually pulled back toward its previous optimal value $\theta^{\star}_{\mathscr{A},i}$ in proportion to its estimated importance. Parameters considered crucial for task $\mathscr{A}$ (i.e., those with large $\mathscr{F}_{\mathscr{A},i}$) are penalized more heavily for deviating, while less important parameters remain free to adapt (see Figure 4.1). Notably, this independent treatment of parameters is not only computationally efficient but also biologically plausible: it enables synapses to preserve task-relevant information without requiring global coordination or awareness of other synapses, in line with the local nature of biological synaptic plasticity.

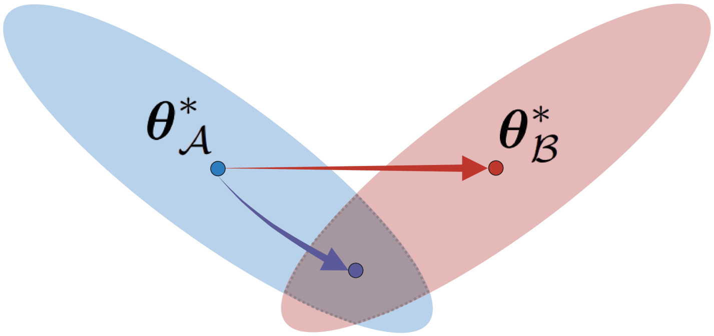

$\boldsymbol{\theta}^*_{\mathcal{A}}$
$\boldsymbol{\theta}^*_{\mathcal{B}}$

Figure 4.1: **Regularization Effect in Elastic Weight Consolidation.** $\boldsymbol{\theta}^{\star}_{\mathscr{A}}$ and $\boldsymbol{\theta}^{\star}_{\mathscr{B}}$ represent the optimal parameters for tasks $\mathscr{A}$ and $\mathscr{B}$, respectively. The blue-shaded region indicates the area of acceptable error for task $\mathscr{A}$, while the red-shaded region represents the same for task $\mathscr{B}$. The grey-shaded region shows the overlap where both tasks have acceptable errors. The red arrow illustrates the optimization path without regularization, leading to catastrophic forgetting as the final point lies outside the grey region. The purple arrow illustrates the optimization path with regularization, where the final point remains within the grey region, ensuring acceptable performance on both tasks. Figure adapted from [22].

4.3.1 Generalizing EWC to Multiple Tasks

The EWC framework naturally extends to scenarios involving more than two tasks [172]. Consider a sequence of tasks $\mathscr{T}_1, \mathscr{T}_2, \ldots, \mathscr{T}_{M+1}$, where the goal is to train on task $\mathscr{T}_{M+1}$ while retaining the knowledge acquired from previous tasks.

According to Bayes' rule, the incorporation of information about tasks $\mathscr{T}_1, \mathscr{T}_2, \ldots, \mathscr{T}_M$ leads to the posterior distribution over parameters $\boldsymbol{\theta}$ given the current task $\mathscr{T}_{M+1}$:

$$P\left(\boldsymbol{\theta} \mid \mathscr{T}_{M+1}\right) = \frac{P\left(\mathscr{T}_{M+1} \mid \boldsymbol{\theta}\right) P\left(\boldsymbol{\theta} \mid \mathscr{T}_1, \ldots, \mathscr{T}_M\right)}{P\left(\mathscr{T}_M \mid \mathscr{T}_1, \ldots, \mathscr{T}_M\right)}. \tag{4.19}$$

To implement this in practice, EWC accumulates the Fisher information across all previous tasks, producing a cumulative importance estimate for each parameter. The resulting multi-task EWC loss takes the form:

$$\mathscr{L}_{\text{EWC}}(\boldsymbol{\theta}) = \mathscr{L}_{M+1}(\boldsymbol{\theta}) + \frac{\lambda}{2} \sum_i \widehat{\mathscr{F}}_{M,i} \left(\theta_i - \theta^*_{M,i}\right)^2, \tag{4.20}$$

where the per-parameter cumulative Fisher information $\widehat{\mathscr{F}}_{M,i}$ is defined as:

$$\widehat{\mathscr{F}}_{M,i} = \sum_{m=1}^{M} \mathscr{F}_{m,i}, \tag{4.21}$$

and $\theta^\star_{M,i}$ denotes the parameter value after training on task $\mathscr{T}_M$.

4.3.2 Eligibility Propagation (e-prop) and EWC

In the context of spiking neural networks (SNNs), the eligibility propagation (e-prop) framework [16] provides a biologically plausible mechanism for computing gradients and updating synaptic weights. Using the techniques introduced in Chapter 3, the Fisher Information Matrix (FIM) can be estimated within the e-prop framework in an event-driven manner. Since the score function corresponds to the gradient of the log-likelihood, the mini-batch approximation in Eq. (4.16) can be efficiently computed using the local gradient estimation provided by e-prop during a dedicated post-training phase:

$$\mathscr{F}_{m,i} \approx \frac{1}{N_{\text{mini}}} \sum_{n=1}^{N_{\text{mini}}} \left(\sum_{t=\tau^{\text{est}}_{m,n}}^{\tau^{\text{est}}_{m,n+1}-1} L_i^t \, e_i^t \right)^2, \tag{4.22}$$

where L_i^t is the learning signal at time step t associated with parameter i, and e_i^t is the corresponding eligibility trace.[1] To ensure that parameters remain fixed during this phase, the learning rate is set to zero ($\eta = 0$) (see Algorithm 1).

Algorithm 1 Eligibility Propagation (e-prop) and EWC

```
Require: task sequence ${\mathscr{T}_m}_{m=1}^{M}$, regularization strength $\lambda$, base learning rate (LR) $\eta_0$
$\boldsymbol{\theta} \leftarrow$ random init, $\boldsymbol{\theta}^\star \leftarrow \mathbf{0}$, $\widehat{\mathscr{F}} \leftarrow \mathbf{0}$
for $m \leftarrow 1$ to $M$ do
    (train on $\mathscr{T}_m$) minimise w/ e-prop (LR $\leftarrow \eta_0$)
        $$\mathscr{L}_m(\boldsymbol{\theta}) + \frac{\lambda}{2}\sum_i \widehat{\mathscr{F}}_i\big(\theta_i - \theta_i^\star\big)^2$$
    (estimate Fisher) (LR $\leftarrow 0$)
    $\mathscr{F} \leftarrow \mathbf{0}$
    for $n \leftarrow 1$ to $N_{\text{mini}}$ do
        run sequence $(x_{m,n}, y_{m,n})$
        $\mathscr{F}_i \mathrel{+}= \left(\sum_{t=\tau^{\text{est}}_{m,n}}^{\tau^{\text{est}}_{m,n}-1} L_i^t\, e_i^t\right)^2 \quad \forall i$
    end for
    $\widehat{\mathscr{F}} \leftarrow \widehat{\mathscr{F}} + \mathscr{F}\,/\,N_{\text{mini}}$
    $\boldsymbol{\theta}^\star \leftarrow \boldsymbol{\theta}$
end for
```

4.4 Synaptic Intelligence (SI)

Synaptic Intelligence (SI) [23] is a regularization-based method for continual learning that mitigates catastrophic forgetting in a biologically inspired manner. In contrast to EWC, which requires a second pass over the dataset to estimate the Fisher Information, SI computes parameter importance *online* during training.

The key idea behind SI is to quantify the contribution of each parameter θ_i to the overall reduction in loss throughout the training of a task. The total change in loss for a task $\mathscr{T}_m$ between the end of the previous task $\mathscr{T}_{m-1}$ and end of training on the current one $\mathscr{T}_m$ can be expressed as the sum of individual parameter contributions:

[1] In standard e-prop notation, a synapse is identified by a pair of indices ji, where j refers to the postsynaptic neuron and i to the presynaptic neuron. Accordingly, the learning signal is denoted L_j^t, and the eligibility trace e_{ji}^t. For simplicity, we adopt a single-index notation i in this chapter to refer directly to individual parameters.

$$\mathscr{L}_m\left(\boldsymbol{\theta}_m^\star\right) - \mathscr{L}_m\left(\boldsymbol{\theta}_{m-1}^\star\right) = -\sum_i \omega_{m,i}, \tag{4.23}$$

where each contribution $\omega_{m,i}$ is given by:

$$\omega_{m,i} = \int_{\tau_m^{\text{train}}}^{\tau_{m+1}^{\text{train}}} \frac{\partial \mathscr{L}(t)}{\partial \theta_i} \cdot \theta_i'(t)\, dt, \tag{4.24}$$

where $\theta_i'(t) = \frac{d\theta_i}{dt}$ is the temporal derivative of the parameter. In discrete form, the contribution is approximated as a sum over training steps:

$$\omega_{m,i} = \sum_{t=\tau_m^{\text{train}}}^{\tau_{m+1}^{\text{train}}-1} \frac{\partial \mathscr{L}(t)}{\partial \theta_i} \cdot \delta_i(t), \tag{4.25}$$

where $\delta_i(t) = \theta_i(t+1) - \theta_i(t)$ is the update to parameter θ_i at step t.

To normalize the estimate and render it dimensionless, SI defines the importance weight as:

$$\Omega_{m,i} = \frac{\omega_{m,i}}{\left(\Delta_{m,i}\right)^2 + \xi}, \tag{4.26}$$

where $\Delta_{m,i} = \theta_{m,i}^\star - \theta_{m-1,i}^\star$ is the total parameter change during training and ξ is a small positive constant introduced to avoid any potential divisions by zero.

Once the importance weights are computed, they are used to regularize the learning of a new task $\mathscr{T}_{m+1}$, resulting in the following SI loss function:

$$\mathscr{L}_{\text{SI}}(\boldsymbol{\theta}) = \mathscr{L}_{m+1}(\boldsymbol{\theta}) + \frac{\lambda}{2} \sum_i \Omega_{m,i} \left(\theta_i - \theta_{m,i}^\star\right)^2, \tag{4.27}$$

where $\theta_{m,i}^\star$ is the value of parameter θ_i at the end of training on task $\mathscr{T}_m$, and λ is a hyperparameter controlling the strength of regularization.

4.4.1 Generalizing SI to Multiple Tasks

Similar to EWC, Synaptic Intelligence can be extended to continual learning across multiple tasks. The generalization is heuristic and involves accumulating importance estimates over time. After training on task $\mathscr{T}_M$, the cumulative importance is updated as:

$$\widehat{\Omega}_{M,i} = \sum_{m=1}^{M} \Omega_{m,i}, \tag{4.28}$$

where $\Omega_{m,i}$ denotes the normalized contribution of parameter θ_i during task $\mathscr{T}_m$.

The updated cumulative importance weights are then used to define the regularized loss for task $\mathscr{T}_{M+1}$:

$$\mathscr{L}_{\mathrm{SI}}(\boldsymbol{\theta}) = \mathscr{L}_{M+1}(\boldsymbol{\theta}) + \frac{\lambda}{2}\sum_i \widehat{\Omega}_{M,i}\left(\theta_i - \theta^\star_{M,i}\right)^2, \tag{4.29}$$

where $\theta^\star_{M,i}$ is the value of parameter θ_i at the end of training on task $\mathscr{T}_M$.

4.4.2 Eligibility Propagation (e-prop) and SI

Since both SI and e-prop operate in an online fashion, their combination allows parameter importance to be estimated during training. Specifically, the per-parameter contribution $\omega_{m,i}$ in SI can be computed using the e-prop learning rule:

$$\omega_{m,i} = \sum_{n=1}^{N_m}\left[\left(\sum_{t=\tau^{\mathrm{train}}_{m,n}}^{\tau^{\mathrm{train}}_{m,n+1}-1} L_i^t\, e_i^t\right)\cdot\left(\theta_i\left(\tau^{\mathrm{train}}_{m,n+1}\right) - \theta_i\left(\tau^{\mathrm{train}}_{m,n}\right)\right)\right], \tag{4.30}$$

where L_i^t and e_i^t are eprop's learning signal and eligibility trace for time step t. The normalization and regularization steps proceed as in the standard SI formulation. An algorithmic outline of the SI method is provided in Algorithm 2.

4.5 Biological Plausibility of EWC and SI

Although Elastic Weight Consolidation (EWC) is inspired by neurobiological principles such as synaptic consolidation, several aspects of its implementation challenge its plausibility from a biological perspective [173]. Specifically, two limitations stand out:

- **EWC requires a second pass over the training data.** After the initial learning phase, a separate traversal of the dataset is necessary to estimate the diagonal of the Fisher Information Matrix. During this phase, no further synaptic updates occur, unlike in biological systems where learning is believed to proceed continuously and online.

- **EWC assumes explicit knowledge of task boundaries.** The method relies on a clear indication of when one task ends and another begins to trigger consolidation. In contrast, biological learning systems typically operate in environments without such neatly defined transitions, suggesting that this assumption may not hold in natural settings.

Algorithm 2 Eligibility Propagation (e-prop) and SI

```
Require: task sequence $\{\mathcal{T}_m\}_{m=1}^{M}$, regularisation strength $\lambda$
$\boldsymbol{\theta} \leftarrow$ random init, $\boldsymbol{\theta}^\star \leftarrow \mathbf{0}$, $\widehat{\boldsymbol{\omega}} \leftarrow \mathbf{0}$
for $m \leftarrow 1$ to $M$ do
    (train on $\mathcal{T}_m$) minimise w/ e-prop
        $$\mathcal{L}_m(\boldsymbol{\theta}) + \frac{\lambda}{2}\sum_i \widehat{\Omega}_i(\theta_i - \theta_i^\star)^2$$
    $\boldsymbol{\omega} \leftarrow \mathbf{0}$, $\boldsymbol{\Delta} \leftarrow \mathbf{0}$
    for $n \leftarrow 1$ to $N_m$ do
        run sequence $(x_{m,n}, y_{m,n})$
        $g_i = \sum_{t=\tau_{m,n}^{\text{train}}}^{\tau_{m,n+1}^{\text{train}}-1} L_i^t e_i^t \quad \forall i$
        $g_i^{\text{reg}} = \lambda \widehat{\Omega}_i(\theta_i - \theta_i^\star) \quad \forall i$
        $\delta_i = -\eta\left(g_i + g_i^{\text{reg}}\right) \quad \forall i$
        $\theta_i \mathrel{+}= \delta_i \quad \forall i$
        $\Delta_i \mathrel{+}= \delta_i \quad \forall i$
        $\omega_i \mathrel{+}= g_i\delta_i \quad \forall i$
    end for
    $\widehat{\Omega}_i \mathrel{+}= \omega_i/(\Delta_i^2 + \xi) \quad \forall i$
    $\boldsymbol{\theta}^\star \leftarrow \boldsymbol{\theta}$
end for
```

Synaptic Intelligence (SI) [173] addresses some of these issues by introducing a single-phase learning algorithm that accumulates synaptic importance on the fly during training. While SI increases biologically plausiblility, it too presents certain limitations:

- **SI requires an external task boundary signal.** Although consolidation occurs within a single phase, it is still assumed that the model receives an explicit signal to trigger the storage of importance weights. This assumption, like in EWC, is difficult to reconcile with the implicit and gradual learning processes observed in animals.

- **SI relies on detailed synaptic history.** To estimate parameter importance, SI integrates the product of instantaneous gradients and weight updates throughout training. This approach assumes that biological systems can track and preserve fine-grained, synapse-specific temporal information over extended periods, a premise we believe lacks clear neurobiological support.

4.6 Proposed Approach

We present a continual learning framework that mitigates the biological implausibility of Elastic Weight Consolidation (EWC) and Synaptic Intelligence (SI). Our central insight is that the quantities used to encode parameter importance, i.e., the Fisher information in EWC and the loss contributions in SI, have the potential to be framed as quantities playing a role analogous to long-term synaptic efficacies. Accordingly, we treat these *saliencies* as trainable variables and update them with the similar biologically motivated rules that govern the model parameters. The framework is founded on five assumptions:

A1 **Convergence.** For each task $\mathscr{T}_m$, the parameters converge to a local maximum-a-posteriori (MAP) solution:

$$\boldsymbol{\theta}^{(n)} \longrightarrow \boldsymbol{\theta}_m^\star = \arg\max_{\boldsymbol{\theta}} \log P\left(\boldsymbol{\theta} \mid \mathscr{T}_m\right), \qquad n \to \infty, \tag{4.31}$$

where $\boldsymbol{\theta}^{(n)}$ is the parameter vector after the n-th sample.

A2 **Score Smoothness.** Near convergence, the per-parameter score

$$S_i(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) := \frac{\partial}{\partial \Theta_i} \log P\left(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\Theta}\right)\bigg|_{\boldsymbol{\Theta}=\boldsymbol{\theta}} \tag{4.32}$$

is smooth and stable, i.e., exhibits low-variance fluctuations with respect to $\boldsymbol{\theta}$.

A3 **Stationarity.** Training samples for task $\mathscr{T}_m$ are drawn i.i.d. from a stationary distribution $P_m(\boldsymbol{x}, \boldsymbol{y})$:

$$(\boldsymbol{x}_{m,n}, \boldsymbol{y}_{m,n}) \sim_{\text{i.i.d.}} P_m(\boldsymbol{x}, \boldsymbol{y}), \qquad \forall n. \tag{4.33}$$

A4 **Shift Sensitivity.** When the data distribution changes, e.g. at task boundaries, internal statistics such as the mean or variance of the score exhibit a measurable shift.

A5 **Detectability.** These statistical shifts can be detected online and can therefore be used to trigger consolidation.

Based on these assumptions, our framework combines two complementary mechanisms:

- **Online Saliency Trace.** Guided by Assumptions **A1**-**A3**, we maintain a per-parameter *Online Saliency Trace*, implemented as an exponential moving average (EMA) of the squared score. This trace provides a continual, online estimate of parameter importance.

- **Novelty Signal.** Motivated by Assumptions **A4** & **A5**, we implement a *novelty neuron* able to emit a binary *novelty signal* whenever a distributional shift is detected. The signal implicitly marks task transitions and initiates consolidation.

4.6.1 Online Saliency Traces

Inspired by *eligibility traces* in biological neurons; i.e., transient markers of recent synaptic activity, we equip every parameter θ_i with an **online saliency trace**. Formally, the trace is an exponential moving average (EMA) of the squared score 4.32:

$$\begin{aligned} f_i^{(n)} &= (1-\beta)\sum_{l=0}^{n} \beta^{n-l} S_i\Big(\boldsymbol{x}_l, \boldsymbol{y}_l, \boldsymbol{\theta}^{(l)}\Big)^2 \\ &= \beta f_i^{(n-1)} + (1-\beta)\, S_i\big(\boldsymbol{x}_n, \boldsymbol{y}_n, \boldsymbol{\theta}^{(n)}\big)^2, \end{aligned} \tag{4.34}$$

where $0 < \beta < 1$ controls the decay rate and $f_i^{(n)}$ is the trace after the n-th sample. At regular intervals (e.g. after each mini-batch) we record

$$f_i^{(c)} = f_i^{(n)}, \qquad \theta_i^{(c)} = \theta_i^{(n)}, \tag{4.35}$$

providing snapshots for later consolidation.

4.6.1.1 Asymptotic Convergence of Online Saliency Traces to Fisher Information

Our core intuition is that the online saliency trace $f_i^{(n)}$ serves as an estimate of the importance of parameter θ_i, analogous to the diagonal elements of the Fisher Information Matrix (FIM). In this section, we formalize this idea by presenting a proposition that establishes the convergence of the online saliency trace to the corresponding diagonal entry of the FIM, under the assumptions previously described.

Before introducing and proving this proposition, we present some preliminary supporting results. We begin with a convergence theorem for the exponential moving average (EMA), which plays a central role in the construction of the saliency trace, followed by Slutsky's theorem, which will be key to establishing the convergence of the saliency trace to the Fisher Information.

Theorem 1 (Convergence of the Exponential Moving Average)**.** *Let $X_1, X_2, \ldots$ be a sequence of independent and identically distributed random variables with finite expectation $\mu = \mathbb{E}[X_n]$. Define the recursive sequence*

$$f_n = \beta f_{n-1} + (1-\beta) X_n, \qquad f_0 \in \mathbb{R},\ 0 < \beta < 1.$$

Then, as $\beta \to 1$ and $n \to \infty$, we have

$$f_n \xrightarrow{\mathsf{P}} \mu.$$

For the proof of Theorem 1, see Appendix 6.

Theorem 2 (Slutsky's Theorem [174])**.** *Let $X_1, X_2, \ldots$ and $Y_1, Y_2, \ldots$ be sequences of random variables. Suppose that*

$$X_n \xrightarrow{\mathsf{D}} X \quad \textit{and} \quad Y_n \xrightarrow{\mathsf{P}} a \quad \textit{as } n \to \infty,$$

where a is some constant. Then

$$\begin{aligned} X_n + Y_n &\xrightarrow{\mathsf{D}} X + a, \\ X_n - Y_n &\xrightarrow{\mathsf{D}} X - a, \\ X_n \cdot Y_n &\xrightarrow{\mathsf{D}} X \cdot a, \\ \frac{X_n}{Y_n} &\xrightarrow{\mathsf{D}} \frac{X}{a}, \quad \textit{for } a \neq 0, \end{aligned}$$

as $n \to \infty$.

Proposition 1 (Online Saliency Trace Convergence to Fisher)**.** *Under Assumptions **A1**-**A3** (convergence, score smoothness, stationarity), the saliency trace converges in probability to the diagonal element of the Fisher information matrix:*

$$f_i^{(n)} \xrightarrow{\mathsf{P}} \mathscr{F}_{m,i}, \quad \textit{as } n \to \infty,\ \beta \to 1, \tag{4.36}$$

where

$$\mathscr{F}_{m,i} \;=\; \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim P_m(\boldsymbol{x},\boldsymbol{y})} \left[\left(\frac{\partial}{\partial \theta_i} \log P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) \right)^2 \right]_{\boldsymbol{\theta} = \boldsymbol{\theta}_m^\star}. \tag{4.37}$$

Proof. By Assumption **A1**, there exists N_0 such that for all $n \geq N_0$:

$$\left\| \boldsymbol{\theta}^{(n)} - \boldsymbol{\theta}_m^\star \right\| < \delta$$

for an arbitrarily small $\delta > 0$. By score smoothness (Assumption **A2**), we conclude $S_i(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}^{(n)})^2$ is smooth as well and therefore locally Lipschitz continuous with Lipschitz constant L:

$$\left| S_i(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}^{(n)})^2 - S_i(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}_m^\star)^2 \right| \;\leq\; L \cdot \left\| \boldsymbol{\theta}^{(n)} - \boldsymbol{\theta}_m^\star \right\|.$$

This means that, for all $n \geq N_0$:

$$\left| S_i(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}^{(n)})^2 - S_i(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}_m^\star)^2 \right| \;\leq\; L \cdot \delta. \tag{4.38}$$

Now, define the sequence

$$X_n \;:=\; S_i\left(\mathbf{x}_n, \mathbf{y}_n, \boldsymbol{\theta}^{(n)}\right)^2 \;=\; Z_n + R_n,$$

where

$$Z_n := S_i\left(\mathbf{x}_n, \mathbf{y}_n, \boldsymbol{\theta}^{\star}_m\right)^2, \quad \text{and} \quad |R_n| \;\le\; L\cdot\delta \quad (n \ge N_0).$$

Because the samples $(\mathbf{x}_n, \mathbf{y}_n)$ are drawn *i.i.d.* from P_m (Assumption **A3**), the variables $\{Z_n\}_{n\ge N_0}$ form an *i.i.d.* sequence with finite mean:

$$\mathbb{E}[Z_n] = \mathscr{F}_{m,i}.$$

Unrolling the recurrence for the saliency trace yields:

$$f_i^{(n)} = (1-\beta)\sum_{k=0}^{n}\beta^{n-k}X_k = g_n + h_n,$$

where we define the truncated sum

$$g_n := (1-\beta)\sum_{k=0}^{n}\beta^{n-k}Z_k$$

and the residual

$$h_n := f_i^{(n)} - g_n = (1-\beta)\sum_{k=0}^{n}\beta^{n-k}R_k.$$

Because $\{Z_k\}$ is an *i.i.d.* sequence with mean $\mathscr{F}_{m,i}$, by Theorem 1 the exponential moving average converges in probability to this mean:

$$g_n \xrightarrow{\mathsf{P}} \mathscr{F}_{m,i}$$

provided that both $n \to \infty$ and $\beta \to 1$.

For $n \ge N_0$, we have:

$$|h_n| \;=\; \left|(1-\beta)\sum_{k=0}^{n}\beta^{n-k}R_k\right| \;\le\; (1-\beta)\sum_{k=0}^{n}\beta^{n-k}|R_k|,$$

now, using the inequality (4.38), we obtain:

$$|h_n| \;\le\; L\cdot\delta\,(1-\beta)\sum_{k=0}^{n}\beta^{n-k}.$$

Let $j = n-k$, so as k goes from 0 to n, j goes from n to 0. This rewrites the sum:

$$\sum_{k=0}^{n}\beta^{n-k} = \sum_{j=0}^{n}\beta^{j}$$

We can upper bound the partial sum with a full geometric series:

$$\sum_{j=0}^{n} \beta^j \le \sum_{j=0}^{\infty} \beta^j = \frac{1}{1-\beta},$$

since $0 < \beta < 1$. Therefore:

$$|h_n| \le L \cdot \delta.$$

Since δ can be made arbitrarily small by choosing N_0 sufficiently large, it follows that

$$h_n \xrightarrow{\mathsf{P}} 0 \quad \text{as } n \to \infty.$$

Combining the decomposition $f_i^{(n)} = g_n + h_n$ with the convergence results above:

$$g_n \xrightarrow{\mathsf{P}} \mathscr{F}_{m,i}, \quad h_n \xrightarrow{\mathsf{P}} 0 \implies f_i^{(n)} \xrightarrow{\mathsf{P}} \mathscr{F}_{m,i},$$

by Slutsky's Theorem. ■

Practical Implications. As the proposed online saliency traces provide a parameter importance measure closely related to the Fisher Information that is computable during training, the need for a second pass over the training data is eliminated. Moreover, because the EMA naturally discounts stale information, the trace *never needs to be reset* between tasks, allowing parameter importance to accumulate in a task-agnostic, biologically plausible manner.

4.6.2 Novelty Neuron and Novelty Signal

Complementing the saliency trace, we attach a **novelty neuron** that continuously monitors summary statistics of the network output. All output units project presynaptically to this neuron, enabling it to sense distributional shifts in real time.

Binary novelty signal. At every time step t the neuron emits a binary *novelty signal*

$$N_t = \begin{cases} 1, & \text{if a distribution shift is detected,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.39}$$

When $N_t = 1$, the signal is broadcast to the entire network and triggers *consolidation* of the current saliency snapshot into a long-term parameter-importance:

$$F_i \leftarrow F_i + N_t f_i^{(c)}(t), \tag{4.40}$$

where F_i accumulates long-term importance for parameter θ_i and $f_i^{(c)}(t)$ denotes the most recent snapshot of the online saliency trace. This process effectively computes the per-parameter cumulative Fisher information $\widehat{\mathscr{F}}_{M,i}$ defined in Equation 4.21 for multiple tasks.

Detection mechanism. We instantiate the novelty neuron with *Bayesian Online Changepoint Detection* (BOCD) [154].

4.6.3 Bayesian Online Changepoint Detection

Bayesian Online Changepoint Detection (BOCD) [154] is a principled statistical framework for detecting, in real time, changes in the generative distribution of a sequence of observations. This kind of detection is particularly relevant in domains such as finance, quality control, and environmental monitoring, where abrupt shifts in data patterns often signal significant underlying events or regime changes [154].

The method considers a sequence of observations

$$\boldsymbol{x}_{1:t} = (x^1, x^2, \dots, x^t), \tag{4.41}$$

which may be segmented into contiguous, non-overlapping partitions (see Figure 4.2a). The boundaries between this partitions, referred to as *changepoints*, mark transitions between statistically distinct regimes. Within each segment, observations are assumed to be independent and identically distributed (*i.i.d.*) from a fixed but unknown distribution.

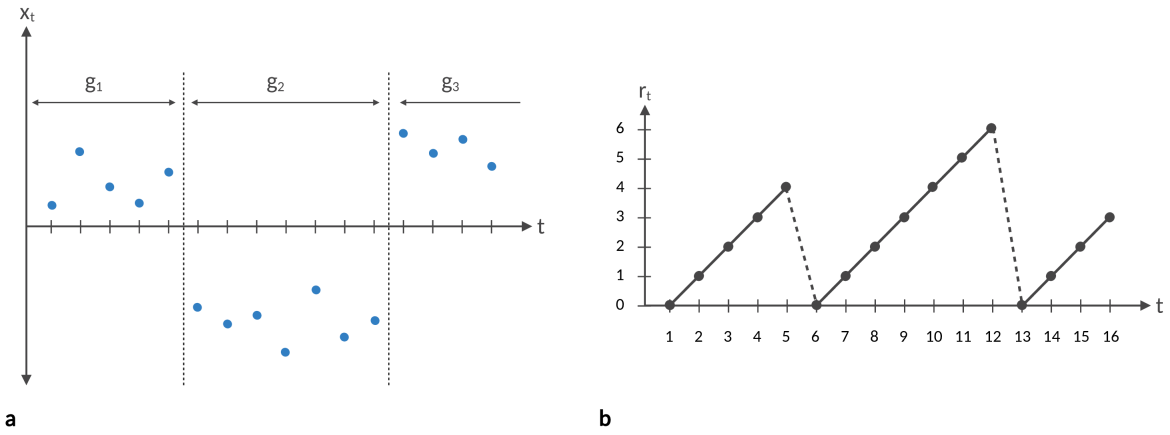


Figure 4.2: **Changepoint Detection and Run Length in BOCD.** (a) The sequence of observations is divided into segments, each characterized by a distinct distribution. The goal is to identify the changepoints τ_t that separate these segments. (b) The run length r_t at time t is defined as the number of observations since the last changepoint. Figure adapted from [154].

The goal of BOCD is to identify, as new data arrives, the time indices t at which the underlying distribution changes. To do so, the method is updated recursively with each new observation, which allows online detection of distributional shifts without the need to store or reprocess the entire history of observations.

4.6.3.1 Recursive Run Length Estimation

A central concept in Bayesian Online Changepoint Detection (BOCD) is the **run length** r_t, which denotes the number of observations since the last changepoint (see Figure 4.2b). Formally, the run length at time t is defined as:

$$r_t = t - \tau_t, \tag{4.42}$$

where τ_t is the time of the most recent changepoint at or before time t. The run length takes values in the range $0 \le r_t \le t$.

The goal of BOCD is to compute the posterior distribution over the run length given the observed data:

$$P(r_t \mid \boldsymbol{x}_{1:t}). \tag{4.43}$$

This posterior is obtained by first computing the joint distribution $P(r_t, \boldsymbol{x}_{1:t})$, from which the posterior follows via normalization:

$$P(r_t \mid \boldsymbol{x}_{1:t}) = \frac{P(r_t, \boldsymbol{x}_{1:t})}{P(\boldsymbol{x}_{1:t})}. \tag{4.44}$$

To efficiently compute the joint distribution, BOCD employs a recursive update derived from the chain rule and marginalization over the previous run length r_{t-1}:

$$\begin{aligned} P(r_t, \boldsymbol{x}_{1:t}) &= \sum_{r_{t-1}} P(r_t, r_{t-1}, \boldsymbol{x}_{1:t}) \\ &= \sum_{r_{t-1}} P(x_t, r_t \mid r_{t-1}, \boldsymbol{x}_{1:t-1}) \cdot P(r_{t-1}, \boldsymbol{x}_{1:t-1}) \\ &= \sum_{r_{t-1}} P(x_t \mid r_{t-1}, \boldsymbol{x}_t^{(r)}) \cdot P(r_t \mid r_{t-1}) \cdot P(r_{t-1}, \boldsymbol{x}_{1:t-1}), \end{aligned} \tag{4.45}$$

where $\boldsymbol{x}_t^{(r)}$ denotes the set of observations associated with the current run length r_{t-1}, i.e., the most recent r_{t-1} data points before x_t.

This recursion decomposes into three interpretable components:

- $\boldsymbol{P(r_t \mid r_{t-1})}$: the **changepoint prior**, which defines the probability of transitioning from run length r_{t-1} to r_t.
- $\boldsymbol{P(x_t \mid r_{t-1}, x_t^{(r)})}$: the **predictive model**, which defines the likelihood of observing x_t conditioned on the current partition of data and its length. This depends on the assumed predictive model used for these partitions (e.g., exponential family models with conjugate priors).

- $\boldsymbol{P(r_{t-1}, x_{1:t-1})}$: the **joint distribution at the previous step**, which has already been computed at time $t-1$. This enables the recursive structure of the algorithm.

Both the changepoint prior and the predictive model are computed or updated at each time step. Since the last factor, $P(r_{t-1}, \boldsymbol{x}_{1:t-1})$, is carried forward from the previous iteration, this recursive formulation gives BOCD online capabilities.

4.6.3.2 Changepoint Prior

The **changepoint prior** $P(r_t \mid r_{t-1})$ plays a crucial role in the BOCD framework. It defines the probability of observing a changepoint at time t, given the current run length r_{t-1}. This prior is derived from the **hazard function** $H(\tau)$, which specifies the probability that a changepoint occurs exactly after τ time steps. Formally, the prior over run lengths is given by:

$$P(r_t \mid r_{t-1}) = \begin{cases} H(r_{t-1}+1), & \text{if } r_t = 0, \\ 1 - H(r_{t-1}+1), & \text{if } r_t = r_{t-1}+1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.46}$$

The hazard function itself is derived from the inter-changepoint interval distribution $P_{\text{gap}}(\tau)$, which encodes the probability that a changepoint occurs exactly τ steps after the previous one. The hazard function is defined as:

$$H(\tau) = \frac{P_{\text{gap}}(\tau)}{\sum_{t=\tau}^{\infty} P_{\text{gap}}(t)} = \frac{P_{\text{gap}}(\tau)}{\Pr(T \geq \tau)}, \tag{4.47}$$

where the denominator is the survival function of the gap distribution. A common and analytically convenient choice for $P_{\text{gap}}(\tau)$ is the geometric distribution [175]:

$$P_{\text{gap}}(\tau) = (1-p)^{\tau-1} p, \tag{4.48}$$

which models the time until the first success in a series of Bernoulli trials with success probability $p = \frac{1}{\lambda}$. This yields a constant hazard function:

$$H(\tau) = \frac{(1-p)^{\tau-1} p}{(1-p)^{\tau-1}} = p = \frac{1}{\lambda}. \tag{4.49}$$

This result reflects the **memoryless property** of the geometric distribution: the probability of a changepoint occurring at the next time step is independent of the time elapsed since the last changepoint. Substituting the constant hazard into the changepoint prior:

$$P(r_t \mid r_{t-1}) = \begin{cases} \frac{1}{\lambda}, & \text{if } r_t = 0, \\ 1 - \frac{1}{\lambda}, & \text{if } r_t = r_{t-1} + 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.50}$$

4.6.3.3 Predictive Model

A useful approach is to model the data within each partitions employing a distribution belonging to the exponential family (e.g., Gaussian, Poisson, Bernoulli or other similar distributions). This allows to express the probability of an observation $\boldsymbol{x}$ given natural parameters $\boldsymbol{\eta}$ in a form that is amenable to efficient computation and inference. The probability density (or mass) function for an observation $\boldsymbol{x}$ given natural parameters $\boldsymbol{\eta}$ takes the form:

$$P(\boldsymbol{x} \mid \boldsymbol{\eta}) \ = \ h(\boldsymbol{x}) \exp\left(\boldsymbol{\eta}^\top \boldsymbol{U}(\boldsymbol{x}) - A(\boldsymbol{\eta})\right), \tag{4.51}$$

where $h(\boldsymbol{x})$ is the base measure, $\boldsymbol{U}(\boldsymbol{x})$ is the vector of sufficient statistics, $\boldsymbol{\eta}$ is the natural parameter vector, and $A(\boldsymbol{\eta})$ is the log-partition function (normalizer), defined as:

$$A(\boldsymbol{\eta}) = \log \int h(\boldsymbol{x}) \exp\left(\boldsymbol{\eta}^\top \boldsymbol{U}(\boldsymbol{x})\right) d\boldsymbol{x}. \tag{4.52}$$

Given this generative model, the predictive distribution for the next observation x^{t+1}, conditioned on the current run length r^t, is obtained by marginalizing over the latent parameters $\boldsymbol{\eta}$ using the posterior distribution inferred from the data segment associated with r^t:

$$P\left(x^{t+1} \mid r^t\right) \ = \ \int P\left(x^{t+1} \mid \boldsymbol{\eta}\right) P\left(\boldsymbol{\eta} \mid r^t, \boldsymbol{x}^t_{(r)}\right) d\boldsymbol{\eta}. \tag{4.53}$$

Assuming a conjugate prior for $\boldsymbol{\eta}$, the posterior can be represented in terms of updated hyperparameters. The predictive distribution then simplifies to:

$$\pi^t_{(t)} = P\left(x^t \mid \nu^t_{(r)}, \boldsymbol{\chi}^t_{(r)}\right), \tag{4.54}$$

where the updated hyperparameters $\nu^t_{(r)}$ and $\boldsymbol{\chi}^t_{(r)}$ are defined as:

$$\begin{aligned} \nu^t_{(r)} &= \nu_{\text{prior}} + r^t, \\ \boldsymbol{\chi}^t_{(r)} &= \boldsymbol{\chi}_{\text{prior}} + \sum_{t' \in r^t} \boldsymbol{U}\left(x^{t'}\right). \end{aligned} \tag{4.55}$$

This sufficient statistics accumulate the information from the current data segment, enabling recursive and efficient updates of the posterior and the predictive distribution at each time step.

Gaussian Likelihood with Unknown Mean In our experiments we focus on one of the simplest members of the exponential family: the univariate Gaussian distribution with *unknown mean* μ and *known variance* σ^2 [176]. In this case the data likelihood is given by:

$$x_1, \dots, x_n \mid \mu \sim \mathcal{N}(\mu, \sigma^2), \tag{4.56}$$

As the mean is *unknown*, a conjugate Gaussian prior over it is placed:

$$\mu \sim \mathcal{N}(\mu_0, \sigma_0^2), \tag{4.57}$$

After observing a segment of n data points $\{x_1, \dots, x_n\}$, the posterior distribution over μ remains Gaussian, with updated parameters:

$$\mu_n = \sigma_n^2 \left(\frac{\mu_0}{\sigma_0^2} + \frac{n\bar{x}}{\sigma^2} \right), \quad \sigma_n^2 = \left(\frac{1}{\sigma_0^2} + \frac{n}{\sigma^2} \right)^{-1}, \tag{4.58}$$

where $\bar{x}$ is the sample mean of the segment. The predictive distribution for the next observation x_{n+1}, marginalizing over μ, is also Gaussian:

$$x_{n+1} \mid x_1, \dots, x_n \sim \mathcal{N}(\mu_n, \sigma^2 + \sigma_n^2), \tag{4.59}$$

with the variance composed of the known observation noise σ^2 and the posterior uncertainty σ_n^2 in the estimate of the mean.

This form makes it easy for the BOCD method to update the necessary predicted probabilities and sufficient statistics in a recursive way. We can save the weighted sum of prior observations for each possible run length r^t to update μ_n and σ_n^2 incrementally without having to keep the whole data history.

4.6.3.4 Changepoint Detection

In BOCD changepoints are detected by monitoring $P(r_t = 0 \mid x_{1:t})$. If this probability exceeds a predefined threshold, it indicates that the current observation x_t is significantly different from the previous observations, suggesting a changepoint has occurred (see Algorithm 3).

4.6.4 Saliency Trace and Novelty Neuron Integration

We conclude the methodological exposition by summarizing how the two core components, the online saliency trace and the novelty neuron, interact within the overall continual learning framework.

Algorithm 3 Bayesian Online Changepoint Detection (BOCD)

```
Input: hazard function $H(\cdot)$, data stream $x_{1:T}$
Initialize:
  $P(r_0 = 0, x_{1:0}) \leftarrow 1$
  initialize parameter-posterior hyperparameters for run length $r = 0$
for $t = 1$ to $T$ do
    for $r = 0$ to $t - 1$ do
        compute predictive probability
            $\pi_t^{(r)} = P(x_t \mid r_{t-1} = r, x_{t-r:t-1})$
        Growth step:
            $P(r_t = r+1, x_{1:t}) = \pi_t^{(r)} [1 - H(r+1)] P(r_{t-1} = r, x_{1:t-1})$
    end for
    Changepoint step:
        $P(r_t = 0, x_{1:t}) = \sum_{r=0}^{t-1} \pi_t^{(r)} H(r+1) P(r_{t-1} = r, x_{1:t-1})$
    Compute run-length posterior:
        $P(r_t = r \mid x_{1:t}) = \frac{P(r_t = r, x_{1:t})}{\sum_{r=0}^{t} P(r_t = r, x_{1:t})} \quad \forall r = 0, \ldots, t$
    Update parameter posteriors:
      for each $r = 0, \ldots, t$, update conjugate-prior hyperparameters using $x_t$
end for
Output: for each $t$, the posterior $P(r_t \mid x_{1:t})$
```

As shown in Figure 4.3, each model parameter θ_i is augmented with an **online saliency trace** $f_i^{(n)}$, implemented as an exponential moving average (EMA) of the squared score (Equation 4.32). This trace continuously estimates the parameter's importance in an online and biologically plausible fashion, converging asymptotically to the Fisher information under reasonable assumptions.

In parallel, a dedicated **novelty neuron** monitors global network activity to detect distributional shifts in the output statistics. This neuron is instantiated using Bayesian Online Changepoint Detection (BOCD), which maintains a posterior over run lengths and emits a binary *novelty signal* $N_t \in \{0, 1\}$ in real time.

The interaction between these two mechanisms is orchestrated as follows:

1. During training, the saliency trace $f_i^{(n)}$ is updated continuously for each parameter (Equation 4.34).

2. The novelty neuron concurrently evaluates whether a shift in the data distribution has occurred, based on BOCD inference (Algorithm 3).

3. Upon detecting a changepoint (i.e., when $N_t = 1$), the network triggers *consolidation*: the most recent saliency snapshot $f_i^{(c)}$ is integrated into the long-term importance estimate F_i (Equation 4.40).

This integration strategy enables the system to accumulate task-agnostic parameter importance in an online, biologically grounded manner, without requiring access to past data or explicit task boundaries.

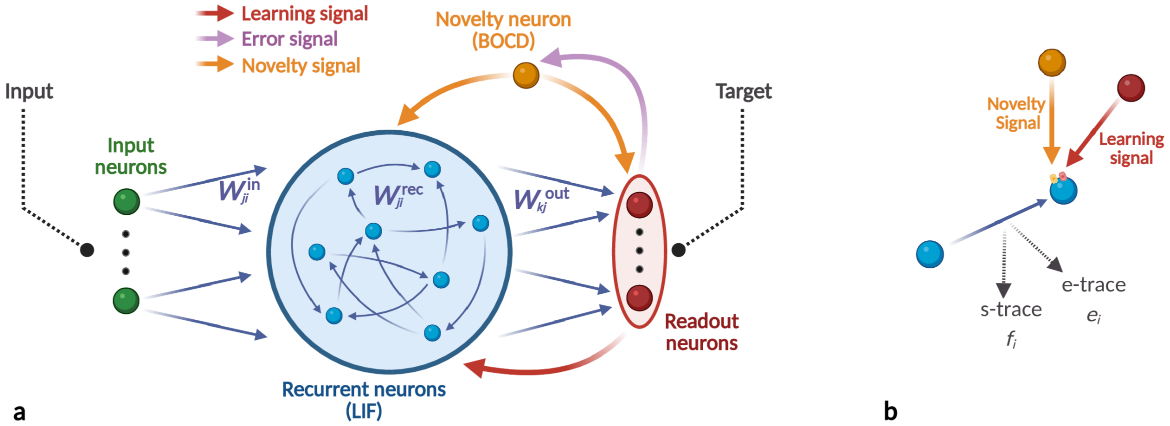


Figure 4.3: **Extended e-prop architecture with Online Saliency Trace and Novelty Neuron.** (a) Standard e-prop architecture (see Figure 3.1) with the addition of a saliency trace for each parameter, updated online during training. The novelty neuron monitors the output statistics and triggers consolidation when a distributional shift is detected. (b) Schematic illustrating the interaction of key components at a single synapse during learning and consolidation: the eligibility and saliency traces, and the modulatory learning and novelty signals.

4.7 Results

To evaluate the effectiveness of our proposed continual learning framework, we conduct experiments on the **Permuted Neuromorphic MNIST** (PN-MNIST) benchmark. For a detailed description of the base dataset, see subsubsection 3.5.1.3. The network configuration, simulation parameters, and hyperparameters are provided in the appendix Tables 7.11 and 7.11.

In the permuted setup, each task corresponds to a different fixed random permutation of the original N-MNIST input pixels. This results in a sequence of tasks that are mutually decorrelated at the input level, requiring the model to learn distinct mappings for each task while preserving knowledge from earlier ones (see Figure 4.4).

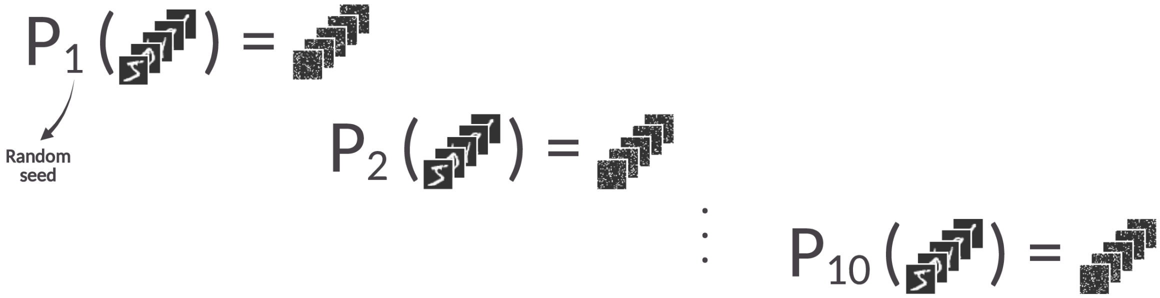


Figure 4.4: **Schematic of Permuted Neuromorphic MNIST (PN-MNIST) Tasks.** In this benchmark, each task is derived from the N-MNIST dataset by applying a different random permutation to the input pixels. This enforces task-specific input statistics, presenting a challenging scenario for continual learning and catastrophic forgetting.

We train models sequentially using e-prop on 10 permuted tasks and compare the performance of our method (**Online Saliency Traces + Novelty Neuron**) against three baselines:

- **No Regularization:** A standard e-prop model trained without any CL strategy,
- **Elastic Weight Consolidation** (EWC) [22] (see section 4.3),
- **Synaptic Intelligence** (SI) [23] (see section 4.4).

Figures 4.5-4.7 provide a task-by-task breakdown of training and retention performance for each method. Each subplot represents one task. The thick colored line corresponds to the accuracy on the current task, while thin gray lines show the accuracy trajectories of the other tasks for reference. Shaded areas indicate the standard deviation across 10 independent runs.

4.7.1 Average Accuracy (ACC)

To provide a compact yet informative summary of the model's ability to retain knowledge across tasks, we compute the **Average Accuracy (ACC)** [177] after training on each task. This

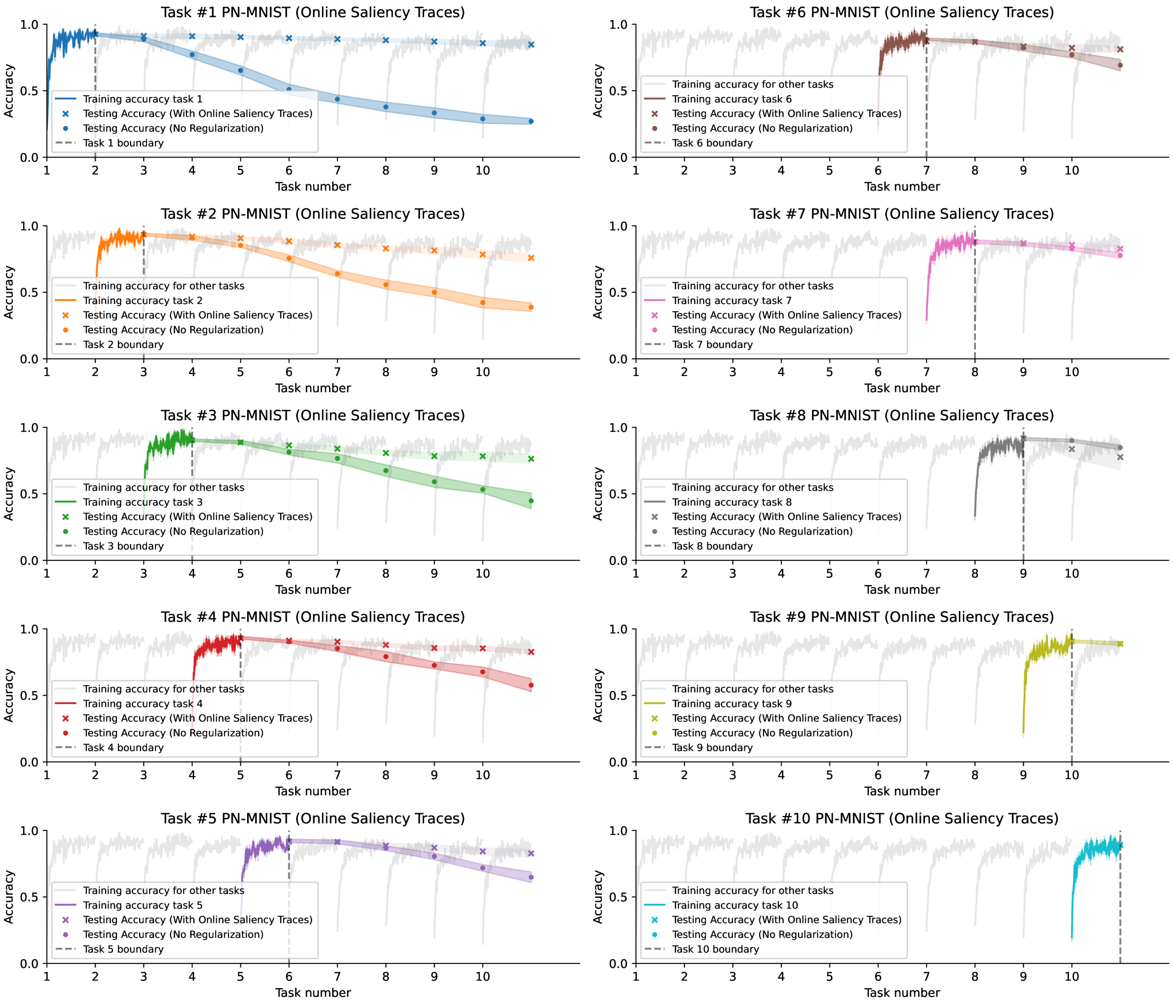


Figure 4.5: **Continual Learning Performance of Online Saliency Traces + Novelty Neuron.** Per-task training and test accuracy for our proposed method on PN-MNIST. The model maintains high accuracy across previously learned tasks, showing strong resistance to catastrophic forgetting.

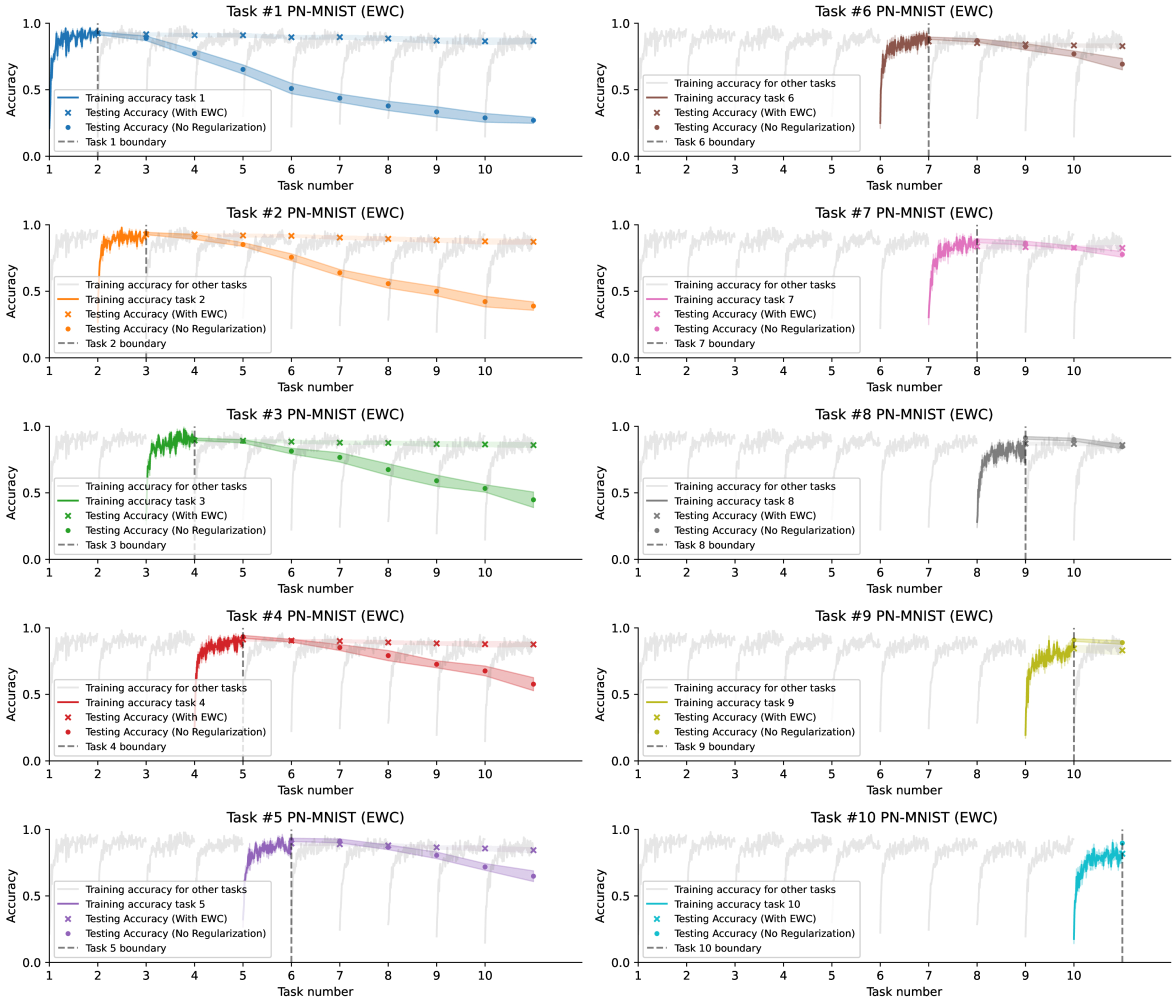


Figure 4.6: **Continual Learning Performance of Elastic Weight Consolidation (EWC).** EWC applies a regularization penalty based on parameter importance (Fisher Information).

metric captures how well the model preserves performance on previously learned tasks as it progresses through the sequence. Formally, for task k, the ACC is defined as:

$$\mathrm{ACC}(k) = \frac{1}{k}\sum_{i=1}^{k} A_{k,i}, \tag{4.60}$$

where $A_{k,i}$ denotes the test accuracy on task i after completing training on task k. This provides a overall measure of the model's cumulative retention ability up to that point in training.

Figure 4.8 compares the ACC trajectories of our method (in red) against EWC (blue), SI (green), and a non-regularized baseline (black). The non-regularized baseline suffers from sharp declines in ACC as new tasks are introduced, indicating severe forgetting. In con-

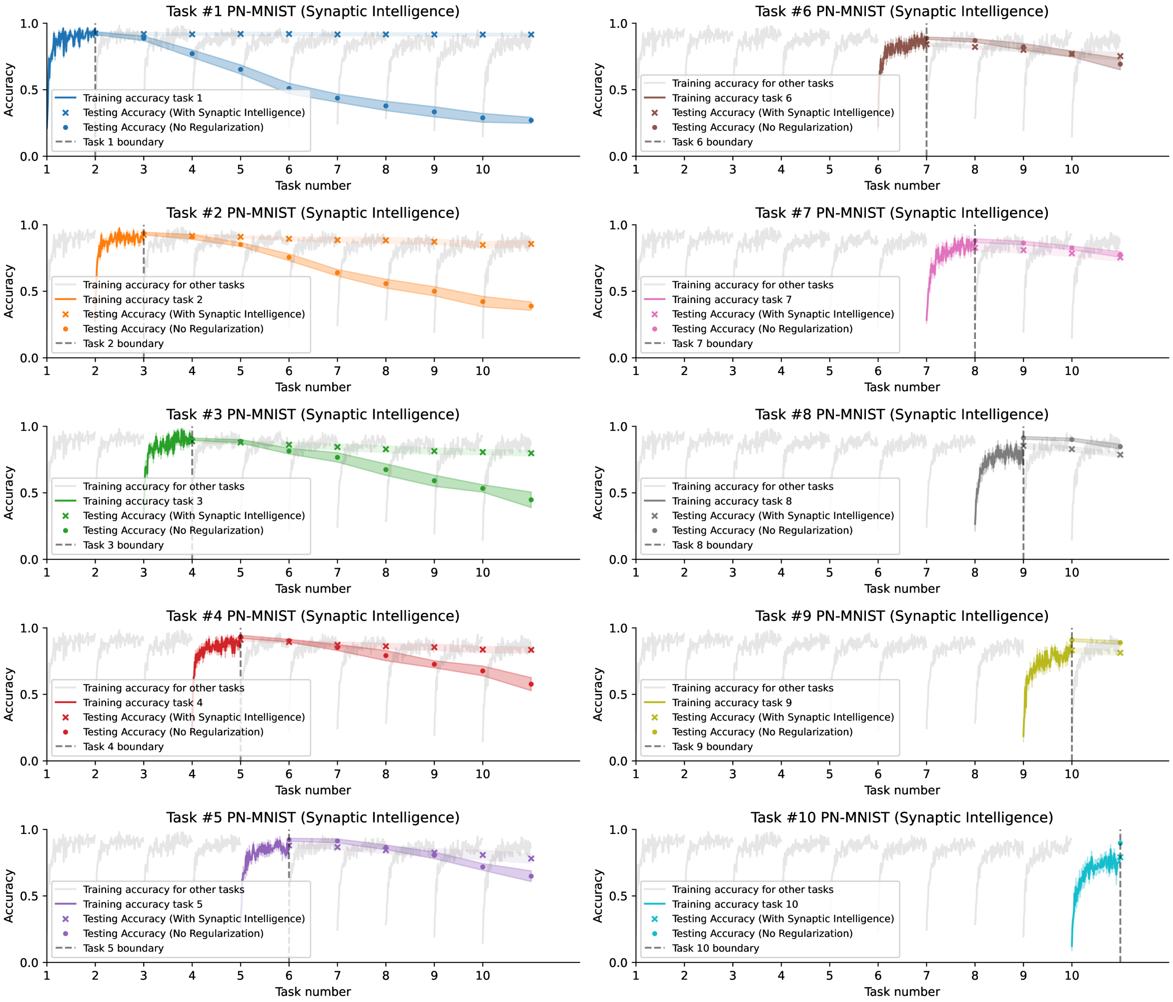


Figure 4.7: **Continual Learning Performance of Synaptic Intelligence (SI).** SI maintains parameter importance via an online computation and performs similarly to EWC in retaining knowledge across permuted tasks.

trast, all three regularized methods, including our approach using online saliency traces and novelty-triggered consolidation, maintain high ACC throughout training.

Notably, our method matches the performance of EWC and SI across the full task sequence, and slightly exceeds that of SI. This demonstrates that biologically inspired mechanisms can be competitive with established regularization-based strategies, while offering advantages such as online updating, task-agnostic consolidation, and biological plausibility. The shaded areas in the plot represent standard deviations across 10 independent runs, indicating the stability of the results.

The slowly declining average classification accuracy (ACC) curves for Elastic Weight Consolidation (EWC) (blue), Synaptic Intelligence (SI) (green), and our method (red) high-light two crucial aspects of these regularized approaches: their ability to continue learning new tasks and their resistance to forgetting previously learned ones. In contrast, a steep drop in ACC, as shown by the non-regularized baseline (black) after learning new tasks, clearly indicates severe catastrophic forgetting.

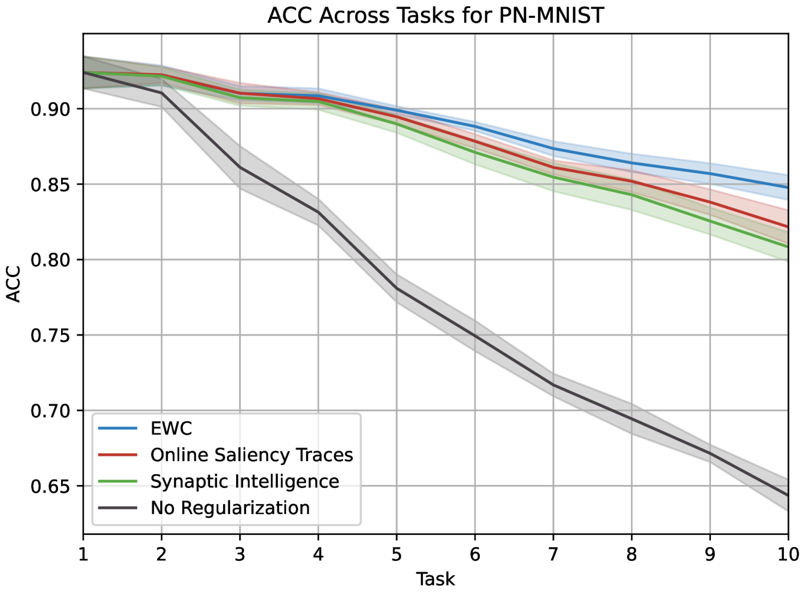


Figure 4.8: **Average Test Accuracy Across Tasks (ACC).** Comparison of average accuracy across all tasks as learning progresses. EWC in blue, SI in green, and our method (Online Saliency Traces + Novelty Neuron) in red. The black line represents the baseline without regularization. Shaded areas indicate standard deviation across 10 runs. The ACC metric shows how well the model retains knowledge from previous tasks while learning new ones. Our method achieves comparable performance to EWC and SI, while the baseline suffers from severe forgetting

4.7.2 Per-Task Accuracy Metrics

To gain insight into how each method performs on individual tasks immediately after learning them, we evaluate both per-task training and test accuracy.

Per-Task Train Accuracy is defined as:

$$A^{\text{train}}(i) = A^{\text{train}}_{i,i}, \tag{4.61}$$

which measures the training accuracy on task i directly after it is learned. As shown in Figure 4.9a, the model without any form of regularization (No Regularization) achieves, although by a small margin, the highest training accuracy across tasks. This is expected: in the absence of constraints on parameter updates, the model retains its full capacity to adapt to each new task.

Interestingly, our method, despite introducing regularization via saliency trace consolidation, performs comparably to the baseline, suggesting that it imposes a lighter constraint on learning dynamics compared to other continual learning methods. In contrast, both EWC and Synaptic Intelligence exhibit noticeably lower per-task training accuracy. This trend arises because these regularization-based methods increasingly restrict parameter updates as the number of tasks grows. Over time, this accumulation of constraints reduces the network's effective plasticity, limiting its ability to fully fit new tasks.

Per-Task Test Accuracy is defined as:

$$A^{\text{test}}(i) = A^{\text{test}}_{i,i}, \tag{4.62}$$

captures how well the model generalizes to task i right after learning it. The results in Figure 4.9b reveal a similar pattern to the training accuracy, but with more pronounced differences between the methods. The No Regularization baseline and our method both achieve higher test accuracy than EWC and SI, confirming that the reduced learning capacity in the latter two also hinders generalization. Notably, our method consistently matches the performance of the unconstrained baseline, despite still preserving knowledge from earlier tasks.

These observations reflect an important trade-off regarding regularization methods: while strong regularization can protect past knowledge, it often does so at the expense of learning new information.

4.7.3 Backward Transfer (BWT) and Forward Transfer (FWT)

Backward Tranfer (BWT) [177] quantifies how the performance on previously learned tasks changes as new tasks are added. Specifically, it compares the final test accuracy[2] on task i after learning all T tasks ($A_{T,i}$) to the accuracy immediately after learning that task ($A_{i,i}$):

$$\text{BWT} = \frac{1}{T-1}\sum_{i=1}^{T-1}\left(A_{T,i} - A_{i,i}\right), \tag{4.63}$$

A negative BWT indicates that the model has *forgotten* some of what it originally learned on earlier tasks due to interference from subsequent learning, which is directly interpretable

[2] Recall that $A_{k,i}$ denotes the test accuracy on task i after completing training on task k

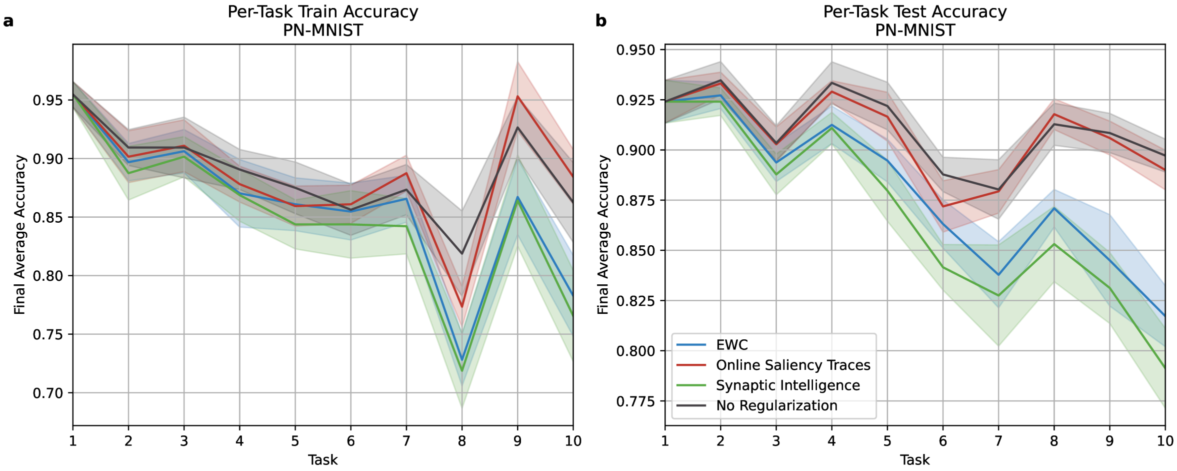


Figure 4.9: **Per-Task Train and Test Accuracy for PN-MNIST. (a)** Per-task train accuracy $A^{\text{train}}(i)$: the baseline (No Regularization) is expected to achieve the highest accuracy due to unrestricted parameter updates. Our method performs similarly, while EWC and SI show reduced training performance due to cumulative regularization constraints. **(b)** Per-task test accuracy $A^{\text{test}}(i)$: the same pattern holds in generalization, with our method and the baseline outperforming EWC and SI. The performance gap is more pronounced in test accuracy, highlighting the impact of over-regularization on both learning and generalization.

as **catastrophic forgetting**. Conversely, a positive BWT would suggest that the model has *improved* on earlier tasks, potentially due to shared structure or other beneficial updates. In practice, most continual learning systems aim to minimize negative BWT, keeping it as close to zero as possible.

Forward Transfer (FWT) [177] captures the model's ability to generalize *forward* across tasks, that is, how much prior learning helps with tasks that have not yet been trained. It compares the test accuracy on task i immediately *before* it is learned $(A_{i-1,i})$ to the model's initial performance on that task at random initialization (b_i):

$$\text{FWT} = \frac{1}{T-1}\sum_{i=2}^{T}\left(A_{i-1,i} - b_i\right), \tag{4.64}$$

A positive FWT suggests that knowledge acquired from previous tasks confers an advantage when the model encounters a new task, indicating the presence of positive forward influence. Negative FWT, would suggest interference or overfitting to earlier tasks that hinders future learning.

Together, BWT and FWT offer complementary insights into how a continual learning system navigates the stability-plasticity trade-off. High BWT implies strong memory retention (stability), while high FWT signals the ability to use past experience for new tasks (plasticity). Ideally, a good continual learning method should aim for minimal forgetting (i.e., BWT close to zero or positive) while achieving at least neutral or positive FWT.

In our experiments (see Figure 4.10), our method demonstrates negative but moderate BWT (compared to the non-regularized baseline), similar to EWC and SI, indicating some forgetting but generally good retention. The FWT values are small but positive across all methods, suggesting limited but measurable generalization to unseen tasks.

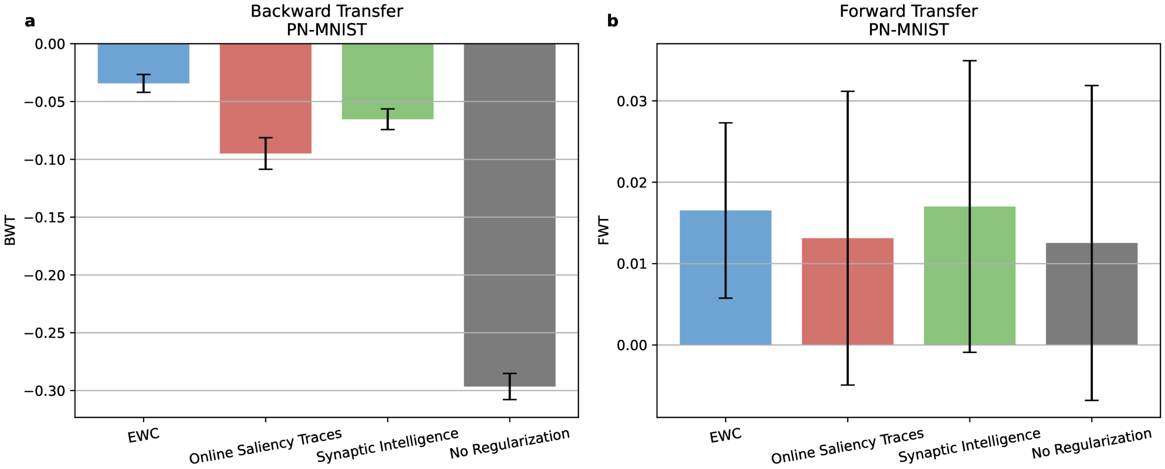


Figure 4.10: **(a) Backward and (b) Forward Transfer for PN-MNIST.** Our method achieves negative BWT (like EWC and SI), indicating mild forgetting, and exhibits small positive forward transfer, showing modest positive benefits in early-stage learning for future tasks.

These results indicate that our biologically inspired approach performs similarly to established regularization-based continual learning approaches, while providing benefits in online applicability and task-agnostic consolidation.

4.8 Discussion

This chapter presents a biologically inspired continual learning paradigm that merges online, per-synapse importance estimation through saliency traces with parameter consolidation based on unsupervised novelty detection. Our method gives a lightweight approximation of the diagonal Fisher information that is computed online during training, without the requirement for task boundaries or second passes over the training data. We do this by updating saliency traces with an exponential moving average of squared local gradients. A specialized novelty-detecting neuron, utilizing Bayesian online changepoint detection

(BOCD), indicates the onset of distributional shifts and triggers synaptic consolidation in real time

This design reflects key neurobiological motifs, most notably synaptic traces and neuromodulatory signals. Importantly, it does so without relying on biologically implausible assumptions found in prior approaches such as EWC and Synaptic Intelligence (SI), in particular, the need for offline Fisher computation or access to explicitly defined task boundaries. Our empirical results on the Permuted Neuromorphic MNIST (PN-MNIST) benchmark confirm that this framework effectively mitigates catastrophic forgetting, maintaining good performance across sequences of tasks without external task-identity information.

In terms of future work, several promising directions emerge. First, validating the framework on a wider range of datasets and learning modalities, including real-world neuromorphic data and reinforcement learning, will help assess its generality and robustness. Second, strengthening the probabilistic modeling by incorporating adaptive variance estimation into the conjugate prior could enhance the accuracy of online saliency estimates under non-stationary dynamics. Third, a deeper analysis of emergent spiking activity patterns could reveal whether functional specialization arises naturally across neuron populations, offering insights into the network's internal organization through the parameter consolidation process. Additionally, exploring the theoretical connection between classical eligibility traces and our proposed online saliency traces may open the door to simplified formulations. Finally, investigating biologically plausible synaptic pruning strategies that exploit persistently low saliency values may allow the model to reduce its computational footprint while preserving accumulated knowledge.

CHAPTER 5

Conclusions and Outlook

This thesis set out to investigate two central questions in the domain of biologically inspired learning with Spiking Neural Networks (SNNs): (1) how to enhance biological plausibility in learning algorithms without compromising performance, and (2) how to support continual learning in a manner consistent with known neurobiological mechanisms. The work presented in Chapters 3 and 4 addressed these questions by developing novel approaches that draw on core principles of brain function, such as local plasticity, asynchronous processing, and neuromodulation, while maintaining an emphasis on computational efficiency and scalability.

Contributions to Biological Plausibility

Chapter 3 addressed the first research question by reformulating eligibility propagation (e-prop) into an event-driven learning rule with additional bio-inspired features. Our implementation supports strictly local updates, asynchronous communication, non-instantaneous signal propagation, and neuron-type-specific dynamics (e.g., Dale's law). These set of changes enable the learning process to better reflect the operational principles of the brain, such as sparse and spike-triggered synaptic plasticity.

Empirical results on benchmark tasks, including N-MNIST, demonstrate that these biologically motivated constraints preserve, and in some cases enhance, training performance. Despite a modest runtime overhead, mainly due to increased activity induced by more active dynamics, learning remained stable, scalable, and robust across a range of configurations.

This work thus shows that it is possible to relax strict mathematical and structural constraints in favor of biologically plausible mechanisms without significantly degrading task performance, thereby affirmatively addressing Research Question 1.

Contributions to Continual Learning

Chapter 4 tackled the second research question by developing a continual learning framework grounded in local plasticity and neuromodulatory control. The proposed method combines Online Saliency Traces, computed from local activity using an exponential moving average of squared gradients, with synaptic consolidation triggered by an unsupervised changepoint detection neuron based on Bayesian Online Changepoint Detection (BOCD). This design mimics the role of neuromodulators in gating plasticity during salient or novel events.

Evaluations on the Permuted Neuromorphic MNIST (PN-MNIST) benchmark show that the framework successfully mitigates catastrophic forgetting without relying on task boundaries or second data passes. The approach preserves model plasticity across task sequences while maintaining knowledge retention on par with conventional regularization techniques such as Elastic Weight Consolidation (EWC) and Synaptic Intelligence (SI). These results provide a positive answer to Research Question 2, indicating that continual learning in SNNs can be achieved through biologically inspired mechanisms that operate in an online, local, and task-agnostic fashion.

Outlook and Future Directions

The results presented in this thesis support the view that biologically inspired learning offers a promising path toward scalable, efficient, and robust neural computation. Several avenues for future research arise naturally from this work:

- **Generalization to Diverse Tasks and Architectures:** Future studies should validate the proposed methods across a broader range of datasets, including reinforcement learning, sequential decision-making, and real-world neuromorphic data. Extending the approach to hierarchical and modular SNN architectures may also reveal additional insights into scalability and specialization.

- **Refinements in Plasticity and Consolidation:** Further work is needed to improve the fidelity of online saliency estimates and enhance the robustness of changepoint detection. Adaptive modeling of uncertainty and context-dependent consolidation strategies may help make the learning process more dynamic and selective.

- **Integration with Neuromorphic Hardware:** The event-driven and locally computable nature of the proposed methods makes them suitable for implementation on neuromorphic platforms. Exploring hardware acceleration and deployment could bridge the gap between biological learning models and real-world, low-power applications.

- **Emergent Dynamics and Interpretability:** A deeper analysis of the internal activity patterns and representations developed during learning could shed light on functional specialization and memory organization.

- **Theoretical Connections:** Investigating the formal relationships between classical learning concepts (e.g., eligibility traces, Fisher Information) and their biologically inspired counterparts (e.g., online saliency traces) could unify existing approaches and lead to more elegant learning rules.

This work demonstrates that bringing together biological realism and algorithmic performance is not only possible, but fruitful. The models and techniques developed herein provide a flexible and extensible foundation for future research at the intersection of neuroscience and artificial intelligence.

CHAPTER 6

APPENDIX A - CONVERGENCE OF THE EXPONENTIAL MOVING AVERAGE

In chapter 4, we introduced the concept of *Online Saliency Traces*, defined as an exponential moving average (EMA) of squared gradients, computed locally at each synapse. While the EMA is a standard tool in time series analysis and signal processing, its statistical properties are also of particular interest in our context of continual learning with Spiking Neural Networks (SNNs).

This appendix provides a self-contained proof of the convergence in probability of the EMA under specific conditions (Theorem 1). Although such convergence results are well established in the literature, typically as part of the analysis of recursive algorithms within the Robbins-Monro framework and the theory of adaptive filters [178, 179], these discussions often cover more general forms of stochastic approximation.

For clarity and completeness, we focus here on the specific recursive structure of the EMA and present a direct proof tailored to our setting.

Proof. We first derive an explicit expression for the sequence $\{f_n\}_{n\geq 0}$. By unrolling the recursion one obtains

$$f_n = \beta^n f_0 + (1-\beta)\sum_{k=1}^{n} \beta^{n-k} X_k. \tag{6.1}$$

Convergence of the mean. Taking expectations and using $\mathbb{E}[X_k] = \mu$ for all k,

$$\mathbb{E}[f_n] = \beta^n f_0 + (1-\beta)\mu \sum_{k=1}^{n} \beta^{n-k} = \beta^n f_0 + \mu\big(1-\beta^n\big) = \mu + \beta^n (f_0 - \mu).$$

Hence, for every fixed $\beta \in (0,1)$, $\lim_{n\to\infty} \mathbb{E}[f_n] = \mu$, because $\beta^n \to 0$. Consequently

$$\lim_{n\to\infty,\, \beta\to 1} \mathbb{E}[f_n] = \mu$$

no matter how the two limits are taken successively.

Bound on the variance. Set $\sigma^2 := \mathrm{Var}(X_1) < \infty$. Because the two summands in (6.1) are deterministic and an average of independent random variables, respectively, the variance of f_n is

$$\mathrm{Var}(f_n) = (1-\beta)^2\sigma^2 \sum_{k=1}^{n} \beta^{2(n-k)} = (1-\beta)^2\sigma^2 \frac{1-\beta^{2n}}{1-\beta^2} \le \sigma^2 \frac{(1-\beta)^2}{1-\beta^2}. \tag{6.2}$$

(Note that the upper bound (6.2) no longer depends on n.)

Vanishing variance as $\beta \to 1$. Rewrite the right-hand side of 6.2 as

$$\frac{(1-\beta)^2}{1-\beta^2} = \frac{1-\beta}{1+\beta} \xrightarrow[\beta\to 1]{} 0,$$

so $\mathrm{Var}(f_n) \to 0$ whenever $\beta \to 1$, uniformly in n.

Convergence in probability. Fix $\varepsilon > 0$. By Chebyshev's inequality and (6.1),

$$\mathbb{P}\big(|f_n - \mu| > \varepsilon\big) \le \frac{\mathrm{Var}(f_n)}{\varepsilon^2} + \frac{\beta^{2n}(f_0-\mu)^2}{\varepsilon^2}.$$

The second term is bounded by $\beta^{2n}(f_0-\mu)^2/\varepsilon^2 \to 0$ as $n \to \infty$ for every fixed $\beta < 1$. The first term can be made arbitrarily small by choosing β sufficiently close to 1. Combining the two observations yields

$$\lim_{n\to\infty,\, \beta\to 1} \mathbb{P}\big(|f_n - \mu| > \varepsilon\big) = 0, \qquad \forall \varepsilon > 0,$$

which is exactly

$$f_n \xrightarrow{\mathsf{P}} \mu.$$

Hence the exponential moving average converges in probability to the true mean as both the window parameter β approaches 1 and the sample size n grows without bound. ■

CHAPTER 7

APPENDIX B - SIMULATION PARAMETERS

This appendix outlines the detailed simulation parameters used in the experiments presented throughout the thesis. These specifications are essential for ensuring reproducibility and for gaining a clear understanding of the experimental setup and conditions.

7.1 Simulation Parameters and Hyperparameters

Table 7.1: Simulation parameters for event-driven e-prop on the pattern generation task (Part 1)

Category	**Parameter**	**Value**
Simulation Setup	Time step resolution (Δt)	1.0 ms
	Number of iterations (n_{iter})	2000
	Sequence duration (T)	1000 ms
	Loss function	Mean-Squared Error
Network Size	Input neurons (n_{in})	100
	Recurrent neurons (n_{rec})	100
	Output neurons (n_{out})	1
Synaptic Optimization	Optimizer type	Gradient descent
	Batch size	1
	Learning rate (η)	1×10^{-4}
	Minimum weight (W_{min})	-100.0 pA
	Maximum weight (W_{max})	100.0 pA

Table 7.2: Simulation parameters for event-driven e-prop on the pattern generation task (Part 2)

Category	Parameter	Value
Output Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	30.0 ms
	External current (I_e)	0.0 pA
	Initial voltage (V_m)	0.0 mV
Recurrent Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	30.0 ms
	External current (I_e)	0.0 pA
	Initial voltage (V_m)	0.0 mV
	Spike threshold (V_{th})	0.03 mV
	Refractory period (t_{ref})	0.0 ms
	Target firing rate (f_{target})	10.0 Hz
	Regularization coefficient (c_{reg})	300.0
	Surrogate gradient width (β)	1.0
	Surrogate gradient height (γ)	0.3

Table 7.3: Simulation parameters for event-driven e-prop on the evidence accumulation task (Part 1)

Category	Parameter	Value
Simulation Setup	Time step resolution (Δt)	1.0 ms
	Number of iterations (n_{iter})	2000
	Sequence duration (T)	2050 ms
	Loss function	Cross-Entropy
Network Size	Input neurons (n_{in})	40
	Recurrent neurons (n_{rec})	100
	Adaptive neurons (n_{ad})	50
	Output neurons (n_{out})	2
Synaptic Optimization	Optimizer type	Adam
	Batch size	32
	Learning rate (η)	5×10^{-3}
	Adam β_1	0.9
	Adam β_2	0.999
	Adam ϵ	1×10^{-8}
	Minimum weight (W_{min})	−100.0 pA
	Maximum weight (W_{max})	100.0 pA

Table 7.4: Simulation parameters for event-driven e-prop on the evidence accumulation task (Part 2)

Category	**Parameter**	**Value**
Output Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	20.0 ms
	External current (I_e)	0.0 pA
	Initial voltage (V_m)	0.0 mV
Recurrent Neuron (Regular)	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	20.0 ms
	External current (I_e)	0.0 pA
	Initial voltage (V_m)	0.0 mV
	Spike threshold (V_{th})	0.6 mV
	Refractory period (t_{ref})	5.0 ms
	Target firing rate (f_{target})	10.0 Hz
	Regularization coefficient (c_{reg})	300.0
	Surrogate gradient width (β)	1.667
	Surrogate gradient height (γ)	0.5
Recurrent Neuron (Adaptive)	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	20.0 ms
	External current (I_e)	0.0 pA
	Initial voltage (V_m)	0.0 mV
	Spike threshold (V_{th})	0.6 mV
	Refractory period (t_{ref})	5.0 ms
	Target firing rate (f_{target})	10.0 Hz
	Regularization coefficient (c_{reg})	300.0
	Surrogate gradient width (β)	0.5
	Surrogate gradient height (γ)	1.667
	Adaptation time constant (τ_{adapt})	2000.0 ms
	Adaptation strength (β_{adapt})	1.664

Table 7.5: Simulation parameters for event-driven e-prop on the N-MNIST task (Part 1)

Category	Parameter	Value
Simulation Setup	Time step resolution (Δt)	1.0 ms
	Number of iterations (n_{iter})	300
	Test iterations ($n_{\text{iter,test}}$)	10
	Group size	100
	Learning window	10 ms
	Sequence duration (T)	300 ms
	Loss function	Cross-Entropy
Network Size	Input neurons (n_{in})	1196
	Recurrent neurons (n_{rec})	150
	Output neurons (n_{out})	10
Synaptic Optimization	Optimizer type	Gradient descent
	Batch size	1
	Learning rate (η)	5×10^{-3}
	Minimum weight (W_{min})	−100.0 pA
	Maximum weight (W_{max})	100.0 pA
Connectivity Sparsity	Input to recurrent	0.75
	Recurrent to recurrent	0.99
	Recurrent to output	0.0

Table 7.6: Simulation parameters for event-driven e-prop on the N-MNIST task (Part 2)

Category	Parameter	Value
Output Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	100.0 ms
	Initial voltage (V_m)	0.0 mV
	External current (I_e)	0.0 pA
Recurrent Neuron (Regular & Adaptive)	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	30.0 ms
	Initial voltage (V_m)	0.0 mV
	External current (I_e)	0.0 pA
	Spike threshold (V_{th})	0.6 mV
	Refractory period (t_{ref})	0.0 ms
	Surrogate gradient width (β)	1.7
	Surrogate gradient height (γ)	0.5
	Regularization coefficient (c_{reg})	2.0
	Target firing rate (f_{target})	10.0 Hz

Table 7.7: Simulation parameters for event-driven e-prop with additional biological features on the N-MNIST task (Part 1)

Category	**Parameter**	**Value**
Simulation Setup	Time step resolution (Δt)	1.0 ms
	Number of iterations (n_{iter})	300
	Test iterations ($n_{\text{iter,test}}$)	10
	Group size	100
	Learning window	10 ms
	Sequence duration (T)	300 ms
	ISI trace cutoff	100 ms
	Loss function	Mean-Squared Error
Network Size	Input neurons (n_{in})	1196
	Recurrent neurons (n_{rec})	150
	Output neurons (n_{out})	10
Synaptic Optimization	Optimizer type	Gradient descent
	Batch size	1
	Learning rate (η)	5×10^{-7}
	W_{min}	-100.0 pA
	W_{max}	100.0 pA
Connectivity Sparsity	Input to recurrent	0.75
	Recurrent to recurrent	0.99
	Recurrent to output	0.0

Table 7.8: Simulation parameters for event-driven e-prop with additional biological features on the N-MNIST task (Part 2)

Category	**Parameter**	**Value**
Output Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	100.0 ms
	Initial voltage (V_m)	0.0 mV
	External current (I_e)	0.0 pA
Recurrent Neuron (Regular & Adaptive)	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	30.0 ms
	Initial voltage (V_m)	0.0 mV
	External current (I_e)	0.0 pA
	Spike threshold (V_{th})	0.6 mV
	Refractory period (t_{ref})	0.0 ms
	Surrogate gradient width (β)	1.7
	Surrogate gradient height (γ)	0.5
	Regularization coefficient (c_{reg})	66.666
	Target firing rate (f_{target})	10.0 Hz

Table 7.9: Simulation parameters for event-driven e-prop with additional biological features on the pattern generation task with larger delays (Part 1)

Category	**Parameter**	**Value**
Simulation Setup	Time step resolution (Δt)	1.0 ms
	Number of iterations (n_{iter})	200
	Sequence duration (T)	1000 ms
	ISI trace cutoff	1000 ms
	Loss function	Mean-Squared Error
Network Size	Input neurons (n_{in})	100
	Recurrent neurons (n_{rec})	100
	Output neurons (n_{out})	1
Synaptic Optimization	Optimizer type	Gradient descent
	Batch size	1
	Learning rate (η)	5×10^{-3}
	Minimum weight (W_{min})	-100.0 pA
	Maximum weight (W_{max})	100.0 pA

Table 7.10: Simulation parameters for event-driven e-prop with additional biological features on the pattern generation task with larger delays (Part 2)

Category	**Parameter**	**Value**
Output Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	30.0 ms
	External current (I_e)	0.0 pA
	Initial voltage (V_m)	0.0 mV
Recurrent Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	30.0 ms
	External current (I_e)	0.0 pA
	Initial voltage (V_m)	0.0 mV
	Spike threshold (V_{th})	0.03 mV
	Refractory period (t_{ref})	0.0 ms
	Target firing rate (f_{target})	10.0 Hz
	Regularization coefficient (c_{reg})	0.3
	Eligibility trace filter (κ)	0.97
	Regularization filter (κ_{reg})	0.97
	Surrogate gradient width (β)	33.3
	Surrogate gradient height (γ)	10.0
	Surrogate gradient (ψ)	piecewise linear

Table 7.11: Simulation parameters for event-driven e-prop on the PN-MNIST tasks (Part 1)

Category	**Parameter**	**Value**
Simulation Setup	Number of tasks	10
	Time step resolution (Δt)	1.0 ms
	Number of iterations (n_{iter})	95
	Test iterations ($n_{\text{iter,test}}$)	5
	Group size	64
	Learning window	10 ms
	Sequence duration (T)	100 ms
	Loss function	Cross-Entropy
Network Size	Input neurons (n_{in})	2312
	Recurrent neurons (n_{rec})	150
	Output neurons (n_{out})	10
Synaptic Optimization	Optimizer type	Adam
	Batch size	64
	Learning rate (η)	5×10^{-3}
	Adam β_1	0.9
	Adam β_2	0.999
	Adam ϵ	1×10^{-8}
	Minimum weight (W_{min})	-100.0 pA
	Maximum weight (W_{max})	100.0 pA
Connectivity Sparsity	Input to recurrent	0.75
	Recurrent to recurrent	1.0
	Recurrent to output	0.0
Regularization Coefficients	Online Saliency Traces + Novelty Neuron	0.5
	Elastic Weight Consolidation (EWC)	0.5
	Synaptic Intelligence (SI)	0.05

Table 7.12: Simulation parameters for event-driven e-prop on the PN-MNIST tasks (Part 2)

Category	**Parameter**	**Value**
Output Neuron	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	100.0 ms
	Initial voltage (V_m)	0.0 mV
	External current (I_e)	0.0 pA
	Online Saliency Trace filter β	0.9999
Recurrent Neuron (Regular & Adaptive)	Membrane capacitance (C_m)	1.0 pF
	Resting potential (E_L)	0.0 mV
	Membrane time constant (τ_m)	30.0 ms
	Initial voltage (V_m)	0.0 mV
	External current (I_e)	0.0 pA
	Spike threshold (V_{th})	0.6 mV
	Refractory period (t_{ref})	5.0 ms
	Surrogate gradient width (β)	1.7
	Surrogate gradient height (γ)	0.5
	Regularization coefficient (c_{reg})	2.0
	Target firing rate (f_{target})	10.0 Hz
	Online Saliency Trace filter β	0.9999
Novelty Neuron (BOCD)	Prior mean (μ_0)	2.0
	Prior variance (σ_0^2)	0.5
	Likelihood variance (σ^2)	5.0
	Hazard constant (H)	1×10^{-5}
	Max run length ($L_{\max}$)	1280

Bibliography

[1] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. T. H. Le, “Spiking neural networks and their applications: A review,” vol. 12.

[2] C. Zhou, H. Zhang, L. Yu, Y. Ye, Z. Zhou, L. Huang, Z. Ma, X. Fan, H. Zhou, and Y. Tian, “Direct training high-performance deep spiking neural networks: A review of theories and methods,”

[3] H. Paugam-Moisy and S. Bohte, *Computing with spiking neuron networks*, pp. 335–376.

Springer Berlin Heidelberg.

[4] C. Han, R. Zihan, and H. Xue, “The future of brain-inspired computing: A survey on large-scale spiking neural networks,”

[5] L. Zanatta, F. Barchi, S. Manoni, S. Tolu, A. Bartolini, and A. Acquaviva, “Exploring spiking neural networks for deep reinforcement learning in robotic tasks,” vol. 14, p. 30648.

[6] N. Rathi, I. Chakraborty, A. Kosta, A. Sengupta, A. Ankit, P. Panda, and K. Roy, “Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware,”

[7] M. S. Hasan, C. D. Schuman, Z. Zhang, T. Rahman, and G. S. Rose, “Spike-based neuromorphic computing for next-generation computer vision,”

[8] S. Shankar, Y. Pan, H. Jiang, Z. Liu, M. R. Darbandi, A. Lorenzo, J. Chen, M. M. Hasan, A. H. Zidan, E. Gelman, J. A. Konfrst, J. Y. Russell, K. Fernandes, T. Yang, Y. Li, H. Zhao, A. Jahin, T. Ganguly, S. Dinesha, Y. Zhou, Z. Wu, X. Li, L. Adusumilli, A. Hussein, S. Nookarapu, J. Hou, K. Jiang, J. Li, B. Heinel, X. Xi, H. Hubbard, Z. Khan, L. Whitaker, I. Cao, M. Allgaier, A. Darby, L. Zhao, L. Zhang, X. Wang, X. Li, W. Zhang, X. Yu, D. Zhu, Y. Abate, and T. Liu, “Bridging brains and machines: A unified frontier in neuroscience, artificial intelligence, and neuromorphic systems,”

[9] D. R. Muir and S. Sheik, “The road to commercial success for neuromorphic technologies,” vol. 16, p. 3586.

[10] M. Pfeiffer and T. Pfeil, “Deep learning with spiking neurons: Opportunities and challenges,” vol. 12, p. 774.
Asynchronous, central clock.

[11] A. E. Schegolev, M. V. Bastrakova, M. A. Sergeev, A. A. Maksimovskaya, N. V. Klenov, and I. I. Soloviev, “Contemporary implementations of spiking bio-inspired neural networks,”

[12] C. Lv, Y. Gu, Z. Guo, Z. Xu, Y. Wu, F. Zhang, T. Shi, Z. Wang, R. Yin, Y. Shang, S. Zhong, X. Wang, M. Wu, W. Liu, T. Li, J. Zhu, C. Zhang, Z. Ling, and X. Zheng, “Towards biologically plausible computing: A comprehensive comparison,”

[13] A. Ororbia, A. Mali, A. Kohan, B. Millidge, and T. Salvatori, “A review of neuroscience-inspired machine learning,”

[14] S. Mazurek, J. Caputa, J. K. Argasiski, and M. Wielgosz, “Three-factor learning in spiking neural networks: An overview of methods and trends from a machine learning perspective,”

[15] N. Frémaux and W. Gerstner, “Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules,” vol. 9, p. 85.

[16] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, “A solution to the learning dilemma for recurrent networks of spiking neurons,” vol. 11, p. 3625, 2020.
Publisher: Nature Publishing Group.

[17] K. Roy, A. Jaiswal, and P. Panda, “Towards spike-based machine intelligence with neuromorphic computing,” vol. 575, pp. 607–617.

[18] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, P. Date, and B. Kay, “Opportunities for neuromorphic computing algorithms and applications,” vol. 2, pp. 10–19.

[19] T. C. Wunderlich and C. Pehle, “Event-based backpropagation can compute exact gradients for spiking neural networks,” vol. 11, p. 12829.

[20] Y. H. Liu, S. Smith, S. Mihalas, E. Shea-Brown, and U. Sümbül, “Cell-type-specific neuromodulation guides synaptic credit assignment in a spiking neural network,” vol. 118, p. e2111821118.

[21] N. J. Mackintosh, *Animal Learning and Cognition.*

Handbook of Perception and Cognition, Elsevier Science.

[22] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," vol. 114, pp. 3521–3526.
arXiv:1612.00796 [cs, stat].

[23] F. Zenke, B. Poole, and S. Ganguli, "Continual Learning Through Synaptic Intelligence."
Comment: ICML 2017 arXiv:1703.04200 [cs, q-bio, stat].

[24] D. Sheynikhovich, S. Otani, and A. Arleo, "Dopaminergic control of long-term depression/long-term potentiation threshold in prefrontal cortex," vol. 33, pp. 13914–13926.

[25] W. Schultz, P. Dayan, and P. R. Montague, "A neural substrate of prediction and reward," vol. 275, pp. 1593–1599.

[26] T. U. Hauser, E. Eldar, N. Purg, M. Moutoussis, and R. J. Dolan, "Distinct roles of dopamine and noradrenaline in incidental memory," vol. 39, pp. 7715–7721.

[27] B. P. Bean, "The action potential in mammalian central neurons," vol. 8, pp. 451–465.

[28] Kandel, Eric R and Schwartz, James H and Jessell, Thomas M and Siegelbaum, Steven and Hudspeth, A James and Mack, Sarah et. al., *Principles of neural science.*
McGraw-hill New York.

[29] D. Purves, G. J. Augustine, D. Fitzpatrick, W. Hall, A.-S. LaMantia, and L. White, *Neurosciences.*
De Boeck Supérieur, 2019.

[30] E. Neher and T. Sakaba, "Multiple roles of calcium ions in the regulation of neurotransmitter release," vol. 59, pp. 861–872.

[31] M. Bear, B. Connors, and M. A. Paradiso, *Neuroscience: Exploring the brain, enhanced edition: Exploring the brain.*
Jones & Bartlett Learning, 2020.

[32] B. Pakkenberg and H. J. Gundersen, "Neocortical neuron number in humans: effect of sex and age," vol. 384, pp. 312–320.

[33] F. A. C. Azevedo, L. R. B. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. L. Ferretti, R. E. P. Leite, W. Jacob Filho, R. Lent, and S. Herculano-Houzel, "Equal numbers of neu-

ronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain," vol. 513, pp. 532–541.

[34] J. Zhang, "Basic neural units of the brain: Neurons, synapses and action potential,"

[35] T. P. Vogels and L. F. Abbott, "Gating multiple signals through detailed balance of excitation and inhibition in spiking networks," vol. 12, pp. 483–491.

[36] A. Citri and R. C. Malenka, "Synaptic plasticity: multiple forms, functions, and mechanisms," vol. 33, pp. 18–41.

[37] Y. Wang, M. Lobb-Rabe, J. Ashley, V. Anand, and R. A. Carrillo, "Structural and functional synaptic plasticity induced by convergent synapse loss in the drosophila neuromuscular circuit," vol. 41, pp. 1401–1417.

[38] R. Lamprecht and J. LeDoux, "Structural plasticity and memory," vol. 5, pp. 45–54.

[39] B. C. Albensi and D. Janigro, "Traumatic brain injury and its effects on synaptic plasticity," vol. 17, pp. 653–663.

[40] D. O. Hebb, *The organization of behavior: A neuropsychological theory.*
Lawrence Erlbaum Associates, 1st edition ed.

[41] E. Oja, "Oja learning rule," vol. 3, p. 3612.

[42] E. L. Bienenstock, L. N. Cooper, and P. W. Munro, "Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex," vol. 2, pp. 32–48.

[43] W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea, "Eligibility traces and plasticity on behavioral time scales: Experimental support of NeoHebbian three-factor learning rules," vol. 12, p. 53.

[44] T. Trappenberg, *Fundamentals of Computational Neuroscience.*
OUP Oxford.

[45] G. Q. Bi and M. M. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," vol. 18, pp. 10464–10472.

[46] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, "Neuronal dynamics: From single neurons to networks and models of cognition,"

[47] A. Foncelle, A. Mendes, J. Jdrzejewska-Szmek, S. Valtcheva, H. Berry, K. T. Blackwell, and L. Venance, "Modulation of spike-timing dependent plasticity: Towards the inclusion of a third factor in computational models," vol. 12, p. 49.

[48] P. W. Glimcher, "Understanding dopamine and reinforcement learning: the dopamine reward prediction error hypothesis," vol. 108 Suppl 3, pp. 15647–15654.

[49] K. Yamaguchi, Y. Maeda, T. Sawada, Y. Iino, M. Tajiri, R. Nakazato, S. Ishii, H. Kasai, and S. Yagishita, "A behavioural correlate of the synaptic eligibility trace in the nucleus accumbens," vol. 12, p. 1921.

[50] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity.*
Cambridge University Press.

[51] G. Lenz, K. Chaney, S. B. Shrestha, O. Oubari, S. Picaud, and G. Zarrella, "Tonic: event-based datasets and transformations."

[52] A. L. Hodgkin and A. F. Huxley, "Propagation of electrical signals along giant nerve fibers," vol. 140, pp. 177–183.

[53] E. M. Izhikevich, "Simple model of spiking neurons," vol. 14, pp. 1569–1572.

[54] F. Saraga, C. P. Wu, L. Zhang, and F. K. Skinner, "Active dendrites and spike propagation in multi-compartment models of oriens-lacunosum/moleculare hippocampal interneurons," vol. 552, pp. 673–689.

[55] K. Ferguson and S. A. Campbell, "A two compartment model of a CA 1 pyramidal neuron,"

[56] D. Salaj, A. Subramoney, C. Kraisnikovic, G. Bellec, R. Legenstein, and W. Maass, "Spike frequency adaptation supports network computations on temporally dispersed information," vol. 10.

[57] S. Rotter and M. Diesmann, "Exact digital simulation of time-invariant linear systems with applications to neuronal modeling," vol. 81, pp. 381–402.

[58] K. Park, S. Kim, M.-H. Oh, and W. Y. Choi, "Resting-potential-adjustable soft-reset integrate-and-fire neuron model for highly reliable and energy-efficient hardware-based spiking neural networks," vol. 590, p. 127762.

[59] F. Devalle, E. Montbrió, and D. Pazó, "Dynamics of a large system of spiking neurons with synaptic delay," vol. 98, p. 042214.

[60] A. Morrison, A. Aertsen, and M. Diesmann, "Spike-timing-dependent plasticity in balanced random networks," vol. 19, pp. 1437–1467.

[61] T. Wang, L. Yin, X. Zou, Y. Shu, M. J. Rasch, and S. Wu, “A phenomenological synapse model for asynchronous neurotransmitter release,” vol. 9, p. 153.

[62] A. Bielecki, P. Kalita, M. Lewandowski, and B. Siwek, “Numerical simulation for a neurotransmitter transport model in the axon terminal of a presynaptic neuron,” vol. 102, pp. 489–502.

[63] J. R. Stiles, T. M. Bartol, Jr, E. E. Salpeter, and M. M. Salpeter, *Monte Carlo simulation of neuro- transmitter release using MCell, a general simulator of cellular physiological processes*, pp. 279–284.
Springer US.

[64] I. D. Mienye, T. G. Swart, and G. Obaido, “Recurrent neural networks: A comprehensive review of architectures, variants, and applications,” vol. 15, p. 517.

[65] X. Xue, R. D. Wimmer, M. M. Halassa, and Z. S. Chen, “Spiking recurrent neural networks represent task-relevant neural sequences in rule-dependent computation,” vol. 15, pp. 1167–1189.

[66] B. Yin, F. Corradi, and S. M. Bohté, “Effective and efficient computation with multiple-timescale spiking recurrent neural networks,” in *International Conference on Neuromorphic Systems 2020*, ACM.

[67] B. Chakraborty and S. Mukhopadhyay, “Heterogeneous recurrent spiking neural network for spatio-temporal classification,” vol. 17, p. 994517.

[68] T. Liu, Y. Chua, Y. Ning, P. Liu, Y. Zhang, T. Li, G. Wan, Z. Wan, W. Chen, and S. Zhang, “motorSRNN: A spiking recurrent neural network inspired by brain topology for the effective and efficient decoding of cortical spike trains,” vol. 99, p. 106745.

[69] Fiveable, “8.1 rnn architecture and the concept of sequential memory - deep learning systems,” July 2024.
Accessed: 2025-07-16.

[70] N. Hiratani, Y. Mehta, T. Lillicrap, and P. Latham, “On the stability and scalability of node perturbation learning,”

[71] T. Chen, E. Fox, and C. Guestrin, “Stochastic gradient hamiltonian monte carlo,” vol. 32, pp. 1683–1691.

[72] Y. Maeda, H. Hirano, and Y. Kanata, “A learning rule of neural networks via simultaneous perturbation and its hardware implementation,” vol. 8, pp. 251–259.

[73] Y. Maeda and M. Wakamura, “Simultaneous perturbation learning rule for recurrent neural networks and its FPGA implementation,” vol. 16, pp. 1664–1672.

[74] P. Cortez, M. Rocha, and J. Neves, "A lamarckian approach for neural network training," vol. 15, pp. 105–116.

[75] J.-M. Yang and C.-Y. Kao, "A robust evolutionary algorithm for training neural networks," vol. 10, pp. 214–230.

[76] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," vol. 21, pp. 335–346.
Publisher: Nature Publishing Group.

[77] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," vol. 521, pp. 436–444.

[78] K. G. Kim, "Book review: Deep learning," vol. 22, p. 351.

[79] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," pp. 1017–1024.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by backpropagating errors," vol. 323, pp. 533–536.

[81] P. J. Werbos, "Backpropagation through time: what it does and how to do it," vol. 78, pp. 1550–1560.

[82] T. P. Lillicrap and A. Santoro, "Backpropagation through time and the brain," vol. 55, pp. 82–89.

[83] J. C. R. Whittington and R. Bogacz, "Theories of Error Back-Propagation in the Brain," vol. 23, pp. 235–250.
Publisher: Elsevier.

[84] J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, "Training Spiking Neural Networks Using Lessons From Deep Learning," vol. 111, pp. 1016–1054.
Conference Name: Proceedings of the IEEE.

[85] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[86] P. Cunningham, M. Cord, and S. J. Delany, *Supervised Learning*, pp. 21–49.
Springer Berlin Heidelberg.

[87] J. Menick, E. Elsen, U. Evci, S. Osindero, K. Simonyan, and A. Graves, "A Practical Sparse Approximation for Real Time Recurrent Learning."
arXiv:2006.07232 [cs, stat].

[88] J. F. Kolen and J. B. Pollack, "Backpropagation without weight transport," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 3, pp. 1375–1380 vol.3, IEEE.

[89] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks,"

[90] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," vol. 36, pp. 51–63.

[91] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," vol. 1, pp. 270–280.

[92] F. Zenke and S. Ganguli, "SuperSpike: Supervised learning in multi-layer spiking neural networks," vol. 30, pp. 1514–1541.
arXiv:1705.11146 [cs, q-bio, stat].

[93] J. M. Murray, "Local online learning in recurrent networks with random feedback," vol. 8.

[94] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)," vol. 14, p. 424.

[95] F. M. Quintana, F. Perez-Peña, P. L. Galindo, E. O. Neftci, E. Chicca, and L. Khacef, "ETLP: Event-based Three-factor Local Plasticity for online learning with neuromorphic hardware."
arXiv:2301.08281 [cs].

[96] T. Bohnstingl, S. Wozniak, A. Pantazi, and E. Eleftheriou, "Online spatio-temporal learning in deep neural networks," vol. 34, pp. 8894–8908.

[97] M. Stimberg, D. F. M. Goodman, V. Benichoux, and R. Brette, "Brian 2 - the second coming: spiking neural network simulation in python with code generation," vol. 14, pp. 1–1.

[98] M.-O. Gewaltig and M. Diesmann, "NEST (NEural Simulation Tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[99] M. Hines, T. Carnevale, and R. A. McDougal, *NEURON Simulation Environment*, pp. 1–7.
Springer New York.

[100] H. Hazan, D. J. Saunders, H. Khan, D. Patel, D. T. Sanghavi, H. T. Siegelmann, and R. Kozma, "BindsNET: A machine learning-oriented spiking neural networks library in python," vol. 12, p. 89.

[101] D. Rasmussen, "NengoDL: Combining deep learning and neuromorphic modelling methods," *arXiv*, vol. 1805.11144, pp. 1–22, 2018.

[102] M. G. K. Williams, P. Plank, and S. B. Shrestha, "Lava - a software framework for neuromorphic computing." `https://github.com/lava-nc/lava`, 2023.

[103] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," vol. 38, pp. 82–99.

[104] O. Rhodes, P. A. Bogdan, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D. R. Lester, M. Mikaitis, L. A. Plana, A. G. D. Rowley, A. B. Stokes, and S. B. Furber, "SPyNNaker: A software package for running PyNN simulations on SpiNNaker," vol. 12, p. 816.

[105] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," vol. 102, pp. 652–665.

[106] C. Pehle, S. Billaudelle, B. Cramer, J. Kaiser, K. Schreiber, Y. Stradmann, J. Weis, A. Leibfried, E. Müller, and J. Schemmel, "The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity," vol. 16, p. 795876.

[107] J. Stapmanns, J. Hahne, M. Helias, M. Bolten, M. Diesmann, and D. Dahmen, "Event-Based Update of Synapses in Voltage-Based Learning Rules," vol. 15.
Publisher: Frontiers.

[108] J. A. Espinoza Valverde, E. Müller, N. Haug, C. M. Schöfmann, C. Linssen, J. Senk, S. Spreizer, T. Trensch, M. Lober, H. Jiang, A. Kurth, J. Acimovic, A. Korcsak-Gorzo, J.-E. Welle Skaar, D. Terhorst, J. Stapmanns, S. Graber, R. de Schepper, J. M. Eppler, S. Kunkel, J. Mitchell, W. Wybo, A. Morrison, J. Vogelsang, M. A. Benelhedi, C. Köhn, F. Schmitt, T. Manninen, S. van Albada, J. Gille, J. Jenke, and H. E. Plesser, "NEST," 2024.

[109] G. G. Turrigiano and S. B. Nelson, "Homeostatic plasticity in the developing nervous system," vol. 5, pp. 97–107.

[110] N. P. Issa, K. C. Nunn, S. Wu, H. A. Haider, and J. X. Tao, "Putative roles for homeostatic plasticity in epileptogenesis," vol. 64, pp. 539–552.

[111] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization,"

[112] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale Machine Learning on Heterogeneous Distributed Systems," *arXiv*, 2016.

[113] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarkar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. Luk, B. Maher, Y. Pan, C. Puhrsch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, M. Suo, P. Tillet, E. Wang, X. Wang, W. Wen, S. Zhang, X. Zhao, K. Zhou, R. Zou, A. Mathews, G. Chanan, P. Wu, and S. Chintala, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*, ACM, 2024.

[114] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. S. Maida, "Deep learning in spiking neural networks,"

[115] C. Iaboni and P. Abichandani, "Event-based spiking neural networks for object detection: A review of datasets, architectures, learning rules, and implementation," vol. 12, pp. 180532–180596.

[116] A. Morrison, C. Mehring, T. Geisel, A. D. Aertsen, and M. Diesmann, "Advancing the boundaries of high-connectivity network simulation with distributed computing," vol. 17, pp. 1776–1801.

[117] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," vol. 98, pp. 459–478.

[118] J. Jordan, T. Ippen, M. Helias, I. Kitayama, M. Sato, J. Igarashi, M. Diesmann, and S. Kunkel, "Extremely scalable spiking neuronal network simulation code: From laptops to exascale computers," vol. 12, p. 2.

[119] A. C. Kurth, J. Senk, D. Terhorst, J. Finnerty, and M. Diesmann, "Sub-realtime simulation of a neuronal network of natural density," vol. 2, p. 021001.

[120] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, Jr, M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani, and A. Destexhe, "Simulation of networks of spiking neurons: a review of tools and strategies," vol. 23, pp. 349–398.

[121] M. Gewaltig and M. Diesmann, "NEST (NEural simulation tool)," vol. 2, p. 1430.

[122] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner, "Connectivity reflects coding: a model of voltage-based STDP with homeostasis," vol. 13, pp. 344–352.

[123] R. Urbanczik and W. Senn, "Learning by the dendritic prediction of somatic spiking," vol. 81, pp. 521–528.

[124] E. M. Izhikevich, "Solving the distal reward problem through linkage of STDP and dopamine signaling," vol. 17, pp. 2443–2452.

[125] Z. Brzosko, S. B. Mierau, and O. Paulsen, "Neuromodulation of spike-timing-dependent plasticity: Past, present, and future," vol. 103, pp. 563–581.

[126] C. Seguin, O. Sporns, and A. Zalesky, "Brain network communication: concepts, models and applications," vol. 24, pp. 557–574.

[127] R. Z. Aviv, I. Hakimi, A. Schuster, and K. Y. Levy, "Asynchronous distributed learning : Adapting to gradient delays without prior knowledge," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 436–445, PMLR.

[128] X. Deng, L. Shen, S. Li, T. Sun, D. Li, and D. Tao, "Towards understanding the generalizability of delayed stochastic gradient descent," vol. PP, pp. 1–11.

[129] L. Hui and M. Belkin, "Evaluation of Neural Architectures Trained with Square Loss vs Cross-Entropy in Classification Tasks."

Comment: An extended version of the paper published at ICLR2021. Added material includes evaluations of Transformer architectures arXiv:2006.07322 [cs, stat].

[130] S. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," vol. 31, pp. 1419–1428.

[131] V. Braitenberg and A. Schüz, "Cortex: statistics and geometry of neuronal connectivity,"

[132] A. Alreja, I. Nemenman, and C. J. Rozell, "Constrained brain volume in an efficient coding model explains the fraction of excitatory and inhibitory neurons in sensory cortices," vol. 18, p. e1009642.

[133] P. J. Sjöström, “Editorial: Horizons in synaptic neuroscience,” vol. 15, p. 1295640.

[134] P. D. King, J. Zylberberg, and M. R. DeWeese, “Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1,” vol. 33, pp. 5475–5485.

[135] P. Li, J. H. Cornford, A. Ghosh, and B. Richards, “Learning better with dale’s law: A spectral perspective,” vol. 36, pp. 944–956.

[136] G. Liu, “Local structural balance and functional interaction of excitatory and inhibitory synapses in hippocampal dendrites,” vol. 7, pp. 373–379.

[137] X. Hu and Z. Zeng, “Bridging the functional and wiring properties of V1 neurons through sparse coding,” vol. 34, pp. 104–137.

[138] E. Marder, “Motor pattern generation,” vol. 10, pp. 691–698.

[139] M. P. Nusbaum and M. P. Beenhakker, “A small-systems approach to motor pattern generation,” vol. 417, pp. 343–350.

[140] B. Engelhard, J. Finkelstein, J. Cox, W. Fleming, H. J. Jang, S. Ornelas, S. A. Koay, S. Y. Thiberge, N. D. Daw, D. W. Tank, and I. B. Witten, “Specialized coding of sensory, motor and cognitive variables in VTA dopamine neurons,” vol. 570, pp. 509–513.

[141] A. S. Morcos and C. D. Harvey, “History-dependent variability in population dynamics during evidence accumulation in cortex,” vol. 19, pp. 1672–1681.

[142] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, “Converting static image datasets to spiking neuromorphic datasets using saccades,” vol. 9, p. 437.

[143] F. Zenke and T. P. Vogels, “The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks,” vol. 33, pp. 899–925.

[144] F. Zenke and S. Ganguli, “SuperSpike: Supervised learning in multilayer spiking neural networks,” vol. 30, pp. 1514–1541.

[145] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, “Deep residual learning in spiking neural networks,” vol. 34, pp. 21056–21069.

[146] W. Potjans, A. Morrison, and M. Diesmann, “Enabling functional neural circuit simulations with distributed computing of neuromodulated plasticity,” vol. 4, p. 141.

[147] D. Plotnikov, B. Rumpe, I. Blundell, T. Ippen, J. M. Eppler, and A. Morrison, “NESTML: a modeling language for spiking neurons,”

[148] J. C. Knight and T. Nowotny, “Easy and efficient spike-based machine learning with mlGeNN,” in *Neuro-Inspired Computational Elements Conference*, pp. 115–120, ACM.

[149] A. Perrett, S. Summerton, A. Gait, and O. Rhodes, “Online learning in SNNs with e-prop and neuromorphic hardware,” in *Neuro-Inspired Computational Elements Conference*, ACM.

[150] A. Rostami, B. Vogginger, Y. Yan, and C. G. Mayr, “E-prop on SpiNNaker 2: Exploring online learning in spiking RNNs on neuromorphic hardware,” vol. 16, p. 1018006.

[151] C. Frenkel and G. Indiveri, “ReckOn: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales,” in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, pp. 1–3, IEEE.

[152] D. Kudithipudi, M. Aguilar-Simon, J. Babb, M. Bazhenov, D. Blackiston, J. Bongard, A. P. Brna, S. Chakravarthi Raja, N. Cheney, J. Clune, A. Daram, S. Fusi, P. Helfer, L. Kay, N. Ketz, Z. Kira, S. Kolouri, J. L. Krichmar, S. Kriegman, M. Levin, S. Madireddy, S. Manicka, A. Marjaninejad, B. McNaughton, R. Miikkulainen, Z. Navratilova, T. Pandit, A. Parker, P. K. Pilly, S. Risi, T. J. Sejnowski, A. Soltoggio, N. Soures, A. S. Tolias, D. Urbina-Meléndez, F. J. Valero-Cuevas, G. M. van de Ven, J. T. Vogelstein, F. Wang, R. Weiss, A. Yanguas-Gil, X. Zou, and H. Siegelmann, “Biological underpinnings for lifelong learning machines,” vol. 4, pp. 196–210.
Publisher: Nature Publishing Group.

[153] A. Robins, “Catastrophic forgetting, rehearsal and pseudorehearsal,” vol. 7, pp. 123–146.

[154] R. P. Adams and D. J. C. MacKay, “Bayesian online changepoint detection,”

[155] G. Kempermann, H. G. Kuhn, and F. H. Gage, “Experience-induced neurogenesis in the senescent dentate gyrus,” vol. 18, pp. 3206–3212.

[156] D. Ji and M. A. Wilson, “Coordinated memory replay in the visual cortex and hippocampus during sleep,” vol. 10, pp. 100–107.

[157] B. Rasch and J. Born, “Maintaining memories by reactivation,” vol. 17, pp. 698–703.

[158] W. C. Abraham, “Metaplasticity: tuning synapses and networks for plasticity,” vol. 9, p. 387.

[159] P. S. B. Finnie and K. Nader, “The role of metaplasticity mechanisms in regulating memory destabilization and reconsolidation,” vol. 36, pp. 1667–1707.

[160] M. E. Hasselmo and J. McGaughy, "High acetylcholine levels set circuit dynamics for attention and encoding and low acetylcholine levels set dynamics for consolidation," vol. 145, pp. 207–231.

[161] X. Zou, S. Kolouri, P. K. Pilly, and J. L. Krichmar, "Neuromodulated attention and goal-driven perception in uncertain domains," vol. 125, pp. 56–69.

[162] G. Bellitto, F. P. Salanitri, M. Pennisi, M. Boschini, L. Bonicelli, A. Porrello, S. Calderara, S. Palazzo, and C. Spampinato, "Saliency-driven experience replay for continual learning," in *Advances in Neural Information Processing Systems* (A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, eds.), vol. 37, pp. 103356–103383, Curran Associates, Inc.

[163] H. Salwa, N. Burhan, and E. Rahel, "Continual learning: Overcoming catastrophic forgetting for adaptive AI systems,"

[164] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: survey and performance evaluation on image classification,"

[165] D. Rolnick, A. Ahuja, J. Schwarz, T. P. Lillicrap, and G. Wayne, "Experience replay for continual learning,"

[166] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," vol. 30, pp. 2990–2999.

[167] G. M. van de Ven and A. S. Tolias, "Generative replay with feedback connections as a general strategy for continual learning,"

[168] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks,"

[169] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks,"

[170] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, openaccess.thecvf.com.

[171] K. P. Murphy, *Probabilistic machine learning: An introduction.*
MIT Press.

[172] F. Huszár, "On Quadratic Penalties in Elastic Weight Consolidation," vol. 115.
arXiv:1712.03847 [cs, stat].

[173] F. Zenke and A. Laborieux, "Theories of synaptic memory consolidation and intelligent plasticity for continual learning."
Comment: An introductory-level book chapter. 34 pages, 14 figures arXiv:2405.16922 [cs, q-bio].

[174] A. Gut, *Probability: A Graduate Course.*
Springer Texts in Statistics, Springer.

[175] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions: Forbes/statistical distributions 4E.*
Wiley-Blackwell, 4 ed.

[176] K. P. Murphy, "Conjugate bayesian analysis of the gaussian distribution," *def*, vol. 1, no. 2σ2, p. 16, 2007.

[177] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning,"

[178] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive algorithms and stochastic approximations*, vol. 22.
Springer Science & Business Media, 2012.

[179] H. J. Kushner and G. G. Yin, *Stochastic approximation and recursive algorithms and applications.*
Springer, 2003.