Synaptic Learning for Neuromorphic Vision

Processing Address Events with Spiking Neural Networks

zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte
DISSERTATION

Von
M.Sc. Jacques Kaiser
aus Strasbourg, France

Tag der mündlichen Prüfung:	15. Juni 2020
Erster Gutachter:	Prof. Dr.-Ing. Rüdiger Dillmann
Zweiter Gutachter:	Prof. Dr. Emre Neftci

Credits

> *How does the biological wetware of the brain give rise to our experience: the sight of emerald green, the taste of cinnamon, the smell of wet soil? What if I told you that the world around you, with its rich colors, textures, sounds and scents is an illusion, a show put on for you by your brain? If you could perceive reality as it really is, you would be shocked by its colorless, odorless, tasteless silence. Outside your brain, there is just energy and matter. Over millions of years of evolution the human brain has become adept at turning this energy and matter into a rich sensory experience of being in the world. How?*
>
> David Eagleman, *The brain: The story of you*, 2015

This PhD thesis was conducted within the FZI research center for information technology and funded by the Human Brain Project (HBP). An additional DAAD FIT Weltweit travel grant enabled my three months visit to the University of California, Irvine, for which I would like to express my gratitude.

First of all, I would like to thank my supervisor, Prof. Rüdiger Dillmann, who made it possible for me to conduct research at his chair. His warm-hearted support since day one helped me build the necessary confidence to complete this thesis. Similarly, I am grateful to Prof. Emre Neftci, who welcomed me in his lab and his house in Irvine and again in Zurich, with the hospitality of his wife Jacqueline, their kids and their chickens. His advanced understanding of how neural dynamics leads to learning was exactly what I was seeking.

Thanks to Arne Roennau for coming up with novel ways to accommodate science at FZI, and to all the colleagues from the House of Living Labs for sharing their knowledge of robotics. This thesis would not have been the same without the help and the cohesion within the team. Both Stefan Ulbrich and his successor Daniel Reichard offered me their guidance which was greatly appreciated. I would like to thank the bouldering and the rosrun groups for balancing my computer-intensive lifestyle. A special thank you goes to Stefan Scherzinger for his Geduld with my German. This thesis could not have been accomplished without the support of the talented students I had the chance to supervise and to whom I am grateful: Rainer Stal, David Zimmerer, Andreas Konle, Gerd Lindner, Michael Hoff, Svenja Melbaum, Fedor Scholz, and in particular Jakob Weinland, Philip Keller and Alexander Friedrich, who helped designing, building, wiring

and programming the robotic head. I realize how much I have learned at FZI when I remember my first exposure to robotics in 2015, where I accidentally destroyed an old drone twice during my master thesis at INRIA Grenoble (the wings broke on the first flight and the propellers caught fire on the second).

A lot of my working hours at FZI were dedicated to the HBP. From the HBP colleagues, I particularly thank J. Camilo V. Tieck, said the "monde carré" in Spanish, who stuck together from the beginning to the end, in good times and bad times. We have had an uncountable amount of fun together, between the intimacy of shared bedrooms during business trips to his memorable wedding in Colombia. I picked up a lot of habits from him, such as note-taking with a fountain pen and abusing the word "dude".

Lastly, I want to thank my family who provided the necessary moral support I wish anyone would benefit. The proximity between Karlsruhe and Alsace allowed me to come back often to my mother Chantal and my sister Thérèse. I loved taking care of the garden in Vendenheim and playing board games in Pfaffenhoffen, together with my brother-in-law Gilles, my goddaughter Jade, my girlfriend Camille and grandfather Pierrot, whose life sadly came to an end recently.

This work is dedicated to my father Jean-Georges, who tragically and suddenly passed away halfway throughout this thesis. He would have done anything for me. His good humor, jokes and life advice are irreplaceable and I miss him deeply.

Abstract (English Version)

The brain outperforms conventional computer architectures in aspects of energy efficiency, robustness and adaptivity. These aspects are also important for emerging technologies. It is therefore worthwhile to investigate what biological processes enable the brain to perform computations and how can they be implemented in silicon. Taking inspiration from how the brain performs computations requires a shift in computational paradigm compared to conventional computer architectures. Indeed, the brain is composed of nervous cells, called neurons, connected with synapses and forming self-organized networks. Neurons and synapses are complex dynamical systems ruled by biochemical and electrical reactions. As a result, they can only base their computations on local information. Additionally, neurons communicate with each other with short electrical pulses, called spikes, which travel across synapses.

Computational neuroscientists attempt to model these computations with spiking neural networks. When implemented on dedicated neuromorphic hardware, spiking neural networks can perform fast, energy efficient computations like the brain. Until recently, the advantages of this technology were limited due to the lack of functional methods for programming spiking neural networks. Learning is one paradigm for programming spiking neural networks, in which neurons self-organize into functional networks.

As in the brain, learning in neuromorphic hardware is based on synaptic plasticity. Synaptic plasticity rules characterize weight updates in terms of information local to the synapse. Learning happens in a continuous and online fashion, while sensory input is streamed to the network.

Conventional deep neural networks are commonly trained with gradient descent. However, the constraints imposed by biological learning dynamics prevent the use of conventional backpropagation to compute the gradients. For example, continuous updates hinder the synchronous alternation between forward and backward phases. Additionally, memory limitations prevent the history of neural activity to be stored as-is at the neuron, prohibiting backpropagation-through-time. Novel solutions to these problems were proposed by computational neuroscientists within the time-frame of this thesis.

In this thesis, spiking neural networks are developed to solve visuomotor neurorobotics tasks. Indeed, biological neural networks evolved to control the body. The field of robotics provides an artificial body to the artificial brain. On one side,

this thesis contributes to the current endeavor in understanding the brain by providing difficult closed-loop benchmarks, similar to what the biological brain experiences. On the other side, novel ways of solving traditional robotic problems are introduced, based on brain-inspired paradigms. The research is conducted in two steps. First, promising synaptic plasticity rules are identified and benchmarked on real-world event-based vision benchmarks. Second, novel methods to map visual representations to motor commands are presented.

Neuromorphic vision sensors mark an important step in shifting towards brain-inspired paradigms. Unlike conventional cameras, these sensors emit address events corresponding to local light intensity changes. The event-based paradigm enables energy efficient and fast visual processing but requires new asynchronous algorithms to be derived. Spiking neural networks constitute a subset of asynchronous algorithms inspired by the brain and suited to neuromorphic hardware technology. With a close collaboration with computational neuroscientists, successful methods to learn spatio-temporal abstractions from address event representation are reported. It is shown that top-down synaptic plasticity rules derived to optimize an objective function outperform bottom-up rules solely based on observations in the brain. With this insight in mind, a new synaptic plasticity rule called Deep Continuous Local Learning is introduced, currently achieving state-of-the-art accuracy on event-based vision benchmarks. This rule was jointly derived, implemented and evaluated during a stay at the University of California, Irvine.

In the second part of this thesis, the visuomotor loop is closed by mapping the learned visual representations to motor commands. Three approaches are discussed to obtain a visuomotor mapping: manual coupling, reward-coupling and prediction error minimization. It is shown how these approaches implemented as synaptic plasticity rules can be used to learn simple policies and movements. This work paves the way towards the integration of brain-inspired computational paradigms into the field of robotics. Indeed, it is suggested that advances in neuromorphic technology and plasticity rules would enable the development of learning robots operating at high speed and low power.

Abstract (German Version)

Das Gehirn übertrifft herkömmliche Computerarchitekturen in Bezug auf Energieeffizienz, Robustheit und Anpassungsfähigkeit. Diese Aspekte sind auch für neue Technologien wichtig. Es lohnt sich daher, zu untersuchen, welche biologischen Prozesse das Gehirn zu Berechnungen befähigen und wie sie in Silizium umgesetzt werden können. Um sich davon inspirieren zu lassen, wie das Gehirn Berechnungen durchführt, ist ein Paradigmenwechsel im Vergleich zu herkömmlichen Computerarchitekturen erforderlich. Tatsächlich besteht das Gehirn aus Nervenzellen, Neuronen genannt, die über Synapsen miteinander verbunden sind und selbstorganisierte Netzwerke bilden. Neuronen und Synapsen sind komplexe dynamische Systeme, die durch biochemische und elektrische Reaktionen gesteuert werden. Infolgedessen können sie ihre Berechnungen nur auf lokale Informationen stützen. Zusätzlich kommunizieren Neuronen untereinander mit kurzen elektrischen Impulsen, den so genannten Spikes, die sich über Synapsen bewegen.

Computational Neuroscientists versuchen, diese Berechnungen mit spikenden neuronalen Netzen zu modellieren. Wenn sie auf dedizierter neuromorpher Hardware implementiert werden, können spikende neuronale Netze wie das Gehirn schnelle, energieeffiziente Berechnungen durchführen. Bis vor kurzem waren die Vorteile dieser Technologie aufgrund des Mangels an funktionellen Methoden zur Programmierung von spikenden neuronalen Netzen begrenzt. Lernen ist ein Paradigma für die Programmierung von spikenden neuronalen Netzen, bei dem sich Neuronen selbst zu funktionalen Netzen organisieren.

Wie im Gehirn basiert das Lernen in neuromorpher Hardware auf synaptischer Plastizität. Synaptische Plastizitätsregeln charakterisieren Gewichtsaktualisierungen im Hinblick auf Informationen, die lokal an der Synapse anliegen. Das Lernen geschieht also kontinuierlich und online, während sensorischer Input in das Netzwerk gestreamt wird.

Herkömmliche tiefe neuronale Netze werden üblicherweise durch Gradientenabstieg trainiert. Die durch die biologische Lerndynamik auferlegten Einschränkungen verhindern jedoch die Verwendung der konventionellen Backpropagation zur Berechnung der Gradienten. Beispielsweise behindern kontinuierliche Aktualisierungen den synchronen Wechsel zwischen Vorwärts- und Rückwärtsphasen. Darüber hinaus verhindern Gedächtnisbeschränkungen, dass die Geschichte der neuronalen Aktivität im Neuron gespeichert wird, so dass Verfahren wie

Backpropagation-Through-Time nicht möglich sind. Neuartige Lösungen für diese Probleme wurden von Computational Neuroscientists innerhalb des Zeitrahmens dieser Arbeit vorgeschlagen.

In dieser Arbeit werden spikende neuronaler Netzwerke entwickelt, um Aufgaben der visuomotorischen Neurorobotik zu lösen. In der Tat entwickelten sich biologische neuronale Netze ursprünglich zur Steuerung des Körpers. Die Robotik stellt also den künstlichen Körper für das künstliche Gehirn zur Verfügung. Auf der einen Seite trägt diese Arbeit zu den gegenwärtigen Bemühungen um das Verständnis des Gehirns bei, indem sie schwierige Closed-Loop-Benchmarks liefert, ähnlich dem, was dem biologischen Gehirn widerfährt. Auf der anderen Seite werden neue Wege zur Lösung traditioneller Robotik Probleme vorgestellt, die auf vom Gehirn inspirierten Paradigmen basieren. Die Forschung wird in zwei Schritten durchgeführt. Zunächst werden vielversprechende synaptische Plastizitätsregeln identifiziert und mit ereignisbasierten Vision-Benchmarks aus der realen Welt verglichen. Zweitens werden neuartige Methoden zur Abbildung visueller Repräsentationen auf motorische Befehle vorgestellt. Neuromorphe visuelle Sensoren stellen einen wichtigen Schritt auf dem Weg zu hirninspirierten Paradigmen dar. Im Gegensatz zu herkömmlichen Kameras senden diese Sensoren Adressereignisse aus, die lokalen Änderungen der Lichtintensität entsprechen. Das ereignisbasierte Paradigma ermöglicht eine energieeffiziente und schnelle Bildverarbeitung, erfordert aber die Ableitung neuer asynchroner Algorithmen. Spikende neuronale Netze stellen eine Untergruppe von asynchronen Algorithmen dar, die vom Gehirn inspiriert und für neuromorphe Hardwaretechnologie geeignet sind. In enger Zusammenarbeit mit Computational Neuroscientists werden erfolgreiche Methoden zum Erlernen räumlich-zeitlicher Abstraktionen aus der Adressereignisdarstellung berichtet. Es wird gezeigt, dass Top-Down-Regeln der synaptischen Plastizität, die zur Optimierung einer objektiven Funktion abgeleitet wurden, die Bottom-Up-Regeln übertreffen, die allein auf Beobachtungen im Gehirn basieren. Mit dieser Einsicht wird eine neue synaptische Plastizitätsregel namens "Deep Continuous Local Learningëingeführt, die derzeit den neuesten Stand der Technik bei ereignisbasierten Vision-Benchmarks erreicht. Diese Regel wurde während eines Aufenthalts an der Universität von Kalifornien, Irvine, gemeinsam abgeleitet, implementiert und evaluiert.

Im zweiten Teil dieser Arbeit wird der visuomotorische Kreis geschlossen, indem die gelernten visuellen Repräsentationen auf motorische Befehle abgebildet werden. Drei Ansätze werden diskutiert, um ein visuomotorisches Mapping zu erhalten: manuelle Kopplung, Belohnungs-Kopplung und Minimierung des Vorhersagefehlers. Es wird gezeigt, wie diese Ansätze, welche als synaptische Plastizitätsregeln implementiert sind, verwendet werden können, um einfache Strategien und Bewegungen zu lernen. Diese Arbeit ebnet den Weg zur Integration von hirninspirierten Berechnungsparadigmen in das Gebiet der Robotik. Es wird sogar prognostiziert, dass Fortschritte in den neuromorphen Technologien und bei den Plastizitätsregeln die Entwicklung von Hochleistungs-Lernrobotern mit geringem Energieverbrauch ermöglicht.

Contents

1. Introduction

This introductory chapter is written as a self-contained overview of the whole thesis. It is intended for readers who are not familiar with the field of computing with spiking neurons and event-based vision. This thesis is motivated by the potential of neuromorphic technologies for robotics, as well as the desire to understand the brain better. This leads to the main research goals of the thesis: how can Spiking Neural Networks (SNNs) learn spatio-temporal representations from event streams, how can robots be controlled from these representations? The approach to address these goals is presented in Chapter 4, together with the developed tools for experimenting with simulated and real event-based data. Subsequently, with close interaction with computational neuroscientists, state-of-the-art synaptic learning rules are derived, integrated, and evaluated on event streams (Chapter 5). In Chapter 6, it is shown how to map these representations to motor commands, closing the visuomotor loop. An outlook on the broader impacts and the future work in the field of Neurorobotics is provided in Chapter 7.

1.1. Motivation

The nervous system has extensively been compared with an artificial computing machine [284]. The center of the nervous system is the brain, where most nerve cells, called neurons, are concentrated and connected with synapses. Similarly to computers, the brain processes information, from afferent receptor neurons – the input – to efferent neurons connected to muscles and glands – the output. While individual neurons are understood in considerable detail, it is still unclear how function emerges from large ensembles of neurons.

Unlike standard computer architectures, communication between neurons in the brain is sparse, asynchronous, and unreliable. This communication is based on rapid depolarization impulses, called spikes. Most neuroscientists believe that all spikes are stereotypical, thus carrying no payload. Recent information theory analyses reveal that precise spike times, in the order of milliseconds, carries a substantial amount of information [246, 247]. Overall, spike-based communication makes biological brains energy-efficient, fast, fault-tolerant, and adaptable. When implemented on dedicated neuromorphic hardware, spiking networks enjoy some of the benefits of biological systems: energy-efficient, fast computations,

reduced communication bottleneck (data is transmitted in an event-based fashion) [212]. As for a biological body, these advantages concern critical aspects in robotics. Energy efficiency is desired for long-term use. Fast computations are needed for real-time operation. High-throughput sensorimotor information is highly redundant and jams computations. Sample efficiency is required to learn in the real world. Despite these potential advantages, spiking network models are overlooked by engineers.

Understanding the computational principles of the brain would allow us to replicate its functions in silico to equip robots. This long-term endeavor spans over multiple research fields, as witnessed by the flagship research projects worldwide: the Human Brain Project in Europe, the BRAIN Initiative in the USA, the China Brain Project and the Brain/MINDS Project in Japan. These ambitious research projects accommodate the collaboration of scientists from various fields of research. This thesis was conducted within the Human Brain Project and subscribes to the following research workflow. Neuroscientists and biologists collect data about the nervous system by performing experiments. Their insights are converted into computational models by theoretical neuroscientists. The requirements of these models drive the development of dedicated brain simulation hardware by neuromorphic engineers. This pipeline converges to the recent field of neurorobotics, which provides an artificial embodiment to the artificial brain.

The goal of neurorobotics is twofold. On one side, neurorobotics aims to improve the autonomy of robots using brain-inspired architectures. On the other side, it aims to foster an understanding of the brain by replicating behavioral experiments. Indeed, biological brains are connected to the world through a body. The body is the only mean by which the brain interacts with the world by developing sensory and motor skills. The main role of the brain is therefore to control the body [289]. Additionally, the embodiment also plays a direct role in cognition, as observed by psychological experiments on humans [233].

This thesis focuses on embodying computational brain models to solve neurorobotic visuomotor tasks. Visual ability is an important skill both in biology and in robotics. The complexity of the mammalian eye highlights the evolutionary advantages granted by an advanced vision system [162]. Vision provides a large amount of important information about the world. This information requires complicated processing systems – either nervous or electronic. On the biological side, it was shown that at least some visual tasks such as stimuli categorization and saccade generation were relying on precise spike-time information [280, 279, 191, 83]. This contrasts with state-of-the-art deep learning networks (later referred to as Analog Neural Networks (ANNs)) which process frames synchronously with neurons which real-valued activities represent rates of spikes [124, 243].

Recent neuromorphic vision sensors engaged a shift to spike-based computational paradigm [110, 20]. Inspired from biological retinas, neuromorphic vision sensors emit events at precise time on local light intensity changes, unlike conventional cameras which provide frames at constant time intervals. These sensors

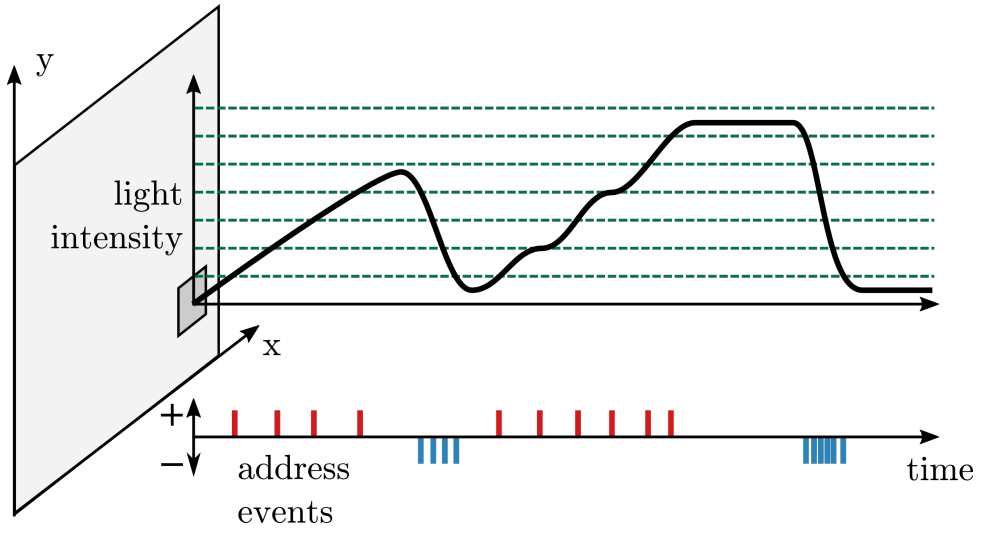


Figure 1.1.: Address Event Representation (AER) in neuromorphic vision sensor. Address events are emitted for a pixel with respect to local light intensity. Increase and decrease in intensity yield positive (top reds) and negative (bottom blues) events respectively. In contrast, a normal camera samples light intensity at discrete time intervals. This leads to redundancies, especially when the light intensity does not change.

convey information in AER, which is better suited to robotics as only the relevant data – what changed in the scene – is transmitted, see Figure 1.1.

The AER is a hardware protocol introduced by Lazzaro et al. in [166], inspired by spike-based communication. Conceptually, it consists of emitting events at precise time, containing the address of the receptor responsible for this event. Data itself is encoded in the time at which the event was emitted. In the case when the latency between the sensor and the host remains constant, individual events do not require to contain timestamps. This protocol defines an energy-efficient communication, enabling less redundant computations and high processing speed. Additionally, AER provides a natural way to encode visual information into spikes.

1.2. Problem Statement

The research goals of this thesis are the following:

Research question 1. Can the synaptic learning rules developed in computational neuroscience learn spatio-temporal representations from event-based data?

These goals are addressed in Chapter 5 and Chapter 6 respectively. The general approach is introduced in Chapter 4.

Research question 2. How can these visual representations map to robot control?

1.3. Approach

The main role of the brain is to control the body by developing sensory and motor skills [289]. Processing in the brain is implemented with neural architectures communicating asynchronously with stereotypical, instantaneous spikes. These architectures are distributed, hierarchical, recurrent, and heterogeneous [182]. In this thesis, it is shown how visuomotor tasks can be solved by combining neuromorphic technology with neural learning techniques.

Learning techniques in spiking neural networks are often referred to as synaptic learning. Unlike conventional backpropagation used in deep learning, these rules take into account biological constraints such as locality of information and neural dynamics. Respecting these constraints enables the development of efficient neuromorphic hardware. These learning rules are often solely evaluated on classification tasks and rarely guide experimentations in a neuroscience lab or improve mainstream computing technology. Collaborations between theoretical neuroscientists and roboticists are required to evaluate such learning rules in real-world tasks. They can learn representations and encodings from a dataset, but can they handle motor control mechanisms and sensorimotor integration in closed-loop like the brain does? What can we learn about the organization of plasticity in the brain from theoretically guided principles? These questions are addressed in this thesis by embodying synaptic learning rules onto robots to solve visuomotor closed-loop tasks, see Figure 1.2.

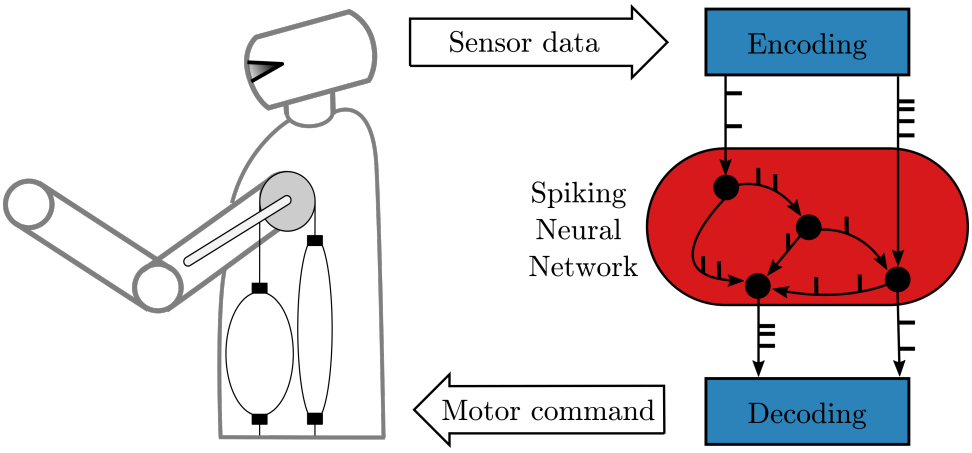


Figure 1.2.: Diagram of a Neurorobotics closed-loop setup. The sensor data is encoded into spike-trains and sent to input neurons of the SNN. The spike-trains of the motor neurons are decoded to motor commands for the robot.

In a closed-loop setup, sensor data is encoded into spike-trains and sent to input neurons of the SNN. The spike-trains of the motor neurons are decoded to motor commands for the robot. As in a biological brain, learning is performed online and locally within the network through synaptic plasticity. Neuromorphic hardware such as SpiNNaker, IBM TrueNorth or Intel's Loihi can be used to simulate the SNN. Conceptually, encoding, decoding and SNN simulation are happening simultaneously instead of sequentially. Therefore, the robot time needs to be synchronized with SNN time, technically handled by neurorobotics tools [4, 285].

In this thesis, visual information is provided in the form of event streams, which can be naturally encoded into spikes, see Figure 1.1. Indeed, previous work on synaptic learning rules often focused on object recognition from a static image encoded with Poisson spike-trains. This encoding converts pixel intensities to rates of spikes, as sampled from a Poisson distribution for a given time interval. By neglecting the role of retinal adaptation, such encoding slows down processing and poorly scales to a continuous stream of visual information. On the other hand, event-based data can capture visual information even for very fast motions such as microsaccadic eye movements.

1.4. Summary of the Contributions

The summary of the contributions of this thesis is listed here. Each contribution is discussed thoroughly in its respective Chapter. Each Chapter corresponds to a series of publications that are outlined in this Section. Specifically, Chapter 5 and Chapter 6 address the two research goals of this thesis: learning spatio-temporal representations from event-based data with synaptic plasticity rules, and control robots from these representations. Addressing these goals required the development of dedicated tools which are themselves contributions, discussed in Chapter 4. Indeed, some of these tools, such as the Gazebo Dynamic Vision Sensor (DVS) plugin, were already used by other researchers in their work [62, 148].

- **Tools for Visuomotor Neurorobotics**: We developed tools that enable simulated and real-world visuomotor experiments to be conducted with event-based data. We additionally showcased these tools in demonstrators. In [15], we introduced the Gazebo DVS plugin, which simulates event-based vision in the closed-loop robotics simulator Gazebo. This simulator is demonstrated in an end-to-end lane following experiment, with a Braitenberg vehicle implemented with a SNN. It was also used in many of our subsequent work [8, 11, 14, 4]. We released the code for the simulator to open-source[1], which was integrated into the Neurorobotics Platform (NRP) and used by other researchers [62, 148]. In [16], the neuromorphic DVS head is introduced to perform eye movements, built from Dynamixel servos and 3D printed parts. Eye movements are relevant in the context of neuromorphic

[1] `https://github.com/HBPNeurorobotics/gazebo_dvs_plugin`

vision since they enable edge extraction from static scenes (microsaccades) and attention shifts (saccades). This head is demonstrated in a stereo vision setup in [16] and used to record a static object classification dataset in [6]. Further, a method to connect event-based data to SNNs is presented and a simple attention mechanism improving accuracy and efficiency is introduced. This mechanism is evaluated in [6].

- **Learning Visual Representations from event-based data**: Many synaptic plasticity rules had already been proposed in the field of computational neuroscience but little was known as to how these rules perform to learn spatio-temporal representations from event-based data. Additionally, being a very active field of research, a strong collaboration with computational neuroscientists (mainly from TU Graz and the University of California, Irvine) was necessary to keep up to date with the latest rules. We show that top-down synaptic plasticity rules derived to optimize an objective function outperform bottom-up rules solely based on observations in the brain. Evidence for this result is provided in Table 5.2, which reports the accuracy of the evaluated methods on the DvsGesture dataset. First, we evaluated Liquid State Machine (LSM) in [14] and Spike-Timing-Dependent-Plasticity (STDP) in [9] for prediction and classification from event-based data respectively. The accuracy obtained with these rules was below state-of-the-art machine learning techniques not relying on SNNs. Second, the rules derived from machine learning – Event-Driven Contrastive Divergence [211] (eCD) in [17] and Event-Driven Random Backpropagation [213] (eRBP) in [6] – have been evaluated. These rules achieved an accuracy comparable to deep learning networks with the addition of convolutions or attention mechanisms. We finished by deriving and evaluating our own rule, Deep Continuous Local Learning [12] (DECOLLE), during my stay at University of California, Irvine in [12]. This rule achieves the best performance ever reported on the DvsGesture dataset in supervised learning fashion. The code implementing this rule was made open-source[2].

- **Visuomotor Coupling**: Closing the visuomotor loop involves learning a mapping from spatio-temporal representations to motor commands. We evaluated three different approaches. First, we manually defined this coupling in [6] by associating a predefined reaching and grasping trajectory to every object class in the dataset. This method is commonly used in robotics for its simplicity but is limited. We then evaluated the reward-learning rule Synaptic Plasticity with Online Reinforcement learning [150] (SPORE) on the lane following task in [8]. This rule could learn performing policy online but is also limited in aspects of network size, learning time and accuracy. Lastly, we evaluated a prediction error minimization technique in [11] relying on the predicting LSMs from [14]. In this work, a simulated robot was able to reproduce movements visually similar to a demonstrated movement.

[2] `https://github.com/nmi-lab/dcll`

1.5. Outlook

Researchers made breakthroughs both in the field of deep learning and neuromorphic engineering over the course of this thesis. The field of deep reinforcement learning, introducing how deep networks could be trained with backpropagation to solve reinforcement learning tasks, was initiated in 2015. This has led to the emergence of generic artificial intelligence learning from experience, capable of beating humans in Atari [200, 47] and Go [267]. Similarly, in neuromorphic engineering, both IBM and Intel – the two major chip manufacturing companies – released their neuromorphic chips, Loihi [84] and TrueNorth [41] respectively. Samsung also released a new neuromorphic vision sensor in [272]. The fact that industry leaders invest in neuromorphic technology witnesses its potential.

Spiking neurons can be seen as conventional artificial neurons, but with temporal dynamics and only communicating with 0s and 1s [214]. In that sense, the accuracy of SNNs is necessarily lower than that of ANNs which can communicate with real-values. So why do large technology companies invest in neuromorphic hardware? Because the constraint of spike-based communication enables hardware optimizations allowing low energy consumption and fast operation. The advantage of low energy consumption especially is critical, as it was shown that training a single deep network on a Graphics Processing Unit (GPU) emits more CO2 than an average car during its whole lifetime [277]. Research in learning with SNNs in sensorimotor loop is therefore essential to retain technological advances in a fossil-fuel-free world with reduced energy availability.

1.6. Thesis Organization

This thesis is structured as follows. In Chapter 2, the important background information to understand neurons, synapses, and vision, from a biological and a computational perspective is presented. Follows a review of the related work in Chapter 3. This review is divided into a computational neuroscience part, on state-of-the-art synaptic plasticity rules, and a robotics part, with state-of-the-art algorithms for event-based vision. The next Chapters are then dedicated to the work that was conducted within this thesis. Chapter 4 introduces the approach and the tools developed to address the research goals, used throughout the thesis as well as by other researchers in the field. The first research goal, learning spatio-temporal representations from event-based data is addressed in Chapter 5. The second research goal, mapping visual representations to motor commands is addressed in Chapter 6. Chapter 7 concludes the thesis. Three appendices are provided documenting SNN simulators (Appendix A), preliminary results on an adapted eRBP rule (Appendix B), and the blueprints of the neuromorphic DVS head developed within the thesis (Appendix C).

2. Background

The subject matter of this study is transdisciplinary. This thesis is intended for readers with a computer science background. This Chapter is therefore dedicated to providing the necessary background knowledge in biology, computational neuroscience, and neuromorphic engineering. The two Sections in this chapter make a parallel between the brain as a computational organ (Section 2.1) and artificial neural networks modeling these computations (Section 2.2). Since this thesis focuses on visuomotor tasks, this parallelism is extended to biological vision and neuromorphic vision sensors.

2.1. The Brain in Biology

Pioneer work that paved our modern understanding of the nervous system was initiated more than 100 years ago [242, 93]. The brain is the main organ of the nervous system, which performs computations leading to cognition. Nowadays, the mechanics of these computations can be observed with an impressive level of detail thanks to the progress in imaging techniques such as Electroencephalography (EEG) and Magnetic Resonance Imaging (MRI). Neuroscientists were able to draw large scale maps of different brain areas with respect to their functions. At the small scale, neurons and their synaptic connections were identified as elementary processing units. The link between these two scales – how individual neurons organize into large functional circuits – is the focus of current research [118].

2.1.1. Biological Neurons

The human brain has around 100 billion neurons. Neurons are cells in the nervous system, the main information processing unit. Information is primarily processed with chemical and electrical signals. A neuron is separated from its surroundings by a cell membrane permeable to some types of ions (electrically charged particles). The difference in ions between the interior and the exterior of the cell leads to a polarization of the membrane, called membrane potential. Specific ions, such as potassium (K^+) and sodium (Na^+) can cross the membrane through specialized channels. The crossing of a channel is driven by two forces:

chemical force (following the concentration gradient) and electrical force (following the voltage gradient). At rest, potassium and sodium ions cross the membrane in both directions [152]. Potassium concentration is higher within the cell, while sodium concentration is higher outside, leading to an equilibrium in membrane potential between $-30\,\mathrm{mV}$ to $-90\,\mathrm{mV}$.

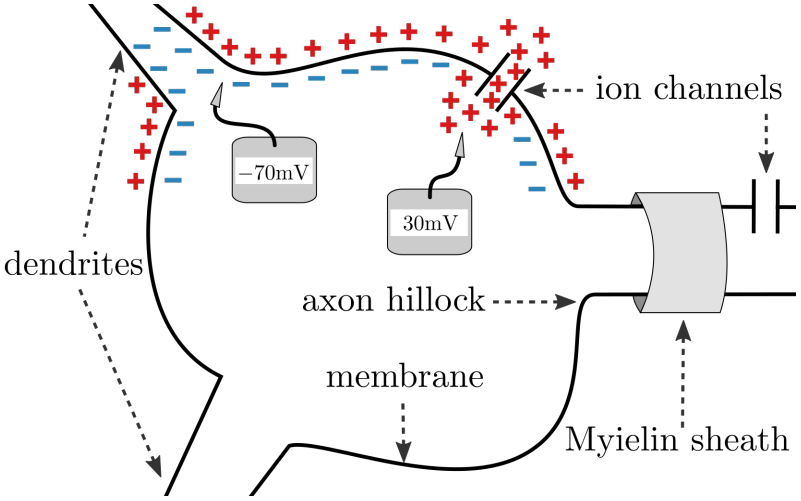


Figure 2.1.: Schema of the soma of a biological neuron (length varies from 10 to 100 microns). Opening and closing of ion channels alter the membrane potential locally.

The cell body of a biological neuron (called soma) is depicted in Figure 2.1. The opening and closing of new ion channels alter the membrane potential. Particularly, neural membranes have ion-specific voltage-gated channels, which open and close depending on the membrane potential. With a sodium voltage-gated channel, once the membrane potential reaches the threshold to open the gate, more sodium ions will flow through the neuron following the chemical and electrical gradient. This will lead the membrane potential to increase even further. Conversely, the presence of potassium voltage-gated channels which opens when the membrane potential is high will allow potassium to flow out of the neuron, leading the membrane potential to fall. These events – brief peaks in membrane potential – are called action potentials, or spikes, and constitute the main mechanism neurons have to communicate with each other. An action potential has a duration of few milliseconds and an amplitude of about $100\,\mathrm{mV}$. It travels across synapses to other neurons, affecting their membrane potential in turn.

The membrane potential is not the same on all the surface of a neuron. The anatomy of a neuron consists of multiple dendrites, a soma and an axon. Action potential at the soma (more precisely, at the axon hillock) will be conducted through the axon to other neurons. On the other hand, action potentials at the dendrites increase the somatic membrane potential in the soma, but may not be sufficient to trigger a somatic action potential, due to the dissipation of the ions, call electrotonic spread. In general, multiple simultaneous dendritic action potentials are required to trigger a somatic action potential.

While the extent of dendrites is limited to about $1\,\mathrm{cm}$ [100], axons can extend above $1\,\mathrm{m}$ in the human body. They are insulated with Myielin sheath preventing a traveling action potential to fade away, interleaved with voltage-gated sodium channels to amplify it. Most connections between neurons are with neighbors, while only a few connections are long-range.

2.1.2. Biological Synapses

Contact points between neurons are called synapses which are complex membrane junctions. The human brain has around 100 trillion synapses so that a neuron can be connected up to 10 000 other neurons. In most cases, the axon of the pre-synaptic neuron is connected to a dendrite of the post-synaptic neuron, although axon-to-soma and axon-to-axon connections also exist. Some synapses are electrical and allow ions to flow directly from a neuron to another. However, most synapses are chemical and transmit signals with molecules called neurotransmitters.

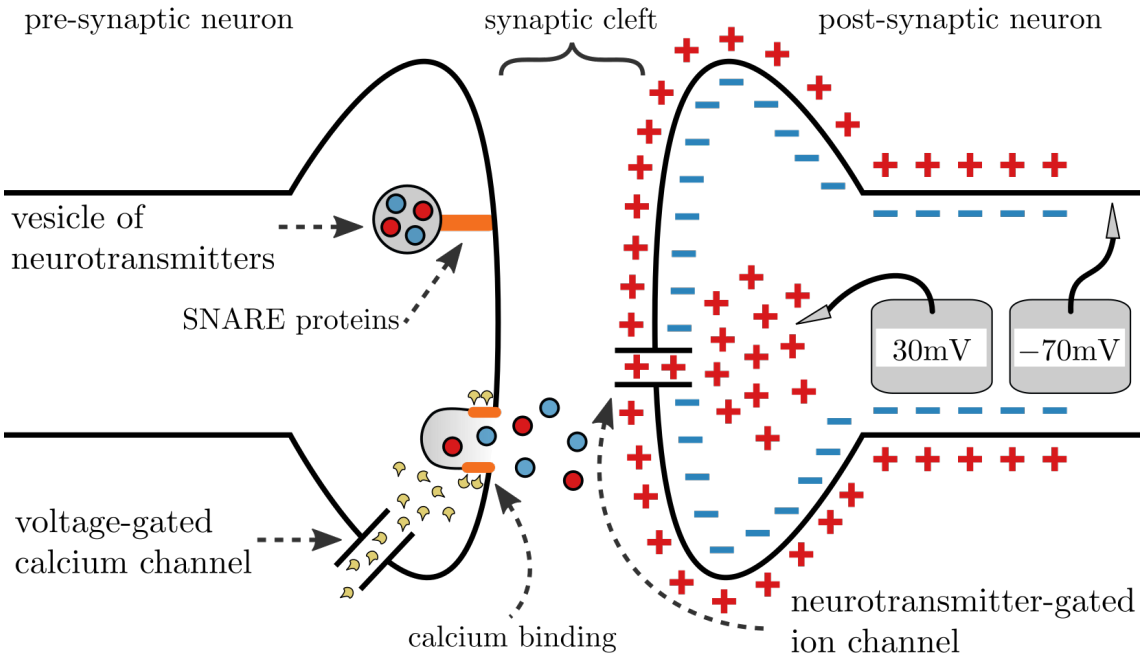


Figure 2.2.: Drawing of a biological synapse. The pre-synaptic neuron releases neurotransmitters which open ion channels on the membrane of the post-synaptic neuron.

A chemical synapse is depicted in Figure 2.2. A terminal-end of an axon contains vesicles of neurotransmitters attached to the membrane with SNARE proteins. In presence of a pre-synaptic action potential, voltage-gated calcium (Ca^{2+}) channels open, letting calcium ions flow into the axon. Calcium binds to the SNARE proteins, triggering the release of the neurotransmitters contained in the vesicles

outside the cell, in the synaptic cleft (the gap between the pre- and post- synaptic neurons). The post-synaptic neuron has neurotransmitter-gated ion channels, which only open in presence of some neurotransmitters. If these channels are sodium channels, sodium will flow in and the membrane potential of the post-synaptic neuron will increase, called excitatory Post-Synaptic Potential (PSP). On the other hand, if the neurotransmitter-gated channels are potassium-specific, then potassium will flow out and the membrane potential of the post-synaptic neuron will decrease, called inhibitory PSP.

2.1.3. Synaptic Plasticity

The amount of neurotransmitters in the pre-synaptic neuron as well as the number of receptors in the post-synaptic neuron varies in time. These two quantities are often grouped together in the term synaptic strength or synaptic efficacy. Changes in synaptic strength are referred to as synaptic plasticity or neuroplasticity. Synaptic plasticity alters the amplitude of the PSP response.

An increase in synaptic strength lasting from minutes to hours is called Long Term Potentiation (LTP). On the other hand, a decrease in synaptic strength is called Long Term Depression (LTD). The mechanisms underlying synaptic plasticity are still under active research. In 1949, Donald Hebb postulated in his book that synaptic efficacy increases when a pre-synaptic neuron repeatedly takes part in firing a post-synaptic neuron [129]. This is often summarized with the sentence "fire together wire together". However, while commonly used in literature, the term "hebbian learning" is vague and captures two separate concepts [49]: locality of information and functional expression. Importantly, synaptic plasticity can only depend on local information, available in their vicinity. Any information useful for learning – such as pain or reward signals – therefore need to be carried through learning channels.

Around the year 1997, it was discovered experimentally *in vitro* that the precise time of spikes plays a crucial role in synaptic plasticity [59, 186, 188, 171, 187]. This has led to the derivation of a variety of rules termed STDP to explain the experimental results. However, further experiments – including *in vivo* – have shown that the play between precise spike-time and synaptic plasticity was not as simple as originally formulated in [104, 177]. It is now assumed that STDP is a manifestation of an underlying learning framework yet to be discovered.

Neuromodulators are molecules in the brain which can also influence neural dynamics by altering neuronal and synaptic properties by targeting ion channels [209]. The effect of neuromodulators can be nonlinear and span over multiple timescales. While fascinating, it is not yet clear how neuromodulators should be incorporated into synaptic learning rules. An interesting research direction would be to investigate the role of neuromodulators in propagating local learning signals. However, this line of research is out of the scope of this thesis.

Aside from synaptic weight changes, new synapses can form between neighboring neurons, and existing synapses can retract. This process is called structural plasticity since it changes the topology of a network. Both synaptic plasticity and structural plasticity are believed to be the main processes responsible for learning [92, 76]. It is also assumed that the genome plays an important role to support learning mechanisms [294, 195], itself shaped by evolution and experience (through epigenetic modifications).

In summary, many forms of plasticity have been observed in the brain. However, it is not clear how these rules are orchestrated for functional learning to emerge. To progress in this field, it was suggested in [49, 264] that learning theory should drive new biological experiments, rather than the other way around. This thesis subscribes to this view by embodying learning rules on physical robot bodies to solve visuomotor tasks.

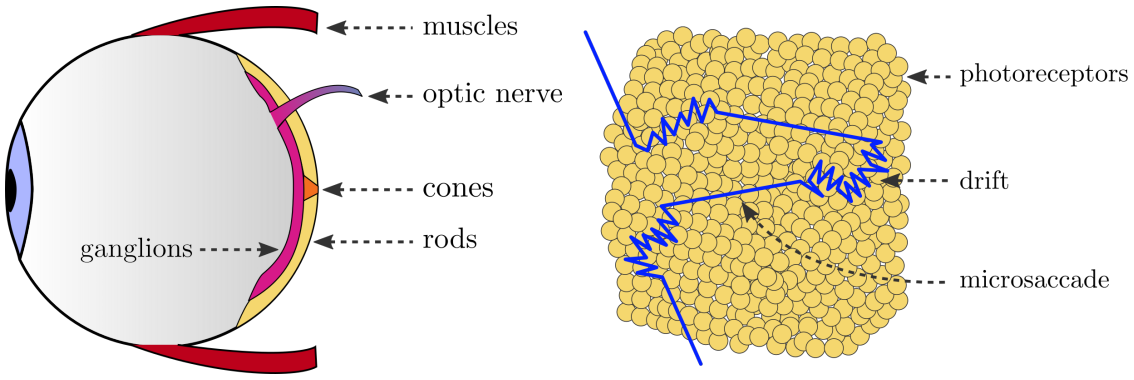
2.1.4. Principles of Biological Vision

The sense of sight is an important evolutionary advantage for all animals living in environments exposed to sunlight. Most animals forms have photoreceptors, and complex eyes evolved independently several times since the Cambrian explosion approximately 541 million years ago [164, 291, 218]. Eyes vary greatly across species, from replication of simple photoreceptors (facet eyes in insects) to very complex multi-pupils structure (as the mantis shrimp). This Section focuses on the mammalian eye and the encoding of visual stimuli to electrical signals in the nervous system through the retina, a process called transduction.

Retina

The eye is the organ responsible for acquiring, encoding, and transmitting important information about the light reflected in the environment. The retina, located on the back of the eye, contains the neural circuits dedicated to these tasks. The first layer of neurons in the retina consists of photoreceptors (cones and rods) that absorb light and transduce this signal into a change of membrane potential. Light is an electro-magnetic wave with an amplitude and a frequency. The human eye has three types of cones absorbing light of three different frequencies, interpreted as red, green, and blue. The cones are therefore responsible for our colored vision. There is only one type of rods, responsible for night vision, motion detection, and peripheral vision. The human retina has about 120 million rods and 6 million cones. Most cones are located in the center of the retina, called the fovea, while most rods reside outside of the fovea. The fovea represents 1% of the retina and only covers 5° in the center of the visual field, hence the necessity of eye movements [198].

The structure of the human retina is depicted in Figure 2.3a. The photo-receptor cells are connected to the bipolar and horizontal cells, which themselves connect



(a) Human eye (b) Fixational eye movements

Figure 2.3.: Human eye, the structure of the retina and eye movements. Fixational eye movements move the sensed light across photoreceptors.

to the amacrine cells and the ganglion cells. The signals from the ganglion cells travel outside of the retina through the optic nerve. There are about 1.5 million ganglion cells in a human retina [210]. These neural circuits form a pre-processing pipeline, from raw points (photoreceptors are equivalent to pixels) to higher-level visual attributes such as simple shapes, edges, and motions.

Neuroscientists distinguish at least 17 different types of ganglion cells which combine into separate parallel processing pathways. Two important pathways are the parvocellular pathway and the magnocellular pathway. The parvocellular pathway consists of specialized cells with high spatial resolution, perceiving details and colors, but a low temporal resolution. On the other hand, the magnocellular pathway consists of specialized cells with low spatial resolution, and no color perception, but a high temporal resolution. It was suggested in [179] that these pathways project to different layers of the visual cortex. The parvocellular pathway would be responsible for processing color, form and blob, and the magnocellular pathway for motion and depth. However, new observations such as the discovery of the koniocellular pathway disproved this hypothesis [268, 210]. It now appears that the different pathways project and combine in complex patterns in the visual cortex, no single pathway having the monopoly for a particular visual property.

Eye Movements

As noted in [163], nearly all animals with good vision – humans included – have a repertoire of eye movements. The goal of this Section is to provide an overview of the role of eye movement in visual perception, outlining applications of this knowledge to artificial, event-based vision systems. Movements of the eyes are generally the same in both eyes [198].

The first types of eye movements evolved to stabilize light on the photoreceptors (vestibulo-ocular reflex and the optokinetic response). These are common to most

animals with eyes.

Human eyes explore the visual scene with rapid movements called saccades. During a saccade, both eyes jump simultaneously with peak velocity around $900\,^\circ\,\mathrm{s}^{-1}$. Our eyes perform up to 3 saccades per second interleaved with fixations. The duration of a fixation varies depending on whether our attention is caught. It is believed that this exploration strategy evolved to prevent blur from rotational photo-receptor motion (1), ease the detection of movement (2) and disambiguate the flow-field to estimate depth from translational motions (3) (enumeration from [163]).

During fixation, the eyes are still in motion. Fixational eye movements refer to the movements of the eyes while we fix our gaze on an object. They are believed to play a major role in our visual perception. Presumably, a central role of fixational eye movements is to prevent visual fading. Visual fading describes the phenomenon of stationary objects fading from perception under perfect retinal stabilization and has been demonstrated in various laboratory experiments. Fixational eye movements generate neural activity at the level of retinal photoreceptors, by moving receptive fields over otherwise stationary stimuli [192]. Thus it is possible that the goal of oculomotor fixational mechanisms is not retinal stabilization, but controlled image motion optimal for visual processing.

Three kinds of fixational eye movements can be distinguished, as depicted in Figure 2.3b:

Tremor — Wave-like motion of the eyes with low amplitude (in the order of the diameter of a cone) and a frequency around $90\,\mathrm{Hz}$. Tremors may be sufficient to maintain retinal activity in the early visual system.

Drifts — Occur simultaneously with tremor and describe slow motions of the eye, across a dozen of photoreceptors.

Microsaccades — Small, fast, jerk-like eye movements during voluntary fixation which carry the retinal image across a range of several dozen to several hundred photoreceptors, and are usually about $25\,\mathrm{ms}$ in duration [192]. The amplitude of microsaccade is usually about 0.5° and there is a clear preference for horizontal and vertical directions [249].

Regarding the role of microsaccades, it has been argued that these jerk-like movements might contribute to perception by generating visual transients, which are important for the transmission of edge information in the human visual system [109]. See [249] for a more recent survey on the role of microsaccades.

Lastly, the eyes can also perform slower, smooth eye movements. Humans can track a moving target with smooth pursuit. This complex mechanism involves predicting the object's motion.

2.2. Modeling The Brain with Neural Networks

In the last years, neuroscientists acquired a broad knowledge about the electrochemical reactions happening in neurons and synapses. Nowadays, it is possible to simulate a neuron at a high level of accuracy, accounting for its geometry, its ion concentration, and its channels. However, this level of detail is impractical when the goal of the simulation is to accomplish function. Indeed, there's a clear trade-off between biological accuracy and required computational resources to simulate a neuron model. A better approach to achieve function with neural simulations is therefore to identify which characteristics of a neuron are important for its dynamics and how can they be simplified. This simplification requires to hypothesize which biological processes are functionally relevant and which ones are biological artifacts. This work is carried out by theoretical neuroscientists, also called "modelers". A neuron model often consists of a set of parameterized dynamical equations describing the evolution of a multi-dimensional state in time with respect to its input.

2.2.1. Neuron Models

The first computational neuron model was developed in 1943 by McCulloch and Pitts [196]. Their model consists of a weighted sum of the inputs followed by a Heaviside function, leading the output to be either 0 or 1 ("all-or-nothing"). This simple model leads to the more general framework of analog neurons currently used in deep learning. Like the McCulloch and Pitts neuron, an analog neuron consists of a linear sum followed by a non-linear activation function. The activation function can be chosen freely and output real numbers. In general, the activation function is nonlinear (so that an ANN can compute nonlinear functions) and differentiable (to allow gradient descent for learning). Despite the simplicity of the analog neuron model, it was proven that ANNs can approximate any function to an arbitrary level of accuracy (universal approximator) [135]. Mathematically, an analog neuron is formalized with the equation:

$$y_i = \phi\left(\sum_{j \in pre} w_{ij} \times y_j\right), \tag{2.1}$$

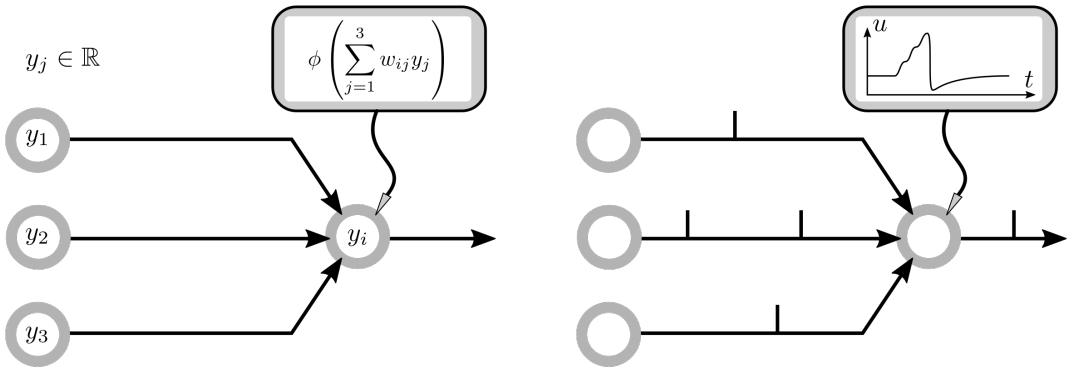
with ϕ the activation function, *pre* the indices of the pre-synaptic neurons, y_j their activations and w_{ij} their associated weights. An analog neuron is depicted in Figure 2.4a.

Which biological processes described in Section 2.1.1 do analog neurons model? For the McCulloch and Pitts neuron with the Heaviside activation function, the binary output of the neuron can be seen as the presence of an action potential in the axon hillock (a spike) since only these are transmitted to other neurons. The threshold at which the neuron changes its output from 0 to 1 is an abstraction for

the geometry of the membrane and its ion channels. Similarly, the real-valued output of an analog neuron is regarded as a rate of spikes.

It is important to note that a feedforward ANN does not take temporal dynamics into account. Indeed, a feedforward ANN is a function that maps a vector space to another, without a state. To solve this problem, recurrent connections can be added to the ANN to form a recurrent ANN. In this case, the computation of a recurrent hidden layer activity relies on its activity in the previous time-step, as well as the current previous layer activity. Such connections can be seen as providing a context for the computations, as in the networks discussed by Elman in [96], later referred to as Elman networks. It was proven that a recurrent ANN can implement any program with a finite number of units (Turing completeness) [266].

Despite the loose modeling of biological neurons, it seems that ANNs share similarities with biological neural networks [94, 290]. Additionally, at the functional level, they can solve a wide variety of tasks at super-human level [267, 200].



(a) Analog neuron (b) Spiking neuron

Figure 2.4.: Analog and spiking neuron models. Unlike spiking neurons, analog neurons do not take temporal dynamics into account. Analog neurons have real-valued activations resulting from a non-linear differentiable function ϕ applied to the sum of their weighted input. Spiking neurons have a membrane potential u evolving with respect to time t and depending on weighted incoming spikes.

On the other side of the spectrum, neuroscientists derive neuron models to understand their dynamics rather than solving computational problems. Some years after the McCulloch and Pitts neuron, Hodgkin and Huxley proposed their neuron model in 1952 [134]. The Hodgkin-Huxley neuron models the dynamics of the membrane potential in the giant axon of a neuron in the squid. This dynamics therefore does not apply to all cells, but gives insights into modeling of biological neurons. It accounts for the permeability of the cell (activation of the gated ion-channels) and the balance of sodium (Na), potassium (K) and unspecific leakage-ions (L), see Section 2.1.1. Specifically, the Hodgkin-Huxley model

is described with the following equations [134, 118]:

$$\begin{aligned} \tau_m \frac{du}{dt} &= -\sum_k I_k + I_{syn}(t), \\ \sum_k I_k &= g_{Na} m^3 h (u - E_{Na}) + g_K n^4 (u - E_K) + g_L (u - E_L), \\ \frac{dx}{dt} &= -\frac{1}{\tau_x(u)} \left(x - x_0(u)\right) \text{ for } x \in \{m, n, h\}. \end{aligned} \tag{2.2}$$

The membrane potential $u(t)$ depends on the internal ion currents I_k and the synaptic input current I_{syn}. The other three state variables m, n and h describe the permeability of the membrane – the current activation of voltage-gated ion-channels. The sodium-channels are controlled by m and h while the potassium-channels are controlled by n. The other variables are constants representing the ion-specific potential at equilibrium (E_{Na}, E_K and E_L, also called reversal potential), the ion-specific conductance (g_{Na}, g_K and g_L) and the membrane conductance τ_m.

The Hodgkin-Huxley neuron closely models biological processes with high accuracy compared to analog neurons. Importantly, with this model, neurons have multi-dimensional states evolving in time, as described by the differential equations in Equation (2.2). This complexity comes at the cost that it takes more resources to simulate, and harder to derive functional networks. All these considerations are reminiscent of spiking neuron models considered in this thesis.

Remarkably, the Hodgkin-Huxley neuron model generates action potentials of similar shape, amplitude and duration as the ones observed in the brain. However, most computational neuroscientists nowadays assume that the particular shape of an action potential is functionally irrelevant. In other words, action potential (or spikes) can be approximated with instantaneous events carrying no additional information but the precise time at which they occur. Neuron models relying on this hypothesis to simplify the neural dynamics are called phenomenological models. They do not provide a precise description of the electro-chemical reactions in the neural substrate but describe the behavior of a neuron in terms of input (currents) and output (spikes). These simplified models focus on the key properties of biological neurons, such as event-based communication and locality. They are employed by computational neuroscientists to reduce the computational cost of their simulation, while still allowing to generalize computational theories of the brain. Such models can also be obtained by reductions of more detailed models, such as the Izhikevich model introduced in [142] which simplifies the Hodgkin-Huxley neuron model to Integrate-and-Fire (IF) dynamics.

The IF neuron model is the most popular phenomenological neuron model, introduced as early as 1907 by Lapicque [165, 38]. As the Hodgkin-Huxley neuron, the IF neuron also hold a state variable reflecting the membrane potential u, but the dynamics of action potentials is neglected. Instead, spikes are generated in a ad

hoc fashion when the membrane potential crosses a threshold. This usually follows a reset of the membrane potential as well as a refractory period (a duration during which the neuron can not spike). As the Hodgkin-Huxley neuron, the IF neuron models the dynamics of the membrane potential as a capacitor [72]:

$$\begin{aligned} \tau_m \frac{du}{dt} &= -I_{leak}(t) + I_{syn}(t), \\ I_{leak}(t) &= u(t) - u_{rest}, \end{aligned} \tag{2.3}$$

with u the membrane potential, u_{rest} the resting membrane potential at equilibrium and $I_{syn}(t)$ a current describing the effect of synaptic input (incoming spikes). With this formulation, the current I_{leak} leaks exponentially.

Conventionally, spikes are emitted when the membrane potential u reaches a threshold θ_{thresh}. This is called a hard threshold: $s = \Theta(u - \theta_{thresh})$, with Θ the Heaviside function[1]. Alternatively, spikes can also be emitted stochastically: $P(s = 1|u) = \sigma(u)$, with σ the sigmoid function. After a spike, the membrane potential resets for a certain refractory time, typically chosen around $5\,\mathrm{ms}$.

The generic differential equation of a IF neuron can be rewritten using filters (temporal convolutions). In this case, we have:

$$u(t) = \eta * s(t) + \epsilon * I_{syn} + u_{rest}, \tag{2.4}$$

where $s(t)$ is the spike-train of the neuron and $*$ denotes a temporal convolution. The kernel η reflects the reset and refractoriness after a spike, while $\epsilon(t) = exp(-1/\tau_m) \times exp(-t/\tau_m)$ models the shape of PSPs. The spike-train of a neuron can be formulated as $s(t) = \sum_{f \in spikes} \delta(t - t_f)$, with δ the Dirac function[2] and t_f the times at which the neuron spiked. This formulation is the basis of the Spike Response Model (SRM) [117, 118]. The dynamics of the different terms of a SRM are depicted in Figure 2.5.

Spiking neuron models such as Hodgkin-Huxley and IF are referred to as point-neuron models, as depicted in Figure 2.4b. Conversely, multi-compartment models account for the geometry of the neuron by computing the membrane potential of dendrites separately. The increased biological plausibility comes at the cost of more computations, although in this case, researchers found functional benefits of modeling dendrites [255, 123]. conversely, mean-field models consist of modeling the activity of a population of neurons. These models are therefore less detailed than point-neuron models. In this thesis, we only relied on point-neuron models of the IF type. These are the most widely used by researchers and supported by neuromorphic hardware.

[1] Also called unit step function: $f(x) = 0$ if $x < 0$, otherwise 1

[2] Mathematical entity only meaningful when integrated: $\delta(x) = 0$ for $x \neq 0$ and $\int \delta(x)dx = 1$

2.2.2. Synapse Models

In ANNs, synapses – connections between neurons – are abstracted as weights. A weight is a one-dimensional real-valued variable often denoted as w. The set of all weights in the ANN are the parameters to train, although training is implemented in an ad hoc fashion with backpropagation. This weight is an abstraction for the amount of neurotransmitters in the pre-synaptic neuron and the number of receptors in the post-synaptic neuron. Indeed, these quantities have been observed to vary with respect to neural activity and are considered crucial for learning, see Section 2.1.3.

In SNNs, synapses model the synaptic current contribution I_{syn}^j of a neuron j after a spike. The synaptic input I_{syn} is the sum of the pre-synaptic current contributions. The simplest type of synapse multiplies the spike-train with a synaptic weight:

$$I_{syn}(t) = \sum_{j \in pre} I_{syn}^j = \sum_{j \in pre} w_j s_j(t), \tag{2.5}$$

with w_j the synaptic weight and $s_{j \in pre}$ the pre-synaptic spike-trains. In this case, a given spike leads to an instantaneous current injection of amplitude w_j in the membrane potential. The PSPs will then fade exponentially with the τ_m time constant of the neuron. Another common type of synapse provide α-shaped PSPs (see Figure 2.5):

$$\tau_{syn} \frac{dI_{syn}}{dt}(t) = -I_{syn}(t) + \sum_{j \in pre} w_j s_j(t). \tag{2.6}$$

This introduces another time constant, τ_{syn}, which can also be adjusted for every synapse. Other synapse types, such as conductance synapse, can also model I_{syn} as a function of the membrane potential u. In general, inhibitory synapses are simply modeled with negative weights.

Moreover, synapses in SNNs have a few more common properties compared to their analog counterparts. Synapses can connect to specific receptor types, having different effects on the post-synaptic neuron. They can also delay the transmission of a spike, accounting for the length of the axon and the length of the dendrite. On top of biological plausibility, these delays also have functional advantages to perform temporal computations such as optical flow and stereo-vision with event-based data [44, 126].

Additionally, synapse models can also consist of multi-dimensional states evolving in time [161]. This state can be used functionally to prevent forgetting [298] or store information for learning [296, 55]. Since the chemical reactions in biological synapses are rather complex (see Section 2.1.2), such a state is biologically plausible. However, maintaining a synaptic state in simulation is computationally expensive since the number of synapses grows quadratically with the number of neurons.

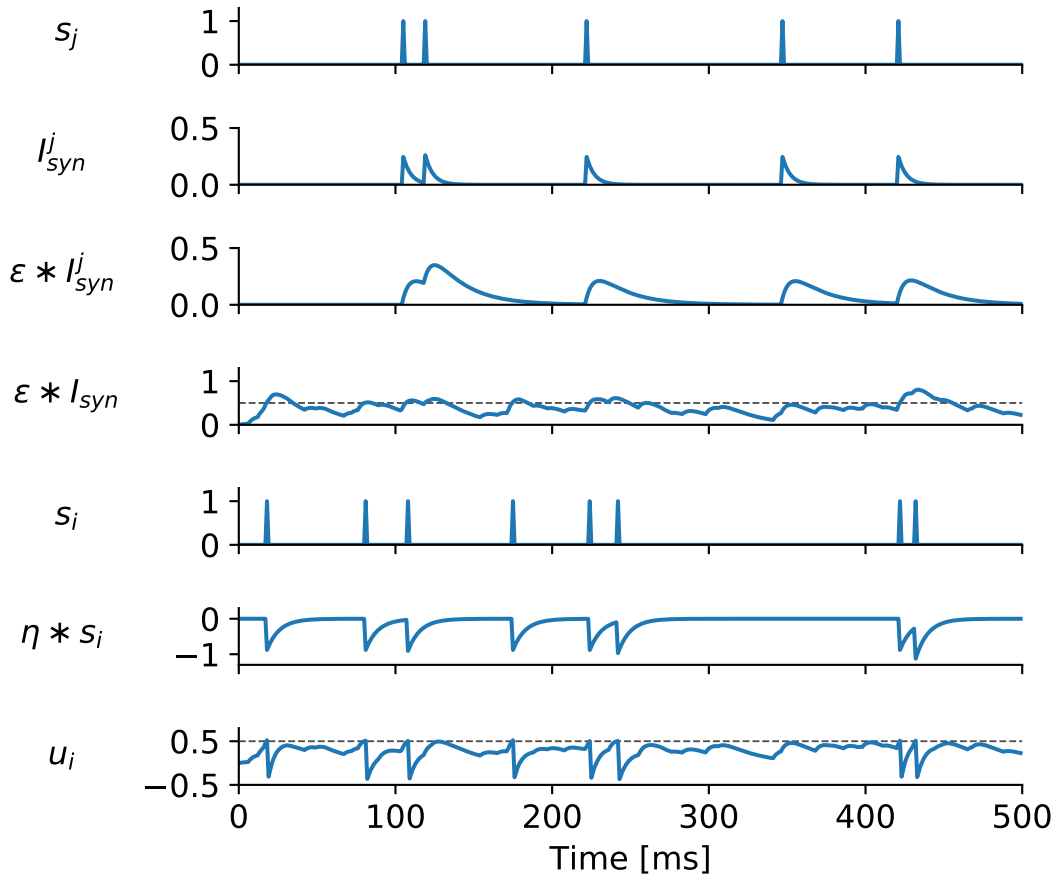


Figure 2.5.: Example dynamics of the individual terms of a SRM neuron i with pre-synaptic neurons j. The PSPs are α-shaped, see Equation (2.6). The threshold $\theta_{thresh} = 0.5$ is depicted with dashed lines.

2.2.3. Synaptic Learning Rules

Synaptic learning rules model synaptic plasticity by formalizing synaptic weight changes. These rules can be evaluated by their biological relevance and their ability to solve a particular task, typically minimization of a loss function. As discussed in [49], synapses in the brain can only access local information. The most generic formalization of a synaptic learning rule is therefore:

$$\Delta w_{ij} = f(\text{local variables}). \tag{2.7}$$

The concept of locality of information is also crucial for neuromorphic hardware. Indeed, while computer memory significantly improved in the last years in terms of storage and density, the latency to fetch information has largely remained the same. Nowadays, this latency is much higher than the processing speed of standard Central Processing Unit (CPU). This problem is referred to as the von Neumann bottleneck. Neuromorphic hardware address this bottleneck by distributing computations to different physical locations (digital chips or analog cir-

cuits). The information required for updating the neural and synaptic dynamics is stored in small and local memory, physically close to the computational process that needs it. Neuromorphic hardware can therefore achieve a remarkable processing speed and energy efficiency.

Which information is considered local? This is ultimately determined by the biological system, but realistic assumptions have to be made by the modeler. It is reasonable to assume that pre-synaptic neuron activity (y_j), post-synaptic neuron activity (y_i) and the synaptic weight itself are local properties. The original hebbian rule, as well as Oja's rule [223] can be formalized with no further information:

Simple Hebbian	$\Delta w_{ij} \propto y_i y_j,$
Oja [223]	$\Delta w_{ij} \propto y_i y_j - y_i^2 w_{ij},$

where $\propto$ denotes proportionality, to account for the learning rate. Oja's rule is a modification of the simple Hebbian rule which regulates weight growth and was shown to compute the principal component of the input [223].

Importantly, if an information is considered local but does not originate from the synapse itself, then this information has to be physically transported to the synapse. This is the concept of learning channels introduced in Baldi et al. [49]. Usually, a learning channel is used to convey an error signal, providing information about the current performance at the task:

Perceptron [251]	$\Delta w_{ij} \propto (T - y_i) y_j,$
Backpropagation [254]	$\Delta w_{ij} \propto B_i y_j,$

with T a supervised target and B_i the post-synaptic backpropagated error. The problem of assigning responsibility to a neuron in the current task performance in the form of an error signal is referred to as the credit assignment problem. Solving this problem is a major challenge when deriving new learning rules, particularly when considering multi-layer networks. Additionally, the credit assignment is not only spatial but also temporal since SNNs have temporal dynamics: a spike leaves a lasting trace in the network state.

The formalization of the learning rules so far introduced was based on ANNs, but can be adapted to SNNs with small adjustments. SNNs distinguish themselves with spike-based communication and continuous time dynamics. This way, pre-synaptic y_j and post-synaptic y_i activity refer to the spikes. Synaptic plasticity rules computing weight updates with respect to precise spike-times of pre-synaptic and post-synaptic neurons are denoted with STDP. These rules were developed after the discovery that precise spike-time plays a crucial role in synaptic plasticity [116, 188]. Usually, STDP rules are formalized as follows [125]:

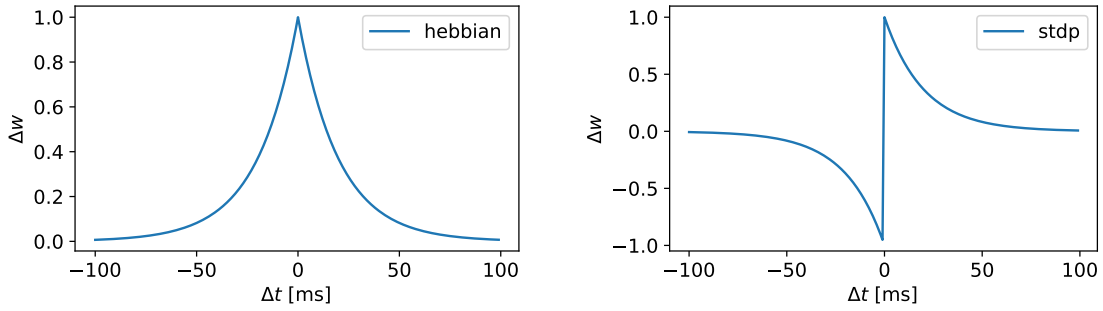
$$\Delta w_{ij} \propto f(w) \times e^{\frac{-|\Delta t|}{\tau}}, \tag{2.8}$$

with $\Delta t = y_i - y_j$ the time difference between a post-synaptic and pre-synaptic spike pair, τ a time constant (usually around $20\mathrm{ms}$) and f a function. Setting

$f(w) = k > 0$ leads to the spiking version of the Simple hebbian learning rule, see Figure 2.6a. A slightly different formulation allows asymmetric weight updates by distinguishing the cases when $\Delta t > 0$ (pre- before post-synaptic spike) and $\Delta t \leqslant 0$ (post- before pre-synaptic spike):

$$\Delta w \propto \begin{cases} f_-(w) \times e^{\frac{-|\Delta t|}{\tau}} & \text{if } \Delta t \leqslant 0 \\ f_+(w) \times e^{\frac{-|\Delta t|}{\tau}} & \text{if } \Delta t > 0 \end{cases}. \tag{2.9}$$

In general, causal firing (pre before post) increases synaptic efficacy, and acausal firing decreases synaptic efficacy [188], see Figure 2.6b. The learning rules relying on this framework are by nature unsupervised and spike-driven. As for ANNs, more advanced learning rules which solve the credit assignment require more information to be considered local.



(a) Simple hebbian

(b) Original STDP

Figure 2.6.: Models of STDP observed in [129, 59, 188] and described by Equations (2.8) and (2.9) with $\tau = 20\text{ms}$.

Since learning rules are derived to minimize errors, they are often solely evaluated on pattern classification tasks. The focus of this thesis is to evaluate them in a visuomotor context.

2.2.4. Neuromorphic Vision

The effort to build *in silico* the first stages of retinal processing started in 1988 with Mead and Mahowald [197]. This effort, still pursued today by many researchers, has led to the development of neuromorphic vision sensors – or silicon retinas. These sensors offer a new paradigm in the field of vision: event-based processing. Conventional cameras sense complete frames at regular time intervals. Conversely, neuromorphic vision sensors emit events upon local light intensity changes. In its simplest form, an event consists of a location (pixel coordinate), a timestamp of when the change occurred and a polarization flag denoting whether light intensity increased or decreased.

The first neuromorphic vision sensor to become available on the market was the DVS presented in 2008 by Lichtsteiner et al. [173]. Unlike traditional cameras, every pixel is integrated into a dedicated circuit consisting of a photoreceptor circuit, a differencing circuit and a comparator circuit. The photoreceptor circuit converts input current from the photodiode into a voltage logarithmically. It has individual pixel gains allowing a high dynamic range and responds quickly to illumination changes. The differencing circuit amplifies these changes and the comparator circuit emits ON and OFF events when the changes cross a threshold.

Nowadays a few research institutes and companies now build such sensors [173, 239, 272], although development is still in an early stage. Compared to conventional cameras, silicon retinas have multiple advantages:

Temporal resolution
: Events are sensed and transmitted in the order of $10\,\mu s$. In contrast, the temporal resolution of standard cameras is in the order of $10\,ms$.

Dynamic range
: Since pixels are independent, the range of sensed light intensity is $140\,dB$ (compared to $60\,dB$ for standard cameras).

Energy efficient
: The stream of events is very sparse compared to frames, leading to low power consumption. The same is true for biological brains and on neuromorphic hardware in general.

Low redundancy
: Only new information – what changed in the scene – is provided. Thus, online event-based algorithms have fewer data to process than their frame-based counterparts.

All these advantages are meaningful for the field of robotics.

The concept of frames (or images) does not exist in event-based representation. Since the field of computer vision mainly focused on processing frames, most vision algorithms can not be used directly to process events. Additionally, a stream of events recorded by an event-based sensor necessarily has a temporal component. Therefore, any event-based algorithm needs to account for time.

The current silicon retina prototypes also have disadvantages. At the time of writing, the spatial resolution does not exceed 640×480 pixels, achieved by Samsung in [272]. The prototype used throughout this thesis, the DVS, has a spatial resolution of 128×128 pixels [173]. Most of these sensors do not sense color and only provide information about light intensity (gray-scale). Additionally, aside from ATIS [239] and DAVIS [67], these sensors do not provide an absolute measurement of the light intensity at all, just relative changes.

3. Related Work in Neuroscience and Event-Based Vision

The field neurorobotics is young and overlaps with computational neuroscience, neuromorphic technology and robotics. This thesis is about processing event streams with SNNs. The methods that this thesis relies on were mostly inspired by the field of computational neuroscience. Therefore, a review of the state-of-the-art synaptic learning rules is given. Subsequently, a review of the state-of-the-art methods to process event streams, with and without SNNs is provided. Finally, an overview of the neurorobotics visuomotor tasks in literature were event-based vision was used to control robots is given.

3.1. Synaptic Learning Rules

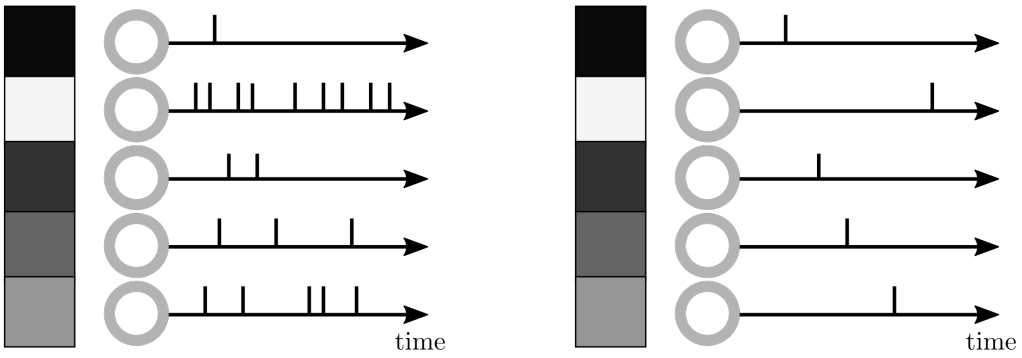
At the time of writing this dissertation, there is still no consensus in the neuroscience community on how the brain learns and what computations it performs. Instead, new synaptic learning rules are regularly derived with various degree of biological plausibility and functionality. These rules are usually evaluated on pattern classification tasks. In this Section, an overview of the state-of-the-art synaptic learning rules that inspired this thesis is given.

3.1.1. Learning Representations with STDP

We introduced STDP in Section 2.2.3, see Equations (2.8) and (2.9). This description yields an unsupervised learning rule where the weight change only depends on pre- and post-synaptic spike times. While the relevance and simplicity of this rule are discussed, it remains a highly influential synaptic learning rule [187]. A recent review on training SNNs with STDP is available in [236].

It has been successfully demonstrated that STDP is capable of learning visual representation from images in SNNs [154, 58, 139, 88], and can be implemented in hardware [262]. Most of these work encode images with Poisson spike-trains fed to the SNN, as depicted in Figure 3.1a. This encoding consists of having one input neuron per pixel which is set to spike at a given frequency depending on its brightness according to a Poisson distribution. An image is presented to the SNN

for a specified duration, usually around $300\,\mathrm{ms}$, with a pause between images for the SNN to return to its resting dynamics.



(a) Rate-coding (b) Rank-order coding

Figure 3.1.: Encoding analog values to spikes

Accuracy above 90% is reached for the MNIST dataset in [154, 58, 139] using a supervised learning signal. Indeed, despite that the STDP rule is by nature unsupervised, a training signal forcing the desired post-synaptic neurons to spikes will increase the synaptic weights from the spiking pre-synaptic neurons. This technique was used in Tieck et al. [21] to learn grasping motions with a 5-finger robotic hand. A major drawback of this approach is that it supports only a single learning layer since such supervision signals are usually not known for the intermediate layers. Therefore, the presented approaches are limited to one or two hidden layers.

In [88], an unsupervised STDP rule is coupled with a winner-take-all circuit to achieve better performance on MNIST than the previous work. The architecture consists of two recurrent layers, one excitatory and one inhibitory. The input neurons are connected to all excitatory neurons. Every excitatory neuron is connected to a corresponding inhibitory neuron which will inhibit all other excitatory neurons in the previous layer. By letting the network process images with STDP, the initial random preferences of the excitatory neurons for particular features are increased through competition imposed by the inhibitory layer. Classification is achieved by associating a label for every excitatory neuron with respect to the digit to which they are the most tuned (obtained by counting the spikes during a presentation of all digits). The method reaches a final accuracy of 95% on the MNIST dataset in an unsupervised fashion with STDP.

STDP in Convolutional Architectures

A series of work focused on training convolutional multi-layered SNNs to learn robust visual representations with STDP in an unsupervised fashion [193, 153,

194]. The latest work formalizes STDP as follows [154]:

$$\Delta w = \begin{cases} a^- w(1-w) & \text{if } \Delta t \leqslant 0 \\ a^+ w(1-w) & \text{if } \Delta t > 0 \end{cases}. \tag{3.1}$$

This formulation of STDP imposes soft bounds on the weights between 0 and 1 and does not depend on the precise value of Δt. In this work, the images were encoded with a latency code (bright pixels spike first) instead of a rate code (bright pixels spike more) used in most other works. This type of encoding is denoted as rank-order coding, depicted in Figure 3.1b. This representation allows the network to classify very fast since no integration time is needed to count spikes, consistent with biology [279, 178, 138]. Additionally, the activity of the network is very sparse with only around 1000 spikes per image, and features are learned from a few samples. The spikes emitted by the network are fed to a Support Vector Machine (SVM) which reaches 98.4% accuracy on MNIST and 82.8% accuracy on the ETH-80 dataset which only contains 410 samples per class for 8 classes. On the same dataset, a convolutional ANN of the same structure reaches 81.9% with supervised backpropagation, demonstrating the ability of the approach to learn from a few samples.

Other works focused on learning generative models with variants of STDP. In [230], a method based on an auto-encoder is presented. The learning rule moves the synaptic weights of the SNN to reproduce the input. With the addition of a supervised dense layer, the network accuracy reaches 99.08% on MNIST.

Voltage-based STDP

Biological experiments have shown that synaptic plasticity in the brain did not only depend on precise spike-times but also other factors such as frequency and cooperativity [270]. This has led modelers to augment the original STDP rule with new local information to explain the observations obtained in biological experiments. Such development has led to the Clopath rule proposed by Clopath et al. in [78, 79]. This influential rule augments STDP with a parameter modeling the post-synaptic membrane potential. Following the notation of this thesis, a simplification of the rule can be expressed as:

$$\frac{dw(t)}{dt} \propto \underbrace{\bar{s}_j(t)[\bar{u}_i^+(t) - \theta^-]^+[u_i(t) - \theta^+]_+}_{\text{LTP}} - \underbrace{s_j[\bar{u}_i^-(t) - \theta^-]^+}_{\text{LTD}}, \tag{3.2}$$

with $u_i(t)$ the post-synaptic membrane potential, $\bar{\cdot}$ denoting low pass filtering and $[\cdot]_+$ a linear rectification. It was shown that this rule could fit a variety of biological synaptic plasticity experiments, highlighting the importance of modeling the post-synaptic potential in the plasticity rule. Interestingly, a number of recent top-down synaptic plasticity rules – presented in Section 3.1.4 – rely on the post-synaptic membrane potential to compute the synaptic gradients.

Reinforcement Learning with STDP

So far, unsupervised and supervised learning rules based on STDP have been presented. By integrating a modulatory reward term in the STDP equations, a reinforcement learning rule can be derived.

Only a small number of work is related to reinforcement learning with spiking neural networks while addressing the previous points. Groundwork of reinforcement learning with spiking networks was presented in [143, 102, 169, 105]. In these papers, a mathematical formalization of how dopamine modulated STDP (DA-STDP) solves the distal reward problem with eligibility traces is introduced. Specifically, the brain needs a form of memory to reinforce previously chosen actions since the reward is received only after a rewarding action is performed. This problem is solved with the introduction of eligibility traces, which assign credit to recently active synapses. This concept has been observed in the brain [106, 229] and is used throughout this thesis. Fewer works evaluated DA-STDP in an embodiment for reward maximization – a recent survey encompassing this topic is available in [63].

3.1.2. Probabilistic Inference through Neural Sampling

Biological experiments have demonstrated that neurons and synapses in the brain have a stochastic behavior, yielding a high trial-to-trial variability [250]. This observation has led computational neuroscientists to study the computations performed by SNNs using probability theory. In particular, a series of work using STDP to learn probabilistic models are based on the neural sampling framework introduced in Buesing et al. [69]. This framework enables SNNs to estimate probability distributions through sampling. Many tasks can be formulated in the context of estimating probability distributions, which intrinsically model the uncertainty about the environment. For example, classification tasks can be reduced to the problem of estimating the probability that the sensory input was caused by a given item.

The authors show how neural dynamics implement Markov chain Monte Carlo (MCMC) sampling. Sampling is accomplished by constructing a Markov chain that has the desired distribution as its equilibrium. A Markov chain consists of a series of states $M = (Z_1, \ldots, Z_t)$, where the transition from one state to another is given by a stochastic transition operator $T(Z_t|Z_{t-1})$. Under reasonable assumptions, and after finitely many steps, it can be shown that the probability of being in a given state $p(Z_t = Z)$ does not depend on the initial state Z_1: the probability distribution over the state has reached the equilibrium.

In a SNN with continuous dynamics, the state of the network $Z(t)$ consists of the state of its neurons: $Z(t) = (z_1(t), \ldots, z_n(t))$. The state of a neuron k is a random

variable with a binary state:

$$z_k(t) = \begin{cases} 1 & \text{if neuron } k \text{ fired recently} \\ 0 & \text{otherwise} \end{cases}. \tag{3.3}$$

An additional time constant is introduced to determine the duration for which a neuron remains active after a spike. Usually, this time constant is set to the refractory period of the neuron. This framework imposes constraints on the neural dynamics. In particular, the membrane potential has to respect the neural computability condition:

$$u_k(t) = log \frac{p(z_k(t) = 1 | Z_{\setminus k}(t))}{p(z_k(t) = 0 | Z_{\setminus k}(t))}, \tag{3.4}$$

with $Z_{\setminus k}(t)$ denoting the state of the rest of the network, aside from neuron k. It is shown that linear neural dynamics correspond to Boltzmann distributions. Additionally, this framework requires the neurons to have a probabilistic spike function of the form $P(s = 1|u) = \sigma(u)$. This framework enables probabilistic inference by clamping spike-trains to a set of neurons and observing the activity of the remaining ones.

Bayesian Computations with STDP

In [216], Nessler et al. show how a variation of the STDP rule implements a stochastic version of the Expectation-Maximization algorithm in winner-take-all circuits, enabling Bayesian computations. The architecture of a winner-take-all circuit consists of two feedforward layers (input and output) of excitatory neurons. Neurons in the output layer mutually inhibit each other, a pattern referred to as lateral inhibition, yielding competition. Output spikes of the winner-take-all circuit are interpreted as the Expectation step and the induced synaptic weight change as the Maximization step. Only the synaptic weights between the input and the output layers are trained, using the following variant of the original STDP rule:

$$\Delta w = \begin{cases} e^{-w} - 1 & \text{if the pre-synaptic neuron fired recently} \\ -1 & \text{otherwise} \end{cases}. \tag{3.5}$$

This rule leads the synaptic weight to converge to the log probability of a recent pre-synaptic spike, assuming a post-synaptic spike. After learning, the synaptic weights associated with a post-synaptic neuron describe a generative model for the input. In this sense, the post-synaptic neuron fires when the synaptic input resembles its generative model. The inference is performed by clamping spike-trains on the input neurons and computing the output spikes. In Bayesian terms, every input spike encodes an evidence for an observed variable, and every output spike is a sample from the posterior distribution. The authors successfully demonstrate the learning rule on multiple unsupervised tasks including handwritten digit recognition.

Event-Driven Contrastive Divergence (eCD)

A synaptic learning rule extending the neural sampling framework and approximating the Contrastive Divergence algorithm is introduced in [211], called eCD. This rule is used to train spiking Restricted Boltzmann Machine (RBM) in an unsupervised fashion. The rule is formulated as a modulatory hebbian STDP as follows:

$$\Delta w \propto g(t) \times e^{\frac{-|\Delta t|}{\tau}} \tag{3.6}$$

with $g(t)$ a global modulatory signal with value -1 (unlearning), 0 (not learning) or 1 (learning). The modulatory signal imposes a temporal structure on the training regime, as distinct phases are required in the presentation of a given sample. Specifically, four sequential phases are described during the presentation of a sample:

1. Burn-in phase: $g(t) = 0$ and the input is clamped on the input neurons;
2. Positive Hebbian phase (LTP): $g(t) = 1$, learning the input distribution;
3. Burn-out phase: $g(t) = 0$;
4. Negative Hebbian phase (LTD): $g(t) = -1$, unlearning the model distribution.

In this case, the network architecture is necessarily based on the RBM structure: two-layers network with bidirectional symmetric weights. Also, the spiking neurons require to be in high conductance state as derived in [235] to sample from the Boltzmann distribution. This is achieved by exposing IF neurons with external high-frequency noise. In this high conductance state, the neuron shows stochastic firing of sigmoidal shape, determined by the input current and the noise frequency. As with analog RBM, trained networks can be stacked on each other to form a deep belief network. This method achieves 91.9% accuracy on MNIST in an unsupervised fashion, against 92.6% with an identical ANN trained with standard Contrastive Divergence.

This rule is improved in [215] by introducing stochasticity in the synapses, implementing a synaptic sampling method. The authors model synapses with a probability ($\approx 50\%$) of dropping spikes from a pre-synaptic to a post-synaptic neuron. It is shown that this simple addition significantly improves learning by decreasing overfitting.

Synaptic Plasticity with Online Reinforcement learning (SPORE)

Synaptic Plasticity with Online Reinforcement learning [150] (SPORE) is an instantiation of the synaptic sampling scheme introduced in [150, 149]. This framework improves on the neural sampling framework of Buesing et al. [69] by additionally introducing stochasticity in the synaptic dynamics. Unlike conventional reinforcement learning objective, SPORE does not converge to a local maximum

policy but instead continuously samples different solutions from a target distribution. The target distribution sampled by SPORE is defined as the expected future discounted reward coupled with a prior term imposing sparsity constraints on the network parameters.

Generally, the prior term is modeled as a Gaussian centered around 0: $\mathcal{N}(0, \frac{1}{c_p})$. The target distribution therefore peaks at biologically plausible parameter vectors θ that likely yield high reward. Additionally, a temperature parameter T allows to make the distribution flatter (high exploration) or more peaked (high exploitation). Let's $r(t)$ denote the received reward at time t and y_i the pre-synaptic spike train of synapse i filtered with a PSP kernel. A SPORE synapse i requires three dynamic variables to perform its local update:

$$\begin{aligned}
&\text{(eligibility trace)} && \frac{de_i(t)}{dt} &&= -\frac{1}{\tau_e} e_i(t) \;+\; w_i(t)\, y_i(t)\, (z_{post_i}(t) - \rho_{post_i}(t)) \\
&\text{(reward gradient)} && \frac{dg_i(t)}{dt} &&= -\frac{1}{\tau_g} g_i(t) \;+\; r(t)\, e_i(t) \\
&\text{(synaptic parameter)} && d\theta_i(t) &&= \beta \left(c_p(\mu - \theta_i(t)) + c_g\, g_i(t) \right) dt \;+\; \sqrt{2T_\theta \beta}\, \mathcal{W}_i
\end{aligned}$$

where $z_{post_i}(t)$ is a sum of Dirac delta functions placed at the firing times of the post-synaptic neuron and $\rho_{post_i}(t)$ is the instantaneous firing rate of the post-synaptic neuron at time t. The synaptic weight w_i is given by the projection:

$$w_i(t) \;=\; \begin{cases} w_0 \, \exp(\theta_i(t) - \theta_0) & \text{if } \theta_i(t) > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{3.7}$$

with scaling and offset parameters w_0 and θ_0, respectively.

In [150], SPORE is evaluated on a binary classification task. The network receives a reward when it makes the correct guess. After $3\,\mathrm{h}$ of learning, the network policy reaches about 82% of the maximal reward. It was shown in [293] that SPORE was able to learn 2D trajectories with a terminal reward. In this setup, the SNN controls a 2D agent initially located in the center of a simulated water-maze. The goal of the agent is to reach a given location in the maze (a platform) upon receiving a go-cue. After $20\,\mathrm{h}$ of learning, the network reaches about 80% of the maximal reward with annealing.

3.1.3. Recurrent Networks

So far, the presented approaches were either feedforward or constrained recurrent networks, such as RBMs [211] or winner-take-all circuits [88, 216]. Only a few approaches were derived to train generic recurrent SNNs.

Liquid State Machines (LSMs)

Liquid State Machines (LSMs) were the first and simplest recurrent architecture for learning with SNNs, presented in [184]. A LSM consists of three distinct parts: an input layer, a liquid, and readouts, as depicted in Figure 3.2. Neurons in the input layer connect randomly to neurons in the liquid. The liquid contains recurrent connections providing a fading memory property. The recurrent connections are classically not trained and seen as a random spatio-temporal kernel projecting the input to a high-dimensional space. The liquid state is defined by the activity of some neurons in the liquid. This liquid state is linearly mapped to the readout neurons. Only this linear mapping is learned, usually in supervised fashion with least-square regression [181]. This paradigm is referred to as reservoir computing. A similar approach was presented simultaneously and independently for ANNs called echo state networks [146].

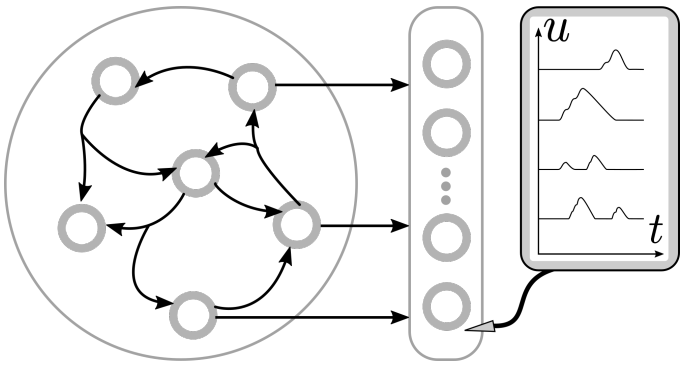


Figure 3.2.: Liquid state machine, also known as reservoir. Recurrently connected pool of neurons with a readout layer.

The core concepts behind reservoir computing are ingeniously demonstrated in [99]. In this work, the authors demonstrate how a linear classifier can solve the Exclusive Or (XOR) problem by projecting input to a high-dimensional space. Specifically, the input consists of two motors that can hit a bucket filled with water. In this case, the liquid is physical water. The liquid state is captured with a camera overseeing the surface of the water. It is shown that a perceptron can learn the XOR problem between the two motors by projecting learning a projection from the high-dimensional camera pixels. This work demonstrates that the water performed a non-linear projection of the input. Indeed, it is well-known that XOR can not be solved solely with a linear projection. Subsequently, LSMs have been used in applications ranging from decoding actual brain activity [217] to control robots [281, 71, 240, 23]. In general, however, tuning hyper-parameters to obtain a well-behaved liquid for the sensory input is not trivial.

In a similar fashion, LSMs have also been used to predict future input in [183]. In this case, the movements of an object are simulated on an 8x8 sensor array. The liquid consists of a SNN built with empirical data from microcircuits in the somatosensory cortex of a rat. The liquid states were sampled every $5\,\mathrm{ms}$. Different readouts are trained to predict whether the object is a ball or a bar, where the

object will exit the frame and the future activity of the sensors $50\,\mathrm{ms}$ in the future. This work demonstrates how the liquid state abstracts a lot of information that can be recovered with linear readouts. This work was extended in [70] with real camera images pre-processed and down-sampled to an 8x6 sensor array.

More recently, it was shown that a set of echo state networks could be trained to learn spatio-temporal patterns from event-based data in [159]. Each echo state network predicts a single motion over a 17x17 sensor array. This is achieved with a winner-take-all circuit ensuring that the networks do not learn redundant predictions. The predictions are provided for one time-step in the future. It is shown that this architecture is capable of learning sparse spatio-temporal patterns.

Lastly, it was also shown in Tieck et al. [23] that a LSM could be used in combination with reinforcement learning methods for a closed-loop robot control target reaching task. In this setup, the LSM is used as a state transformation of the input, which is fed to a reinforcement learning algorithm called Proximal policy optimization [259]. The readouts of the LSM are trained together with the reinforcement learning method.

Simple Recurrent Neural Networks

Recurrent ANNs consist of classical ANNs with recurrent connections leading to memory. Generic recurrent ANNs can be trained with Backpropagation-Through-Time (BPTT), which consists of unrolling the temporal computations into spatial computations and applying backpropagation. In other words, the activity of a recurrent hidden layer at a given time-step t is represented as a node in the computational graph. This node receives input from the previous layer at time t, and from the hidden node at time $t-1$. It provides the input to the hidden node at time $t+1$. This requires to store the activity of the recurrent hidden layer for all times in the sequence.

Simple recurrent ANNs consist of ANNs with limited recurrent connections. Elman networks [96] are a subset of recurrent neural networks with one bipartite hidden layer. The hidden layer is split between hidden neurons connected to the input and output, and context neurons only connected to the hidden neurons. Elman networks are trained with a learning rule applied at each time-step.

Long Short-Term Memory (LSTM)

The recurrent connections of a recurrent ANN can also be redefined to explicitly layout meaningful temporal computations. The Long Short-Term Memory [132] (LSTM) is a new building block aside from neurons, introduced in Hochreiter et al. [132]. An Long Short-Term Memory [132] (LSTM) unit consists of a cell, an input gate, an output gate, a forget gate, and a cell state. The cell is the persistent state of the unit. The three gates control this state depending on the instantaneous

input. This normalized propagation of the activation from a time-step to another addresses the problem of keeping or resetting the memory. It was shown in [133] that LSTM could solve non-trivial algorithmic problems.

The concept of LSTM was only very recently adapted to SNNs in Bellec et al. [54]. In later work, an alternative approach to Backpropagation-Through-Time (BPTT) more suited to SNNs was proposed in [55]. This rule, called Eligibility Propagation [55] (e-prop), is of the same family as the DECOLLE rule presented in Section 5.5, but additionally handles recurrent networks of adaptive-threshold neurons.

3.1.4. Spiking Backpropagation

Almost all successful deep learning applications rely on the backpropagation algorithm [254] for learning with gradient descent. The first equivalent derivation for SNNs was introduced in Bohte et al. [64]. However, the original formulation of the backpropagation rule is biologically implausible [50] and not suitable for SNNs. The main arguments against the biological plausibility of backpropagation are listed in Bengio et al. [57]:

1. The feedback phase consists of only linear computations (while in biological neurons, linear and non-linear computations are interleaved);
2. The feedback phase requires knowledge of the forward activation functions;
3. The feedback phase requires knowledge of the forward weights;
4. Computations are not based on spikes;
5. The forward and feedback phases alternate synchronously;
6. It is not clear where the supervised targets come from.

To these points, we can add the following:

7. It is not clear how to solve the temporal credit assignment locally.

This last point refers to the fact that spiking neurons have temporal dynamics, leading network output to depend on previous neural activity. Therefore, computing the gradient implicitly requires information about previous network activities, a non-local operation in time.

In the same work, they propose modifications to the backpropagation rule addressing points (1) to (6). Indeed, more recent work has shown that backpropagation (as well as BPTT) can be implemented in a biologically plausible manner for SNNs [213, 296, 55], including our own work [12]. All these rules address the previous points in a similar fashion, including the extra point (7). Point (2) is solved by realizing that the precise knowledge of the forward activation is not necessary and can be approximated, a method called pseudo-derivative or surrogate gradient, see Neftci et al. [214]. This is crucial since the activation function of a IF

neuron is the Heaviside function which is not differentiable (another solution is to use a soft threshold [137]). Point (3) is addressed with random feedback weights, a method introduced by Lillicrap et al. and Nøkland et al. [174, 219]. Point (5) is solved by storing the information needed for learning at the synapse directly, allowing spike-driven weight updates. Different approaches were proposed for point (6), currently under very active research. Point (7) is addressed with Real-Time Recurrent Learning (RTRL), a technique introduced in 1989 in Williams et al. [287] and recently adapted to SNNs in Zenke et al. [296]. The remaining points are solved intrinsically. A full derivation of our learning rule DECOLLE is provided in Section 5.5.1 which formally shows how these techniques are used in practice.

In most of the recent spiking backpropagation derivations for IF neurons, the resulting synaptic learning rules have the form:

$$\Delta w_{ij} \propto F(y_j, u_i, E_i), \tag{3.8}$$

with u_i the membrane potential of the post-synaptic neuron and E_i an error signal. The symbol $\propto$ refers to a proportionality relation – the multiplicative constant is the learning rate which can be chosen freely. These types of rules are called voltage-based since they assume that synapses have access to the voltage of the post-synaptic neuron. Commonly, these three terms would be multiplied together, in which case we refer to these rules as three-factor, consistent with biology [237, 261].

These recent developments have led the community to reconsider whether backpropagation could be used in the brain and how. This hypothesis is supported in the latest work from Lillicrap et al. [176], published in Nature Neuroscience in April 2020. In the next sections, state-of-the-art alternatives to backpropagation for SNNs are discussed.

Event-Driven Random Backpropagation (eRBP)

In Neftci et al. [213], the authors demonstrated Event-Driven Random Backpropagation [213] (eRBP) which is a form of approximate gradient backpropagation in SNNs that translates into a three factor rule reminiscent of an error-modulated Hebb rule. For analog networks with a mean square error loss, the backpropagation rule is described as follows for a weight from neuron j to neuron i:

$$\Delta w_{ij}(t) \propto y_j \times \phi' \left(\sum_{j' \in pre} w_{ij'}(t) y_{j'}(t) \right) \times E_i(t) \tag{3.9}$$

with y_j the output of neuron j, *pre* the set of pre-synaptic neurons, ϕ the activation function of neuron i and $E_i(t)$ the error for neuron i. This rule is interpreted for spiking neurons with y_j the spiking output (0 or 1) of neuron j and ϕ the spike

function mapping the membrane potential to spikes, which is the unit-step function Θ [213]. The weighted sum of input spikes $\sum_{j' \in pre} w_{ij'}(t) y_{j'}(t)$ is interpreted as the membrane potential I_i of neuron i. This leads to a voltage-based rule, as the weight update depends on the membrane potential of the post-synaptic neuron, similar to the Clopath rule presented in Equation (3.2). This rule is a three-factor rule with y_j the pre-synaptic term, $I_i = \sum_{j' \in pre} w_{ij'}(t) y_{j'}(t)$ the post-synaptic term and $E_i(t)$ an error term. However, the computation of the error term $E_i(t)$ is non-local and involves backpropagating errors from the output layer back to the input layer. This backpropagation requires knowledge of the forward weights, causing a weight transport problem.

In the last years, it became apparent that the computations of $E_i(t)$ could be relaxed, solving the weight transport problem [174, 219, 144]. Specifically, it was noted in Lillicrap et al. [174] that the feedback weights to backpropagate the errors can be decoupled from the forward weights propagating the activations. It was additionally shown that the network could adapt to fixed random feedback weights, a method called (). Subsequently, it was shown in Nøkland et al. [219] that the feedback weights could be directly connected from the network's output to all hidden neurons, bypassing the previous hidden layers. This technique, appropriately called (), computes the error term as follows:

$$E_i(t) = \sum_{k \in out} e_k(t) g_{ik}, \tag{3.10}$$

with e_k the prediction error for output neuron k, *out* the set of output neurons and g_{ik} a fixed random feedback weight. This rule was adapted to SNNs for the first time in [213], named eRBP, requiring $\mathcal{O}(n_{neurons})$ extra synapses implementing the random feedback. In other words, the neurons in eRBP calculate their error from the spikes of the error neurons, which are connected back to all neurons in the network, see Figure 3.3. These synapses have random fixed weights, and the error is integrated in a dedicated compartment of the neuron.

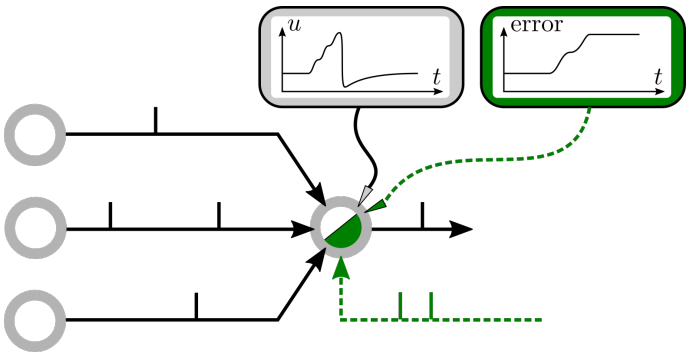


Figure 3.3.: eRBP IF neuron. The credit of this neuron is stored in a dedicated error compartment. It evolves with respect to error spikes generated at the output layer of the network.

Additionally, the non-differentiability of Θ induced by the hard-threshold of spiking neurons for emitting spikes was solved with surrogate gradients [214], see

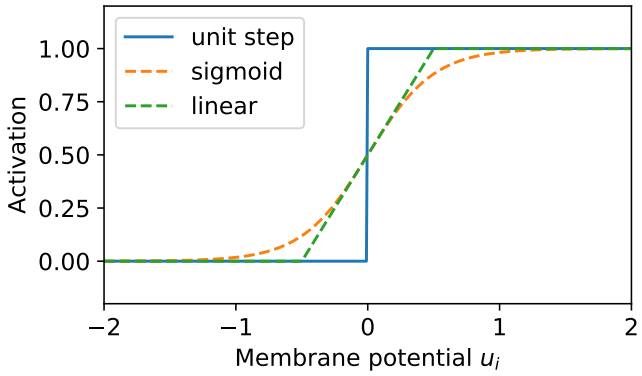


Figure 3.4.: Common approximations to the unit-step function to compute the gradient of the spike by the membrane potential.

Figure 3.4. Specifically, an approximate derivative of Θ was used – in this case a boxcar function, equal to 1 between b_{min} and b_{max} and 0 otherwise. Therefore, this rule can be efficiently implemented in a SNN with weight updates triggered by pre-synaptic spikes y_j as:

$$\Delta w_{ij}(t) \propto \begin{cases} \sum_{k \in out} e_k(t) g_{ik} & \text{if } b_{min} < I_i(t) < b_{max} \\ 0 & \text{otherwise} \end{cases}, \tag{3.11}$$

Since eRBP only uses two comparisons and one addition for each pre-synaptic spike to perform the weight update, it allows a real-time, energy-efficient and online learning implementation running on neuromorphic hardware. Note that the temporal dynamics of the IF neuron – PSPs and refractory period – are not taken into account in Equation (3.11).

SuperSpike

SuperSpike – derived in Zenke et al. [296] – improves over eRBP by mathematically deriving backpropagation from the equations of the IF neuron. The authors show that the errors of every synapse can be integrated online as part of the neural dynamics. This method, called RTRL, was originally introduced in 1989 for recurrent ANN [287]. It enables the differentiation of temporal dynamics by storing states (interpreted as eligibility trace) instead of a history of previous activities. Temporal correlations of different duration can therefore be learned with constant memory requirements. This contrasts with BPTT, where space complexity scales with respect to the duration of the temporal sequence.

Using the SRM formulation in Equation (2.4), the authors calculate the eligibility trace as pre-synaptic spikes convolved with the post-synaptic kernel ϵ. Relying on the van Rossum distance as a loss function, SuperSpike can train multi-layer networks to learn a target spike-train. However, this setup requires one eligibility trace per synapse, leading SuperSpike to scale temporally and spatially as

$O(N^2)$, where N is the number of neurons. While the complex biochemical processes at the synapse could account for the quadratic scaling, it prevents an efficient implementation in digital hardware. In Section 5.5, we introduce DECOLLE which improves over SuperSpike by requiring only one eligibility trace per neuron. Additionally, this eligibility trace is integrated into the forward dynamics computations, leading the spatial complexity of DECOLLE to be constant ($O(1)$), see Algorithm 3. Another improvement over SuperSpike is proposed with e-prop derived in Bellec et al. [55], which computes the gradient of adaptive threshold neurons, providing a long-lasting memory to the SNN.

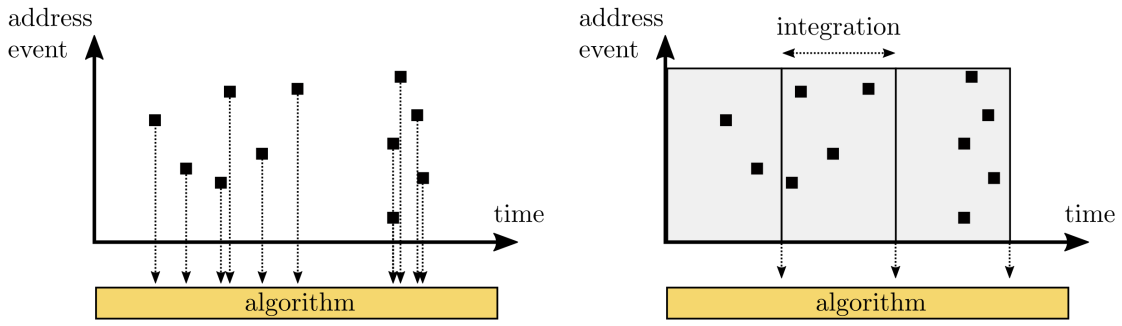
3.2. Neuromorphic Vision Applied to Robotics

The field of computer vision and its application to robotics is tailored to frame-based representation provided by conventional cameras. On the other hand, event-based vision sensors rely on AER, see Section 2.2.4. The divergence between frame-based data and event-based data entails that traditional computer vision algorithms can not be used to directly process events. Neuromorphic engineers have three possibilities to resolve this problem:

- Convert event-based data to frames by integrating events, enabling traditional computer vision algorithms and ANNs to be used. This integration prevents low latency computations.
- Derive new asynchronous algorithms to process events directly and individually, allowing low latency computations.
- Process events with SNNs, a subgroup of asynchronous algorithms, biologically inspired and potentially running on dedicated neuromorphic hardware.

In some sense, even the conversion approach still relies on an event-based algorithm to integrate events into frames. Ingenious integration methods have been derived to minimize the loss of consequent temporal information. A frame-based algorithm is then fed the current integration frame at given intervals. On the other hand, asynchronous event-based algorithms do not explicitly create frames but process events directly. Such approaches still need a form of memory of the previous events, which can be stored as an asynchronously updating frame. SNNs can be considered as a subgroup of asynchronous algorithms but deserve their own section (Section 3.2.3) in the scope of this thesis. A recent survey on event-based vision proposing a similar – although not identical – categorization is provided in [111]. We depict these categories of algorithms in Figure 3.5.

AER can reduce the amount of data to process compared to conventional video frames. This is an important advantage for embedded applications and robotics. Learning visual representations from event streams is a field of growing importance as can be witnessed by the increasing number of event-based datasets



(a) Event-by-event algorithms (b) Frame-based algorithms

Figure 3.5.: Event-by-event and frame-based methods for processing event-based data. Frame-based methods integrate events over time intervals, inducing latency.

[292, 263, 43, 172, 136, 226, 299, 269]. These datasets contain labeled event streams of either scenes involving motion (such as DvsGesture [43]) or static images perceived by a moving sensor (such as N-Caltech101 [226]).

3.2.1. Conversion from address events to frames

The simplest method to convert events to a frame consists of aggregating them during a determined time interval. This integration can be seen as a histogram since the number of events is quantized per pixel for a given time interval. Usually, such integration leads to a two-channel frame – one channel for each polarity. A frame-based algorithm is then run on every frame, thus at a constant frequency defined by the integration time interval. This simple approach has some obvious drawbacks. Mainly, there is a trade-off in the selection of the integration time interval. Longer intervals contain more temporal information per frame since more distant events are integrated. Conversely, longer intervals also destroy more temporal information, since individual event timestamps are not considered. Especially, the same movement performed at different speeds will yield drastically different frames.

Many simple but effective alternatives have been derived to resolve some of these drawbacks. In [180], frames are created by integrating a fixed number of events instead of integrating events for a fixed amount of time. This technique has the advantage to provide similar frames for the same movements performed at different speeds. The frames are then provided to a convolutional ANN for gesture classification. With this integration method, the network automatically runs at a higher frequency when there is more motion in the scene, from 1 to 1000 Hz.

More complex approaches often rely on computing Surface of Active Events, as introduced in [39]. The Surface of Active Events is a spatio-temporal structure storing the last event timestamp for every pixel. Usually, two such surfaces are

needed – one per event polarity. In this space, moving edges will appear as planes with slope dependent of their velocity, and noisy events will appear isolated. This technique was used in [204] to estimate the "life-time" of every event with respect to their velocity on the image plane. Sampling a frame at a given time consists of reporting all events alive at this time.

Recently, a generic framework was presented in [115] which can be used to characterize multiple event-integration methods. The method consists of filtering the events with a kernel (similar to the computation of the membrane potential, see Section 2.2.1) for a given measurement. The filtered event stream is then sampled on a regular grid, to obtain a spatio-temporal tensor (called Event Spike Tensor). For example, considering event timestamp as the measurement leads to the Surface of Active Events. Multiple measurements can then be stacked together. An advantage of this method is that all the steps are differentiable, allowing an ANN to learn at the event level by updating the filters. This method achieves state-of-the-art classification on N-Cars [269] and N-Caltech101 [226] using the pre-trained ResNet-34 architecture [128], as well as state-of-the-art optical flow estimation on the MVSEC dataset [299]. One drawback of the approach is that it requires events to be aggregated, preventing low latency computations as could be provided by a SNN.

In [245], a recurrent ANN is proposed to convert event streams to reconstructed gray-scale video frames. It is shown that the network can learn from simulated data as obtained by the ESIM simulator [244]. This technique allows classic computer vision methods to be used on event-based data.

In [114], an optimization method is proposed to find point trajectories on the image plane by warping events. The optimal warping aligning events in spatio-temporal space is obtained by minimizing the spread in the warped image. The parameter of this warping can later be used to perform various computations, such as estimating the optical flow, the depth, or the ego-motion of the sensor. This framework is extended in [112] by providing a thorough evaluation of the different loss functions that can be used to perform the alignment.

3.2.2. Asynchronous event-by-event processing

In this Section, we briefly report some of the problems and methods from the literature processing address events asynchronously. Most of these approaches are based on probabilistic filters since filters integrate information and operate asynchronously by design [111]. Since these methods work without integrating events, they can run with lower latency and are therefore preferred for high-speed control loops [86, 82].

The problem of tracking corners in an event stream with event-by-event methods is tackled in [203, 283, 77]. Corners do not suffer from the aperture problem and are therefore a feature of choice to leverage event-based algorithms. In [203],

corner detection is performed with few computations, allowing millions of events to be processed per second on a CPU. The method consists of finding cliffs in the Surface of Active Events, a characteristic trace of corners. Cliffs are detected if arcs of surrounding pixels can be formed so that pixels on these arcs are higher (more recent) than all others.

Tracking corners can be used to build and update a map of the environment, crucial for the localization problem. Estimating the pose of the sensor from an event stream using event-by-event methods was undertaken in [286] for planar motions and [205, 207, 113] for 6-DOF (more approaches were proposed to solve this problem by integrating events into frames). Generally, the observed events are matched to entities (edges or corners) in a known map, and the pose transformation is updated. In [205], a continuous-time formulation is proposed by modeling the pose with cubic splines. This allows the pose to be interpolated mathematically with respect to discrete control points estimated by the method. Additionally, it is demonstrated that the absolute scale can be jointly estimated using an inertial measurement unit (IMU).

Recognizing objects from event streams is proposed in [160]. In [160, 269], the concept of time surfaces is introduced. Time surfaces describe the recent history in the spatial neighborhood of an event. Patterns are then learned from the time surfaces in a supervised fashion with feature hierarchy in [160] and histograms in [269]. Time surfaces are equivalent to the membrane potential of input neurons as defined in Section 2.2.1. Indeed, synaptic dynamics also exponentially filter their input events and play the role of time surfaces. However, SNNs can have a hierarchy of layers allowing the sharing of representations, which is not the case when learning time surface prototypes.

Since event-by-event methods can process event-based data with lower latency than frame conversion approaches, they were already used in fast control loops. In [86], an asynchronous algorithm is presented to control a small robot goalkeeper with $3\,\mathrm{ms}$ reaction time at a 4% CPU load. The method relies on tracking clusters of events. In [82], two DVSs are used to balance a pencil upright. The method consists of evaluating the inclination of the pencil with filters using the Hough transform. In [120], a similar event-based Hough transform is proposed, combined with an optical flow method, for tracking a sphere with a moving DVS.

3.2.3. Neuromorphic Vision from Spikes

SNNs are a natural fit to process event-based data since events can be easily converted to spikes. In this sense, SNNs belong to the category of asynchronous event-by-event processing algorithms. SNNs have only been applied to a limited set of problems compared to other event-by-event algorithms discussed in Section 3.2.2. The particularity of SNNs are:

- computations are distributed across neurons;
- computations are ruled by dynamical equations;
- communication is based on instantaneous, stereotypical spikes.

Additionally, neural networks can represent any functions or algorithms by altering their parameters, making them predisposed to learning. The NRP together with the DVS Gazebo plugin developed in Kaiser et al. [15] and described in Section 4.2.2 ease prototyping of visuomotor neurorobotics experiments.

In this Section, we discuss how SNNs were used in literature to process events. Three distinct categories of approach can be identified:

- *Hand-tuned networks*: SNN with static weights (non-learning) engineered to solve a particular task;
- *Converted networks*: conventional ANNs that were trained with backpropagation synchronously and subsequently converted to SNN;
- *Plastic networks*: SNNs which learn from spikes with synaptic plasticity rules.

The last category is the most ambitious since it requires the derivation of novel biologically plausible learning rules. This field is beyond the scope of neuromorphic engineering and robotics alone and is mainly studied in computational neuroscience. In this Section, we will only describe the synaptic learning rules that were applied to the field of event-based vision. A broader overview in the field of biological learning rules is given in Section 3.1.

Hand-tuned Spiking Neural Networks

Some basic features of the visual cortex are believed to be hard-wired from evolution rather than learned from interactions with the environment. This assumption is verified in insects that rely on pre-wired microcircuits to compute optical flow [185]. Similarly, many hand-tuned networks were derived to solve specific tasks such as stereo vision [44, 238, 90, 227] and optical flow [252, 225] since SNNs are computationally powerful but difficult to train. In both case, event-based sensors provide a clear advantage over conventional frame-based approaches: precise time of events is used as an additional constraint for matching. For stereo vision, a recent survey of such approaches is available in Steffen et al. [20].

SNNs solving the stereo vision problem have been proposed in [44, 238, 227, 90] and are based on Poggio and Marr's cooperative algorithm for stereo matching, which was published in 1982 [189]. They consist of a three-dimensional spiking network where output neurons describe one unique point in the observed 3D-space. In other words, an output neuron emits a spike when the location in 3D-space becomes occupied or unoccupied.

The stereo-matching process is described in [190]. It is also referred to as the correspondence problem. First, a point of interest is selected in one image. Second, the same point is identified in the other image. Third, the disparity between the two points is measured, yielding the distance of the object. In practice, matching a point from one frame to another is a difficult problem. This problem is simplified by considering the following physical constraints. *Uniqueness constraint* (C1): for every given point seen by one area of one eye, at a specific time, there can be at most one corresponding match in the other. *Continuity constraint* (C2): physical matter is cohesive and generally has a smooth surface. *Compatibility constraint* (C3): the interest point should look similar in both images. These constraints were translated into a SNN performing disparity computations from event-based data in [90] by relying on micro-ensembles.

Hand-tuned SNNs were also proposed to approximate other functions. Particularly, the Neural Engineering Framework (NEF) provide a simple method to approximate any function with a two-layer SNN of IF neurons. The method consists of encoding multi-dimensional analog values in populations of neurons and computing the optimal decoder approximating the function. The resulting networks can then be combined to form complex cognitive architectures such as Spaun in [274]. Similarly, a SNN approximating the Fast Fourier Transform was proposed in [148] to detect cylinders on a snake robot from event-based data in simulation. This method relies on the NRP and the DVS Gazebo plugin developed in Kaiser et al. [15] and presented in Section 4.2.2.

Similarly, hand-tuned SNNs are also commonly used by neuroscientists to build models of brain regions to approximate a biological function. In [65], a SNN modeling the early regions of the visual cortex is proposed for a segmentation task with crowding and uncrowding. This work is extended in Bornet et al. [3] with a bottom-up attention mechanism transmitting segmentation signals. An embodied evaluation in the NRP on the iCub robot shows that the model performs comparably to humans in behavioral experiments. Recently, in [48], a hand-tuned SNN controller modeling the regions of the human brain involved in eye movements was presented This controller is demonstrated in a stereo tracking task where a robotic head equipped with two cameras centers a moving laser dot in the fovea of both eyes.

Converted Spiking Neural Networks

The other method to go around training a SNN from spikes consists of converting a conventional ANN. Following this approach, training can rely on traditional deep learning methods and backpropagation. Compared to the original ANN, the SNN resulting from the conversion has lower accuracy, but higher energy efficiency with dedicated neuromorphic hardware. This workflow is the core concept behind IBM TrueNorth [41], which is energy efficient but does not support

on-chip learning. The goal then becomes to make the conversion without unacceptable performance loss. This involves tailoring the training process of the ANN to ease the conversion.

Some common challenges when converting an ANN to a SNN involves dealing with negative values. ANNs might use the sigmoid or the tanh activation functions, which produce negative values. Negative outputs could be implemented by inhibitory neurons, but inhibited neurons do not spikes thus do not communicate their value to subsequent layers. In practice, this problem is solved by relying on positive activation functions, such as Rectified Linear Unit (ReLU), which these days are the most commonly used activation function anyway.

In [73], a convolutional ANN with ReLU activation is tailored by adding an absolute value function to the pre-processing step. Furthermore, all biases are set to zero at all times during training since biases are hard to translate to spiking neurons. The Max Pooling operation is substituted with Average Pooling, which is easier to implement in a SNN. The conversion results in a small loss of performance. Similar approaches have been used for event-based vision in [234, 221, 89, 98, 253].

Spiking Neural Networks with Synaptic Plasticity

State-of-the-art synaptic learning rules are discussed in Section 3.1. Some of the presented methods were successfully applied for learning visual representations from images, relying on rate-based or latency-based encodings. Which methods were successfully applied to learn vision from event-based data?

In [60], a variation of STDP is used to learn features from simulated and real event-based data. All synapses are depressed on a post-synaptic spike, except the ones that recently sent a pre-synaptic spike. The authors show that neurons tune themselves to specific chunks of the motion observed by the event-based sensor in an unsupervised manner. This tuning only happens with a proper balance of lateral inhibition – allowing neurons to learn different parts of the motion – and refractory period – preventing a neuron to learn multiple chunks. The method is evaluated in a multi-layer network, with two different strategies: global learning (all layers learn together) or layer-by-layer. By optimizing the hyper-parameters with a genetic algorithm, it is shown that the method successfully learns to detect cars on specific lanes with high accuracy.

In [151], a new method for learning spatio-temporal features with SNNs is introduced. The method consists of evolving the network by adding neurons and synapses to capture new features, jointly with a variation of STDP. It is demonstrated that the method can learn to classify two types of motions – "crash" and "no crash" – from event-based data in an unsupervised fashion.

A spiking backpropagation method called SLAYER is introduced in [265] and trained on event-based dataset in a supervised fashion. The method shows state-of-the-art classification accuracy, 93.64% on DvsGesture with an 8-layers architecture. However, this method learns offline and scales spatially as $O(NT)$, where T is the number of time-steps. The method to be introduced in Section 5.4 and Section 5.5 achieves equivalent or better results, learns online and does not scale with the number of time-steps.

In [276], a set of reflexes for obstacle avoidance was implemented in a SNN using the NEF for a mobile neuromorphic robot, the pushbot. These reflexes include going back when too close to an obstacle, turn left or right at a medium distance from an obstacle, go forward if there is no obstacle. The distance with obstacles is estimated with a laser pointer mounted on the pushbot, visible from an embedded DVS also mounted on the pushbot [81]. The SNN receives as input the position of the laser pointer in the event stream, computed with classical algorithms. Complex behaviors are learned offline by gathering the sensorimotor data when the pushbot performed desired actions and training a new behavior from this data in a supervised fashion. The pushbot learns to turn either left or right in a T-maze depending on the presence of a visual clue, here a mirror.

The simulated lane following experiment that we describe in Section 4.3.1 was improved in [62]. The authors evaluate DA-STDP (referred to as R-STDP for reward-modulated STDP) in a similar lane following environment. Their approach outperforms the hard-coded Braitenberg vehicle presented in Section 4.3.1. The two motor neurons controlling the steering receive different (mirrored) reward signals whether the vehicle is on the left or the right of the lane. This way, the reward provides information on what motor command should be taken, similar to a supervised learning setup. The method is improved in [61] with a Q-learning strategy and in [122] with a deep convolutional architecture for feature extraction.

3.3. Summary and Conclusion

In this Chapter, it has been shown that neuromorphic vision sensors with AER open the door to a new family of asynchronous algorithms. These asynchronous algorithms either collect events into frames or process events directly and individually. The former achieves higher accuracy in general, whereas the latter enables very low latency, important for high-speed robotics. The field of robotics is increasingly adopting the event-based paradigm for various tasks such as localization, mapping, object avoidance, optical flow, depth perception, and action recognition. SNNs constitute a subset of asynchronous algorithms but are underrepresented to process event-based data in robotics.

Simultaneously, the field of computational neuroscience regularly develops advanced synaptic learning rules relying on spike-based communication. Initially,

the modeled learning rules were solely accounting for some types of plasticity observed in *in vivo*, such as STDP and Hebbian. These type of learning rules are capable of training shallow networks to solve specific tasks with appropriate tuning, but can not explain on their own the complexity of biological learning. Enabled by recent findings in deep learning, new synaptic learning rules were derived inspired by the success of backpropagation for ANNs. Strikingly, these rules provide a framework orchestrating plausible types of plasticity as modulated by an error signal. Additionally, these learning rules are capable of training deep SNNs to achieve high accuracy in a sample-efficient manner.

The link between computational neuroscience, event-based computations and robotics is still in its early stage. Only a few synaptic learning rules have been evaluated on event-based data and integrated into a robotics closed-loop setup. Indeed, there is a performance gap between approaches based on SNNs and other approaches based on algorithmic or ANNs. However, the latest developments in computational neuroscience as well as the brain itself are good indicators that this gap can be closed.

4. Basic Approach to Visuomotor Neurorobotics

Neuromorphic vision sensors convey visual information differently than conventional cameras, as discussed in Section 2.2.4. Silicon retinas emit address events upon local light intensity change at the precise time of the change. Time itself represents information in event-based representations. Likewise, SNNs process information in an asynchronous and distributed manner, where precise spike-time matters. These computational paradigms contrast with conventional algorithms and computer vision methods.

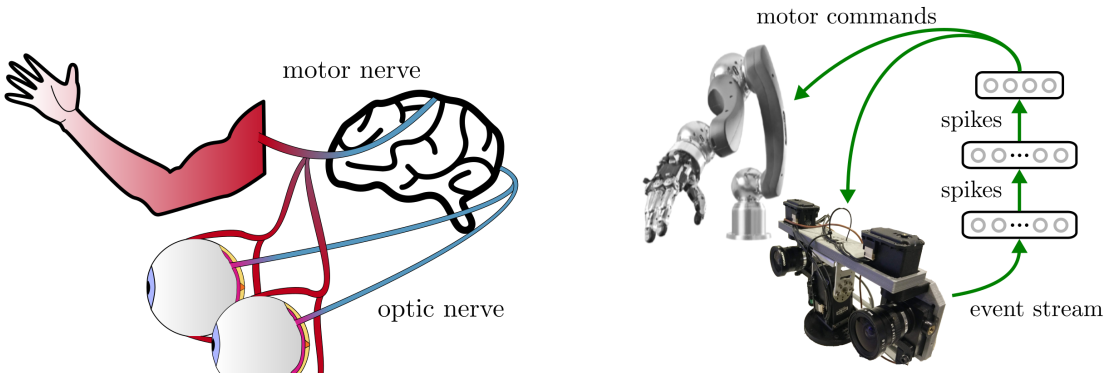
In this Chapter, the approach to address the research goals introduced in Section 1.2 is presented. Subsequently, requirements are identified, leading to the development of tools and datasets. Further, these tools are used in two proof-of-concept experiments. These experiments rely on engineered SNNs exhibiting functional behavior by processing address events. More advanced self-organizing approaches are benchmarked using the same tools in the next chapters (Chapters 5 and 6). The material covered in this Chapter was originally published by the author in [15, 16, 6].

4.1. Our Approach and its Requirements

Throughout this thesis, experiments will be performed by modeling the brain with a SNN, the body with a robot, and the eyes with DVSs, as depicted in Figure 4.1. The research questions are addressed respectively by evaluating promising synaptic learning rules on event-based data and closing the loop with visuomotor mappings. In this Section, the approach to benchmark synaptic learning rules and visuomotor mappings is outlined and discussed. These benchmarks require novel tools to be developed to facilitate evaluation with the help of simulations and experiments within the real world, as presented in Section 4.2.

4.1.1. Addressing the Research Goals

In the following, the two fundamental research questions of this thesis as introduced in Section 1.2 are to be discussed:



(a) Schema of biological visuomotor setup

(b) Structure of the artificial visuomotor setup

Figure 4.1.: Modeling biological visuomotor experiments.

1. Can synaptic learning rules developed in computational neuroscience learn spatio-temporal representations from event-based data?
2. How can these visual representations map to robot control?

We propose to address the first goal by evaluating the recent synaptic learning rules on event-based data. Two questions follow from this proposition: how are the synaptic learning rules selected, and how are they evaluated? The synaptic learning rules to be evaluated are selected based on their performance in the field of computational neuroscience, estimated by a continuous review of the state-of-the-art. With a strong collaboration with computational neuroscientists of TU Graz and University of California, Irvine, novel rules were evaluated on event-based data after their initial introduction in the field. This allowed staying up-to-date with the most recent rules – especially the ones based on backpropagation – which were introduced in computational neuroscience during the studies for this thesis.

The accuracies of the learning rules were evaluated on event-based datasets, as presented in Section 4.2.1. This allows a direct comparison with other methods introduced within the neuromorphic community, not necessarily based on SNNs. Since the introduction of MNIST in 1998 by [167], datasets have become an essential tool to compare different approaches in computer vision research and development. Object and motion recognition are classical tasks to evaluate the performance of a learning rule. In the case of event-based data, both object and motion recognition are performed on sensor-driven event streams. Therefore, the same SNN can be applied to both tasks. This contrasts with conventional computer vision, where object recognition is performed on a single frame and motion recognition on a sequence of frames. The results of this evaluation are summarized in Table 5.2 on the DvsGesture dataset which served as the primary benchmark of this thesis, see Section 4.2.1.

Addressing the second proposed research goal is more subtle. Evaluating a sensorimotor mapping requires a closed-loop control structure. Closed-loop experi-

ments have the particularity that selected actions influence the upcoming sensory input stream, unlike visual representation learning which can be evaluated on datasets. Additionally, such closed-loop control setups have to be simulated to be reproducible and compared with other approaches. In the field of deep learning, this requirement resulted in the development of a variety of simulators especially to evaluate closed-loop policies, such as OpenAI Gym [68] and DeepMind lab [52], both released in 2016. This thesis relies on the Neurorobotics Platform (NRP), which is especially designed and shaped for SNNs and which builds on the robotic framework Robot Operating System (ROS). For the experiments with visual event streams, a simulator for the DVS was developed, which integrates in Gazebo and the NRP, as presented in Section 4.2. The NRP was also used as a tool for teaching students about robotic embodiments in Tieck et al. [22].

Furthermore, it is of high interest to investigate how eye movements can be combined with neuromorphic vision sensors. As discussed in Section 2.1.4, eye movements are an important characteristic of biological vision. While neuromorphic vision sensors are not perfect replicates of biological retinas, they exhibit a similar fading property: no information is transmitted when light remains constant. Therefore, we rely on a repertoire of eye movements inspired by biological vision, including microsaccades to perceive static scenes. This requires an active neuromorphic robotic head that can control its eye movements. For this purpose, a new eye-head configuration has been designed and assembled.

Consequently, it was necessary to connect the event streams provided by the neuromorphic vision sensors to SNNs. All approaches presented in this thesis rely on the IF neuron model (see Equation (2.3)) since most of the proposed synaptic learning rules are derived from it. The connection from address events to IF neurons is discussed in the following Section.

Requirements of the Approach

To summarize the requirements of the proposed approach which yields to the development of the tools presented in Section 4.2; the following specifications have to be considered in the neuromorphic tool set:

- Use of an event-based dataset to evaluate synaptic learning rules;
- Integration of a closed-loop DVS simulator to evaluate visuomotor policies;
- Use of an active neuromorphic head with a capability to move the eyes to imitate macro- and micro-saccadic eye movements.

4.1.2. From Address Events to Spikes

The spike representation used in SNNs and the AER used in neuromorphic sensors are very similar. This allows a seamless connection between the silicon retina and the artificial SNN-based brain.

The main difference between AER and spikes is that all spikes are stereotypical, unlike events which carry a boolean value denoting an increase or decrease in light intensity (ON-OFF). For a particular scene, the polarity of an event depends on the brightness of the moving edge relative to the background. At first glance, the polarity information can therefore seem to be negligible. Indeed, in many early studies, this information was not propagated to the network to reduce the number of neurons. In this case, either ON and OFF events are blended [16], or a single polarity is fed to the network while the other is dropped.

In many cases, event polarity does contain important information. For some dataset such as DvsGesture (see Section 4.2.1), event polarity carries information about direction of motion (see Figure 4.5). It is therefore useful and beneficial to convey this information to the network. The simplest method to propagate the event polarity to the network is to double the number of input neurons, similar to a two-channel image. ON-events are propagated to one channel while OFF-events are propagated to the other.

Covert Visual Attention

In some of the presented methods, the addresses of input events have been translated depending on a moving attention window. A similar phenomenon is witnessed in biology for frontal eye field neurons, referred to as visual receptive field remapping [300, 271]. The attention window has a fixed size and its center is calculated as the median event of the last $n_{attention}$ events, see Figure 4.2. Thanks to the sparseness of event-based data, this computation is efficient and can be performed online. This method belongs to the category of covert attention mechanism, signifying an attention shift which was not marked by eye movements.

This mechanism re-addresses the events relative to the center of the motion. The same motion performed at two different locations in the sensor space will therefore lead to the same input neuron activity. In other words, this mechanism provides translation invariance at a very low computational cost. Conventionally, convolutional networks are used to achieve translation invariance by processing the whole frame with the same weight-sharing kernels. Convolutional layers therefore require many neurons and many computations. In contrast, the proposed method does not require additional neurons and has a low computational cost. It is shown how this simple biologically motivated covert attention mechanism boosts the performance of a dense network in Section 5.4.

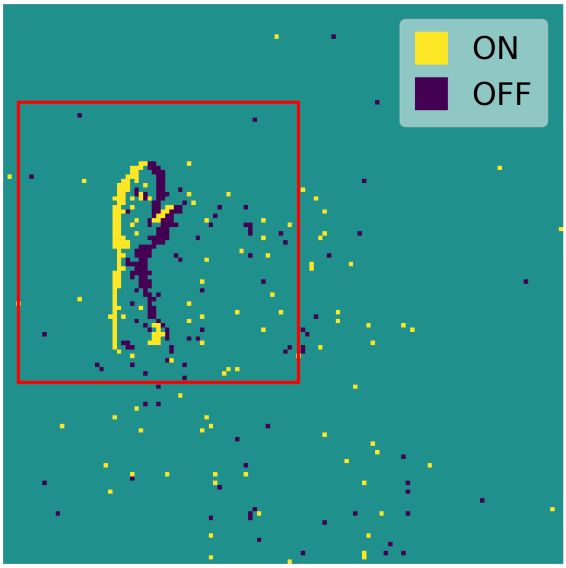


Figure 4.2.: Visualization of the attention window. Aggregation of 1000 events for a sample of the DvsGesture dataset. Yellow pixels symbolize ON-events, blue pixels are OFF-events. The red square represents the attention window of size 64×64, calculated as the median of the last 1000 events.

4.2. Tools for Visuomotor Neurorobotics

In the previous Section, the proposed approach to process event streams with SNN was outlined, and its requirements were identified. In this Section, the tools that address these requirements are discussed. These tools were mainly developed within this thesis and have been made available in open-source, such as the DVS simulator, which was already reused by other researchers in Bing et al. [62] and Jiang et al. [148]. Concerning the datasets, we relied both on popular open-source datasets such as DvsGesture to compare our methods against state-of-the-art, and recorded a new dataset, suited to robotics tasks.

4.2.1. Event-Based Datasets

Datasets are important to evaluate and compare the performance of learning methods in a reproducible manner. Constituting datasets of images was an important contribution to the field of computer vision. Likewise, in recent years, many event-based datasets recorded with neuromorphic vision sensors have been proposed[1] Among the event-based datasets for classification, some consist of converted image datasets as in Orchard et al. [226] for MNIST, while others consist

[1] see `https://github.com/uzh-rpg/event-based_vision_resources` for a curated list

of new motion recordings as in Amir et al. [43] for DvsGesture. Generally, images are converted by relying either on microsaccadic eye movements or on a flashing display. With the event-based paradigm, both types of datasets – converted images or motion – can be classified with the same method since even the converted images have a temporal aspect (despite not bringing much information, see [141]). In other words, both object recognition and motion recognition methods operate on the same input: event streams.

Microsaccadic Datasets

To evaluate some of the learning methods proposed in this thesis, the open microsaccade dataset N-Caltech101 (converted from Caltech101) recorded in Orchard et al. [226] was applied. The same work also recorded N-MNIST, a conversion of the MNIST digit dataset, on which we evaluated DECOLLE in [12]. A sample from this dataset is depicted in Figure 4.3. The full N-Caltech101 dataset consists of 101 object classes and a total of 8709 event stream samples. For our early experiments with STDP, we relied on a subset of N-Caltech101 consisting of few samples from three classes: airplanes, motorbikes, faces (see Section 5.2).

We additionally recorded a microsaccadic dataset of different graspable objects with the developed neuromorphic head (Section 4.2.3). This dataset consists of less than 100 samples for four classes: ball, bottle, pen and background. It was used in the reaching and grasping experiments evaluated in Section 6.1. Some samples from this dataset are depicted in Figure 4.4.

(a) Caltech101 (b) N-Caltech101

Figure 4.3.: Samples from Caltech101 and N-Caltech101 (events are integrated over a 100ms time window) dataset. These datasets are used in Section 5.2. Copyright ©2018, IEEE [9].

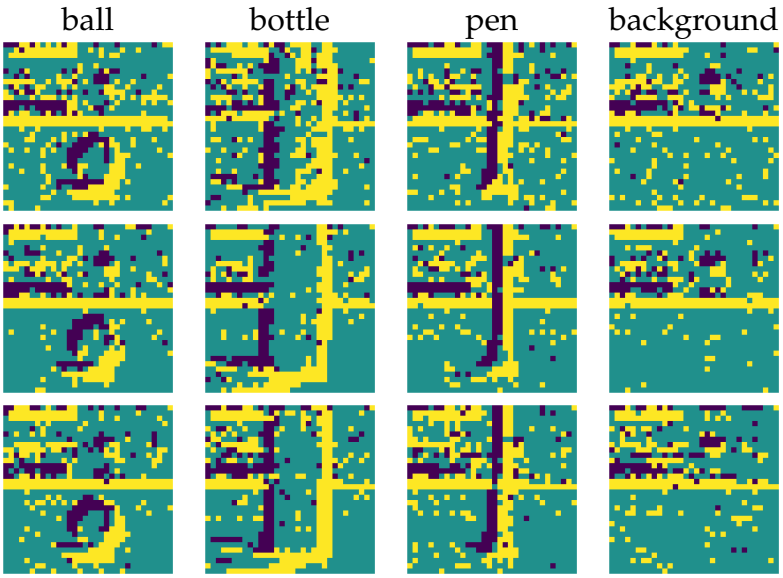
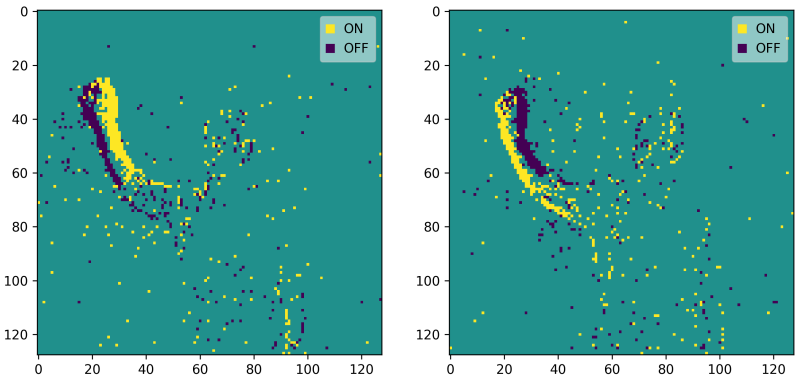


Figure 4.4.: Samples of the ball-bottle-pen-background microsaccade dataset. Events were aggregated into frames and cropped to 32x32 pixels. Yellow denotes ON-events and purple denotes OFF-events.

Motion Recognition Datasets

DvsGesture is an action recognition dataset recorded by IBM using a DVS [43, 173]. It consists of 29 subjects performing 11 diverse actions in three different illumination conditions, see Figure 4.5. The duration of the actions varies considerably across motions, see Table 4.1 and Figure 4.6. Single motions may be about $1\,\mathrm{s}$ or up to a duration of $18\,\mathrm{s}$. We heavily relied on this dataset to evaluate and compare the learning rules proposed in this thesis. Especially, comparisons have been elaborated for: the spiking HMAX architecture trained with STDP as well as its associated LSTM in Section 5.2, a dense feedforward network trained with eRBP in Section 5.4 and a convolutional feedforward network trained with DECOLLE in Section 5.5. An adapted version of eRBP is evaluated on this dataset a second time in Appendix B. The results of all these rules on the DvsGesture dataset are succinctly summarized in Table 5.2.

For HMAX and DECOLLE, samples were obtained from the dataset by extracting time intervals of $100\,\mathrm{ms}$ and $500\,\mathrm{ms}$ respectively. For eRBP, a sample consists of a full motion regardless of its duration. These differences emerge from the capabilities of the SNN simulators that were used to simulate the networks, see Appendix A for more explanations about simulating SNNs.



(a) Arm movement circling clockwise

(b) Arm movement circling anticlockwise

Figure 4.5.: Each image shows the aggregated event stream over 20ms. The left image is created from a sample of label 4, the right one from a label 5 sample. Yellow pixels symbolize ON-events, blue pixels are OFF-events. The difference in motion direction is visible in the event polarity. Adapted from [6].

Label	#Training	#Test	Description
1	97	24	Clapping
2	98	24	Right hand waving
3	98	24	Left hand waving
4	98	24	Right arm circling clockwise
5	98	24	Right arm circling anticlockwise
6	98	24	Left arm circling clockwise
7	99	24	Left arm circling anticlockwise
8	196	48	Arms rolling
9	98	24	Air drum
10	98	24	Air guitar
11	98	24	Other gestures

Table 4.1.: Labels of all the different classes in the DvsGesture dataset and their description. The amount of samples of label 8 is doubled since arms rolling were recorded and labeled with 8 for both rotation directions. Source: [6].

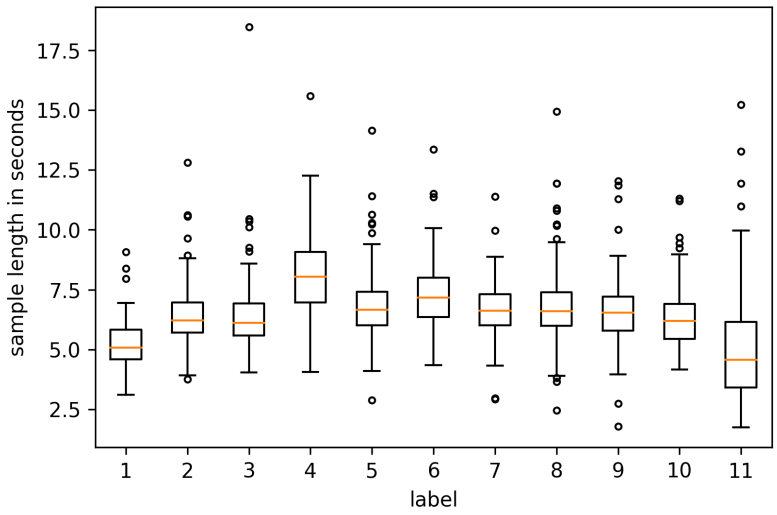


Figure 4.6.: Statistics on sample duration for each label in the DvsGesture dataset. The red horizontal lines represent the median sample duration. Each box indicates the interquartile range ($IQR = Q_3 - Q_1$) per label, which is the range between the first Q_1 and the third quartile Q_3. Whisker pairs show the range of all sample durations within $Q1 - 1.5 \times IQR$ and $Q3 + 1.5 \times IQR$. Outliers are represented by small circles. Source: [6].

4.2.2. Closed-loop DVS Simulator

Simulating sensor data is a convenient computational technique to support efficient algorithm development. It reduces the need for real experiments, allows parallel simulations, and provides ground truth relevant for evaluation and learning. In a closed-loop robotics scenario, such sensor simulation should be able to consider and react to changes in the environment. This constraint is particularly relevant for the DVS which only reacts to local light changes.

We proposed, implemented and open-sourced[2] a DVS simulator in Kaiser et al. [15]. This simulator is integrated as a Gazebo plugin, allowing closed-loop simulations with other ROS components and robots. It relies on the same ROS message type as the ROS DVS driver developed in [206][3], enabling a seamless transition between simulated and real DVS data.

The purpose of our simulator is not to model a DVS with a high degree of accuracy, but instead to allow prototyping event-based closed-loop algorithms. Other DVS simulators have been developed since 2016, notably in [208, 244]. These simulators can reproduce a DVS signal with a high degree of accuracy, but are not integrated in off-the-shelf robotic simulators such as Gazebo, and do not support closed-loop interactions.

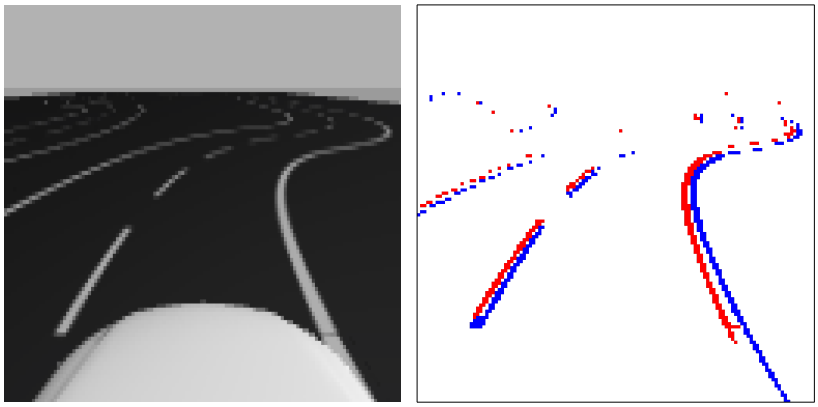
[2] https://github.com/HBPNeurorobotics/gazebo_dvs_plugin
[3] https://github.com/uzh-rpg/rpg_dvs_ros

Both the high-accuracy simulators and the implemented Gazebo DVS plugin rely on a frame differentiation technique. In this case, a DVS is simulated by subtracting new camera frames with an internal one storing the last light intensity in memory. After the subtraction, a threshold is applied to only retain pixels that have a significant difference in light intensity, positive or negative. Differentiated pixels that have a value higher than a threshold θ are reported as ON-events, while differentiated pixels with a value lower than $-\theta$ are reported as OFF-events. After a differentiation, all generated pixel events are reported with the time-stamp of the current frame, see Figure 4.8. The internal stored light intensity is updated for the pixels that emitted events.

The threshold value θ is equivalent to the DVS bias, or to the slow light adaptation parameter in biology [282]. By varying the threshold value with respect to the global background luminosity, light adaptation can be obtained.

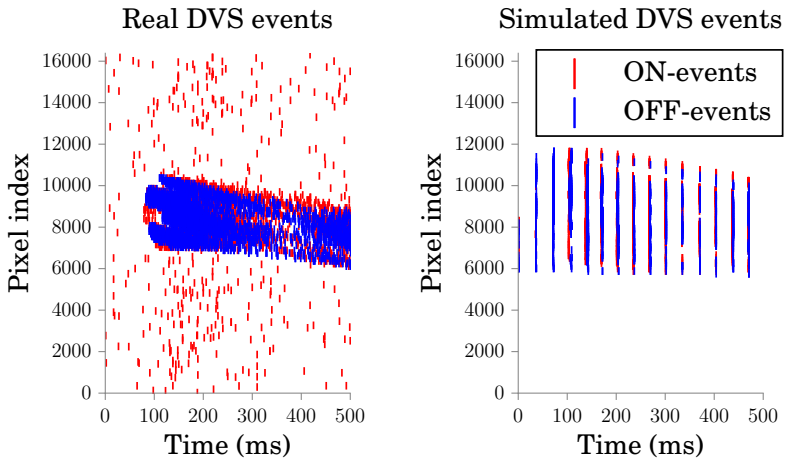
As seen in Figure 4.8, the DVS simulation is less noisy than the real DVS but has a much lower temporal resolution. A drawback of this simple approach is the inability to distinguish which address event should be emitted first when many pixels are reported within the same differentiation step. Indeed, events are emitted by batches when a new frame is received (Figure 4.8b). Even if this drawback makes a poor DVS simulation, it is argued that a robust SNN should be able to cooperate with such inputs. Certainly, the human visual system can still perceive fluid motion when presented discrete frames with a sufficient frame rate.

This type of encoding conveys motion information and differs considerably from previous image encoding methods used in literature, which converts pixel values to Poisson spike rates [88]. It also differs from rank-order coding as it can work continuously without synchronization, see Figure 3.1. In biology, it was noted that motion is processed separately from color and shape, and is strongly coupled with the pre-motor areas of the cortex [157]. It is therefore a relevant encoding for robotic control tasks.



(a) Camera image (b) Aggregated DVS events

Figure 4.7.: Simulated camera image and DVS events in the lane following experiment. Red pixels are ON-events, blue pixels are OFF-events. Copyright ©2016, IEEE [15].



(a) Real DVS events (b) Simulated DVS events

Figure 4.8.: Generated address events of a real DVS and a simulated DVS from webcam images. Both cameras are observing the same stimuli of a ball entering the field of view. The 2 dimensional image structure is flattened to a 1 dimensional pixel index. ON-events are drawn in red, OFF-events are drawn in blue. (a) With the real DVS, the events have continuous time-stamps. (b) With the simulated DVS, the events are batched in time-steps when a new frame is received. Copyright ©2016, IEEE [15].

4.2.3. Neuromorphic Head for Oculomotor Control

As discussed in Section 4.2.2, the proposed DVS simulation does not simulate a real DVS accurately. In this Section, an active visual robotic head with two real

DVSs in stereo configuration mimicking saccadic eye movements is presented. The blueprints of the 3D printed parts for building this head are provided in Appendix C.

The DVS only detects local changes in light intensity, conveying motion information. Therefore, a stationary scene with no dynamic changes and no moving objects can not be detected by the DVS. For typical robotics applications, sensing of static scenes is also important. To solve this issue, we equipped the pair of DVSs with pan-tilt axis driven by electric motors in analogy to human eyes, see Figure 4.9a. The obtained motion is similar to oculomotor eye movements, necessary for basic scene perception by humans. This robot head can perform microsaccadic eye movements, depicted in Figure 4.9b, inspired by the fixational eye movements described in Section 2.1.4. Actively moving vision sensors allows also to refocus the center of attention with saccadic eye movements. This can be used in conjunction with the covert attention model introduced in Section 4.1.2 remapping the receptive fields without moving the eyes.

The head is built with custom 3D printed parts, one Dynamixel MX-64AT servomotor to tilt both DVSs simultaneously and two additional MX-28AT servomotors to pan each DVS independently. The center of all rotations is approximately the optical center of each DVS. This robotic head can be mounted on HoLLiE [130], a humanoid robot developed at FZI.

(a) Neuromorphic head with two DVSs (b) Microsaccadic eye movements

Figure 4.9.: (a): The developed robotic head, reproducing eye movements with two DVSs. (b): Microsaccadic motion of the DVS performed by the robotic head. Source: [6].

4.3. Simulated and Real Experiments

In this Section, two early experiments connecting event-based data with SNNs are presented – one in simulation, one in the real world. These early experiments

rely on the tools presented previously, the Gazebo DVS plugin and the neuromorphic robotic head.

Specifically, an end-to-end lane following network is introduced in Section 4.3.1, and a stereo vision network in Section 4.3.2. The lane following experiment is useful to understand the dynamics of a closed-loop neurorobotic visuomotor task and consequent requirements for learning. The neuromorphic stereo vision experiment shows how event-based processing and SNN can ingeniously solve building blocks of robotics. Both experiments do not learn – instead, they show what SNN can achieve with hand-tuned weights. Learning these weights is the focus of Chapter 5.

Throughout this thesis, these basic experiments will be extended with advanced synaptic learning rules. The simulated lane following experiment is used to embody SPORE in Section 6.2. The neuromorphic robotic head is used to record a visual grasping affordance dataset, presented in Section 4.2.1 and evaluated with eRBP in Section 5.4.

4.3.1. Simulated Visuomotor Lane Following Experiment

The material covered in this Section has been originally published in Kaiser et al. [15]. In this Section, the first proof-of-concept closed-loop experiment combining all the components on which this thesis builds is presented. This experiment consists of a visuomotor lane following task, where a simulated autonomous vehicle has to drive and follow a track. The vehicle is controlled by a SNN agent that receives address events from the simulated DVS and outputs on time steering commands.

The experiment is neither too easy – as a random network is unable to follow the lane, neither too hard – since the task can be solved with simple networks. the difficulty of the task can be controlled by evaluating on different circuits, assessing the generality of an agent. Circuits are designed using the world builder provided by Zofka et al. in [301]. The world builder provides predefined tiles containing straight road segments, curves with different radius and several intersection types. With this method, new circuit layouts can be designed and modified easily to get more variation in training data. Each street segment template features two lanes, one in each direction with equal lane width, separated by a dashed center line. The simulated roads are flat and without any external visual disturbances.

Additionally, metrics can easily be defined to evaluate the performance of the agent or to provide learning signals (see Section 6.2). For the circuits, lanelet maps are generated as ground truth. Lanelets are a representation of the drivable environment defined by the left and right boundaries of a lane segment [56]. They can be used to calculate distances between the ego vehicle and the center of a lane,

as well as the offset between vehicle heading and lane orientation. Following this strategy, it is possible to obtain a metric for the quality of the vehicle pose.

The framework for the visuomotor lane following experiment is shown in Figure 4.10. ROS is used as communication middle-ware between the different modules. Gazebo is the physics simulator and represents the environment, including the vehicle. The DVS simulation provides visual input to the network in form of address events, see Section 4.2.2. Address events from the DVS are converted to spikes in spike generator neurons, see Figure 4.11b. There are 12 spike generator neurons organized in two rows and six columns, covering the whole DVS pixel array with equally sized, non-overlapping receptive fields. These neurons emit a spike whenever an address event (ON or OFF) is emitted by the DVS within their receptive field.

The SNN has two layers (direct input to output mapping) and implements a Braitenberg-vehicle [278], see Figure 4.11a. The architecture of the SNN is depicted in Figure 4.11b. It processes the events and communicates asynchronously with spikes. Spikes of the two output neurons are decoded to motor commands with an agonist-antagonist muscle model. Specifically, steering commands are decoded from output spikes as a ratio between spike rate of the left and right motor neuron:

$$r = k \times \frac{a_L - a_R}{a_L + a_R}, \tag{4.1}$$

with a_L and a_R the spiking rate of left and right motor neurons. The constant k defines the range of the steering angle commands.

The SNN is depicted in Figure 4.11, together with a taxonomy of simple Braitenberg vehicles. The behavior of a Braitenberg vehicle depends on the wiring of the network. Excitatory connections increase wheel speed whereas inhibitory connections decrease it. The sensor and motor neurons of the lane following network implements an aggressive Braitenberg. Spike generators propagate the spikes with respect to the address events to the sensor neurons. The connection pattern between spike generators and sensor neurons represents an hard-coded simple lane detector with weights $\left(\begin{smallmatrix} 25 & 20 & 0 & 0 & 20 & 25 \\ 70 & 5 & 0 & 0 & 20 & 70 \end{smallmatrix}\right)$.

The proof-of-concept SNN depicted in Figure 4.11 allows to follow the lane in closed-loop for simple circuits, see Figure 4.12. Unlike conventional methods, all processing happens asynchronously in the spike-domain and are characterized by the temporal dynamics of the neurons and synapses, see Figure 4.13. However, the Braitenberg network does not solve the task perfectly: intersections are not handled, the vehicle wiggles within the lane and higher speeds are unstable (Figure 4.12). For slow and medium speed, the car follows the lane but wiggles. It completes the lap approximately at the same time both with medium and fast maximum speed. In the latter, the car drives off the road a few times and activates the brakes more often.

Other visual disturbances such as pedestrians, other vehicles, or buildings would also disturb the hard-coded 16 neurons network. More complex visuomotor tasks

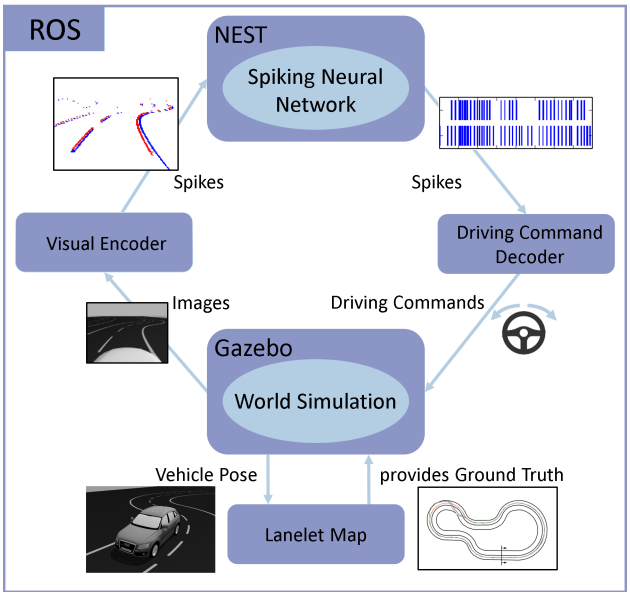
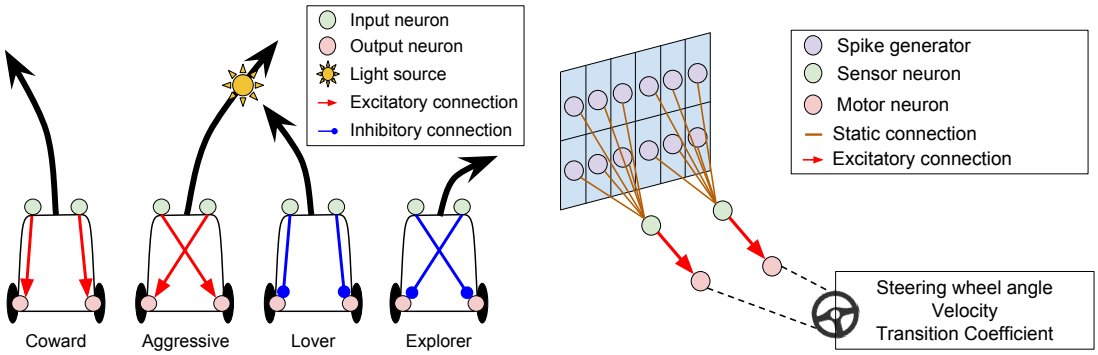


Figure 4.10.: Framework for the visuomotor lane following experiment. ROS is used as a communication middle-ware between the different modules. Gazebo enhanced with the DVS plugin emits address events and simulate the physics. The SNN processes the events and output spikes are converted to motor commands. Copyright ©2016, IEEE [15].



(a) Different Braitenberg-vehicles (b) SNN for lane following

Figure 4.11.: (a): Simple Braitenberg-vehicles with two light-sensitive sensor neurons and two motor neurons controlling the wheels. (b): The feed-forward SNN controlling the vehicle end-to-end. Spike generators cover the DVS pixel array. Motor neurons are used to decode steering angle and velocity. Copyright ©2016, IEEE [15].

require more complex, self-organizing networks. The objective of this thesis is to research on the synaptic learning rules capable of training such networks to solve similar visuomotor tasks.

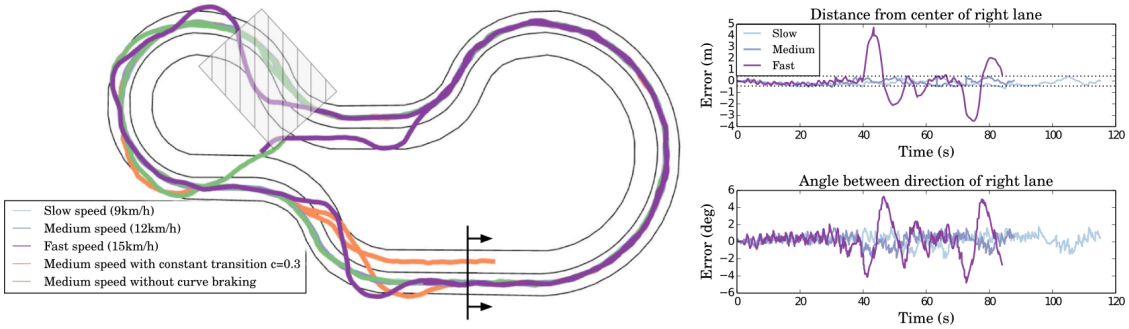


Figure 4.12.: Evaluation of the Braitenberg network on the lane following task. Left: positions of the car on a road at three different speeds. The car manages to stay on the right lane of the road for all configurations except at high speed. Right: metrics to evaluate the performance of the car during one lap. The dotted lines represent the boundary of the right lane. Copyright ©2016, IEEE [15].

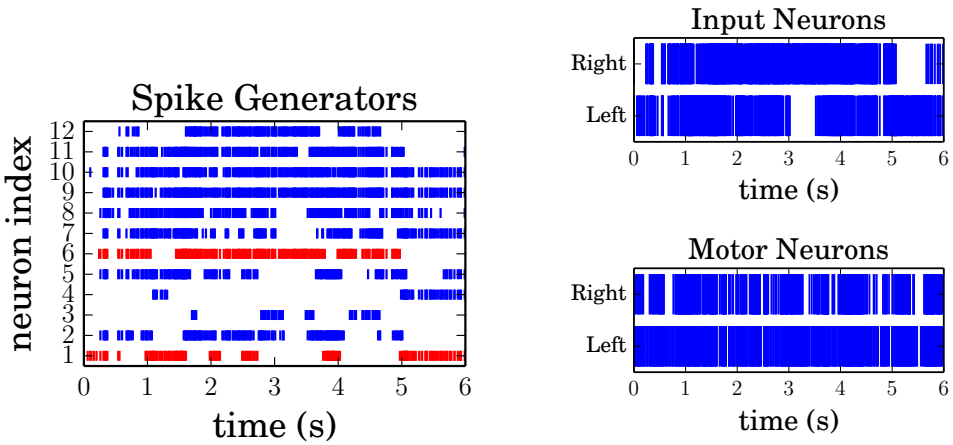


Figure 4.13.: spike-trains of the lane-following Braitenberg network during a 800 ms time period. The vehicle drives on the strong curvature of the circuit without intersection at medium speed (12km/h). The neuron 1 and 6 (in red) are responsible for the bottom corners of the image and are assigned strong weights. Copyright ©2016, IEEE [15].

4.3.2. Microsaccades for neuromorphic stereo vision

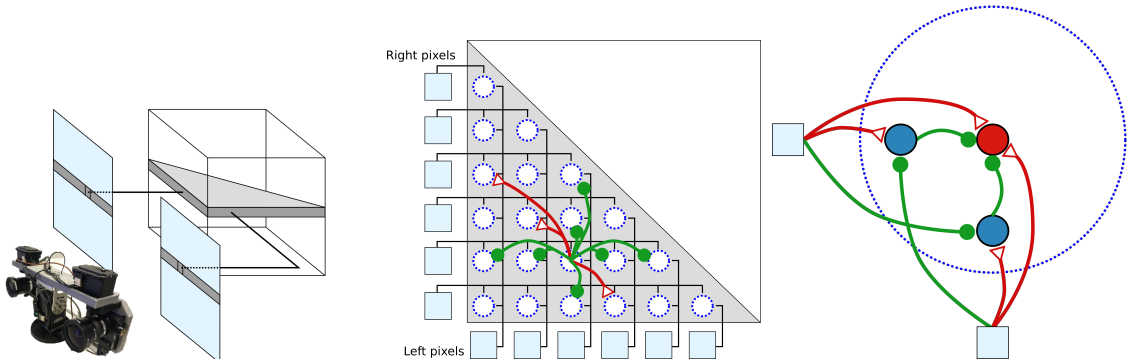
The material covered in this Section was originally published by the author in Kaiser et al. [16]. In this Section, the neuromorphic DVS head is demonstrated in a stereo vision setup. Specifically, it is shown how the method can be used to perceive depth from motionless static scenes through microsaccadic eye movements with a SNN computing disparities.

The structure of our SNN computing disparity from stereo event-based vision sensors is based on Osswald et al. [227] and Dikov et al. [90]. The network consists of a three-dimensional grid of disparity-sensitive neurons (see Figure 4.14a). Address events from the two DVSs lying on to the same epipolar plane are fed to the corresponding epipolar layer in the network (see Figure 4.14b). Each of the disparity-sensitive neurons describes one unique point in the observed 3D-space, relative to the common fixation point of the cameras [228].

For each disparity neuron, a micro-ensemble ensures hetero-lateral matching (see Figure 4.14c). The micro-ensembles are connected to each other with respect to the constraints mentioned in [16]. If the timing of the events projected by the retinal pixels into the neural ensemble is temporally congruent, the signal reaches the disparity-sensitive neuron. However, if the temporal offset of the incoming signals between the left and right pixels is too large, the blockers prevent the activation of the disparity-sensitive neuron. The C3 constraint (*compatibility constraint*) could be implemented by separating ON and OFF events in two separate pathways. As this would double the number of neurons, the C3 constraint is often ignored so that ON and OFF events can match each other. The number of micro-ensembles can be reduced by bounding the minimum and maximum detectable disparities.

This network has been evaluated observing natural scenes with the neuromorphic DVS head performing horizontal and vertical microsaccades. The DVS head is positioned on a table with two objects (a ball and a thermos bottle) and both DVSs observe in parallel mode, see Figure 4.15. The network supports the computation of the disparity of the different objects in the scene with a horizontal microsaccade including the garage door in the background, see Figure 4.16. Note the peaks around disparity 7 which correspond to the vertical garage door, around 20 to the thermos bottle, and around 29 to the ball. Because most contrast lines in the scene are vertical, the tilting microsaccade does not trigger many events, leading to few disparity detections. Additionally, extracting disparity of horizontal edges is harder for the network, because many events will share the same epipolar layer.

The stereo network is a proof-of-concept showing how conventional robotics building blocks can be solved with SNNs. These techniques can take advantage of an additional constraint for matching: time. However, the asynchronous spike-based communication as described by the neural dynamics is complicated



(a) Disparity SNN (b) Epipolar layer (c) Micro-ensemble

Figure 4.14.: Structure of the stereo network for detecting all possible positive disparities (schemas inspired by [90]). Triangular red edges denote excitatory synapses, rounded green edges denote inhibitory synapses. (a): Three-dimensional structure of the stereo network. (b): Organization of micro-ensembles within an epipolar layer. For clarity, only the outgoing connections of a single micro-ensemble are drawn. (c): Schematic representation of a neural micro-ensemble. The two blue neurons on the left and bottom of the micro-ensemble are the blockers, while the red neuron in the middle is the disparity-sensitive collector neuron. Source: [16].

to regulate with hand-tuned parameters. This concern is addressed in the coming Section with synaptic learning rules.

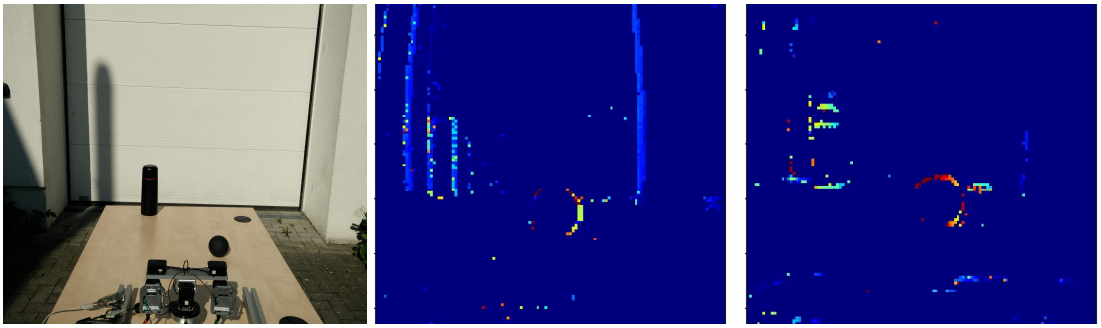
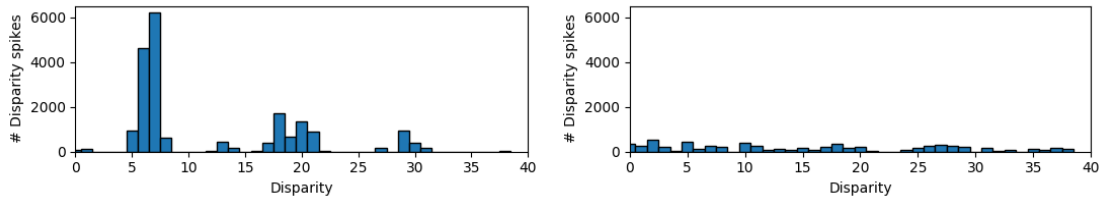


Figure 4.15.: Evaluation of the stereo-network on a static scene perceived with microsaccades. Overview of the setup and rendering of the computed disparities during panning and tilting. Source: [16].



(a) Disparity histogram during paning

(b) Disparity histogram during tilting

Figure 4.16.: Histogram of the computed disparities during panning and tilting. Source: [16].

4.4. Summary and Conclusion

In this Section the basic approach that is taken to address the research questions of this thesis has been discussed and presented. The IF was selected as the base neuron model and presented methods to connect address events to SNN. Especially, it was shown that separating ON and OFF events is important since they carry information about the direction of motion. In addition, a covert attention mechanism providing translation invariance has been introduced, which will be evaluated in Section 5.4.

Describing the approach allows us to identify the requirements, leading to the development of tools that will be used throughout this thesis. Importantly, the Gazebo DVS plugin supports the design of simulated experiments, and the neuromorphic head to perform real-world experiments. These two tools were demonstrated in proof-of-concept experiments.

5. Learning Visual Representations

In Chapter 4, we showcased how hand-tuned SNNs were capable of solving visuomotor tasks. Manually selecting the network parameters is a tedious process and is only viable for small, highly structured networks. A more scalable approach is to learn these parameters from the data, through self-organization. Deriving a learning rule for the self-organization of large SNNs of arbitrary architectures is an important goal of computational neuroscience. For ANNs, such a rule has arguably been found: gradient descent using backpropagation. However, the classic formulation of backpropagation violates biological constraints which prevents its implementation in SNNs, see Section 3.1.4. This has led the computational neuroscientists to formulate other rules capable of learning from spikes. In the last three years, it became apparent that a different formulation of backpropagation – relaxing a precise gradient computation – could be implemented by SNNs and the brain [174, 219, 144]. Since then, a variety of new learning rules approximating backpropagation in SNNs emerged [213, 296, 55], including ours [12] (see Neftci et al. [214] for a review). In this Chapter, it is outlined how the rules proposed by computational neuroscientists during the course of this thesis have been adapted to support alternative backpropagation mechanisms for event stream benchmarks. As we will see, backpropagation rules achieve much higher accuracy on event stream benchmarks.

5.1. Liquid State Machines

The material covered in this Section was originally published by the author in Kaiser et al. [14]. It has been shown by Thorpe et al. in [280, 191] that many tasks in the visual cortex including perceptual categorization could be solved by solely relying on feedforward network architectures, see Section 1.1. However, the visual cortex has a majority of feedback connections, leading to a highly recurrent structure [182]. Especially relevant for event streams, these recurrent structures could provide a mechanism to build and maintain a scene representation from purely transient information. Indeed, as address events signal changes, a memory is required to aggregate these changes and retain the current state of the scene. Recurrent architectures could play this role but only a few methods to train recurrent SNNs were presented in the computational neuroscience community, see Section 3.1.3. Liquid State Machines (LSMs) are SNNs with recurrently and randomly connected neurons, referred to as the liquid, which receives an input data

stream. Learning in LSMs consists of computing the linear mapping from the state neurons in the liquid (the liquid state) to an arbitrary target signal.

5.1.1. Method

In this Section, a LSM is trained to predict address events in a similar fashion than Maass et al. [183] and Burgsteiner et al. [70] predicted images, see Figure 5.1. By using an event-based sensor, the challenge of predicting frames is circumvented by only predicting changes in pixel intensity. Moreover, compared to previous works, the input dimensionality that the liquid operates is scaled by two orders of magnitude. This method implicitly leads to learning spatio-temporal features as was demonstrated in Lagorce et al. [159] for echo state networks.

The contribution of this work, originally presented in Kaiser et al. [14] are:

- The proposed metric based on centroid computation to evaluate the performance of the prediction by taking spatial structure into account;
- The proposed technique to scale the LSM to different input dimensionality by conserving a constant number of connections between input and liquid.

Filtering address events with an exponential filter to generate a target signal was already introduced in [159].

For simplicity, the mathematical notations introduced in Kaiser et al. [14] are reused, where vectors are denoted with bold lowercase letters and matrices with bold uppercase letters. The event stream is fed in recurrently connected spiking neurons, called the liquid. A subset of n_{rec} excitatory liquid neurons are connected to each readout neuron in the output layer, which has the same dimensionality as the input. The activity relayed by these connections is called the liquid state, and is denoted as $\boldsymbol{x}(t) \in \mathbb{R}^{n_{rec}}$ for a given time t. Only these connections are trained. They are parameterized with the weights $\boldsymbol{W} = [\boldsymbol{w_1}, \ldots, \boldsymbol{w_p}] \in \mathbb{R}^{n_{rec} \times p}$, with p the number of readout neurons. The training consists of a simple supervised linear regression from the liquid states to the target signals, defined as:

$$\text{argmin}_{\boldsymbol{W}}(\boldsymbol{X} \cdot \boldsymbol{W} - \boldsymbol{B}), \tag{5.1}$$

with

$$\boldsymbol{X} = \begin{bmatrix} | & | & & | \\ \boldsymbol{x}(t_1) & \boldsymbol{x}(t_2) & \ldots & \boldsymbol{x}(t_{n_{samples}}) \\ | & | & & | \end{bmatrix}^{\intercal} \in \mathbb{R}^{n_{samples} \times n_{rec}} \tag{5.2}$$

the accumulated sampled activities, and

$$\boldsymbol{B} = \begin{bmatrix} | & | & & | \\ \boldsymbol{b}(t_1) & \boldsymbol{b}(t_2) & \ldots & \boldsymbol{b}(t_{n_{samples}}) \\ | & | & & | \end{bmatrix}^{\intercal} \in \mathbb{R}^{n_{samples} \times p} \tag{5.3}$$

the sampled target signals for all p readout neurons. The target signals are obtained by applying an exponential filter on the input event stream time-shifted by Δ_{pred}:

$$\boldsymbol{b}(t) = \left[\begin{array}{c} \\ exp\left(-\frac{t + \Delta_{pred} - t_i^{spike}}{\tau}\right) \\ \\ \end{array} \right], \tag{5.4}$$

where $t_i^{spike} \in]-\infty, t]$ denotes the last spike time of input neuron i, and τ a global fading term. At a given time-step t, the visual prediction $\boldsymbol{p}(t)$ is:

$$\boldsymbol{p}(t) = \boldsymbol{x}(t)^\intercal \cdot \boldsymbol{W}. \tag{5.5}$$

As noted in Kaiser et al. [14], only a single training motion is sufficient to provide predictions for similar motions when the scene is not crowded by many moving objects. A visualization of how target signals are computed is provided in Figure 5.2.

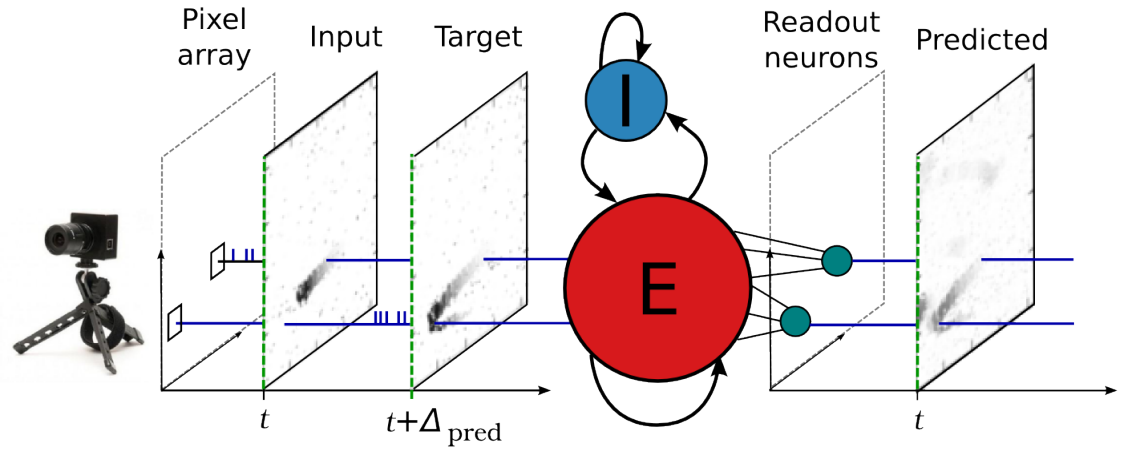


Figure 5.1.: Predicting address events from a DVS using a liquid state machine. Events from the DVS are converted to spikes and directly fed to the excitatory pool of neurons in the liquid. A layer of readout neurons of the same dimensionality as the input is connected to n_{rec} excitatory neurons of the liquid. Those connections are trained at time t with the target image sampled at time $t + \Delta_{pred}$. In the test phase, unseen events are streamed to the liquid and the activity of the readout neurons encodes the predicted future input. The input image is just shown for visualization purposes, only address events are streamed to the liquid. Source: [14].

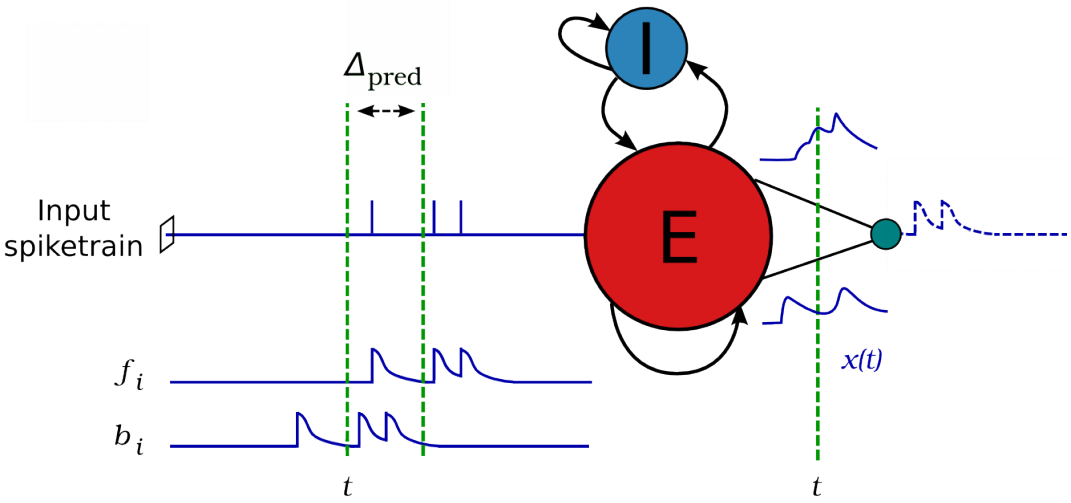
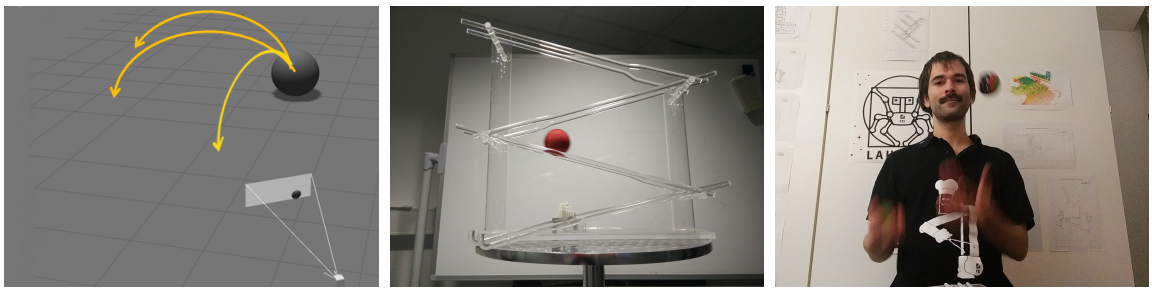


Figure 5.2.: Visualization of the training of readout neuron i. To generate a target signal, the input spike-train is convolved with an exponential filter (denoted f_i). The target signals are obtained by shifting the convolved signal by Δ_{pred}, see Equation (5.4). During training, the readout neuron i learns a mapping between the liquid state $\boldsymbol{x}(t)$ to $b_i(t)$ for all times t. Source: [14].

5.1.2. Experimental Setup

A LSM predicting future visual input from event-based data provided by a DVS has been implemented and evaluated. The evaluation was performed both in simulation and with a real DVS, with a prediction time Δ_{pred} set to $200\,\mathrm{ms}$. The three scenarios are visualized in Figure 5.3.



(a) The first scenario (b) The second scenario (c) The third scenario

Figure 5.3.: The three scenarios against which the method is validated. (a) A 32x32 simulated DVS observes a ball jumping from left to right and from right to left of the pixel array. (b) A real DVS (128x128) observes a ball rolling down a structure. This image depicts the point of view of the DVS. (c) A real DVS (128x128) observes a person juggling with three balls. This image depicts the point of view of the DVS. Source: [14].

The liquid consists of 1250 randomly connected leaky integrate-and-fire neurons, 80% of which are excitatory and 20% inhibitory. There are 102400 random synapses from the DVS input layer to the liquid. The number of connections between the input and the liquid remains constant across the different experiments despite the change in input dimensionality from scenario 1 (32x32) to scenarios 2 and 3 (128x128), see Figure 5.3. 500 excitatory liquid neurons are connected all-to-all to perfect linear readout neurons in a feedforward fashion. The readout weights are trained with a regularized linear regression to predict the future address events convolved, see Equations (5.1) to (5.5).

Each scenario starts by presenting the emitted address events for a time t_{train} to the liquid and recording the liquid states. Then the readout weights are trained with the procedure described in 5.1.1. Finally, the LSM is presented yet unseen address events for a time t_{test}. The sampling time interval of the liquid is set to $\Delta_{sample} = 10$ms. Predictions are generated by reading from the readout neurons at the same sample rate $1/\Delta_{sample}$ used for training.

The performance of the method is evaluated with respect to the predictions generated by the LSM. Two metrics to evaluate the predictions are defined. The first metric is general and consists of computing the normalized error for all predictions:

$$e_1(\boldsymbol{W}) = \frac{1}{n^{test}_{samples} \cdot p} \cdot ||\boldsymbol{X}^{test} \cdot \boldsymbol{W} - \boldsymbol{B}|| \tag{5.6}$$

with $\boldsymbol{X}^{test}$ the accumulated liquid states during the test period, and $n^{test}_{samples}$ the number of samples in the test set and p the number of pixels, used for normalization. The residual error is the one implicitly minimized when solving the linear regression in Equation (5.1).

A drawback of this error is that it does not take the spatial structure of the sensor array into account. Indeed, if the LSM wrongly predicts an activated pixel, the loss should consider the target activity of surrounding pixels. A second metrics which takes into account this spatial structure is introduced: centroid activation.

The second metric is the distance between the centroid of the predicted activity and the centroid of the target activity. This metric is intended for scenes with only a single object in motion – it is therefore calculated only for the first two scenarios. The centroid of an image is defined as the average position of the activation:

$$\boldsymbol{c}(\boldsymbol{I}) = \begin{bmatrix} r_c \\ c_c \end{bmatrix} = \frac{1}{\sum\limits_{r,c} \boldsymbol{I}(r,c)} \cdot \begin{bmatrix} \sum\limits_{r,c} \boldsymbol{I}(r,c) \cdot r \\ \sum\limits_{r,c} \boldsymbol{I}(r,c) \cdot c \end{bmatrix}, \tag{5.7}$$

with $\boldsymbol{I}$ an input image. This is similar to the centroid computation introduced in Section 4.1.2 to guide an attention mechanism. The second error metric is calcu-

lated as:

$$\begin{aligned} e_2(\boldsymbol{W}) &= \frac{1}{n^{test}_{samples}} \cdot \sum_{t \in \boldsymbol{t^{test}}} ||\boldsymbol{c}(predicted) - \boldsymbol{c}(target)|| \\ &= \frac{1}{n^{test}_{samples}} \cdot \sum_{t \in \boldsymbol{t^{test}}} ||\boldsymbol{c}(\boldsymbol{x}(\boldsymbol{t}) \cdot \boldsymbol{W}) - \boldsymbol{c}(\boldsymbol{b}(\boldsymbol{t}))|| \end{aligned} , \tag{5.8}$$

with $\boldsymbol{t^{test}}$ the sampled times during test, and $\boldsymbol{b}(\boldsymbol{t})$ the target signals for all readouts at time t. Here, it is assumed that both the prediction $\boldsymbol{x}(\boldsymbol{t}) \cdot \boldsymbol{W}$ and the target signals $\boldsymbol{b}(\boldsymbol{t})$ are reshaped to images of size rows $\times$ columns before the computation of the centroid.

5.1.3. Results

For each scenario, a representative prediction from the LSM is shown (see Figures 5.4a, 5.6a and 5.8). *Input* and *Target* denotes the encoded input spike-train at a given time t and $t + \Delta_{pred}$. *Predicted* is the output from the LSM at time t and *Error* refers to the residual error (Equation 5.6), which is the difference between *Predicted* and *Target*. Both error metrics (Equation 5.6, solid blue line and Equation 5.8, dashed red line) are presented with varying amount of training data and increasing prediction times Δ_{pred} in Figures 5.5, 5.7 and 5.9.

First scenario: simulated projectile trajectories

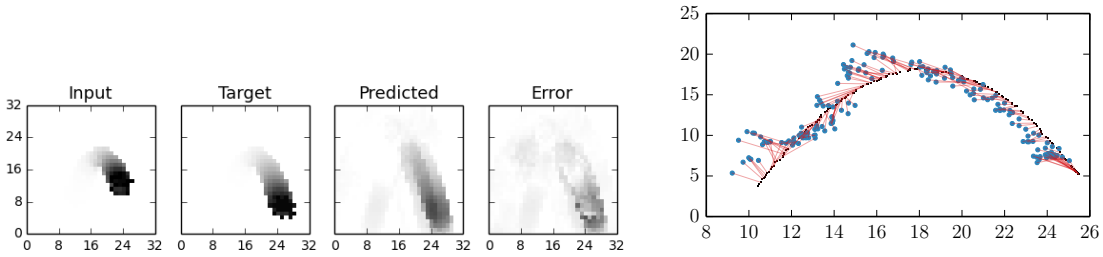
There are 10 ball trajectories in the training set, altogether lasting $27.6\,\mathrm{s}$. In 5 samples, the ball goes from left to right, and in 5 others, from right to left.

A single test sample consisting of the ball going from left to right is evaluated within a time interval of $1.9\,\mathrm{s}$. The LSM never saw this particular sample. Since the LSM was trained with both left-to-right and right-to-left samples, it needs to rely on its memory to identify the motion.

The LSM correctly predicts future motion on the right side of the image, see Figure 5.4a. It therefore accurately identified that the motion was from the left-to-right kind and not of the right-to-left kind, equally represented in the training set. This identification can only be based on memory since both left-to-right and right-to-left motions cover the same pixels.

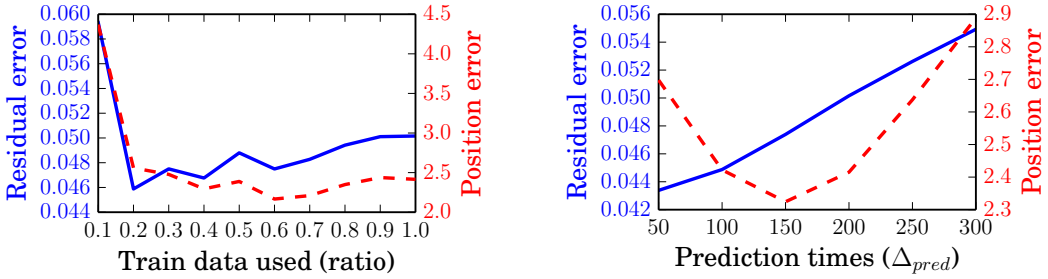
The predicted centroids are close to the target centroids, see Figure 5.4b. However, the ambient noise of the LSM drives the centroids towards the average motion seen during training. Competition across readouts could help reduce this noise.

Both error metrics decrease with respect to the amount of training data used, as seen in Figure 5.5a. The first two samples are entirely consumed after 20% ratio,



(a) Activity on the 32x32 sensor array (b) Target and predicted centroids

Figure 5.4.: (a) Selected test samples from the first scenario at prediction time $\Delta_{pred} = 200$ms The ball starts descending after reaching its highest point, the liquid predicts the fall. (b) Targets (black) and predicted centroids (blue) for the first scenario across the whole test motion. Source: [14].



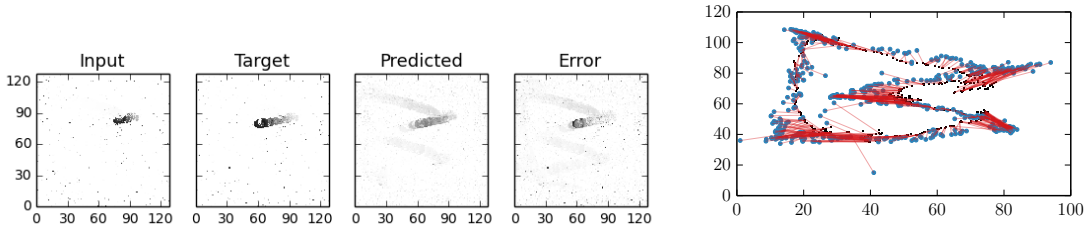
(a) (b)

Figure 5.5.: Residual error and positional error obtained in the first scenario: simulated ball jumping left and right. (a) Error with respect to the amount of training data used with $\Delta_{pred} = 200$ms. (b) Error with respect to prediction time Δ_{pred}. Source: [14].

corresponding to the first drop in error. Similarly, the two error metrics increase with respect to the amount of time predicted in the future, see Figure 5.5b. Initial decrease of centroid error until $\Delta_{pred} = 150$ms is due to the difficulty of providing good predictions at the beginning of the motion when only a few events have been received.

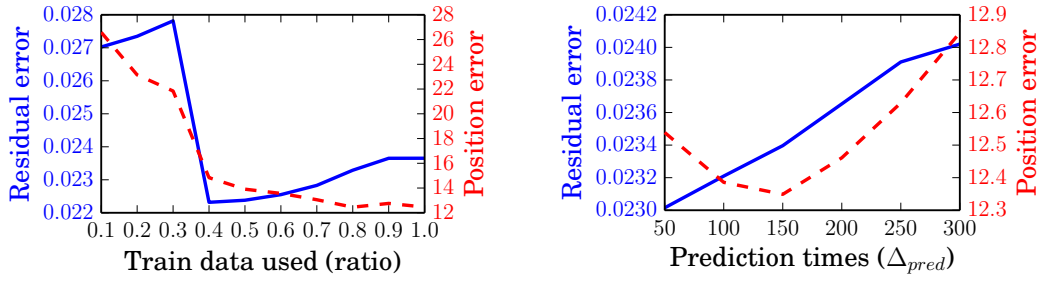
Second scenario: rolling ball on structure

The training set consists of the ball rolling two and a half times down the structure. When the ball arrives at the end of the structure, it is manually replaced on top of it. The total duration of the training set is 15 s. The test set consists of the ball rolling down the structure a single time lasting 6.8 s.



(a) (b)

Figure 5.6.: (a) Selected test samples from the from scenario at prediction time $\Delta_{pred} = 200$ms The ball is rolling on the structure, the liquid predicts its path. (b) Target and predicted centroids for the second scenario across the whole test motion. Source: [14].



(a) (b)

Figure 5.7.: Residual error and positional error obtained in the second scenario: ball rolling on a structure. (a) Error with respect to the amount of training data used with $\Delta_{pred} = 200$ms. (b) Error with respect to prediction time Δ_{pred}. Source: [14].

The sample in Figure 5.6a show that the same liquid used for the 32x32 simulated DVS can also perform prediction over the whole 128x128 sensor array using the real DVS.

The error metrics have the same shape as in the first scenario, the same observations can be drawn, see Figures 5.5 and 5.7. It can be noted that the residual error is smaller with the real 128x128 DVS than with the simulated 32x32 DVS. This is because the residual error is normalized by the number of readouts (see Equation (5.6)), and that the activation is sparser for the second scenario than the first one.

Third scenario: juggling three balls

The training set consists of a person juggling with three balls for $29.7\,\mathrm{s}$. The test set consists of the same person juggling for another $5.5\,\mathrm{s}$.

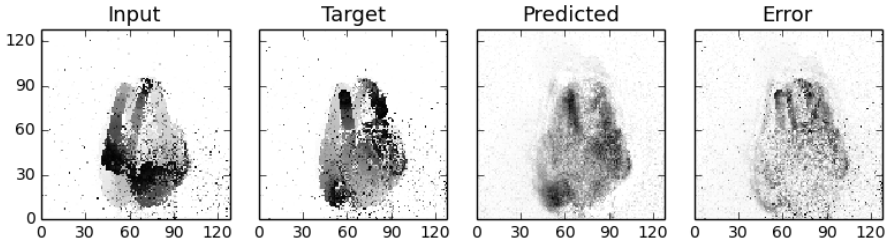
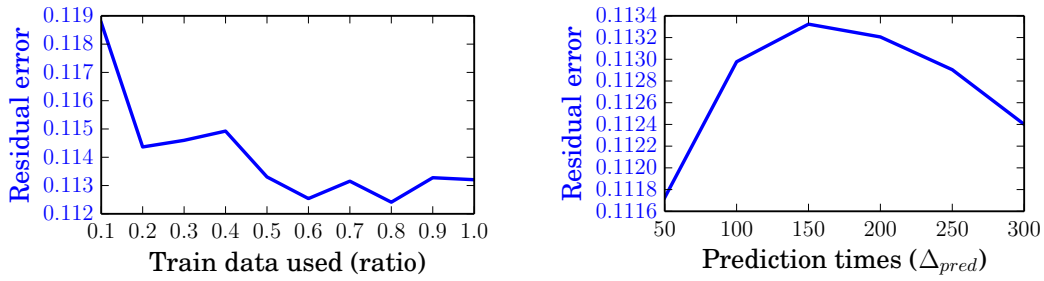


Figure 5.8.: Activity of the LSM during presentation of a juggling motion with prediction time $\Delta_{pred} = 200\,\mathrm{ms}$ projected on the 128x128 sensor array. In the liquid output, the position of the hands of the juggler and the two balls present in the target image can be recognized. Source: [14].



(a) (b)

Figure 5.9.: Residual error obtained in the third scenario: juggling with three balls. The positional error is not relevant in this case: since the juggler himself does not move across the image, the centroid of activation is static. (a) Error with respect to the amount of training data used with $\Delta_{pred} = 200$ms. (b) Error with respect to prediction time Δ_{pred}. Source: [14].

Since there is more than a single moving object in the third scenario, the activation is more chaotic. The juggler and the balls' silhouettes can still be recognized in the liquid's predictions, see Figure 5.8. However, these predictions are probably not sufficiently precise to be used in an actual robotic application. Increasing learning data might not help, as suggested by the error not decreasing anymore in Figure 5.9a. Since multiple objects are in motion, the centroid error metric is not calculated, as it is only suited to a single object in motion. This problem should be solved with a more advanced learning rule which also trains the recurrent weights, such as the recently derived e-prop rule.

5.1.4. Summary and Conclusion

The LSM can reasonably predict future visual input from address events when a single object is in motion. By evaluating the approach on various scenarios of increasing complexity, it has been shown that the method was able to learn different motions on the full scale 128x128 pixel arrays of the DVS, without any knowledge about its environment or physical laws. On the other hand, the presented method can not tackle complex scenes with multiple moving objects. Overall, the performance of the method remains below ANNs predicting images trained with backpropagation.

The most interesting aspect of the LSM resides in its genericity and the simple learning procedure. However, the simple learning procedure also admits drawbacks. A LSM has many hyper-parameters, important for regulating the dynamics of the SNN, which are complicated to tune. Besides, only the readout weights are trained, which is not sufficient for learning a hierarchy of visual representations as in the visual cortex. Lastly, training relies on a supervised off-line linear regression which requires data to be collected and labeled beforehand. To solve these problems, the synaptic learning rules presented next will be evaluated in multi-layered SNNs, with online weight updates.

5.2. Two-Factor Spike-Timing-Dependent-Plasticity

The material covered in this Section was originally published by the author in Kaiser et al. [9]. In this Section, the subject of learning with a synaptic plasticity rule is addressed, in contrast to the linear regression introduced in the previous Section with LSMs. Since the original STDP rule formulated in Bi et al. [59] and Markram et al. [188] does not include an error term (see Section 2.2.3), a synapse in a hidden layer can not receive information about the current performance on the task. Therefore, most experiments are conducted with STDP learning a single layer. To leverage this approach, the learning STDP layer can be stacked on top of another non-learning layer providing higher-level, predefined features.

The HMAX architecture introduced in Riesenhuber et al. [248] is the most commonly used approach to perform such experiments. It is based on a convolutional architecture, which has become in the last years the most popular methods to learn from images. The convolutional architecture takes advantage of the structure of the data to reduce the number of learned parameters and provide translation invariance. Indeed, when dealing with images, it is reasonable to assume that all the areas of the image should be processed in the same manner. Convolutional networks rely on learnable spatial filters that convolve the whole image. This operation can be seen as an ANN with shared weights representing the convolutional kernels.

The convolutional structure can be implemented with SNNs the same way it is implemented in analog networks, see Section 3.1.1. However, weight sharing needs to be adapted since synapses in SNNs are dynamical systems (see Section 2.2.1). Most often, the weights of a given kernel are harmonized at regular time intervals throughout the whole network. A biological phenomenon that could support this process is not known in the brain, which may be a hint to the non-plausibility of convolutional learning structures. It is suggested that such structures could be replaced by attentional ones in Section 4.1.2.

The contribution of this Section is to evaluate STDP in a HMAX architecture with the same implementation for images in Peric et al. [18] and for event-based data in Kaiser et al. [9]. Additionally, local layer classifiers in Kaiser et al. [9] allow us to monitor the quality of the features throughout the network. In Section 5.5, similar local layer classifiers are used to provide local loss functions to train the weights.

5.2.1. Method

Visual representations to be learned from address events are discussed in the following Section with the HMAX architecture and the original STDP rule, in a similar fashion than Masquelier et al. [194] and Peric et al. [18] for images. The architecture is depicted in Figure 5.10. The neurons in the S1 layer convolve the input spikes and are responsive to four edge orientations. Max pooling from S1 to C1 reduces dimensionality and contributes to translation invariance. Neurons in the S2 layer learn patterns out of preceding feature maps belonging to all orientations through STDP. A weight sharing mechanism ensures translation invariance across learned S2 features. Each S2 prototype has one max-pooling C2 neuron. Therefore, the firing of a C2 neuron indicates the occurrence of one specific combined-edges feature.

The same setup is evaluated on images encoded as Poisson spike-trains in Peric et al. [18] and on event streams in Kaiser et al. [9]. This allows a fair comparison of the performance between frame representation and AER. Additionally, it will be shown how to train classifiers for the different layers of the network in Kaiser et al. [9]. This allows us to evaluate the quality of the visual representations at different stages of the processing pipeline.

5.2.2. Experimental Setup

For the unsupervised STDP framework presented in Section 5.2, its potential to learn visual representations from images with a HMAX architecture has been demonstrated in Masquelier et al. [194] and Peric et al. [18]. Is a similar architecture capable of learning representations from event-based data? This study allows a direct comparison between the performance of a SNN trained on image

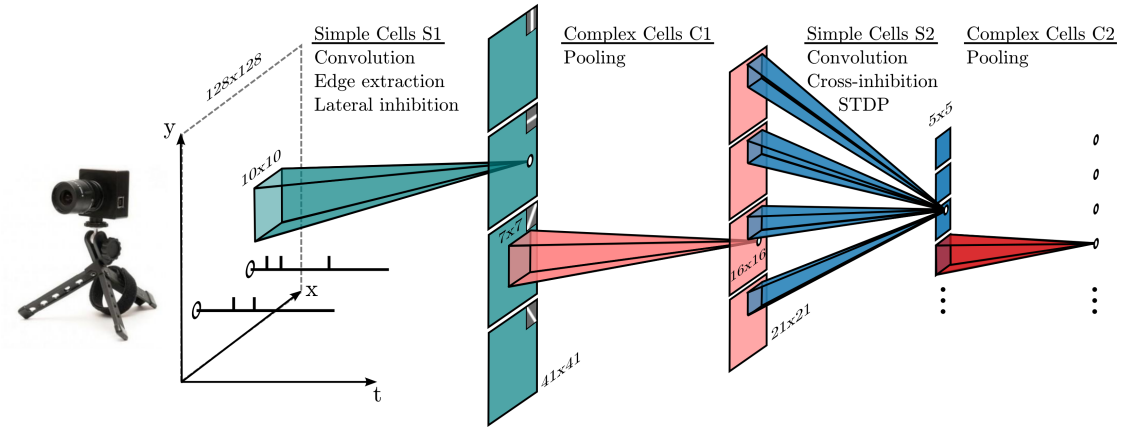


Figure 5.10.: Overview of the HMAX-based architecture presented in Peric et al. [18] and Kaiser et al. [9] and inspired by Masquelier et al. [194]. This architecture was adapted to event streams provided by a DVS (picture from iniVation AG). The S1 edge detector layer on their own combined with microsaccades achieve competitive feature extraction. copyright ©2018, IEEE [9].

data encoded with Poisson spike-trains and on event-based data. Additionally, three classifiers – Histogram, linear SVM, and LSM – are attached to every layer the HMAX architecture, see Figure 5.11. This allowed us to assess the quality of the representations learned in the layer hierarchy, a method similar to Alain et al. [42] or Kaiser et al. [12] for learning. Additionally, this provides a figure for the classification accuracy of these classifiers on the raw event stream, allowing comparison across the methods introduced in the following Sections.

The same HMAX architecture as described in Masquelier et al. [194] and Peric et al. [18] is used. Specifically, the first layer of simple cells consists of hard-coded edge features, while the second one learns with the STDP framework defined in Equation (2.9), and repeated here for convenience:

$$\Delta w \propto \begin{cases} f_-(w) \times e^{\frac{-|\Delta t|}{\tau}} & \text{if } \Delta t \leqslant 0 \\ f_+(w) \times e^{\frac{-|\Delta t|}{\tau}} & \text{if } \Delta t > 0 \end{cases}. \tag{5.9}$$

with $\Delta t = y_i - y_j$ the time difference between a post-synaptic and pre-synaptic spike pair, τ a time constant (usually around 20ms). The complete HMAX architecture is depicted in Figure 5.10. The hyper-parameters of this architecture were optimized with the black-box optimization method Covariance Matrix Adaptation Evolution Strategy (CMA-ES) against F1 score.

The main difference between the presented approach and the previous work is the encoding of the input. Instead of encoding visual information with Poisson spike-trains as Peric et al. [18] or variations of rank-order coding as Kheradpisheh et al. [153, 154], visual input consists of an event stream. In this case, one continuous stream is fed to the network without arbitrary delimitations between the

samples contained in the stream. This makes unsupervised learning even harder since the network does not know the beginning and end of an object presentation. Along with the event stream, a continuous label signal is used for training the classifiers to evaluate the features learned by the network.

This approach has been evaluated with the help of two datasets: N-Caltech101, a microsaccadic dataset, and DvsGesture, a motion dataset (see Section 4.2.1). Only three classes of N-Caltech101 are considered: faces, airplanes and motorcycles. This allows a direct comparison with the previous approaches in Masquelier et al. [194] Peric et al. [18] and Kheradpisheh et al. [153, 154], which evaluate on the same classes using the Caltech101 dataset.

Three different classifiers are used for each layer: histogram, linear SVM and LSM. All classifiers are trained to map corresponding layer activations to target labels in a supervised fashion, see Figure 5.11. This allows us to estimate the quality of the learned features at different stages of the processing pipeline.

For both the histogram and SVM, classification happens on a per-block basis. A block consists of the spikes emitted during the presentation of a given class, integrated in time to form a vector. In this case, the order of the spikes in the block does not influence the classifiers.

For the LSM, the corresponding spikes are streamed directly to the liquid. In this case, the order of the spikes during the presentation of a label influences the classification. The LSM is trained with the same procedure as described in Section 5.1.1, to predict a one-hot encoding classification vector.

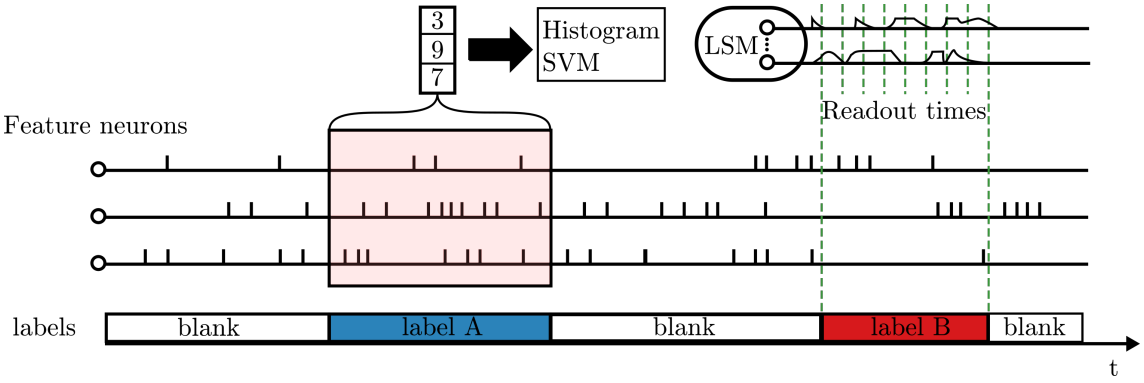


Figure 5.11.: Evaluation of learned features in an HMAX architecture by classifying spike-trains. Three classifiers are trained: histogram, linear SVM, and LSM. In the histogram and SVM case, time blocks are sliced from the event stream and aggregated into feature vectors. These time blocks are delimited with changes of labels, similar to classical samples. Conversely, the LSM is fed with the spikes directly.

5.2.3. Results

The experiments show that the edge extraction layers of the HMAX architecture provide a good representation of the event streams on the microsaccade dataset. Indeed, an F1 score of 98% is achieved by the SVM classifier attached to the S1+C1 layer with only 100 training samples per class, see Figure 5.12. The same SVM only achieves a F1 score of 89% when trained on raw events instead of extracted edges. This score outperforms the previously reported F1 score of 97% by Peric et al. [18] on Caltech101 encoded with Poisson spike-trains obtained with 350 training samples per class.

Interestingly, the F1 score for the SVM drops to 81% after the learning STDP layer. A similar performance drop between edge features and learned features is observed for all three classifiers, for both datasets. This denotes that the learning STDP layer fails to build qualitative visual representations from the extracted edges in the event stream. Indeed, in this case, the features learned with STDP seem to explode over the course of learning for both datasets, see Figures 5.13 and 5.15. This contrasts with other work relying on different encodings. Replacing the classic STDP formulation (Equation (5.9)) with the self-regulating formulation introduced by Kheradpisheh et al. [154] (see Equation (3.1)) could alleviate this issue.

For the motion recognition dataset, the classification of raw events has been consistently more accurate than the classification of extracted edge features, see Figure 5.14. This denotes that static edge features do not support the efficient building of visual representations for scenes containing natural human motions. This experiment also provides the accuracy of the LSM (42.78%) and the histogram method (45.83%) on the raw DvsGesture dataset. In contrast, the full HMAX architecture with STDP and SVM classifier only achieves 37.50% accuracy. These accuracies are compared with the rules presented in the next Sections in Table 5.2.

5.2.4. Conclusion

The performed experiments suggest that the original STDP formulation (Equation (5.9)) does not enable learning of features from event-based data in an unsupervised manner. These results are contradictory with previous experiments on images using the same framework in Peric et al. [18]. This denotes a difference in nature between Poisson spike-trains and event-based data. With Poisson spike-trains, a high-contrast pixel is guaranteed to spike at high-rate, each spike triggering the STDP update rule. However, high-contrast pixels do not spike at a high rate with event-based encoding, triggering fewer STDP updates. This lack of redundancy leads to instability in learning. This problem could be solved with the STDP formulation introduced in Kheradpisheh et al. [154], which implements soft bounds on the weights thus preventing weight explosion. In the next Section, more structured synaptic learning rules will be introduced.

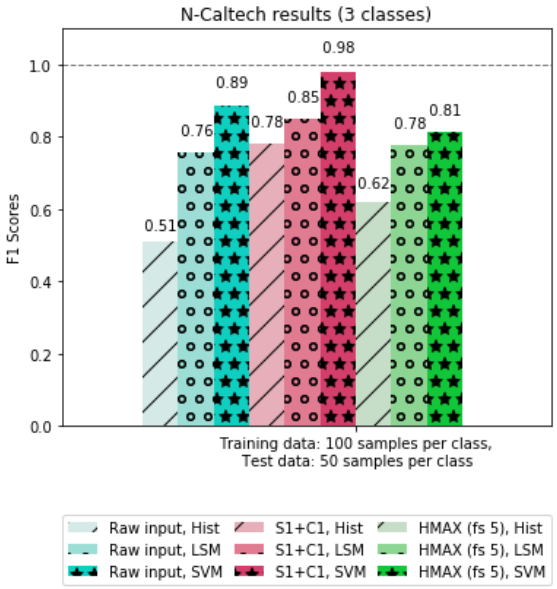
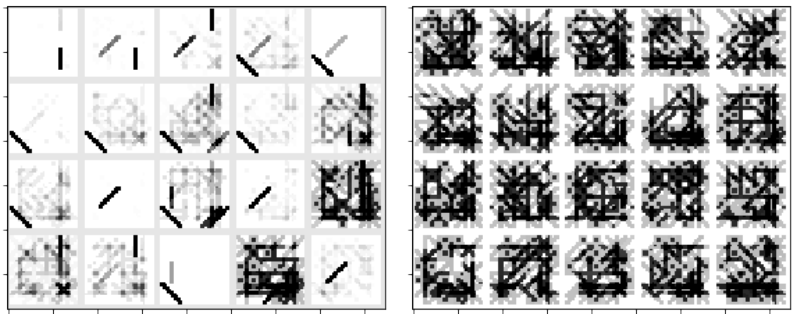


Figure 5.12.: Results of HMAX trained with STDP on N-Caltech101. F1 score results for a random subset of the N-Caltech101 dataset with three classes. The best F1 score of 98% is achieved with edge features and SVM classifier. copyright ©2018, IEEE [9].



(a) 5 learning samples (b) 83 learning samples

(c) 165 learning samples (d) 250 learning samples

Figure 5.13.: Features learned with STDP on N-Caltech101. The 20 kernels of the S2 neurons are projected back to the input layer after the presentation of a given number of learning samples of the N-Caltech101 dataset.

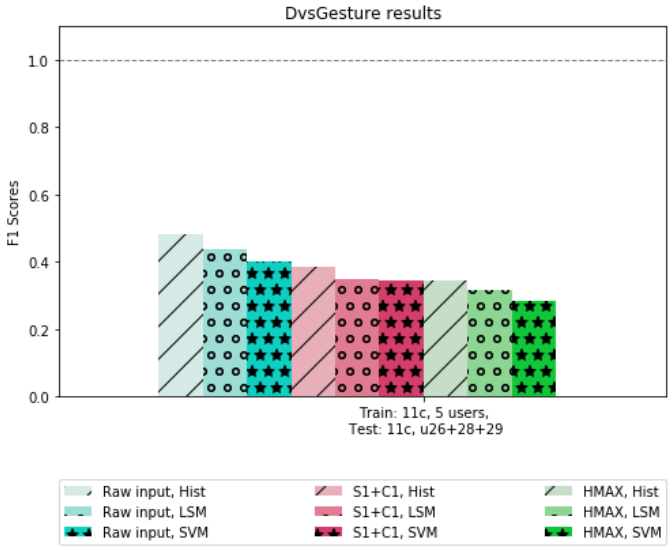
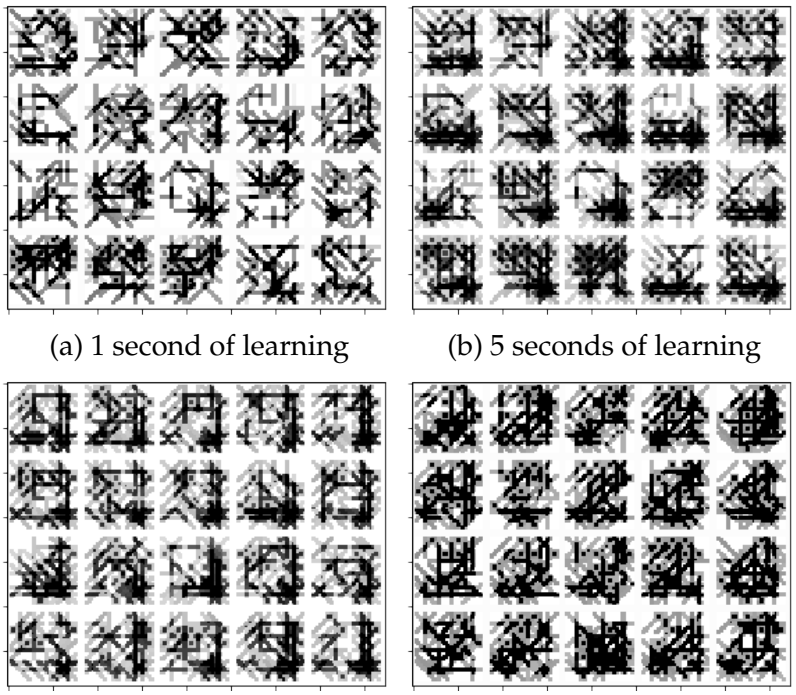


Figure 5.14.: Results of HMAX trained with STDP on DvsGesture. F1 score results on a subset of the DvsGesture dataset with 11 classes, five train users and three test users. The best F1 score of 45.83% is achieved with the raw event stream and a histogram classifier. LSM achieves 42.78% on the raw event stream. copyright ©2018, IEEE [9].



(a) 1 second of learning

(b) 5 seconds of learning

(c) 15 seconds of learning

(d) 41 seconds of learning

Figure 5.15.: Features learned with STDP on DvsGesture. The 20 kernels of the S2 neurons are projected back to the input layer after a given time of learning on the event stream *user01_lab*. copyright ©2018, IEEE [9].

5.3. Event-Driven Contrastive Divergence

The material covered in this Section was originally published by the author in Kaiser et al. [17]. In the previous Section, it has been shown that learning visual representation from AER in an HMAX architecture with the standard STDP formulation failed. A hypothesis for this failure is that event-based data, unlike images, is not suitable for this rule, leading to weight explosion and instability. In this Section, another synaptic learning rule is proposed: Event-Driven Contrastive Divergence [211] (eCD), see Section 3.1.2.

Similarly to STDP, eCD is unsupervised, and relies on spike-times to compute weight updates. This learning rule approximates the Contrastive Divergence rule to train spiking RBM. A RBM is a two-layers network with bidirectional symmetric weights. One aspect of this contribution is to bring the convolutional structure to the spiking RBM in Kaiser et al. [17].

5.3.1. Method

In this Section, it is shown that visual representations can be learned from address events with the eCD synaptic learning rule, in a similar fashion than Neftci et al. [211] for images.

The rule is formulated as a hebbian learning rule (see Figure 2.6a), with a global modulatory signal g indicating the current phase:

$$\Delta w \propto g(t) \times |f_+(w)| \times e^{\frac{-|\Delta t|}{\tau}}. \tag{5.10}$$

Five phases are described in this implementation (against four in the original formulation in Neftci et al. [211], see Section 3.1.1) – determined by the value of $g(t)$ – for a given sample. The different phases are depicted in Figure 5.16. The input is clamped on the visible layer only during the first phase (burn-in).

One of the contribution of this work is to bring the convolutional structure to the spiking RBM. Convolutional RBMs were already proposed for analog networks in [87, 168, 220]. In our spiking implementation, max pooling layers are replaced with lateral inhibition to enforce sparseness. Specifically, two types of lateral inhibition are introduced: between all neurons within the same feature map, and between all neurons having the same location across different feature maps, see Figure 5.17b. The former increases sparsity in a feature map by reducing the probability of a neighbor neuron to fire when another already fired, similar to probabilistic max pooling. The latter enforces learning of discriminative features by reducing the correlation between different kernels, see Figure 5.17b. Let $w_{ijk}^{i'j'k'}$ be the weight between two hidden-layer neurons x_{ijk} and $x_{i'j'k'}$ at position i, j

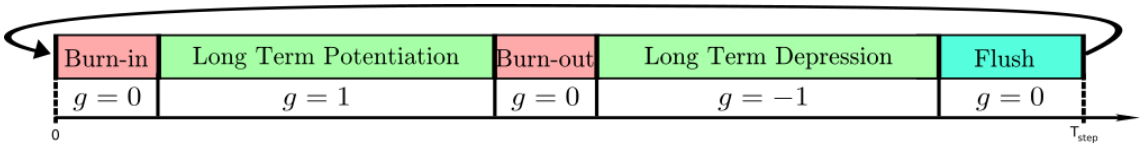
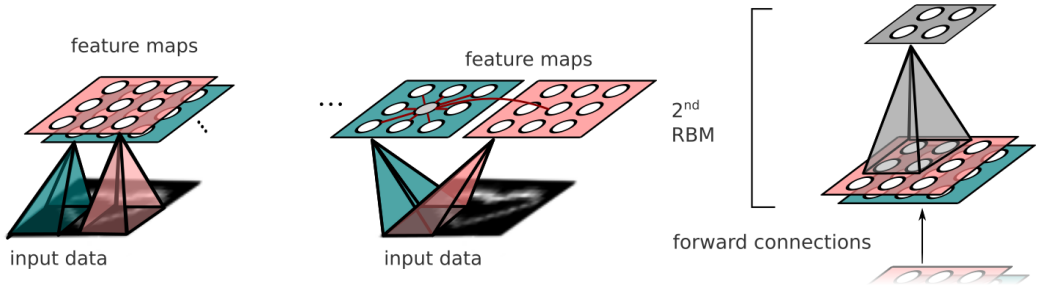


Figure 5.16.: The five phases of eCD for a training step. The input stream is only clamped to the visible units during the burn-in phase. In the second phase, the data is learned by triggering weight increase on spike correlation (LTP). Learning is then deactivated in the burn-out phase. In the fourth phase, the model is unlearned by triggering weight decrease on spike correlation (LTD). Unlike Neftci et al. [211], a fifth phase is added to flush the network. A training step is simulated for $T_{step} = 168\text{ms}$, the learning phases last 8%, the burn phases 36% and the flush phase 12%. Source: [17].

and i', j' at feature map k and k' respectively in the following equation:

$$w_{ijk}^{i'j'k'} = \begin{cases} \beta_1, & \text{for } |i-i'| < b_d \text{ , } |j-j'| < b_d \text{ and } k \neq k' \\ \beta_2, & \text{for } |i-i'| < b_s \text{ , } |j-j'| < b_s \text{ and } k = k' \\ 0, & \text{otherwise} \end{cases}, \tag{5.11}$$

where b_d and b_s are the neighborhood size in different feature maps and within the same feature map respectively, and $\beta_1, \beta_2 \leq 0$ are inhibitory weights. On top of the required synchronization eCD requires for bidirectional weights, the convolutions induce another required synchronization within filters.



(a) feature maps (b) lateral inhibition (c) deep belief network

Figure 5.17.: Schema of the architectures for the proposed Spiking Convolutional Deep Belief Network (SCDBN). (a) The hidden layer of a SCRBM is organized in feature maps convolving the input. (b) The feature maps have inhibitory connections from one to another to encourage feature discrimination, and across the local neighborhood for sparsity. (c) A SCDBN consisting of two stacked SCRBMs, connected with a feedforward layer, see Kaiser et al. [17]. Source: [17].

5.3.2. Experimental Setup

eCD has been evaluated in a 2-layers RBM architecture, referred to as SCDBN. The first layer is convolutional with ten filters of size 10×10 and performs the feature extraction. The second layer is fully connected and performs the association between the extracted features and the correct labels. The complete architecture is depicted in Figure 5.18.

This network has been evaluated on two event-based datasets: the open Poker-DVS in Serrano et al.[263] and an early version of the self-recorded ball bottle and pen dataset, later referred to as Ball-Can-Pen dataset, see Section 4.2.1. The Ball-Can-Pen dataset was recorded with a DVS looking at images flashing on a screen (unlike the next version recorded with microsaccades). In this case, the dataset consists of only three classes, without a background class. There are 90 samples for each of the three classes, each sample of dimension 16×16 pixels lasting 100ms.

To underline the benefits of the proposed convolutional architecture, The performance is compared against a similar architecture but without any convolutions, with ten fully connected hidden units for extracting features. This non-convolutional architecture is similar to the one used in Neftci et al. [211], with an additional layer.

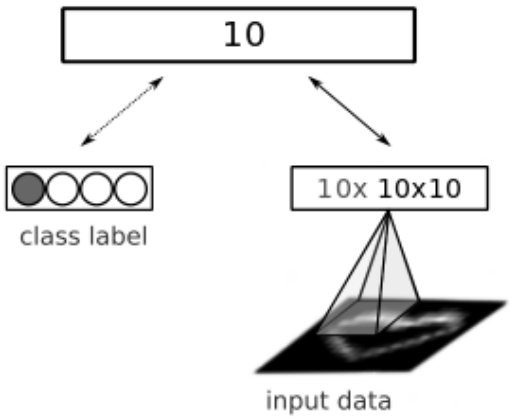


Figure 5.18.: The SCDBN consists of 16×16 visible neurons, 10 feature maps of size 10×10 and 10 association neurons. Source: [17].

5.3.3. Results

The number of parameters and the classification accuracy and runtime per sample can be seen in Table 5.1 and in Figure 5.19. Since the classification is obtained without external SVM, and features are not hard-coded as in the first layer of the HMAX architecture, the accuracy is only influenced by the synaptic learning rule for a given architecture. Unlike the pure STDP experiment presented in

Section 5.2, eCD manages to learn representations from event-based data in an unsupervised fashion without exploding weights.

The accuracy is higher than the pure STDP experiment (Section 5.2.3). With fewer learning parameters, the convolutional architecture has a higher classification performance. Moreover, the model reaches 90% accuracy on Poker-DVS, comparable to state-of-the-art methods learning offline.

	Accuracy	#Parameters 1st layer		Runtime
Ball-Can-Pen	0.82	2560		4.8 s
Ball-Can-Pen convolution	1.0	1000		6.2 s
Poker-DVS convolution	0.90	1000		6.2 s
Poker-DVS Spiking CNN [263]	0.91	600	(offline)	-
Poker-DVS H-First [263]	0.975	0	(hard-coded)	-

Table 5.1.: Classification accuracy of the SCDBN on Poker-DVS and Ball-Can-Pen dataset. Due to the small amount of training samples, testing is performed on the training set. Despite fewer parameters for the convolutional architecture, runtime per sample is longer due to the shared synapses having their own dynamics. Source: [17].

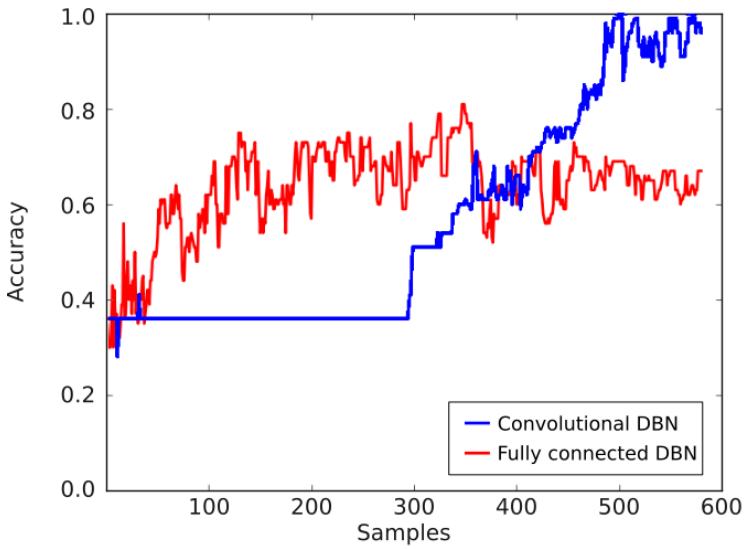


Figure 5.19.: Evaluation of the SCDBN on the Ball-Can-Pen dataset. Classification accuracy on the Ball-Can-Pen dataset of our method against a similar network without convolutions, as presented in Neftci et al. [211]. Source: [17].

5.3.4. Conclusion

It has been shown that eCD was capable of learning representations from event streams. However, this rule has many constraints, which makes its implementation complicated, especially in neuromorphic hardware. Mainly, this rule relies on synchronization at different levels, while SNNs communicate asynchronously. Synchronicity is necessary for the different learning phases implemented by the global factor $g(t)$. The duration of these phases is an important meta-parameter. This means that input can not be streamed continuously to the network, but instead samples need to be clearly delimited in time. This is not practical for event-based processing, which defines a continuous stream by design.

Additionally, the bidirectional weights between forward and backward connections are biologically not plausible and also require synchronization. This problem is referred to as the weight transport problem, and is also relevant in other learning rules such as backpropagation. Lastly, deep networks are trained in a greedy fashion, where a first RBM is trained, then a second, and so on. This type of training is not suitable for online learning.

5.4. Event-Driven Random Backpropagation

Some of the material covered in this Section was originally published by the author in Kaiser et al. [6]. In the previous Section, eCD – an interpretation of Contrastive Divergence for SNNs – has been evaluated. While this rule was capable of learning representations from event-based data, it had many cumbersome constraints with respect to weight sharing and synchronicity, seriously hindering the advantages of SNNs.

In this Section, Event-Driven Random Backpropagation [213] (eRBP) – an interpretation of the random backpropagation introduced in Lillicrap et al. [174] and Nøkland [219] for SNNs – is discussed. This rule is a three-factor voltage-based learning rule and requires an error term. This error term prescribes a supervised learning rule, unlike STDP and eCD which were unsupervised. However, advances in deep learning suggest that supervised learning rules such as backpropagation can also solve unsupervised [101, 75] and reinforcement learning [259, 200] tasks. These paradigms are achieved with a definition of the loss function with respect to a reward or to the state of the network itself.

The contribution of this Section shows that spatio-temporal representations can be learned from real visual event-based sensor data with eRBP in SNNs. Additionally, it is shown that the covert attention mechanism introduced in Section 4.1.2 further improves the performance by providing translation invariance at low computational cost, without convolutions. Indeed, convolutions require a weight sharing mechanism that is neither biologically plausible nor easy to implement in neuromorphic hardware. Lastly, this rule has been integrated into a

real-world closed-loop robotic grasping setup involving a robotic head, arm and a five-finger hand, as shown in Section 6.1.

5.4.1. Method

The original formulation of eRBP from Neftci et al. [213] that has been reported in Section 3.1.4 is used. The learning rule can be generally expressed as:

$$\Delta w_{ij}(t) \propto \sum_{k \in out} e_k(t) g_{ik} \times box(u_i(t)) \times s_j(t), \tag{5.12}$$

with s_j the pre-synaptic spike-trains, *box* the boxcar function[1] with bounds b_{min} and b_{max}, e_k the prediction error for output neuron k and g_{ik} a fixed random feedback weight. The term *out* refers to the set of output neurons. This generic formulation is used in Appendix B to provide a comparison between eRBP and DECOLLE and derive an updated version of eRBP.

Since $s_j(t)$ is either 0 or 1 at a given time t, the learning rule can be triggered on pre-synaptic spikes instead of continuously applied. This yields the simplified weight update equation from Equation (3.11):

$$\Delta w_{ij}(t) \propto \begin{cases} \sum_{k \in out} e_k(t) g_{ik} & \text{if } b_{min} < u_i(t) < b_{max} \\ 0 & \text{otherwise} \end{cases}. \tag{5.13}$$

The term $\sum_{k \in out} e_k(t) g_{ik}$ is the credit of neuron i, computed as a random combination of the network errors, a method referred to as direct feedback alignment. To avoid the transmission of analog data for learning, the errors $e_k(t)$ can be encoded as spikes. In this case, dedicated error neurons spike at a rate proportional to the value of the error. These error neurons are connected to all hidden neurons with the random feedback weights g_{ik}. In every hidden neuron, the credict is subsequently integrated and stored in a dedicated compartment, see Section 3.1.4. This technique is used for the evaluation in this Section. In other words, both processing data and learning from data is exclusively based on spikes, no continuous values are communicated from a neuron to another.

5.4.2. Experimental Setup

The eRBP rule has been evaluated on the full DvsGesture dataset. Unlike the evaluation of STDP with HMAX, the architecture consisted of two hidden layers of 200 neurons densely connected in a feedforward manner. The network is depicted in Figure 5.22a. All the connections are trained with eRBP. Additionally, the ON- and OFF-events obtained from the DVS are separated into two distinct input channels, as described in Section 4.1.2. All synapses are stochastic, with

[1] $box(x) = 1$ if $b_{min} < x < b_{max}$, 0 otherwise

a 35% chance of dropping each spike, making the network activity sparser and learning more robust. This method is comparable to dropout and has been reported to improve learning accuracy [212]. The voltage gates for learning are set to $b_{min} = -0.6$ and $b_{max} = 0.6$. The presented experiments used on the open-source implementation of eRBP[2] based on the neural simulator Auryn [297]. The code used to obtain the results presented in this Section has been open-sourced[3]. The evaluation was performed with and without the attention mechanism described in Section 4.1.2. The number of events to calculate the position of the attention window was set to $n_{attention} = 1000$, see Figure 4.2. The same architecture and method is later used to learn discrete visual reaching and grasping affordances in Section 6.1.

5.4.3. Results

The evaluation on the DvsGesture dataset shows that eRBP efficiently learns to classify motions from raw event streams with the attention mechanism. The accuracy of 92.7% is achieved after only 60 epochs, corresponding to approximately 127h of training data, see Figure 5.20. This accuracy is comparable to state-of-the-art deep networks (IBM EEDN [43], 94.49%) and other synaptic learning rules taking temporal dynamics into account (DCLL [12], 94.19%), both relying on convolutions. Without the attention mechanism, the accuracy of the network drops to 86.1%. The dimension of the event stream is 64×64 in both cases. These results confirm that our simple covert attention mechanism provides translation invariance without convolutions, at a low computational cost. The drop of accuracy in the early stages of learning in the attention window case is due to the stochastic synapses, which have a 35% chance of dropping spikes. The rescaling approach is resilient to this stochasticity since all events in the original stream are squeezed into macro-pixels, leading to redundant events.

Additionally, unambiguous samples are classified in under $0.1\,\mathrm{s}$, with increasing confidence over time, see Figure 5.21. The rhythm of the "left hand waving" motion is visible in the input spike-train. The neurons in the hidden layers spike close to their maximum frequency, as limited by the refractory period.

Since DvsGesture is a classification task, not accounting for neural temporal dynamics in the learning rule (Equation (3.11)) does not impact the performance considerably. Indeed, the target output signal as encoded by label neurons is constant across a training sample for durations of several seconds. We expect this omission to decrease performance significantly for a temporal regression task – such as learning a time sequence – where the temporal dynamics of the target signal is relevant.

[2] `https://gitlab.com/eneftci/erbp_auryn`
[3] `https://github.com/HBPNeurorobotics/auryn`

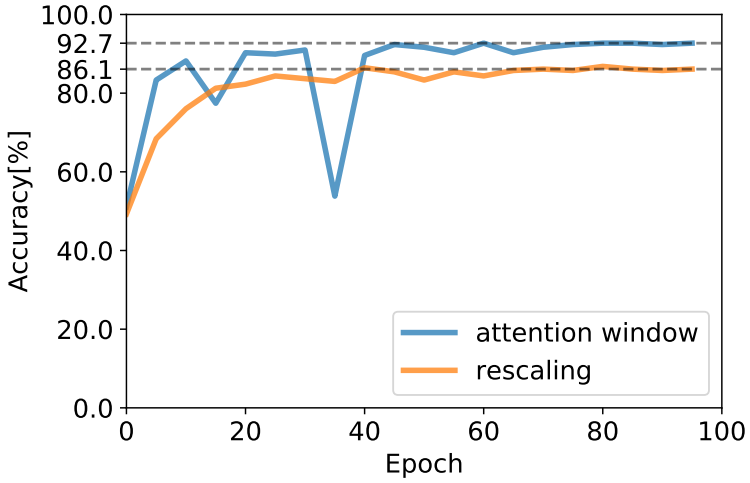


Figure 5.20.: Classification accuracy for the DvsGesture task during learning with eRBP. Source: [6].

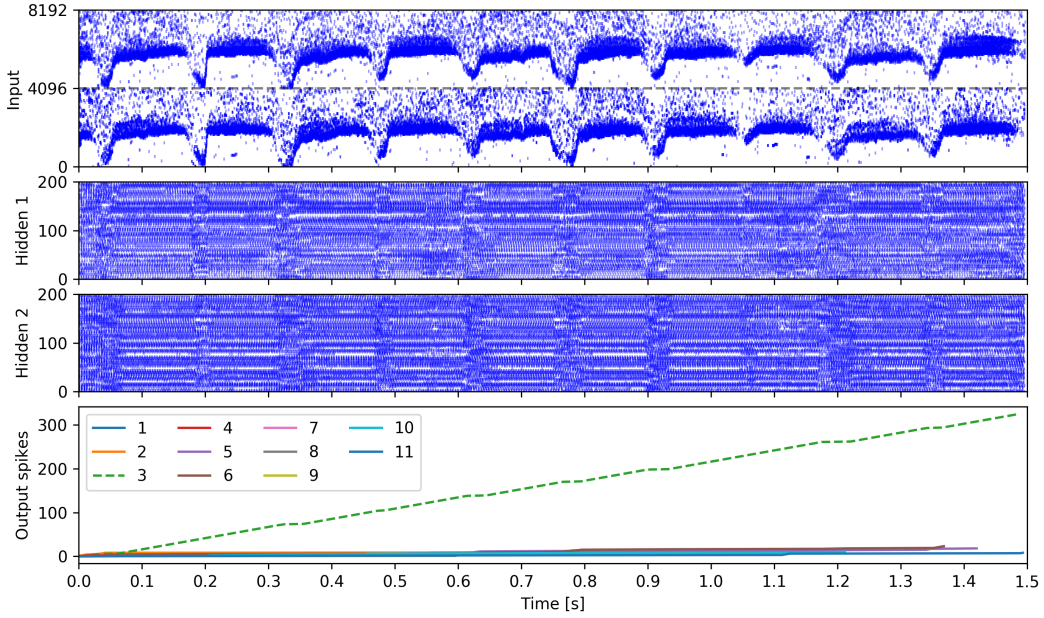


Figure 5.21.: spike-trains for a test sample of class “left hand waving” from DvsGesture dataset. The network manages to correctly classify the sample in less than $0.1\,\mathrm{s}$, with increasing confidence over time. Source: [6].

5.4.4. Conclusion

It has been shown that eRBP is capable of learning representations from event streams resulting from both dynamic scenes and fixational eye movements. Additionally, the simplicity of this rule (two comparisons and one addition) makes it a good candidate for neuromorphic hardware implementation.

However, while efficient on classification tasks, eRBP is not capable of learning time sequences since the weight update described in Equation (3.9) does not take the temporal dynamics of the IF neuron into account (PSPs and refractory period). This omission was discussed in the following publications of Zenke et al. [296] and Bellec et al. [55], as well as in our own work in Kaiser et al. [12] which is derived and evaluated in the next Section. Subsequently, an updated version of eRBP taking into account the PSP dynamics was derived in Appendix B.

5.5. Deep Continuous Local Learning

The material covered in this Section was originally published by the author in Kaiser et al. [12]. Like eRBP, DECOLLE is another gradient descent method for multi-layers SNNs. Unlike eRBP, the temporal dynamics of the spiking neurons are taken into account. This can be accomplished with the addition of eligibility traces matching the time constant of the PSPs. This technique was originally proposed for SNNs by Zenke et al. [296] with SuperSpike.

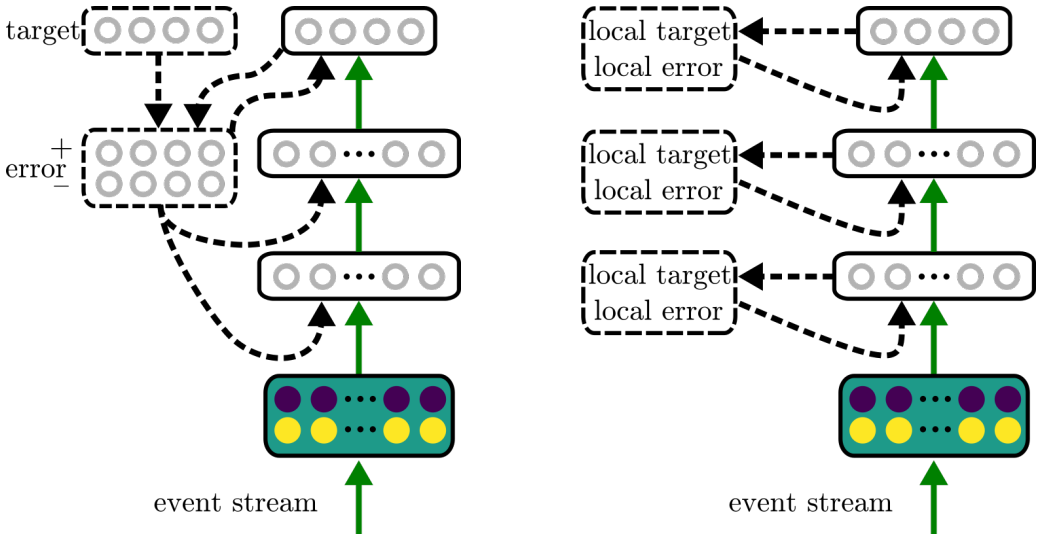
The difference between SuperSpike and DECOLLE resides in the formulation of the loss function maximized by the network. In SuperSpike, the loss function is the Van Rossum distance, formulated for an output neuron i as:

$$L(T) = \frac{1}{2} \int_{-\infty}^{T} \left[\alpha * s_i(t) - \alpha * \hat{s}_i(t)\right]^2 dt, \tag{5.14}$$

with $*$ denoting a temporal convolution and α a smooth kernel, s_i and $\hat{s}_i$ the output and target spike-trains of output neuron i respectively.

This allows SuperSpike to learn specific patterns of precise spike-times. However, as will be seen in the DECOLLE derivation, this loss function implies that eligibility traces are specific to the synapse. The amount of additional memory required to implement the SuperSpike learning rule is therefore proportional to the number of synapses. On the other hand, the loss function with DECOLLE is a sum of layer-specific loss functions defined against an analog value. This allows the eligibility traces to be stored in the neurons, leading DECOLLE to scale in space with respect to the number of neurons. This difference is depicted in Figure 5.22b.

The main contribution of this Section is the derivation of DECOLLE, a new synaptic learning rule which outperforms state-of-the-art rules on event-based dataset with fewer iterations.



(a) eRBP learning architecture (b) DECOLLE learning architecture

Figure 5.22.: Learning architectures for eRBP and DECOLLE. In eRBP, errors are calculated from network-level output and network-level targets. Error-related computations are spike-based. In DECOLLE, errors are calculated from layer-level output and layer-level targets. Error-related computations are analog. Source: [6, 12].

5.5.1. Method

In the following the derivation of DECOLLE is provided and explained. Let's consider the SRM formulation of a post-synaptic IF neuron i as described in Equation (2.4):

$$\begin{aligned} u_i(t) &= \sum_{j \in pre} w_{ij}(\epsilon * s_j(t)) + \eta * s_i(t), \\ s_i(t) &= \Theta(u_i(t)), \end{aligned} \tag{5.15}$$

where u_i is the membrane potential, s_i the spike-train of neuron i, w_{ij} the synaptic weight and Θ the unit-step function. The temporal convolution kernels ϵ and η reflect PSP and refractory period dynamics, respectively.

We can use the chain rule to express the gradient of a loss function L with respect to the weight w_{ij}:

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial s_i} \times \frac{\partial s_i}{\partial u_i} \times \frac{\partial u_i}{\partial w_{ij}}. \tag{5.16}$$

References to time are omitted by an abuse of notation. In the following, the calculation of these terms is provided.

Calculation of $\frac{\partial u_i}{\partial w_{ij}}$: The refractory kernel η is difficult to account for because of its temporal dependence on previous activity. As a simplification, it is therefore

traditionally ignored in the derivation. This omission only has a weak impact on the performance, as long as the loss function incorporate a regularization term enforcing low spiking rates.

$$\begin{aligned}\frac{\partial u_i}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \sum_{j \in pre} w_{ij}(\epsilon * s_j) + \eta * s_i \\ &= \frac{\partial}{\partial w_{ij}} \sum_{j \in pre} w_{ij}(\epsilon * s_j) + \frac{\partial}{\partial w_{ij}} \eta * s_i \\ &= \epsilon * s_j + \frac{\partial}{\partial w_{ij}} \eta * s_i \\ &\approx \epsilon * s_j.\end{aligned} \tag{5.17}$$

We therefore have to compute and store the convolved pre-synaptic spikes for learning. This is done online with eligibility traces – a dynamical state maintained in time. Since this term only depends on the pre-synaptic activity and not on the post-synaptic activity, it can be stored in pre-synaptic neurons and reuse by other synapses.

Calculation of $\frac{\partial s_i}{\partial u_i}$: This term is problematic since the unit-step function Θ is not differentiable. The solution is to approximate it with a surrogate derivative (also called pseudo-derivative) [214], see Section 3.1.4. Here Θ is replaced with a sigmoid function σ for gradient computation, as if the neuron had a stochastic spiking behavior:

$$\begin{aligned}\frac{\partial s_i}{\partial u_i} &= \frac{\partial}{\partial u_i} \Theta(u_i) \\ &\approx \sigma'(u_i).\end{aligned} \tag{5.18}$$

This term relates to the post-synaptic voltage.

Calculation of $\frac{\partial L}{\partial s_i}$: This term represents the error and depends on the choice of the loss function L. In DECOLLE, we calculate pseudo-outputs y_k with linear fixed-weights readouts mapping the post-synaptic spikes: $y_k = \sum_{i \in post} g_{ki} s_i$. Assuming a mean square error loss, we can write: $L = \frac{1}{2} \sum_{k \in rdout} (\hat{y}_k - y_k)^2$, with $\hat{y}_k$ the pseudo-targets for the pseudo-outputs. The terms *post* and *rdout* refer to the set of post-synaptic and readout neurons respectively. In this case, the calculation

of the error signal term is:

$$\begin{aligned}
\frac{\partial L}{\partial s_i} &= \frac{\partial}{\partial s_i}\frac{1}{2}\sum_{k\in rdout}(\hat{y}_k - y_k)^2 \\
&= \frac{1}{2}\sum_{k\in rdout}\frac{\partial}{\partial s_i}(\hat{y}_k - y_k)^2 \\
&= \sum_{k\in rdout}(\hat{y}_k - y_k)\frac{\partial}{\partial s_i}(\hat{y}_k - y_k) \\
&= -\sum_{k\in rdout}(\hat{y}_k - y_k)\frac{\partial}{\partial s_i}y_k \\
&= -\sum_{k\in rdout}(\hat{y}_k - y_k)\frac{\partial}{\partial s_i}\sum_{i\in post} g_{ki}s_i \\
&= \sum_{k\in rdout}\underbrace{(y_k - \hat{y}_k)}_{e_k} g_{ki}.
\end{aligned} \tag{5.19}$$

This equation implies that the weights g_{ki} used for the forward computations have to be known by the feedback process used for learning. This constraint is not biologically plausible, it is referred to as the weight transport problem. Using fixed random weights instead does not decrease the performance significantly. This technique referred to as direct feedback alignment was originally introduced by Nøkland et al. [219].

Final Expression: The rule in Equation (5.16) is therefore a multiplication of three terms related to pre-synaptic activity, post-synaptic voltage and an error term. It is a three-factor voltage-based rule, as discussed in Section 3.1.4. Injecting the calculated terms back into the learning equation Equation (5.16) results in:

$$\frac{\partial L}{\partial w_{ij}} = \sum_{k\in rdout} e_k g_{ki} \times \sigma'(u_i) \times \epsilon * s_j. \tag{5.20}$$

This equation is valid for a mean square error loss assuming s_i are the spikes of the output layer. With DECOLLE, we train multi-layer networks by computing pseudo-outputs with linear readouts for all layers, see Figure 5.22b. Gradient descent is therefore achieved with the following weight update equation, which stands for all synapses:

$$\Delta w_{ij}^l(t) \propto \sum_{k\in rdout^l} e_k^l(t) g_{ki}^l \times \sigma'(u_i^l(t)) \times \epsilon * s_j^{l-1}(t), \tag{5.21}$$

with superscript l denoting the neural layer to which the variable belongs. The three terms of Equation (5.21) required to compute the gradients are computed as part of the neural dynamics (propagated forward), enabling weight updates to be performed online. This method, named RTRL, enables learning of temporal sequences of different duration with constant memory, because the information

to compute the gradient is integrated and stored in a neural state. This contrasts with BPTT which stores the history of all previous neural activity.

To compute the weight updates online, $\epsilon * s_j$ has to be maintained dynamically. Since this term only depends on the pre-synaptic neuron, this variable can be stored as an eligibility trace in the neurons. In SuperSpike [296] and SLAYER [265], the loss function is a Van Rossum distance, which introduces an additional temporal convolution α over the spike-trains, see Equation (5.14). This additional temporal convolution prevents the eligibility traces to be shared across neurons. Therefore, these rules scale spatially with respect to the number of synapses. In Kaiser et al. [12], it is shown how the neural dynamics of DECOLLE can be expressed with respect to the eligibility traces, leading the spatial complexity of DECOLLE to be constant.

Similar to a LSM, the readouts of every layer can be seen as decoders of the spiking activity to analog signals. Unlike a LSM, the decoder weights are fixed, and the rule consists of learning the encoding weights to decrease the loss with respect to the local error. This rule was inspired by the realization that layer-wise errors can be synthesized by an external module [202, 144]. In this case, the loss function is the sum of the layer-wise loss functions, *i.e.*:

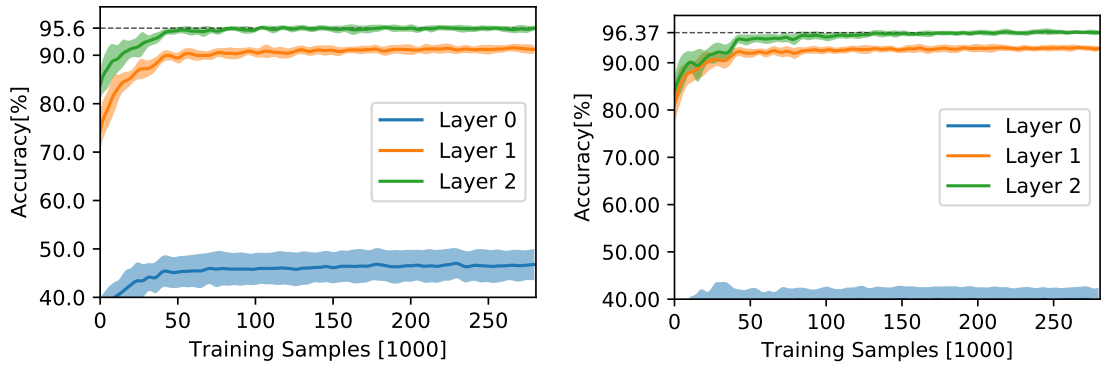
$$\mathcal{L} = \sum_{n=1}^{N} L^n(\mathbf{y}^n, \hat{\mathbf{y}}^n), \tag{5.22}$$

where $\hat{\mathbf{y}}^n$ is the pseudo-target for layer n, plus a regularizer on neural activity and membrane potential. For classification tasks, the pseudo-targets for all layers are simply the labels of the samples. In other words, each layer in the network will try to maximize its own local classification score using the representation from the layer below.

5.5.2. Experimental Setup

DECOLLE has been implemented with the PyTorch deep learning framework and has been made open-source[4]. This implementation allows us to rely on out-of-the-box automatic differentiation tools, optimization methods, convolutional architectures, GPU simulations and mini-batch iterations. The IF neuron dynamics in Equation (5.15) are therefore implemented in discrete time – one forward pass corresponds to $1\,\mathrm{ms}$ of biological time. This implementation expresses SNNs as binary recurrent networks, as noted in Neftci et al. [214]. The recurrence represents the dynamics of a IF neuron, which carries its leaking membrane potential and refractory period from a state to another. Within the PyTorch framework, synaptic delays have not been implemented – spikes are transmitted directly from a layer to another as they happen.

[4] `https://github.com/nmi-lab/decolle-public`



(a) Input rescaling to 32x32

(b) Attention window 32x32

Figure 5.23.: Classification accuracy for the DvsGesture task during learning with DECOLLE. Shadings indicate standard deviation across the 7 runs. Source: [7, 12].

The evaluation of DECOLLE has been done on the DvsGesture dataset. The network receives as input event streams of dimension 32×32, binned in frames of $1\,\mathrm{ms}$. The experiment is repeated both with input rescaling and with the attention mechanism presented in Section 4.1.2, which was implemented and released as part of the torchneuromorphic library [5]. For DvsGesture, the network is trained with $500\,\mathrm{ms}$ sequences and tested on $1800\,\mathrm{ms}$ sequences. The predicted class is calculated by counting spikes at the output after an initial "burn-in period" of $50\,\mathrm{ms}$. The network consists of three convolutional layers slightly adapted from the architecture in Springenberg et al. [273] with 7x7 kernels. The experiment is run 7 times with different random seeds to evaluate the standard deviations, both with and without the attention mechanism.

5.5.3. Results

The achieved performance of 95.60% with rescaling and 96.37% with the attention mechanism is better than other known related SNN implementations that rely on gradient descent for training (Figure 5.23). DECOLLE reached the reported accuracy after a significantly smaller number of iterations compared to the IBM EEDN case reported in Amir et al. [43], see Table 5.2. Furthermore, the proposed network achieved these results on the basis of a less complex and smaller network compared to the results reported in other related work.

5.5.4. Conclusion

A new synaptic plasticity rule called DECOLLE has been introduced, providing online gradient descent to SNNs. This rule extends eRBP with the introduction

[5] `https://github.com/nmi-lab/torchneuromorphic`

of eligibility traces, solving the temporal credit assignment problem. It also improves over SuperSpike, as its spatial complexity is linear instead of quadratic.

Out of all the approaches evaluated in this thesis, DECOLLE is the one that provided the best accuracy on event stream benchmarks. Compared to eRBP, this rule can also learn temporal regression, as demonstrated in an additional experiment in Kaiser et al. [12]. This rule is also the first one considered in this thesis which has been implemented with a generic computational framework – PyTorch – instead of a dedicated neural simulator. Such frameworks considerably simplify the prototyping of synaptic learning rules and have now been applied and extended by the computational neuroscience community. Additionally, the concept of local errors (pseudo-targets) that we introduced in this learning rule is an actual research topic in neuroscience. Such learning signals are assumed to exist in the brain, but which information they encode and how they are orchestrated is a topic of actual research.

5.6. Summary and Conclusion

Summarizing, state-of-the-art synaptic plasticity rules have been derived, integrated and evaluated to efficiently learn spatio-temporal visual representations from event-based data. This Chapter started with simple rules that were derived in a bottom-up strategy – from observations in the brain – such as STDP. It continued with top-down rules that have been derived from gradient descent and predicting modulatory synaptic plasticity. It could be shown with a set of standard experiments, that there is a large accuracy gap between bottom-up rules such as STDP and top-down rules derived from backpropagation. This is best demonstrated with the obtained accuracy on the DvsGesture dataset, reported in Table 5.2. The derivation of the Continuous Random Backpropagation [7] (cRBP) rule is provided in Appendix B – the evaluation procedure (network architecture, optimizer, regularization, learning rate) is identical to that of DECOLLE.

Similarly, spiking backpropagation closes the accuracy gap between ANNs and SNNs for spatio-temporal pattern recognition tasks. The benefit of SNNs in terms of power efficiency and low latency computations are considerably retained in comparison to existing related approaches. This underlines the current trend of computational neuroscientists moving from SNNs to generic computational frameworks such as PyTorch. This effectiveness also questions whether backpropagation is used in the brain and how, as suggested recently in Lillicrap et al. [176].

Model	**Accuracy**	**#Iterations**
DECOLLE (conv)	$95.60 \pm 0.56\%$	Online .16M
DECOLLE (conv+attention)	$\mathbf{96.37 \pm 0.51\%}$	**Online .16M**
cRBP (conv)	$92.48 \pm 0.89\%$	Online .16M
cRBP (conv+attention)	$95.34 \pm 0.78\%$	Online .16M
cRBP (2L dense)	$77.93 \pm 2.09\%$	Online .16M
cRBP (2L dense+attention)	$90.80 \pm 1.16\%$	Online .16M
eRBP (64×64 dense)	86.1%	Online .11M
eRBP (64×64 dense+attention)	92.7%	Online .11M
Histogram	45.83%	Offline (Closed-form)
LSM	42.78%	Offline (Closed-form)
HMAX-STDP-SVM	37.50%	Online $<$ .001M
SLAYER (conv) [265]	93.64 ± 0.49 %	Offline .27M
IBM EEDN (conv) [43]	$91.77\%(94.59\%)$	Offline 64M

Table 5.2.: Accuracy of the rules on the DvsGesture dataset. The number of iterations refers to the number of training samples that were fed to the network. Top-down gradient descent rules outperform bottom-up rules. EEDN increases its accuracy with output filtering.

6. Closing The Loop: Visuomotor Coupling

In the previous Chapter, plasticity rules to learn high-level representations from event-based data with SNNs have been introduced and discussed. In this Chapter, the loop is closed by defining SNN-based visuomotor coupling methods. Visuomotor coupling can be seen as a functional mapping of learned high-level visual representations to motor commands. Three methods are discussed in the following Chapter to derive such SNN-based mapping: manually, using a reward, and using predictions.

6.1. Manual Visuomotor Coupling

The material to be presented in this Section was partly already published by the author in Kaiser et al. [6]. In the initially introduced lane following experiment (Section 4.3.1), the weights from the input neurons to the motor neurons were manually composed to implement the desired behavior. A similar SNN-based approach to control a robot from learned visual representations is introduced in this Section. First, it is shown how visual representations relevant to reaching and grasping can be learned in a supervised manner with eRBP. Second, a predefined robot grasping trajectory in joint space is manually associated with every learned grasp object class. This method can only be applied when the visual representation space is low-dimensional and temporal dynamics are simple. Nonetheless, the simplicity of this approach makes it suitable and proposing in robotics.

6.1.1. Method

A SNN is trained in a supervised manner with eRBP to classify four object classes from event-based data: ball, bottle, pen and background. The visual event stream is emitted by a DVS mounted on a neuromorphic head (described in Section 4.3.2) performing microsaccades. In this case, the SNN maps from the event-based data of dimension $n_{\text{row}} \times n_{\text{col}} \times 2$ to the four output neurons, corresponding to the four output classes, at all time. For each class, a predefined reaching and grasping motion to be taught by the user is associated. This mapping can be seen as a SNN-based visuomotor coupling between high-level learned visual representations and high-level motor trajectories. This definition allows to train the object

classification network just with the objects and without the robotic arm, as motor control is not learned. At test time, the detected class triggers the respective reaching and grasping motion. This method is depicted in Figure 6.2.

The main advantage of defining the visuomotor coupling by hand is that the data can be recorded and the SNN can be trained without the robot. This advantage is very practical since the learning process necessarily explores sub-optimal configurations of the SNN. If the SNN was directly controlling the robot, evaluating the sub-optimal configurations in the real-world would be unsafe for the robot as well as its user. However, manually defining the visuomotor coupling has severe disadvantages. Mainly, such coupling is only feasible and may only be useful for low-dimensional representations with simple temporal dynamics. For instance, accounting for the action's impact on the environment and estimating the next sensor readings is non-trivial. It is also cumbersome to define such coupling for a SNN trained differently than with supervised learning, as the emerged representations can be arbitrary.

6.1.2. Experimental Setup

For a demonstration of a visuomotor coupling designed manually, a real robot sequence of reaching and grasping experiments has been performed. A SNN was trained with eRBP to categorize four static objects perceived with an event-based vision sensor performing microsaccades: ball, bottle, pen and background, see Figure 6.2. During training, objects of particular classes are placed on a table at a specific position. The robotic head performs microsaccadic eye movements to extract visual information from static objects, similar to the N-Caltech101 dataset, see Section 4.2.1. The event streams are recorded together with the corresponding object affordances. To each object class, a predefined reaching and grasping motion has been assigned. During testing, microsaccades are performed and the detected object affordances trigger the adequate predefined reaching and grasping motions of a Schunk LWA4P arm equipped with a five-finger Schunk SVH gripper.

The same type of microsaccadic eye movements was used for training and testing. The particular type of microsaccadic movement was not an important parameter in the experiment. Indeed, the SNN was capable of correctly classifying the objects before termination of the movement, as will be seen in the next Section. In this experiment, the microsaccadic motion consisted of an isosceles triangle in joint space, see Figure 6.1. The motion has three phases. A negative tilt of α and negative pan of $\alpha/2$, followed by a tilt of α and negative pan of $\alpha/2$, followed by a final pan of α moving the DVS back to its initial position. The angle $\alpha = 1.833°$ is much larger than in humans, to compensate for the size of the DVS pixels, see Section 2.1.4. Each motion is effectuated in $0.2\,\mathrm{s}$.

The input stream is cropped to the dimension 32×32 pixels and is fixed to match the position of the objects on the table, see Figure 6.3. This demonstrator was

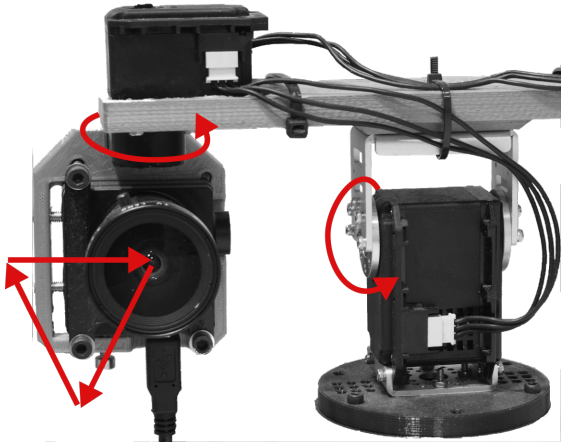


Figure 6.1.: Microsaccadic motion of the DVS performed by the robotic head. Source: [6].

implemented with the ROS Framework [241] and the ROS-DVS driver introduced in [207].

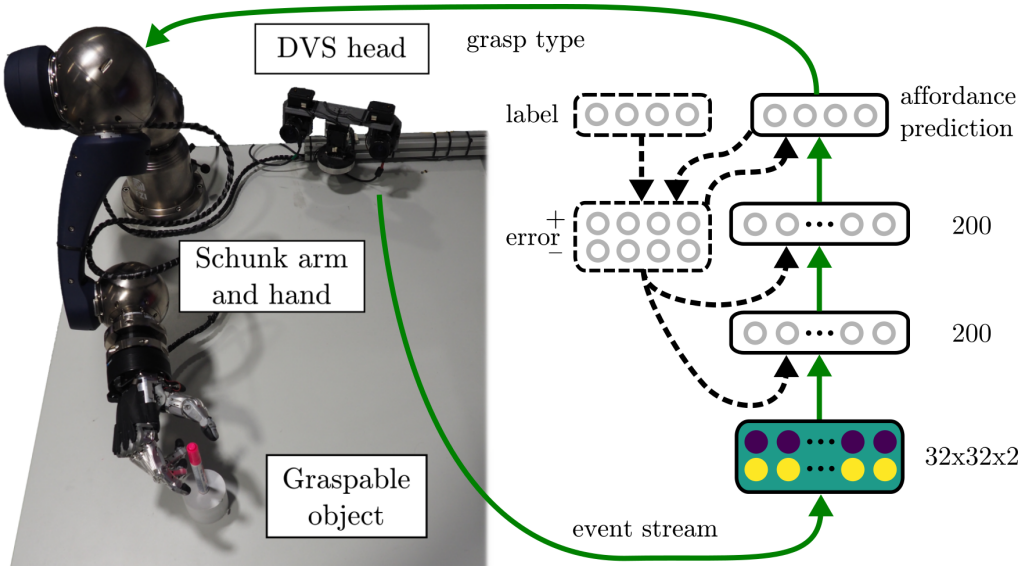


Figure 6.2.: Real-world robotic setup embodying the synaptic learning rule eRBP [213]. The DVS is mounted on a robotic head performing microsaccadic eye movements. The SNN is trained with eRBP (Section 5.4) to classify four types of affordances: ball-grasp, bottle-grasp, pen-grasp or do nothing. Reaching and grasping trajectories executed by a Schunk LWA4P arm equipped with a five-finger Schunk SVH gripper are manually assigned to the corresponding four classes. Source: [6].

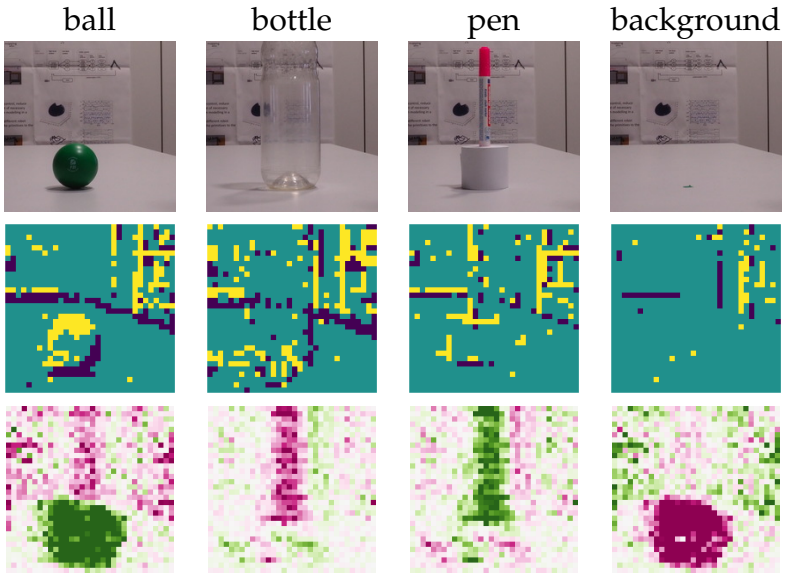


Figure 6.3.: Example samples and learned weights for the grasp-type recognition experiment. Top row: camera image of the objects. Middle row: integration of the address events during 15ms after microsaccade onset. Bottom row: projection of the synaptic weights of the 4-layers network for each label neuron onto the input after training. Green denotes excitation (positive influence) and pink denotes inhibition (negative influence). Source: [6].

6.1.3. Results

The learning performance of eRBP has already been demonstrated in Section 5.4 on the DvsGesture Dataset. It has been shown that eRBP is also capable to learn from few samples and static scenes perceived through microsaccades.

The learning accuracy of the four object affordances has been evaluated after 30 epochs on 20 samples per class. The spike-trains and the classification results can be seen in Figure 6.4. The network is confident when classifying balls and pens but less confident between bottles and background. This is because the transparent bottles are visually similar to the background, see Figure 6.3. This uncertainty can be reduced with more samples and more training iterations. Correct classification is obtained by counting the spikes of the output neurons for a period of about $100\,\mathrm{ms}$ after microsaccade onset, long before the termination of the microsaccadic motion.

Despite the small number of training samples, the network is also capable of moderate generalization to other objects of the same shape. This was demonstrated by using different balls for training and testing. Since the DVS does not sense color, the network can only rely on shape information for its prediction, which is desired for affordances. Additionally, slightly moving the objects does not disrupt the classification as much as modifying the background. This is due

to the background being learned as an additional class for the "do nothing" affordance.

The detected affordances trigger the corresponding taught reaching and grasping motions. A complementary video[1] shows the dataset collection and the performed trajectories. Since the objects are always placed at the same location on the table, the reaching and grasping motions do not fail. Incidentally, this method can not be applied to random object locations on the table.

6.1.4. Conclusion

The simplest method to control a robot from learned visual representations consists of manually defining what the robot should do when a particular class is recognized. An important advantage of this method is that the robot is not required to be in the loop during training. This method is suitable for low-dimensional representation spaces, simple temporal dynamics and simple tasks. For instance, the method introduced in this Section allows the grasping of objects located at a precise location on the table. Significant modifications would be required to enable the grasping of objects anywhere on the table. This could be achieved using a higher-lever control interface, such as the motion primitives introduced in Tieck et al. [24], demonstrated on a target reaching task in [25, 26].

Finally, this method also prevails open-loop control over closed-loop control since accounting for the action's impact on the environment and estimating the next sensor readings is non-trivial. These limitations are discussed in the next Section with reward-based learning.

[1] https://neurorobotics-files.net/index.php/s/sBQzWFrBPoH9Dx7

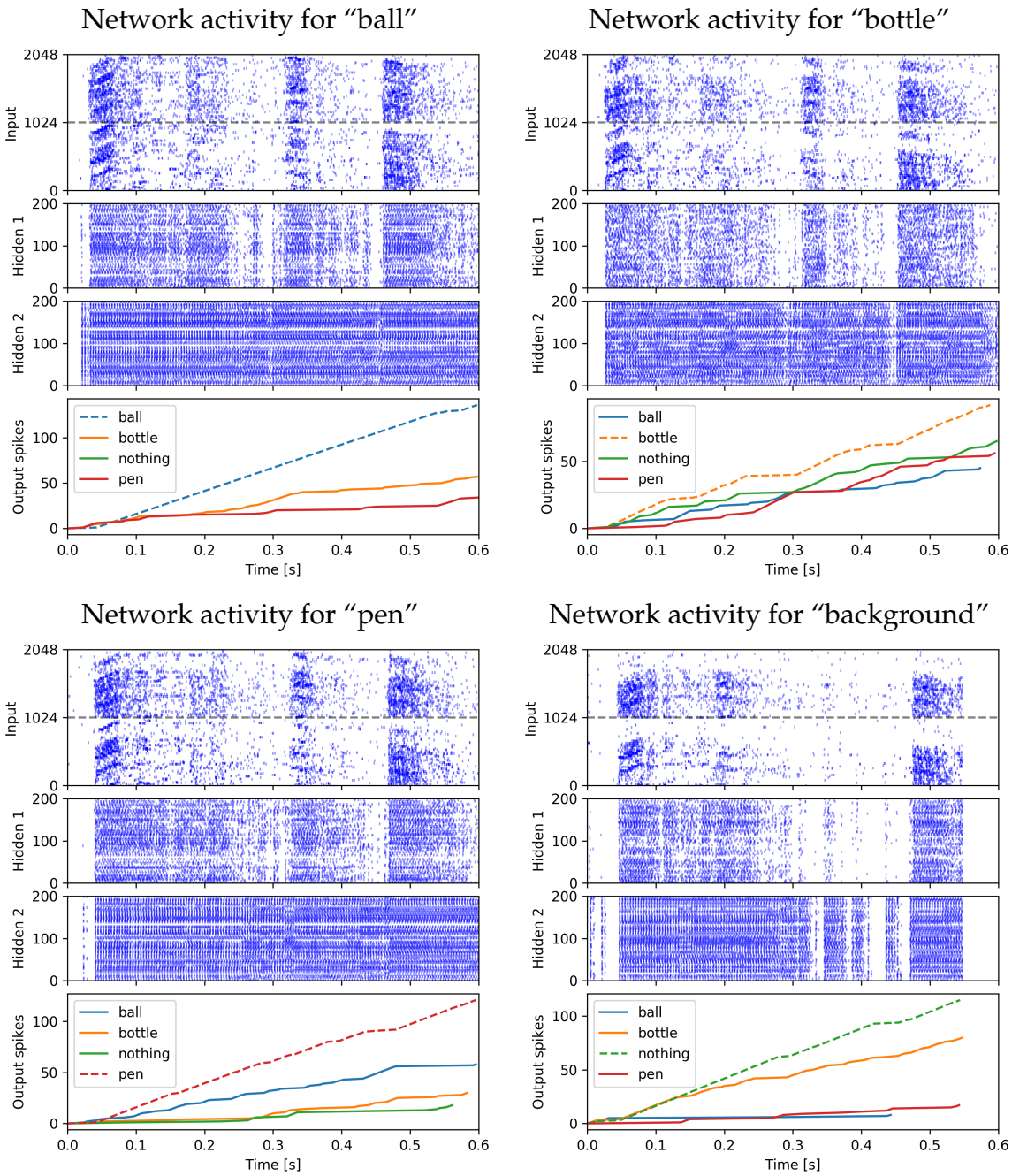


Figure 6.4.: spike-trains and classification results for four test samples of the grasp-type dataset. The network manages to correctly classify the four test samples. The ball and the pen are easily classified despite the small amount of training data. However, the transparent bottle generates few events, yielding to more uncertainty in the classification with the background. The three phases of the microsaccadic motion are visible in the input spike-trains (first row of each plot). However, the activity of the hidden layers does not drop instantaneously even when the input is sparse, indicating a form of short-term memory induced by the neural dynamics. The neurons in the hidden layers spike close to their maximum frequency, as limited by the refractory period. Source: [6].

6.2. Learning Visuomotor Coupling with a Reward

The material covered in this Section was originally published by the author in [8]. Reinforcement learning has become the standard framework for learning visuomotor couplings. The goal of reinforcement learning is to learn a policy that maximizes the expected reward. The reward function is generally defined by the user with respect to the state of the environment or the action of the robot. The manual involvement to provide a well-behaved reward function guiding the robot to a performing policy is referred to as reward shaping [155].

Few synaptic learning rules have been proposed for reinforcement learning tasks, see Section 3.1.1. Some approaches, such as Synaptic Plasticity with Online Reinforcement learning [150] (SPORE), are inherently derived for reinforcement learning tasks. In conventional deep learning, most approaches rely on supervised gradient backpropagation, with a loss function formulated with respect to the reward. With the recent development of spiking backpropagation rules (Sections 5.4 and 5.5), these approaches could also be ported to SNNs. In this Section, the synaptic sampling rule SPORE discussed in Section 3.1.2 is evaluated with the help of two visuomotor scenarios. The main contribution of this Section is the elaboration of a dedicated closed-loop framework allowing reward-based learning rules such as SPORE to be evaluated in a robotic visuomotor context.

6.2.1. Method

Usually, synaptic learning rules are solely evaluated on open-loop pattern classification tasks. To evaluate SPORE in a closed-loop visuomotor task with event-based data, a framework was implemented by connecting many open-source components together. This framework evaluating the performance of biologically plausible plasticity models in closed-loop robotics environments. This framework has been applied to evaluate the synaptic sampling rule SPORE, as depicted in Figure 6.5.

This framework has been implemented for evaluating spiking network learning rules grounded in real-world embodiment experiments. Visual sensory input is sensed, encoded as spikes, processed by the network, and output spikes are converted into motor commands. The motor commands are executed by the agent, which interacts with the environment. This modification of the environment is continuously observed and sensed by the agent. Additionally, a continuous reward signal is emitted from the environment. SPORE tries to maximize this reward signal online by steering the ongoing synaptic plasticity processes of the network towards configurations which are expected to yield more overall reward. Unlike classical reinforcement learning setups, the spiking network is treated as a dynamical system continuously interacting with the environment. This allows to report learning progress with respect to biological time, unlike classical reinforcement learning which reports learning progress in number of iterations. Similarly,

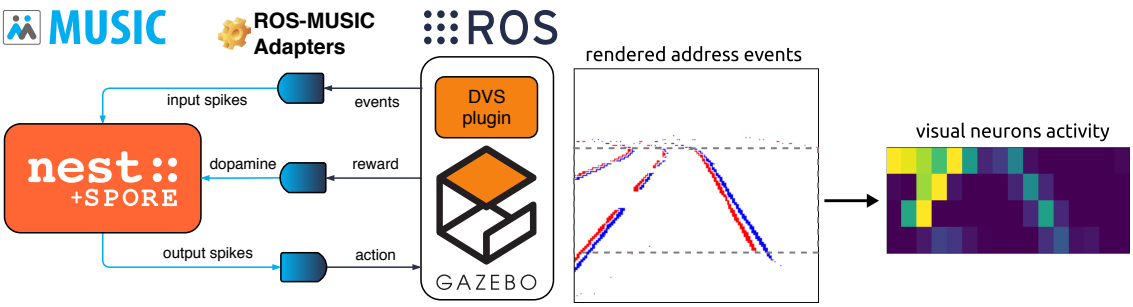


Figure 6.5.: Closed-loop framework to evaluate SPORE. Left: our asynchronous framework based on the open-source software components NEST neural simulator, SPORE, MUSIC, ROS, ROS-MUSIC tool-chain, Gazebo and DVS plugin. Right: Encoding visual information to spikes for the lane following experiment, see Section 4.3.1 for more information. Address events (red and blue pixels on the rendered image) are down-scaled and fed to visual neurons as spikes. Source: [8].

new episodes are not delimited by resetting the environment. Instead, the agent learns continuously and without interruption, which significantly complicates the learning process.

This framework is configured by many open-source software components: The neural simulation relies on NEST, introduced in Gewaltig et al. [119], combined with the open-source implementation of SPORE by Kappel et al. [150][2]. The robotic simulation is performed by Gazebo [156] and ROS [241], and visual perception has been implemented with the help of the open-source DVS plugin for Gazebo, see Section 4.2.2. The robotic simulator and the neural network run in different processes. MUSIC, introduced by Djurfeldt et al. [91] and Ekeberg et al. [95], is used to communicate and transform the spikes and the ROS-MUSIC tool-chain by Weidel et al. [285] is used to bridge between the two communication frameworks. The latter also synchronizes ROS time with neural simulator time. This work contributed to the Gazebo DVS plugin by integrating it to ROS-MUSIC, and to the SPORE module by integrating it with MUSIC. These contributions enable the of design new ROS-MUSIC experiments using event-based vision to evaluate SPORE or other biologically-plausible learning rules.

6.2.2. Experimental Setup

In this Section the evaluation of SPORE is presented on the lane following task introduced in Section 4.3.1. Steering commands are decoded from output spikes

[2] https://github.com/IGITUGraz/spore-nest-module

as a ratio between the following linear decoders:

$$\begin{aligned} a_L &= \sum_{i=1}^{N/2} a_i, \\ a_R &= \sum_{i=N/2}^{N} a_i, \\ r &= \frac{a_L - a_R}{a_L + a_R}. \end{aligned} \tag{6.1}$$

With N the number of decoder neurons, a_L and a_R the relative coefficients for steering left and right respectively and r the computed steering ratio. The first $N/2$ neurons pull the steering on the left side, while the remaining $N/2$ neurons steer to the right side. With $N = 8$, there are 4 left motor neurons and 4 right motor neurons. The steering command is obtained by discretizing the ratio r into five possible commands: hard left (-30°), left (-15°), straight (0°), right (15°) and hard right (30°). The decision boundaries between these steering angles are $r = \{-10, -2.5, 2.5, 10\}$ respectively. This discretization is similar than the one used in Wolf et al. [288]. It resulted with better performance than directly applying r (multiplied with a scaling constant k) as a continuous-space steering command, see Section 4.3.1.

The reward signal transmitted to the vehicle is equivalent to the performance metrics used in Section 4.3.1 to evaluate the policy. The reward depends on two terms – the angular error β_{err} and the distance error d_{err}. The angular error β_{err} is the absolute value of the angle between the right lane and the vehicle. The distance error d_{err} is the distance between the vehicle and the center of the right lane. The reward $r(t)$ is computed with:

$$r(t) = e^{-0.03\ \beta_{err}^2} \times e^{-70\ d_{err}^2}. \tag{6.2}$$

The constants are chosen so that the score is halved every 0.1m distance error or 5°angular error. Note that this reward function is comprised between $[0, 1]$ and is less informative than the error used in Bing et al. [62]. In this case, the same reward is transmitted to all synapses, and a particular reward value does not indicate whether the vehicle is on the left or the right of the lane. The decay of the learning rate is $\lambda = 8.5 \times 10^{-5}$.

A reset of the position and orientation of the agent is initiated only in the case when it goes off-track in the lane following task. Finite-time episodes are not enforced and neither the agent nor SPORE are notified of the reset.

6.2.3. Results

The achieved results show that SPORE is capable of learning policies online for moderately difficult embodied tasks within some simulated hours. In the experiments, regulation of the learning rate β played an important role in retaining

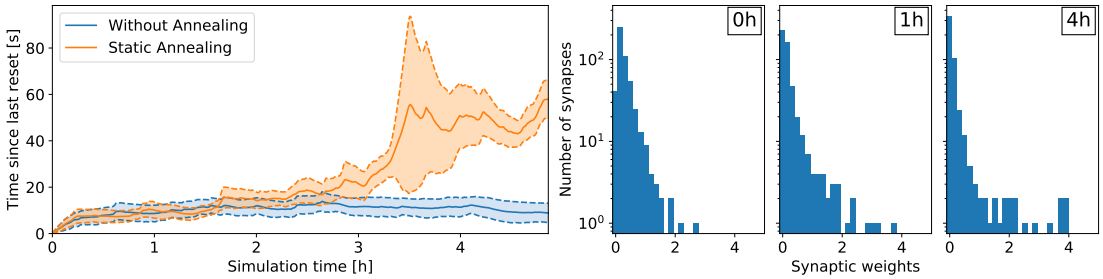


Figure 6.6.: Results of SPORE with and without annealing the learning rate β on the lane following task. Left: performance averaged over 6 trials. Right: Development of the synaptic weights over the course of learning with annealing. Source: [8].

policy improvements. Specifically, when the learning rate β remained constant over the course of learning, the policy did not improve compared to random, see Figure 6.6. In this case, the vehicle remains about $10\,\mathrm{s}$ on the right lane until triggering a reset. On the other hand, exponentially decaying the learning rate β over time enables the policy to retain improvements. After $3\,\mathrm{h}$ of learning, β decreased to 40% of its initial value and the policy starts to improve. After $5\,\mathrm{h}$ of learning, β approaches 20% of its initial value and the performance improvements are retained. Indeed, while the weights are not frozen, the amplitude of subsequent synaptic updates are drastically reduced. In this case, the policy is significantly better than random and the vehicle remains on the right lane about $60\,\mathrm{s}$ on average.

6.2.4. Conclusion

SPORE represents one of the few synaptic learning rules that have been proposed for reinforcement learning tasks. It has been shown that SPORE is capable of learning shallow feedforward policies online for moderately difficult embodied tasks within some simulated hours. On a functional scale, deep reinforcement learning methods still outperform biologically plausible learning rules such as SPORE. This performance gap should be addressed in future work by taking inspiration from deep learning. Indeed, synaptic learning rules based on backpropagation could be reformulated in a reinforcement learning context, as discussed in previous Chapters.

6.3. Learning Visuomotor Coupling with Prediction Error

The material covered in this Section was originally published by the author in Kaiser et al. [11]. In the previous Section, a reward-learning plasticity rule has

been evaluated on a visuomotor task. This is the standard approach applied to learn policies. This Section presents a different approach by learning a visuomotor coupling from visual prediction.

In [107], Karl Friston introduced the free-energy principle, a unified brain theory accounting for perception and behavior. The theory is based on the observation that biological systems must avoid surprises to ensure that state representation stays within the physiological realm. Avoiding surprise is achieved by minimizing free-energy, which is a function of sensory input and the recognition density. The recognition density is an approximate probability distribution of the causes of the sensory input, as encoded by the agent's internal state. In order to decrease free energy, an agent has two possibilities: change its sensory input by acting on the world, and change its recognition density by updating its internal model. These two possibilities point to action and perception modules respectively. In other words, the free-energy principle implies that an agent selects the actions that minimize the prediction error of its internal representation of the world.

6.3.1. Method

Inspired by the free-energy principle, a method is proposed for learning the reproduction of observed movements from a visual prediction model. Specifically, a LSM is trained to provide short-term visual prediction from event streams as described in Section 5.1. This LSM is trained with a limited set of movements it can predict, with the procedure outlined in Kaiser et al. [14] and reported in Section 5.1. In the second phase, an adequate action is searched by minimizing the prediction error with the help of an optimization process. Note, this technique only reflects one side of the free-energy principle: selection of actions that minimize prediction errors. The other side consists of adapting the predictions to new sensory input, which is out of the scope of this thesis.

The method is depicted in Figure 6.7. The action consists of a vector of goal joint positions. The robot always starts from the same initial position. The robot performs a movement by controlling its joints from the initial positions to the goal positions. No trajectory planner nor a robot configuration space transformation is used. By training the prediction model with a first-person view of arm movements, the robot learns motions visually similar to the demonstrated one.

First, a visual predictive model is learned from a demonstration with a LSM, see Figure 5.1. A single demonstration is sufficient, but it should be provided in first-person view. Second, the robot tries different movements iteratively in an attempt to minimize the prediction error. The prediction eror is computed at every time with the negative Pearson correlation:

$$\rho(t) = -\frac{\mathrm{cov}(\boldsymbol{p}(t), \boldsymbol{b}(t))}{\sqrt{\mathrm{var}(\boldsymbol{p}(t))\mathrm{var}(\boldsymbol{b}(t))}}, \tag{6.3}$$

with $\boldsymbol{p}(t)$ and $\boldsymbol{b}(t)$ the predicted and perceived visual stream respectively, see Section 5.1.1. cov and var denote the covariance and variance. The prediction error of a time sequence is the averaged correlation error:

$$e = \tanh\left(\frac{1}{n_{samples}^{test}} \cdot \sum_{t \in \boldsymbol{t}^{\boldsymbol{test}}} \operatorname{arctanh}\left(\rho(t)\right)\right). \tag{6.4}$$

The correlation coefficients are tansformed with Fisher-z before summation, then transformed back. The choice for the negative Pearson correlation for the optimization process instead of a mean square error is motivated in Kaiser et al. [11]. Mainly, a mean square error is more sensitive in the case when few events are generated by the movements. In this case, the prediction as well as the perceived event stream consists of few events leading to a small mean square error. This is demonstrated in Kaiser et al. [11] with an additional experiment. The negative Pearson correlation is less sensitive to this problem. The optimization of the movement to minimize the prediction error is performed with CMA-ES, introduced in Hansen et al. [127].

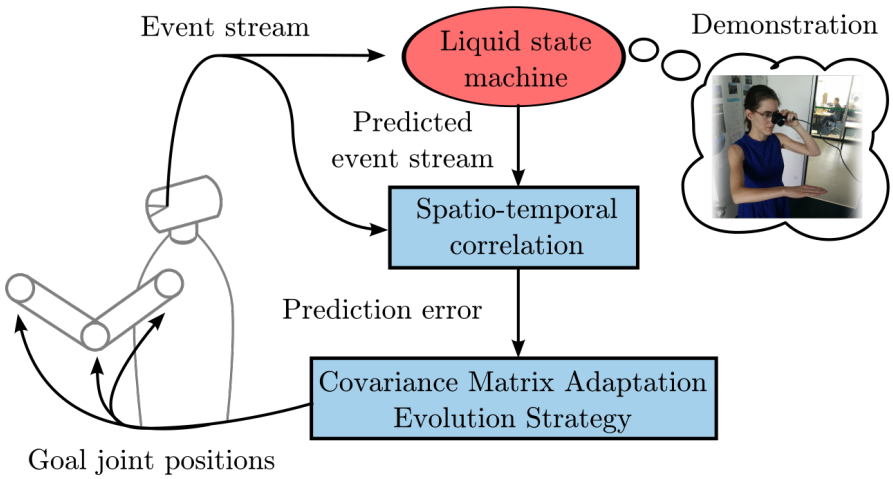


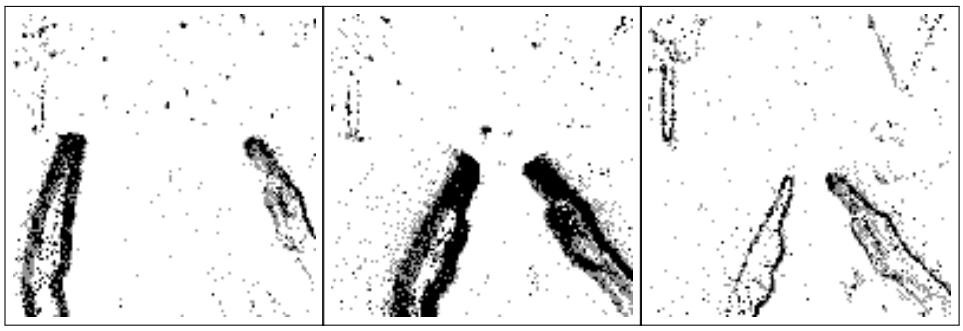
Figure 6.7.: Schema of the method to learn movements by imitation from event-based visual prediction. Copyright ©2018, IEEE [11].

6.3.2. Experimental Setup

In the following, the ability of the proposed network to learn movements from visual prediction errors is evaluated with the help of the NRP. In the first phase, a visual prediction model is learned with a LSM as presented in Section 5.1. In a second phase, adequate control commands for the robot that minimize the visual prediction error are searched. This describes an imitation learning setup, which

allows the robot to learn and to execute a movement visually similar to what it previously perceived.

Two experiments are performed. In the first experiment, a simulated iCub robot learns to move its arm as demonstrated by a human teacher recorded with a real DVS, see Figure 6.8. The human demonstrator has a real DVS positioned on his front side. The movement consists of positioning the two arms closer together.



(a) (b) (c)

Figure 6.8.: Visualization of the demonstration for the first experiment. The images are rendered by aggregating DVS address events at the beginning (a), during (b) and at the end (c) of the demonstration. Copyright ©2018, IEEE [11].

In the second experiment, a simulated robot (Schunk arm LWA 4P) learns to move with respect to a visual cue (a flashing ball) present during demonstration, see Figure 6.9. The demonstration consists of moving the arm left when a ball is flashing on the left and moving right when a ball is flashing on the right. The arm returns to a straight upright position when the ball does not flash. The experiments have been implemented in the NRP using the Virtual Coach. The Virtual Coach is a software library enabling experiments to be started, paused, stopped and configured programmatically, allowing to model such optimization processes.

Learning the movement from the trained predictive LSM consists of the following steps. A robot is initialized with similar initial joint positions as in the demonstration. The initial solution μ of CMA-ES is set to the initial joint positions, and the initial variance σ to 0.5 radians per joint. Therefore, the first movements generated by the optimization process will be small. For each iteration, 15 offspring are generated for the iCub and 10 for the Schunk arm LWA 4P. More offspring for the iCub are recommended since the movement consists of 12 joints, and only 3 for the Schunk arm. An offspring is a vector of goal joint positions. For each offspring, a movement is executed, going from the initial joint positions to the goal joint positions. During the execution, the perceived event stream is recorded. The recorded event stream is fed to the LSM to obtain a prediction. This prediction is compared with the recorded event stream with the negative Pearson correlation. The resulting prediction error is assigned to the fitness value for this offspring.

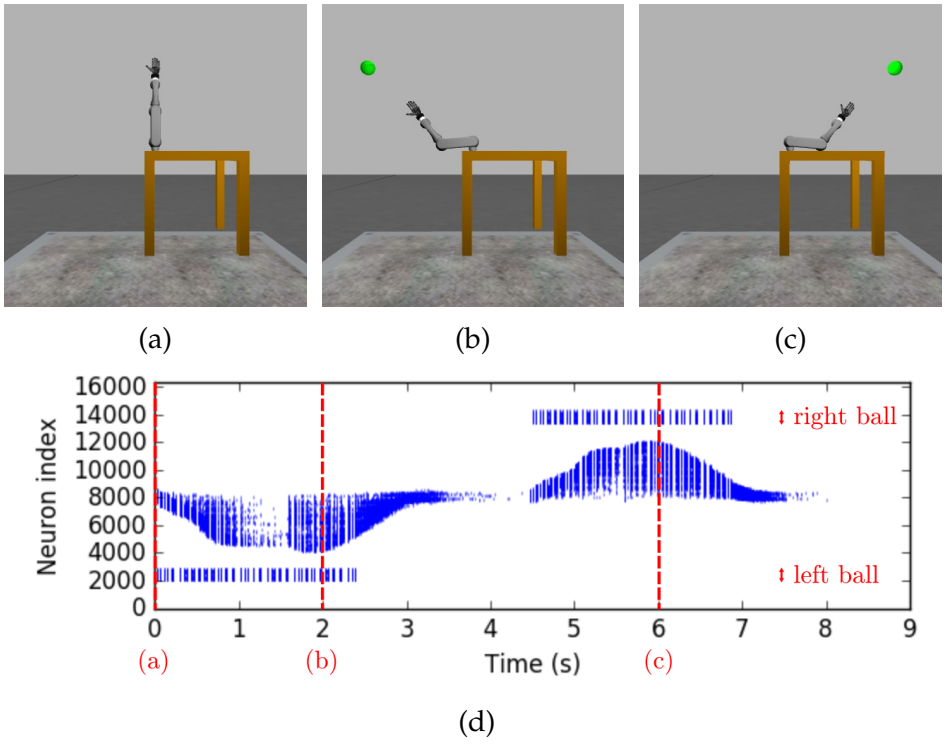


Figure 6.9.: Visualization of the demonstration for the second experiment. First row: third-person view of the demonstrated motion at 0 s (a), 2 s (b) and 6 s (c). (d): Input spike-train of the demonstration. Copyright ©2018, IEEE [11].

At the end of the execution, the robot goes back to its initial joint positions until another offspring is generated. The learning process is terminated after a given number of iterations.

6.3.3. Results

For the first experiment, the simulated iCub manages to learn a similar movement to the one demonstrated by the human, see Figure 6.10c and Figure 6.8. The optimization process with CMA-ES provides a steady decrease in error with respect to the learning iterations (Figure 6.11). Despite that the joints of the two arms are treated independently in the optimization process, the learned movement is symmetric as shown in the demonstration, see Figure 6.8. This has been also shown in the learned joint goal positions, which are similar for left and right arm, except for the elbow and the shoulder roll, see Figure 6.11.

Additionally, since the demonstration was provided by a human with a real DVS, it shows that the visual prediction model learned with the real DVS can also be used for the simulated DVS, despite the inaccurate simulation discussed in Section 4.2.2.

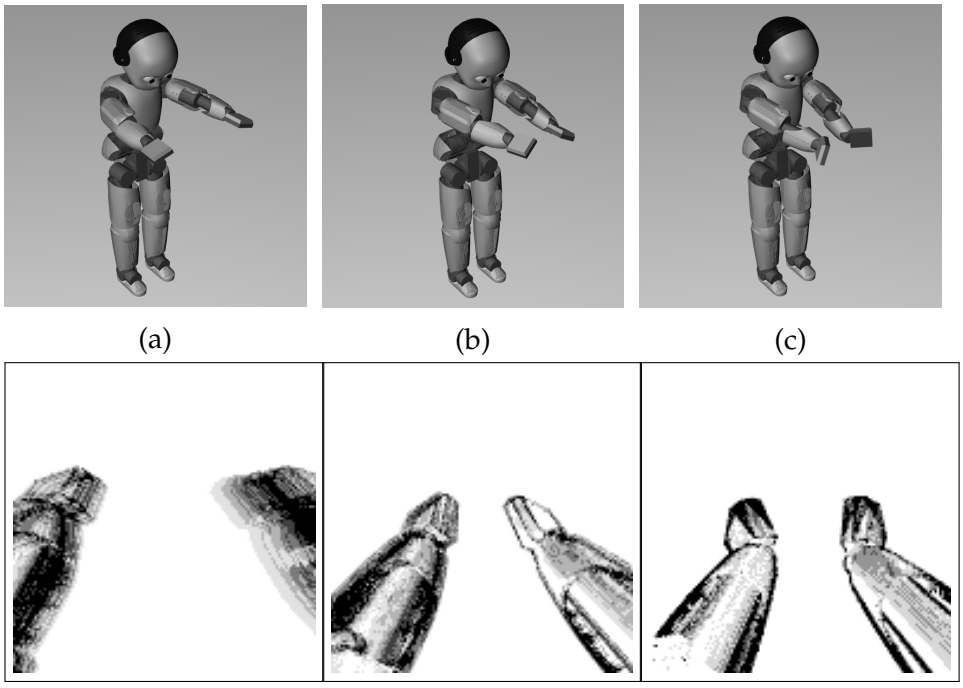


Figure 6.10.: Learned movement for the first experiment. First row: third-person view (visualization purpose only, not used for training). second row: aggregated DVS address events. Images rendered at the beginning (a,d), during (b,e) and at the end (c,f) of the learned motion. Copyright ©2018, IEEE [11].

The second experiment shows that multiple movements can be learned from a single demonstration depending on the visual cues. This experiment is repeated two times: the first time, the ball flashes on the left during the execution of each offspring, the second time it flashes on the right.

The optimization process recovers similar movements to the demonstration depending on the visual cue, see Figure 6.9. This shows that the LSM managed to associate the visual cue with the arm movement performed at the same time during the demonstration, see Figure 6.9. Conversely, if the arm moves to one side, the LSM predicts the ball flashing on this side: the proposed method does not distinguish what it can control from what it can not. The visual cue therefore needs to be sufficiently strong to prevent exploratory movements from predicting other cues. In this case, the flashing ball is a strong visual cue for an event-based sensor since both flash-in and flash-out generate events. In both runs, the error metric decreases, meaning that the robot has learned the movement with confidence, see Figure 6.13.

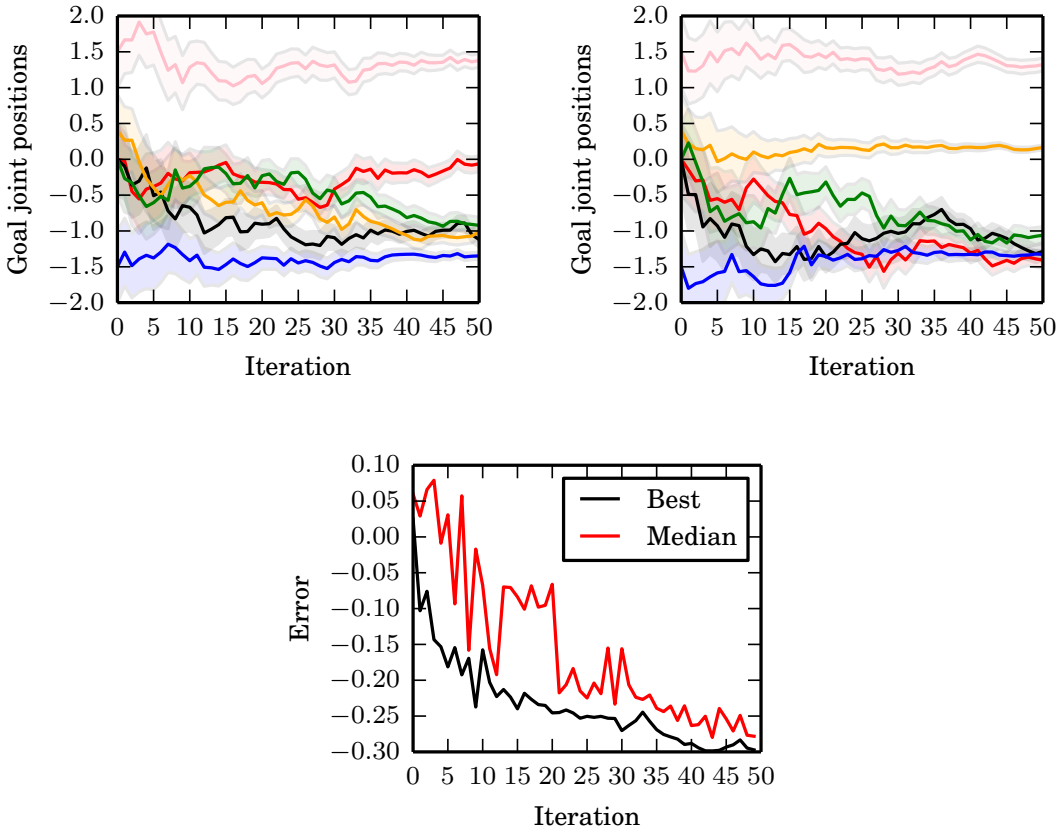


Figure 6.11.: Learning process for the first experiment. Means and standard deviations of goal joint positions for the six joints of the left and right arms (top), and decrease of the negative Pearson correlation (bottom). Copyright ©2018, IEEE [11].

6.3.4. Conclusion

The free-energy principle is based on the assumption that the goal of the brain is to minimize its surprise by adapting its model of the world and taking predictable actions. This framework was decoupled, with a focus on action generation from a prediction model. Specifically, the ability of a robot to learn a movement by minimizing its visual prediction error was investigated. Despite the simplicity of the presented method, the preliminary results suggest that a robot can learn new movements from visual demonstrations in less than 1000 trials.

As reinforcement learning, the free-energy principle is a proposing approach for future research to learn policies. The proposed method could be enhanced with different representations for motions, such as motion primitives introduced in Tieck et al. [24], allowing more complex movements to be performed. In addition, an agent learning online would require the perceptive part of the free-energy principle to be integrated. To be more specific, the predictive model (here the LSM) should learn online from experience instead of an initial demonstration.

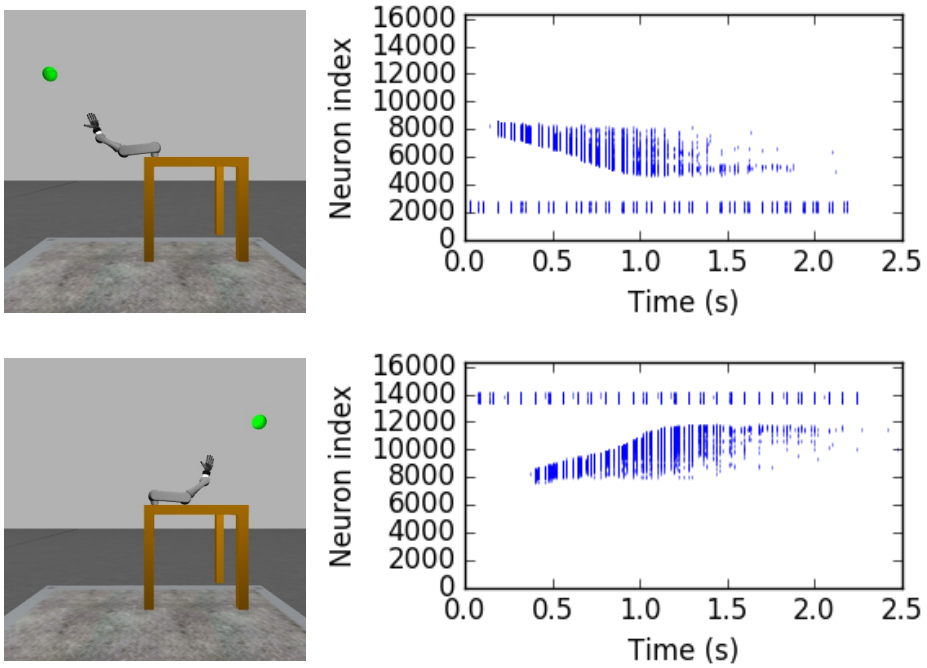


Figure 6.12.: Learned movements for the second experiment. The movements learned by the arm depend on the visual cue and are visually similar to the demonstrated ones. The input spike-trains of the learned movements are similar to the two distinct portions of the demonstration, see Figure 6.9. Copyright ©2018, IEEE [11].

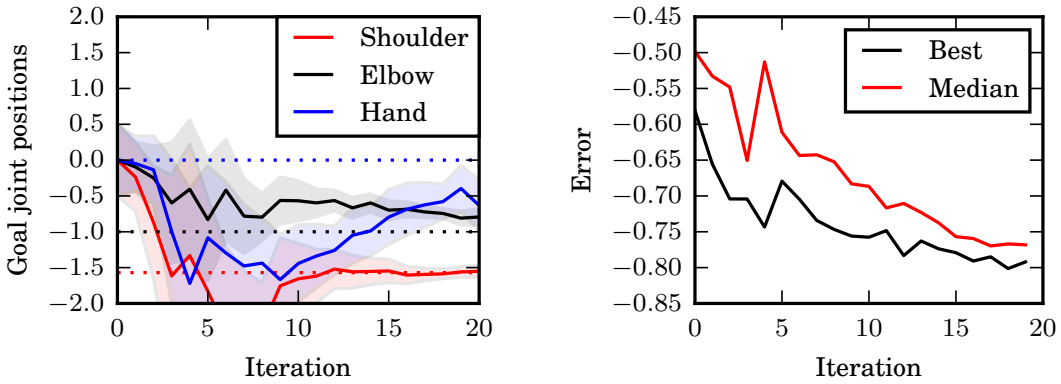


Figure 6.13.: Learning process for the second experiment in case of a right visual cue. Means and standard deviations of goal joint positions for shoulder, elbow and hand joints (left), and decrease of the negative Pearson correlation (right). Ground truth is shown in dashed lines. Copyright ©2018, IEEE [11].

6.4. Summary and Conclusion

In this Chapter, it has been shown how to close the visuomotor loop with the visual representations learned from event-based data. Three methods were introduced: manual coupling and reward coupling, which are common in robotics and machine learning, and the proposing prediction error coupling.

Manual coupling is the simplest and most restricted technique since it consists of manually assigning motor commands to the learned visual representations. With this technique, it is possible to trigger predefined reaching and grasping motions in a real robot setup corresponding to classified visual affordances. On the other hand, reward-maximization can learn such motor mappings and was discussed and implemented with the synaptic plasticity rule SPORE. SPORE was able to learn performing policies in shallow networks for moderately difficult tasks. Lastly, prediction error minimization can learn such mapping by assuming that actions are selected to minimize surprise. Following this approach, it was possible to show how to teach robots new movements from visual demonstrations.

Unlike supervised learning tasks, the performance gap between SNNs and ANNs is still large for motor learning tasks. The aspect of learning motor policies with SNNs should therefore be investigated further. Following deep reinforcement learning, a promising direction of research would be to apply spiking backpropagation rules such as DECOLLE to reward-maximization problems.

6.4.1. Synaptic Gradients for Learning Policies

In Chapter 5, it was concluded that top-down synaptic plasticity rules derived from theoretical considerations outperform bottom-up rules derived solely from biological observations. Such rules implement efficient approximations of backpropagation (and BPTT), adapted to continuous neural dynamics, enabling online weight updates. While originally considered as a supervised learning method, backpropagation was adapted to reinforcement learning problems in Mnih et al. [200], initiating the field of deep reinforcement learning. In this Section, the adaptation of top-down synaptic learning rules (such as DECOLLE) to deep reinforcement learning tasks is briefly discussed, along with the related challenges.

Reinforcement Learning Objective

Reinforcement learning tasks are formalized as (partially observable) Markov decision processes (S, A, T, R), with S the state space, A the action space, $T : S \times A \times S \rightarrow \mathbb{R}$ the transition probabilities from one state to another upon an action, and $R : S \times A \rightarrow \mathbb{R}$ a reward function. The objective of reinforcement

learning is to parameterize a policy $\pi_\theta : S \to A$ maximizing the expected return of a trajectory τ:

$$\begin{aligned} p_\theta(\tau) &= p_\theta(s_1, a_1, ..., s_T, a_T) = p(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) T(s_{t+1}|s_t, a_t) \\ \theta^\star &= \text{argmax}_\theta \, E_{\tau \backsim p_\theta(\tau)} \left[\sum_t \gamma^t R(s_t, a_t) \right], \end{aligned} \tag{6.5}$$

with $\theta^\star$ denoting the optimal parameters, $E_{\tau \backsim p_\theta(\tau)}$ the expectation over the trajectories τ distributed by p_θ and $0 < \gamma \leq 1$ a discount factor.

There are two family of approaches to formulate gradients in order to solve this optimization problem: actor-critic and Q-learning. Actor-critic approaches learn two functions simultaneously: the policy π_θ and a critic $V_{\theta'} : S \to \mathbb{R}$, associating a value (expected return) to every state. Q-learning approaches learn a single function $Q : S \times A \to \mathbb{R}$, associating a value (expected return) to every state-action pairs. The optimal policy $\pi^\star$ is then formulated with respect to the optimal state-action value function $Q^\star$ as $\pi^\star(s) = \text{argmax}_a \, Q^\star(s, a)$.

New algorithms in both families are regularly derived [259, 199, 175, 258, 200], a thorough description is beyond the scope of this thesis. These algorithms differ from one another in the formulation of the gradients and in the collection and usage of the sensorimotor data. From a theoretical standpoint, DECOLLE can be used in-place of backpropagation in the aforementioned deep reinforcement learning methods. In practice, some challenges related to the nature of synaptic gradient computation need to be overcome.

Reinforcement Learning with Spikes

The first challenge is a mismatch between the continuous dynamics of SNNs and the standard, discrete formulation of Markov decision processes. In practice, SNNs are simulated at around $1000\,\text{Hz}$ (resolution of $1\,\text{ms}$). On the other hand, standard reinforcement learning tasks such as Atari games from OpenAI Gym [68] are simulated at around $20\,\text{Hz}$. Therefore, the input provided by the environment needs to be encoded into spikes, and the SNN simulated for multiple time-steps. Similarly, the action needs to be computed by decoding the output spikes over multiple time-steps.

This process complicates the definition of a state s_t, the next state s_{t+1} and the action a. This definition is, however, crucial for learning since most deep reinforcement learning methods require a memory of experienced trajectories (either in form of short-term rollouts for on-policy methods or a long-term replay buffer for off-policy methods). This memory is usually implemented as a traditional data structure, aside from the network. While similarities have been drawn with the sharp-wave ripple events in the hippocampus [46], a functional implementation of such memory with SNNs would be difficult.

Online Learning

As presented in Section 5.5, DECOLLE is capable of learning online, in the sense that weights can be updated as part of the neural dynamics, while the sensory input is streamed into the SNN. This contrasts with standard deep reinforcement learning methods that perform batch updates at the end of episodes. It was shown in Section 5.5 that this online learning strategy can achieve high performance on supervised learning tasks. In a supervised learning setup, the loss function is defined against ground truth targets which are fixed. In reinforcement learning, however, the estimation of the objective function depends on past experiences and the network parameters. Constantly changing parameters introduce variance in this estimation, hindering the learning of a performing policy. This observation was also noted in Mnih et al. [200], which introduced the concept of target networks – a copy of the learning network used to estimate the objective, but updating at a slower pace. The importance of a slow learning process is also discussed in [66].

Real-World Experiments

Neuromorphic hardware enables the realization of autonomous learning robots with embedded computing running on battery. However, the approaches presented in the field of deep reinforcement learning are complicated to apply in the real world.

One method consists of modeling the real-world environment in simulation so that an agent can be trained with virtual data. With sufficient randomization, the learned policy also generalizes to the real-world. This method was successfully applied to learn in-hand object manipulation [45].

Another method consists of learning directly on real-world robots. Because of the large number of required trials for a performing policy to be learned, the environment has to be automated to mitigate human supervision. Specifically, external off-board sensors are required to evaluate the reward. In addition, the environment needs to be resettable in a random initial state at the end of a trial. A successful instance was presented in [295], in which a robot learns to throw arbitrary objects in defined bins. The environment resets automatically with a motor lifting the plate in the throwing area to slide the objects back next to the robot. In [170], multiple identical robots are used to collect data in parallel to learn the grasping of arbitrary objects from images. In both cases, the reward is delivered by external cameras. On the other hand, neuromorphic learning robots would benefit from an approach that can be used without instrumentalization of the environment (without externally computed rewards and without reset). The field of intrinsic motivation, which enables the learning of policies with rewards generated by the robot's sensors, addresses this topic [256].

Local Losses

The DECOLLE rule relies on local loss functions at the hidden layers to learn intermediate representations. In a supervised learning setup, it was shown that a mean square error loss as the one used at the network output enabled the learning of performing visual representations. In a reinforcement learning setup, it is not clear how the local losses should be formulated to enable the learning of intermediate representations useful to the policy. It was shown in Jaderberg et al. [145] that auxiliary tasks – such as learning to predict future rewards or learning what action can modify the observation in a certain way – could yield useful intermediate representations. Other interesting approaches to learn policies without a reward are presented in [103, 80, 74, 158]. These approaches are referred to as curiosity or intrinsic motivation. Whether applying these loss functions locally in DECOLLE would yield the learning of useful intermediate representations should be investigated.

7. Conclusion and Outlook

This Chapter marks the end of this thesis. A summary of the approach and the contributions presented in the previous Chapters is provided. This thesis is then closed with a concluding statement, followed by a note on the lessons learned by the author.

7.1. Summary of the Approach

The two research goals of this thesis were to learn spatio-temporal representations from event-based data with synaptic plasticity rules, and control robots from these representations. These two goals were addressed separately, after familiarization with the field of computing with SNNs. First, synaptic plasticity rules derived in computational neuroscience were selected and evaluated on event-based classification benchmarks. This enabled the identification of the most functional rules, capable of training multi-layer SNNs, based on gradient descent. A novel rule of this form was proposed – DECOLLE – which currently outperforms other rules on the DvsGesture benchmark, and is suitable for neuromorphic hardware implementation. Second, three methods were investigated to couple visual representations to motor commands with SNNs. These methods were based on different assumptions about the task. The manual mapping required the user knowledge of what action should the robot perform for a given network output. The reward mapping required a reward function to be shaped. The prediction error minimization required a predictive model, here trained from a user demonstration. Concerning visuomotor coupling, the results obtained with SNN methods are below state-of-the-art ANNs trained with deep reinforcement learning. In the future, this gap could be closed with the application of gradient descent methods for SNNs to reinforcement learning.

7.2. Summary of the Contributions

Addressing the research goals in this thesis has led to contributions and to the development of novel tools to ease the prototyping of closed-loop experiments relying on event-based data and SNNs.

- **Tools for Visuomotor Neurorobotics**: The ability to test visuomotor experiments in simulation increases research productivity. To this end, the Gazebo DVS plugin – a closed-loop event-based simulator – was developed as part of this thesis. This plugin was made open-source, integrated in the NRP, and was already used by other researchers. A simulated lane following experiment relying on this plugin was designed and used to evaluate the reward-learning rule SPORE in Section 6.2. Similarly, a neuromorphic DVS head performing eye movements enabling the design of real-world experiments involving oculomotor control was introduced. This head was showcased in a stereo-setup and for recording a dataset of objects perceived with microsaccadic eye movements. This dataset was used to detect visual grasping affordances with eRBP in Section 6.1. It was also discussed how to connect event-based data to a SNN and introduced a novel attention mechanism evaluated in Section 5.4.

- **Learning Visual Representations from event-based data**: Relying on the tools developed within this thesis, the ability to learn spatio-temporal representations from AER with synaptic plasticity rules was investigated. One complexity of this task was to keep up with the newly proposed rules within the course of this thesis, requiring a close collaboration with computational neuroscientists. Initially, two bottom-up rules – STDP and LSM – were evaluated, and obtained unsatisfactory results compared to standard machine learning approaches. Subsequently, the top-down rules eCD and eRBP were evaluated, derived from contrastive divergence and random backpropagation. These rules matched ANN accuracy and retained the advantages associated to SNN, see Table 5.2 for summarized results. This Chapter closed with DECOLLE, derived during a stay at the University of California, Irvine. This rule obtained state-of-the-art results on the DvsGesture and N-MNIST benchmarks. Based on this derivation, a new version of eRBP was proposed, with similar performance to DECOLLE.

- **Visuomotor Coupling**: Closing the visuomotor loop involves a mapping from visual representations to motor commands. Three different methods to close the visuomotor loop were introduced: manual coupling, reward learning and prediction error minimization. Manual coupling is simple but limited and was demonstrated in a reaching and grasping experiment. Reward learning and prediction error minimization consist of learning this mapping. The reward-learning rule SPORE was evaluated in the lane following experiment. A method for visual imitation learning based on prediction error minimization computed from a LSM trained on event-based data was presented. These two methods provide initial results that should be improved to catch up with standard machine learning approaches.

7.3. Concluding Statements

The brain outperforms computer architectures in aspects of energy efficiency, robustness and adaptivity. The computational paradigms of the brain are vastly different from modern computer architectures. Biological neural networks base their computations on local information and communicate asynchronously with spikes. Understanding how these paradigms can be implemented in hardware would enable the design of autonomous learning robots operating at high speed for a fraction of the energy budget of current solutions.

Neuromorphic vision sensors mark an important step in shifting towards brain-inspired paradigms. The AER enables energy efficient and fast visual processing but requires new asynchronous algorithms to be derived. SNNs constitute a subset of asynchronous algorithms inspired by the brain and suited to neuromorphic hardware.

Learning in the brain is believed to be based on synaptic plasticity. Unlike conventional machine learning methods, weight updates are characterized in terms of information local to the synapse. Synaptic learning enables an efficient neuromorphic hardware implementation, asynchronous updates and online learning.

In this thesis, computational neuroscience was linked with robotics by exploring synaptic plasticity rules for learning spatio-temporal representations from event-based data. A top-down derivation of gradient descent for spiking neurons was provided, leading to plausible synaptic plasticity rules. It was shown that such rules were orders of magnitude more functional than bottom-up rules formalized from observed synaptic behaviors *in vitro*.

We additionally presented how representations could be mapped to motor commands, closing the visuomotor loop. While a few functional reward-learning synaptic plasticity rules have been proposed, there is a large performance gap with deep reinforcement learning. As with deep learning, this gap could be addressed by formalizing spiking backpropagation as a reward-maximization rule.

Advances in neuromorphic technology and computational neuroscience are important to reduce the power consumption of machine learning. Globally, electric energy is mostly generated from fossil fuels (in 2017, 58.97% from coal and oil alone [40]), which are limited and emit greenhouse gases. At the same time, the amount of energy required to train deep ANNs is considerable [277]. The increasing amount of computations in deep learning [224] might lead to represent a significant portion of global energy consumption. It is therefore important to research on methods to reduce the number of computations and the amount of energy per computation. It was proposed in [260] that the financial cost (related to the energy cost) of developing, training, and running neural networks should be reported alongside their accuracy.

7.4. Lessons Learned

A major complication encountered during the elaboration of this thesis was the rapidly changing tools and methods in both machine learning and computational neuroscience. The field of learning with SNNs considerably changed between 2015 when this thesis started and today in early 2020. The release of generic deep learning frameworks such as Tensorflow in 2015 and PyTorch in 2016 significantly simplified the prototyping of learning methods. However, these tools were not meant for SNNs simulations and it took some years to realize their potential, see Appendix A. The first experiments presented in this thesis were developed with PyNN-NEST, which – at the time of writing – was not ideal for prototyping functional synaptic plasticity rules. Such changes in software tools also complicated the reuse of software prototypes from a contribution to another. From the theoretical side, many contributions important to this thesis were published while the work was being conducted. Especially, feedback alignment (Lillicrap et al. [174]) and direct feedback alignment (Nøkland et al. [219]) published both in 2016 and the SNN adaptation in Neftci et al. [213] published in 2017 marked the introduction of biologically plausible gradient descent rules for SNNs. Before such rules, the performance of SNNs on learning benchmarks was not competitive against backpropagation with ANNs.

An important lesson learned during this thesis has been to evaluate prototypes first on simple tasks. Learning online in closed-loop control is difficult because the observations received by the network will depend on the network output. When possible it is therefore much simpler to first validate the learning rule in a reproducible supervised manner, by training on a dataset. Similarly, keeping code tidy and open-sourcing it enhances collaborations in research. Many open-source prototypes were released throughout this thesis, although it has not always been possible when the code relied on proprietary components.

Appendix

A. Simulating Spiking Neural Networks

Processing in SNNs happens in a distributed and asynchronous manner. All neurons and synapses in the SNN have their own dynamics, requiring frequent, simultaneous updates. The asynchronous and sparse communication of spikes requires these dynamics to be synchronized across neurons. From these particularities emerged a variety of techniques and hardware to simulate SNNs, very different from one another. In this Section, the main options for simulating SNNs are outlined.

A.1. Dedicated Neural Simulators

There are many open-source software for simulating SNNs using standard computer architectures. In general, these simulators implement different types of neurons and synapses that can be instantiated and wired into networks at setup time. The network can then be simulated for a given duration. Some neurons can spike spontaneously to account for the sensory input (by specifying spike-times, injecting currents or with a Poisson spike-train of a given rate). The activity (spikes or membrane potential) of other neurons can be recorded to serve as the SNN output.

Within this category, some simulators focus on reproducing biology accurately (NEST [119] and NEURON [131]). Other simulators focus on learning with spikes (Brian [121], Auryn [297]) and interactivity (Nengo [53]). The PyNN library enables to use many of these simulators with the same python code, by providing a common frontend and implementing the interface to multiple backends. With these frameworks, the user describes in code the structure of the SNN, see Algorithm 1. In this case, learning has to be implemented with custom neuron and synapse types.

Since they are implemented on digital hardware, these simulators have to update the neural dynamics at discrete time intervals. Generally, the differential equations are approximated with variations of the Euler method. Formally, a differential equation describing the evolution of the membrane potential, of the form:

$$\tau \frac{du}{dt}(t) = -u(t) + u_{rest} + I_{syn}(t), \tag{A.1}$$

ALGORITHM 1
Pseudo-code to setup and simulate a network with PyNN

```
input_neurons ← create_neurons(n=10, type=Poisson(rate=10))
hidden_neurons ← create_neurons(n=200)
output_neurons ← create_neurons(n=10)
synapse(input_neurons, hidden_neurons, weights=W1)
synapse(hidden_neurons, output_neurons, weights=W2)
run_network(T)
```

can be numerically approximated in a computer with the Euler method:

$$\begin{aligned} u(t+\Delta_t) &= u(t) + \Delta_t \times \frac{du}{dt}(t) \\ u(t+\Delta_t) &= u(t) + \Delta_t \times \frac{1}{\tau} \times (-u(t) + u_{rest} + I_{syn}(t)) \\ u(t+\Delta_t) &= u(t) \times \underbrace{\left(1 - \frac{\Delta_t}{\tau}\right)}_{\alpha} + \frac{\Delta_t}{\tau} \times u_{rest} + \frac{\Delta_t}{\tau} \times I_{syn}(t), \end{aligned} \tag{A.2}$$

with $\alpha\ \Delta_t$ the simulation time-step, usually $1\,\mathrm{ms}$ or below.

All the mentioned simulators can run on classical CPUs, with multi-threading. Additionally, NEST can run on super-computer clusters and Nengo can run on GPU. When simulated on a CPU, only spike events are transmitted from a neuron to another. However, all neurons need to be updated sequentially. A standard optimization is therefore to simulate n time-steps at once, with $n \times \Delta_t <= min_{delay}$ and min_{delay} denoting the smallest synaptic delay of the SNN in seconds. In this case, spikes are generally computed and transmitted to other neurons at intervals of min_{delay} seconds.

A.2. Generic Machine Learning Frameworks

Two popular generic machine learning frameworks were recently released: Tensorflow in 2016 by Google [37] and PyTorch in 2017 by Facebook [231]. Unlike previous deep learning frameworks such as Caffe [147], Tensorflow and PyTorch support many operations and are not limited to ANNs. Instead of instantiating neurons and synapses of specific types and connecting them into networks (as with dedicated neural simulators), the user describes the mathematical operations on the data (tensors) directly, see Algorithm 2. This enables a direct access to the operations performed and to the simulation loop.

Computational neuroscientists are now increasingly relying on such frameworks to experiment SNNs [214, 55, 12]. Indeed, the equations of the IF neuron can be easily implemented within these frameworks, see Algorithm 3. Since operations are defined on tensors for an efficient vectorization, the absence of spikes

ALGORITHM 2
Pseudo-code to setup and simulate a network with PyTorch

```
input_spikes ← Tensor(shape=(T, 10))
hidden_potentials ← Tensor(shape=(200))
output_potentials ← Tensor(shape=(200))
for t ← 1...T do
    hidden_potentials ← hidden_potentials ×α + W1 · input_spikes
    hidden_spikes ← hidden_potentials > threshold
    output_potentials ← output_potentials ×α + W2 · hidden_spikes
    output_spikes ← output_potentials > threshold
end for
```

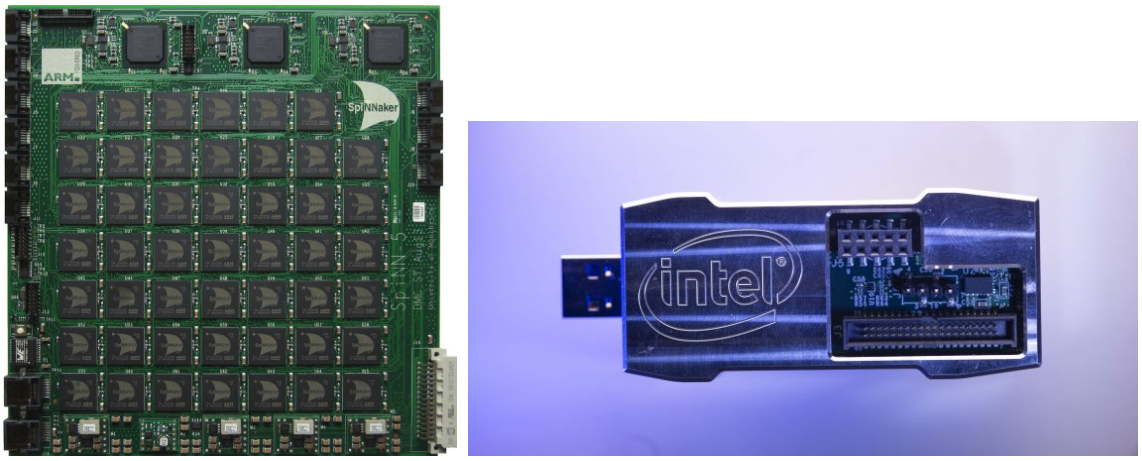
is also communicated. In other words, a layer of 1000 IF neurons will output a 1000-dimensional vector on every simulation time-step, with some 1s representing spikes but mostly 0s representing the absence of spikes. While somewhat counter-intuitive, such implementation has many advantages for computational neuroscientists researching on learning rules. First, both Tensorflow and PyTorch support automatic differentiation. Therefore, any variable in the computation graph can be optimized by gradient descent with respect to an arbitrary loss function. Second, large networks can be simulated on GPU, and learning can take advantage of a batch dimension (the same network is simulated multiple times at once). Third, since these frameworks are used by machine learning researchers, they offer many powerful features, such as advanced optimization schemes, native support for convolutions, normalization and dropout. They also ease collaborations across these fields of research and profit from an active community.

A.3. Dedicated Neuromorphic Hardware

Until now, only software frameworks to simulate SNNs on conventional computers were discussed. However, conventional computer architectures do not accommodate the regime of SNNs particularly well. Since neurons and synapses only have access to local information, and that the communication between neurons is sparse, dedicated hardware can distribute neurons on different cores with their own local memory. In some cases, this allows a significant reduction of power consumption as well as an increased processing speed. Such hardware designed specifically for the simulation of SNNs are referred to as neuromorphic hardware. They generally offer a high-level programming interface to describe the architecture of the SNN based on PyNN or similar, see Algorithm 1.

The most common type of neuromorphic hardware is digital, such as SpiNNaker [108] and Intel Loihi [84], see Figure A.1. As with dedicated neural simulators, the user provides a description of the network by specifying neuron types and

synaptic connections. This type of hardware can guarantee that the SNN runs in real-time. With the addition of live spike injection and retrieval, they are a good match for robotics applications.



(a) SpiNNaker. Source: [108] (b) Kapoho Bay, Loihi. Source: [140]

Figure A.1.: Example digital neuromorphic hardware.

Neuromorphic engineers also developed analog hardware, such as BrainScaleS [257] and DYNAP [201], see Figure A.2. In this case, the neurons are physically implemented with electrical circuits, and the neural properties directly mirrored by physical properties. The advantages of such hardware are extreme power efficiency (DYNAP neurons are implemented with transistors on sub-threshold regime) and extreme processing speed (BrainScaleS simulate SNNs 10000 times faster than biological time). However, these advantages are traded-off with noise and non-reproducibility since electrical signals are subject to variations in the surrounding environment (such as room temperature).

(a) BrainScaleS. Source: [222] (b) DYNAP-SE. Source: [51]

Figure A.2.: Example analog neuromorphic hardware.

B. Continuous Version of eRBP

In this Chapter, the main differences between eRBP and DECOLLE are discussed. A new version of eRBP is derived, named Continuous Random Backpropagation [7] (cRBP), which accounts for temporal dynamics as DECOLLE. Importantly, this new version is implemented using PyTorch, allowing a fair comparison with DECOLLE, with an identical neuron model, training regime, network architecture, and optimizer. Some of the material covered in this Section was originally published by the author in Kaiser et al. [7].

B.1. Differences Between eRBP and DECOLLE

For convenience, the generic eRBP equation defined in Equation (5.12) is reported below:

$$\Delta w_{ij}(t) \propto \sum_{k \in out} e_k(t) g_{ik} \times box(u_i(t)) \times s_j(t), \tag{B.1}$$

with s_j the pre-synaptic spike-trains, *box* the boxcar function, e_k the prediction error for output neuron k and g_{ik} a fixed random feedback weight. Similarly, the DECOLLE rule defined in Equation (5.21) is formulated as follows:

$$\Delta w_{ij}^l(t) \propto \sum_{k \in rdout^l} e_k^l(t) g_{ki}^l \times \sigma'(u_i^l(t)) \times \epsilon * s_j^{l-1}(t). \tag{B.2}$$

with superscript l denoting the neural layer to which the variable belongs.

The following differences between the two rules can be noted:

1. Unlike eRBP, DECOLLE filters the pre-synaptic spikes with ϵ to account for PSP dynamics;
2. The chosen surrogate functions for eRBP and DECOLLE are different (boxcar and derivative of sigmoid);
3. DECOLLE errors are locally computed at the layer with the local readouts $rdout^l$, while eRBP errors are the network errors computed from the network output *out*.

The first two differences result from different approximation techniques in the derivation, while the third point is the core difference between the two rules. DECOLLE and eRBP assign credits of hidden neurons differently for a given error. This is the core difference depicted in Figure 5.22.

B.2. Continuous Random Backpropagation

Unlike eRBP, DECOLLE takes the PSP dynamics into account in the weight update equation. This behavior can be adapted to eRBP by reformulating the equation:

$$\Delta w_{ij}(t) \propto \sum_{k \in out} e_k(t) g_{ki} \times \sigma'(u_i(t)) \times \epsilon * s_j(t). \tag{B.3}$$

In this case, the weights are continuously updated along with the neural dynamics, unlike the original eRBP formulation which triggered weight updates on pre-synaptic spikes, see Section 5.4. For a better comparison, the boxcar function is also replaced by the derivative of the sigmoid. The only remaining difference with DECOLLE is captured in the e_k errors, which are obtained locally for DECOLLE (with local readouts $rdout^l$ – specific to the layer l), and at the network level for cRBP (with network output *out*). From these equations, an efficient implementation of both cRBP and DECOLLE can be leveraged using machine learning frameworks and automatic differentiation (see Appendix A.2). A pseudo-code of this implementation is provided in Algorithm 3.

In Algorithm 3, f_{loss} denotes the loss function, r the learning rate, GRAD an operation to compute the gradient with automatic differentiation and DETACH an operation that prevents the flow of gradients by setting them to zero. The instructions of the form DETACH$(A-B)+B$ enable the forward computations to return A while the derivative will be computed with respect to B. In line 11, it is used in the forward dynamics to implement the surrogate gradient.

The implemented neural dynamics to advance the state of the neural simulation are expressed in matrix formulation and described in [12]. It is ensured that gradients are not propagated backward from a layer to another through the whole network with the DETACH operation on line 9, enabling RTRL instead of BPTT. The described dynamics is shared between the cRBP and DECOLLE implementations. Both cRBP and DECOLLE compute pseudo-outputs Y^l with a random combination G^l of the spikes S^l for every l layer. In the case of DECOLLE, pseudo-targets $\hat{Y}^l$ are assumed for every hidden layer l. The gradients of the local weights W are calculated with respect to the loss function between the pseudo-targets and the pseudo-outputs. In the case of cRBP, the loss function is only applied to calculate the network loss at the output, see line 21. The pseudo-output of the hidden layers are subsequently multiplied with the network loss on line 24. In effect, this provides an implementation of direct feedback alignment. In practice, the weight updates are nested in the forward dynamics computations – it is here split for clarity.

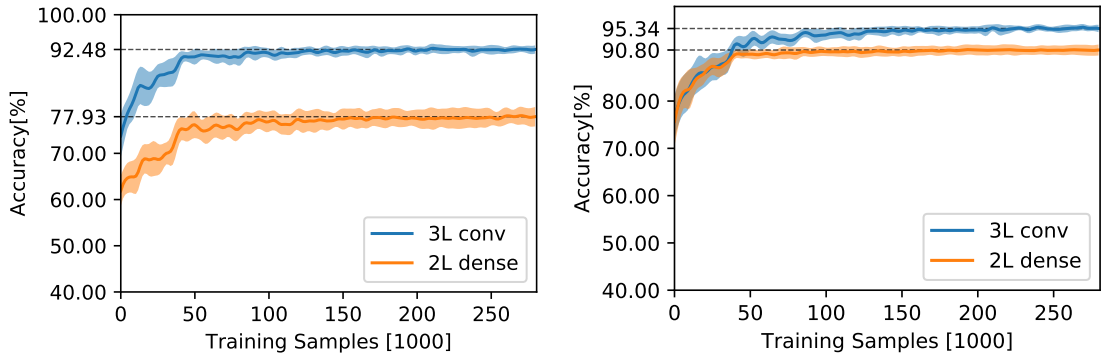
ALGORITHM 3
Pseudo-code for cRBP and DECOLLE

Initialization
$P^l, Q^l, U^l, S^l, R^l, Y^l \leftarrow [0...0]$ for $l \in [1, ...L]$
$S^0 \leftarrow$ *input spikes*
$\hat{Y}^l \leftarrow$ *pseudo-targets for layer l*
for $t = 1...T$ **do**
  Advance State
  for $l = 1...L$ **do**
    $P^l = \alpha P^l + (1 - \alpha)Q^l$
    $Q^l = \beta Q^l + (1 - \beta)\text{DETACH}(S^{l-1})$
    $U^l = W^l \cdot P^l - R^l$
    $S^l = \text{DETACH}(\Theta(U^l) - \sigma(U^l)) + \sigma(U^l)$
    $R^l = \gamma R^l + (1 - \gamma)\text{DETACH}(S^l)$
    $Y^l = G^l \cdot S^l$
  end for
  Weight Update DECOLLE
  for $l = 1...L$ **do**
    $L^l = f_{loss}(Y^l, \hat{Y}^l)$
    $W^l = W^l + r \times \text{GRAD}(L^l, W^l)$
  end for
  Weight Update cRBP
  $L^L = f_{loss}(Y^L, \hat{Y}^L)$
  $W^L = W^L + r \times \text{GRAD}(L^L, W^L)$
  for $l = 1...(L - 1)$ **do**
    $L^l = Y^l \cdot \text{DETACH}(L^L)$
    $W^l = W^l + r \times \text{GRAD}(L^l, W^l)$
  end for
end for

B.3. Performance Comparison

The same experimental setup to evaluate DECOLLE is used, see Section 5.5. Unlike the Auryn implementation of eRBP presented in Section 5.4, the input dimension is set to 32x32 instead of 64x64. Additionally, synaptic and axonal delays are not simulated – spikes are transmitted from a layer to another within the same time-step. The errors are computed and communicated as analog values to the neurons for the synaptic updates. Both the 2-layers dense architecture introduced in Section 5.4 and the 3-layer convolutional architecture introduced in Section 5.5 are benchmarked. The results are presented for cRBP on the DvsGesture dataset in Figure B.1, and included in Table 5.2.

The accuracy of the cRBP rule with the 2-layer dense architecture on a 32x32 input is slightly lower than the accuracy of eRBP on a 64x64 input. This accuracy increases significantly when replacing the input rescaling with the attention



(a) Input rescaling to 32x32

(b) Attention window 32x32

Figure B.1.: Classification accuracy for the DvsGesture task during learning with cRBP. Shadings indicate standard deviation across the 7 runs. Source: [7].

mechanism: from 77.93% to 90.80%. Similarly, the accuracy of the cRBP rule with the convolutional architecture is slightly lower than the accuracy of DECOLLE. This accuracy moderately increases when replacing the input rescaling with the attention mechanism: from 92.48% to 95.34%.

C. DVS Head Blueprints

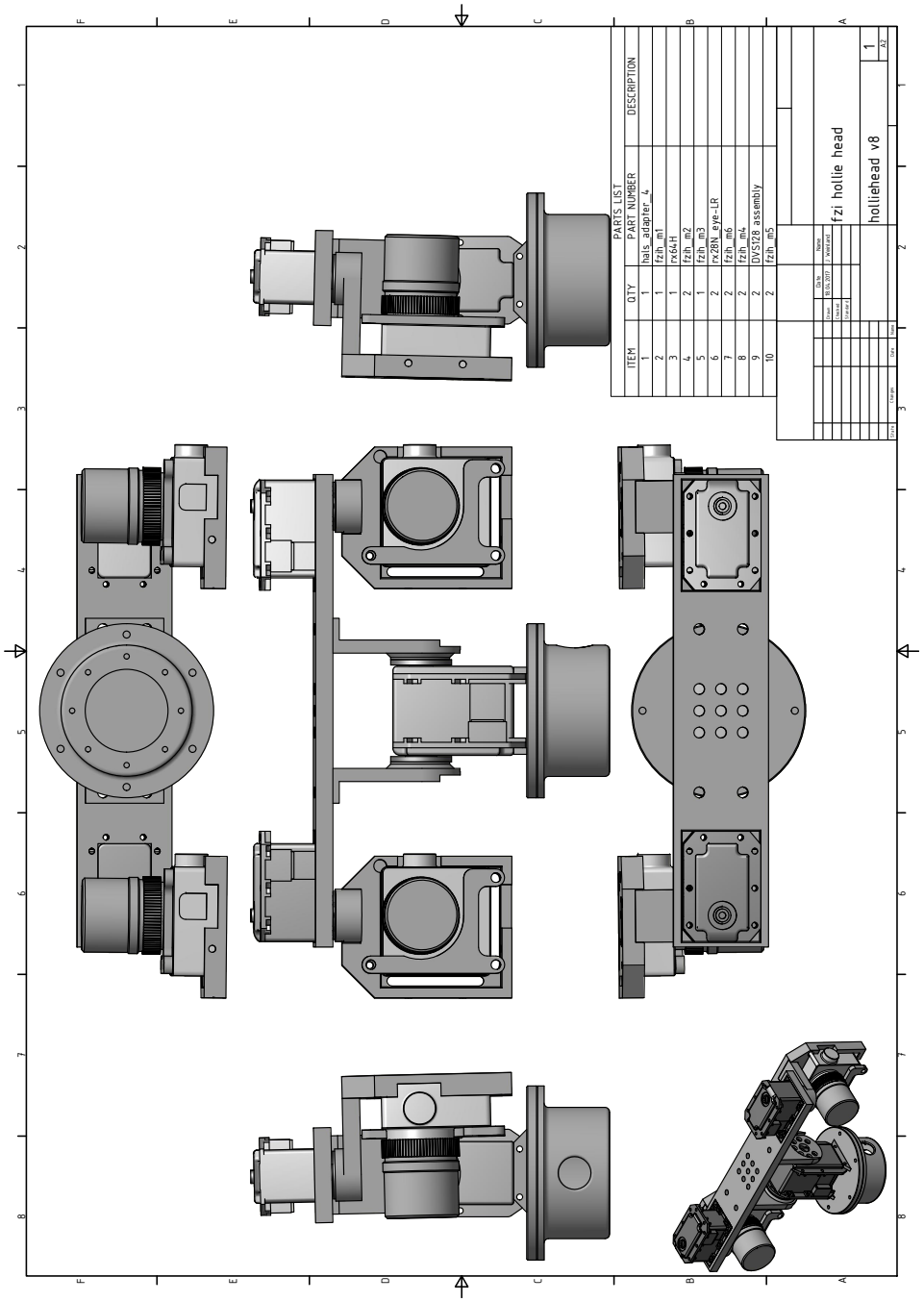


Figure C.1.: Blueprints of the 3D printed robotic stereo-DVS head.

Acronyms

AER Address Event Representation. Denote the hardware protocol to communicate data in event-based hardware, introduced in Lazzaro et al. [166]. Events are emitted at precise time, each event contains the address of a receptor and some payload. vii, 3, 38, 45, 50, 77, 84, 124, 125, 143, 145

ANN Neural networks where neurons communicate synchronously and with continuous values (as opposed to SNNs). At time of writing, this type of network is by far the most commonly used in the field of deep learning In literature, the acronym ANN ambiguously refers to either Analog Neural Network or Artificial Neural Network, but always denotes the same concept. Since SNNs are also artificial, the term Analog Neural Network is preferred.. vii, 2, 7, 16, 17, 20, 22, 23, 27, 30, 32, 33, 37–40, 42–44, 46, 67, 76, 98, 118, 123–126, 130

BPTT Backpropagation-Through-Time. vii, 33, 34, 37, 96, 118, 134

CMA-ES Covariance Matrix Adaptation Evolution Strategy [127]. Black-box optimization method based on Evolution Strategy. At each iteration, offspring are generated by sampling a multivariate Gaussian distribution. A weighted combination of the best offspring survive an iteration, and covariance matrices are ingeniously updated . vii, 78, 112–114

CPU Central Processing Unit. vii, 21, 41, 130

cRBP Continuous Random Backpropagation [7]. vii, 98, 99, 133–136, 146

DECOLLE Deep Continuous Local Learning [12]. vii, xi, 6, 34, 35, 38, 52, 53, 89, 92–99, 118–121, 123, 124, 133–136, 146

DVS Dynamic Vision Sensor. Biologically inspired vision sensor which emit address events upon local light intensity changes. vii, x, xi, 5, 7, 24, 41–43, 45, 47, 49, 51, 53, 55–61, 63, 65, 69–71, 74, 76, 78, 86, 89, 101–104, 108, 113, 114, 124, 137, 145, 146

e-prop Eligibility Propagation [55]. vii, 34, 38, 75

eCD Event-Driven Contrastive Divergence [211]. vii, 6, 30, 84–88, 124, 146

eRBP Event-Driven Random Backpropagation [213]. vii, xi, 6, 7, 35–37, 53, 59, 88–93, 97–99, 101–104, 124, 133–136, 145, 146

FFT Fast Fourier Transform. vii

GPU Graphics Processing Unit. vii, 7, 96, 130, 131

IF Integrate-and-Fire. Family of phenomenological spiking neuron models. Incoming spikes alter the membrane potential, and the neuron spikes (fires) when its membrane potential reaches a threshold value. The membrane potential leaks in the absence of spikes to a resting potential. vii, 18, 19, 30, 34–37, 43, 49, 65, 92, 93, 96, 130, 131, 140, 145

LSM Liquid State Machine. Pool of neurons recurrently connected exhibiting a fading memory property, such as the waves in a bucket of water. vii, x, 6, 32, 33, 67–73, 75, 76, 78–80, 83, 96, 99, 111–113, 115, 116, 124, 145, 146

LSTM Long Short-Term Memory [132]. vii, 33, 34, 53

LTD Long Term Depression. Instantiation of a STDP rule where correlations in spike times trigger weight decrease. vii, 12, 27, 30, 85

LTP Long Term Potentiation. Instantiation of a STDP rule where correlations in spike times trigger weight increase. vii, 12, 27, 30, 85

NEF Neural Engineering Framework [275]. Framework allowing to approximate functions with two-layer SNNs of IF neurons implemented with the Nengo simulator. The SNNs can be combined together to form complex cognitive architectures. vii, 43, 45

NRP Neurorobotics Platform [4]. Open-source[1] simulation environment to design and evaluate experiments with robots controlled with SNNs. vii, 5, 42, 43, 49, 112, 113, 124

PSP Post-Synaptic Potential. Conversion at the synapse of an instantaneous spike into a voltage contribution to the membrane potential of a neuron. vii, 12, 19–21, 31, 37, 92, 93, 133, 134

RBM Restricted Boltzmann Machine. vii, 30, 31, 84, 86, 88

ReLU Rectified Linear Unit. vii, 44

ROS Robot Operating System. vii, 49, 55, 60, 61, 103, 108

RTRL Real-Time Recurrent Learning [287]. vii, 35, 37, 95, 134

SCDBN Spiking Convolutional Deep Belief Network. vii, 85–87, 146, 147

[1] `https://bitbucket.org/hbpneurorobotics/neurorobotics-platform`

SNN Spiking Neural Network. Artificial neural network where neurons communicate asynchronously with spikes. Both neurons and synapses are stateful, their dynamics are described with respect to time. vii, xi, 1, 4–7, 20, 22, 25–28, 31, 32, 34–38, 40–51, 53, 56, 58–61, 63–65, 67, 76, 77, 88, 92, 96–98, 101–103, 107, 118–120, 123–126, 129–132, 139, 140, 144

SPORE Synaptic Plasticity with Online Reinforcement learning [150]. vii, 6, 30, 31, 59, 107–110, 118, 124, 146

SRM Spike Response Model. vii, 19, 21, 37, 93, 145

STDP Spike-Timing-Dependent-Plasticity Learning rule where weight update depends on precise pre-synaptic and post-synaptic spike times. vii, ix, 6, 12, 22, 23, 25–30, 44–46, 52, 53, 76–78, 80–84, 86–89, 98, 99, 124, 140, 145, 146

SVM Support Vector Machine. vii, 27, 78–80, 86, 99

Glossary

credit assignment Denotes the formulation and computation of the credit of a given neuron or synapse in the current task performance . vii, 22, 23, 34, 98

direct feedback alignment Variation of feedback alignment. The random feedback weights are connected from the network output to the hidden neurons directly, bypassing previous hidden layers. Introduced in [219] . vii, 36, 89, 95, 126, 134

eligibility trace Trace in a neuron or a synapse of its previous activity, also referred to as synaptic tag. vii, 37, 38, 92, 94, 96, 98

event-based Sensor, data, algorithm or computation relying on AER. For example, conventional sensors provide readings of all receptors at regular frequency. Instead, event-based sensors emit individual events upon change in readings from a particular receptor. vii, 1–3, 5–7, 14, 20, 23–25, 33, 38–52, 55, 58, 59, 63, 68, 70, 77, 78, 80, 84, 86–88, 92, 98, 101, 102, 107, 108, 112, 115, 118, 123–125, 139, 145, 146

feedback alignment The realization that a network can adapt through learning to fixed random feedback weights, decoupled from the forward weights, thus solving the weight transport problem. Introduced in [174] . vii, 36, 126, 143

HMAX Hierarchical models for object recognition based on alternating layers of simple cells detecting features and complex cells pooling the dominant features with the MAX operation, first introduced in [248]. vii, 53, 76–81, 83, 84, 86, 89, 99, 146

mean square error Common loss function used in machine learning. It is expressed as $L = \frac{1}{2}(\hat{y} - y)^2$, with y the network output and $\hat{y}$ the targets . vii, 35, 94, 95, 112, 121

NEST Spiking Neural Network Simulator presented in [119], with python bindings introduced in [97] . vii, 126, 144

Poisson spike-train spike-train in which the spikes for a given neuron are emitted at a given rate independently of one another, as drawn from a Poisson Distribution. vii, 5, 25, 77, 78, 80, 129

PyNN Python library offering a common wrapper over common SNNs simulators (backends) such as NEST, SpiNNaker and others, introduced in [85] . vii, 126, 129–131

PyTorch Generic deep learning framework presented in [232]. vii, 96, 98, 130, 131, 133

spike-train Spikes emitted by a neuron over a time period. vii, 4, 5, 20, 29, 37, 62, 70, 72, 79, 89–93, 96, 104, 106, 114, 117, 144–146

weight transport problem This problem denotes the dependence on the forward weights in the computation of the weight update when training a neural network. This dependence is biologically implausible and impractical for a spiking network implementation . vii, 36, 88, 95, 143

winner-take-all circuit Two-layers network in which every input pattern is matched to a single, competing output. vii, 26, 29, 31, 33

List of Figures

List of Tables

Publications by the author et al.

This directory lists all publications for which the author of this dissertation is either the first author or has contributed significantly to the publication as co-author (in the form of problem definition, solution, discussion or experimental evaluation).

[1] Alessandro Ambrosano, Lorenzo Vannucci, Ugo Albanese, Murat Kirtay, Egidio Falotico, Georg Hinkel, Jacques Kaiser, Stefan Ulbrich, Paul Levi, Christian Morillas, et al. Retina color-opponency based pursuit implemented through spiking neural networks in the neurorobotics platform. In *Conference on Biomimetic and Biohybrid Systems*, pages 16–27. Springer, 2016.

[2] Gabriele Bolano, Pascal Becker, Jacques Kaiser, Arne Roennau, and Ruediger Dillmann. Advanced usability through constrained multi modal interactive strategies: The cookiebot. In *2019 19th International Conference on Advanced Robotics (ICAR)*, pages 213–219. IEEE, 2019.

[3] Alban Bornet, Jacques Kaiser, Alexander Kroner, Egidio Falotico, Alessandro Ambrosano, Kepa Cantero, Michael H Herzog, and Greg Francis. Running large-scale simulations on the neurorobotics platform to understand vision-the case of visual crowding. *Frontiers in neurorobotics*, 13:33, 2019.

[4] Egidio Falotico, Lorenzo Vannucci, Alessandro Ambrosano, Ugo Albanese, Stefan Ulbrich, J. Camilo Vasquez Tieck, Georg Hinkel, Jacques Kaiser, Igor Peric, Oliver Denninger, Nino Cauli, Murat Kirtay, Arne Roennau, Gudrun Klinker, Axel Von Arnim, Luc Guyot, Daniel Peppicelli, Pablo Martínez-Cañada, Eduardo Ros, Patrick Maier, Sandro Weber, Manuel Huber, David Plecher, Florian Röhrbein, Stefan Deser, Alina Roitberg, Patrick van der Smagt, Rüdiger Dillman, Paul Levi, Cecilia Laschi, Alois C. Knoll, and Marc-Oliver Gewaltig. Connecting artificial brains to robots in a comprehensive simulation framework: The neurorobotics platform. *Frontiers in Neurorobotics*, 11:2, 2017.

[5] Jacques Kaiser and Rüdiger Dillmann. Learning movements by imitation from event-based visual prediction. In *2nd Human Brain Project Student Conference*, 2018. (Extended Abstract).

[6] Jacques Kaiser, Alexander Friedrich, J. Camilo Vasquez Tieck, Daniel Reichard, Arne Roennau, Emre Neftci, and Rüdiger Dillmann. Embodied neuromorphic vision with event-driven random backpropagation. *arXiv preprint arXiv:1904.04805*, 2019.

[7] Jacques Kaiser, Alexander Friedrich, J. Camilo Vasquez Tieck, Daniel Reichard, Arne Roennau, Emre Neftci, and Rüdiger Dillmann. Embodied neuromorphic vision with continuous random backpropagation. In *International Conference on Biomedical Robotics and Biomechatronics (BIOROB)*. IEEE, 2020.

[8] Jacques Kaiser, Michael Hoff, Andreas Konle, J. Camilo Vasquez Tieck, David Kappel, Daniel Reichard, Anand Subramoney, Robert Legenstein, Arne Roennau, Wolfgang Maass, et al. Embodied synaptic plasticity with online reinforcement learning. *Frontiers in Neurorobotics*, 13:81, 2019.

[9] Jacques Kaiser, Gerd Lindner, J. Camilo Vasquez Tieck, Martin Schulze, Michael Hoff, Arne Roennau, and Rüdiger Dillmann. Microsaccades for asynchronous feature extraction with spiking networks. In *International Conference on Development and Learning and Epigenetic Robotics (ICDL-EPIROB)*. IEEE, 2018.

[10] Jacques Kaiser, Agostino Martinelli, Flavio Fontana, and Davide Scaramuzza. Simultaneous state initialization and gyroscope bias calibration in visual inertial aided navigation. *IEEE Robotics and Automation Letters (RA-L)*, 2(1):18–25, 2016.

[11] Jacques Kaiser, Svenja Melbaum, J. Camilo Vasquez Tieck, Arne Roennau, Martin V. Butz, and Rüdiger Dillmann. Learning to reproduce visually similar movements by minimizing event-based prediction error. In *International Conference on Biomedical Robotics and Biomechatronics (BIOROB)*. IEEE, 2018.

[12] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

[13] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in neuroscience*, 2020.

[14] Jacques Kaiser, Rainer Stal, Anand Subramoney, Arne Roennau, and Rüdiger Dillmann. Scaling up liquid state machines to predict over address events from dynamic vision sensors. *Bioinspiration & Biomimetics*, 12(5):055001, 2017.

[15] Jacques Kaiser, J. Camilo Vasquez Tieck, Christian Hubschneider, Peter Wolf, Michael Weber, Michael Hoff, Alexander Friedrich, Konrad Wojtasik, Arne Roennau, Ralf Kohlhaas, et al. Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, pages 127–134. IEEE, 2016.

[16] Jacques Kaiser, Jakob Weinland, Philip Keller, Lea Steffen, J. Camilo Vasquez Tieck, Daniel Reichard, Arne Roennau, Jörg Conradt, and Rüdiger Dillmann. Microsaccades for neuromorphic stereo vision. In *International Conference on Artificial Neural Networks (ICANN)*, pages 244–252. Springer, 2018.

[17] Jacques Kaiser, David Zimmerer, J. Camilo Vasquez Tieck, Stefan Ulbrich, Arne Roennau, and Rüdiger Dillmann. Spiking convolutional deep belief networks. In *International Conference on Artificial Neural Networks (ICANN)*, pages 3–11. Springer, 2017.

[18] Igor Peric, Robert Hangu, Jacques Kaiser, Stefan Ulbrich, Arne Roennau, Johann Marius Zoellner, and Ruediger Dillman. Semi-supervised spiking neural network for one-shot object appearance learning. In *Proceedings of the 5th International Congress on Neurotechnology, Electronics and Informatics - NEUROTECHNIX,*, pages 47–53, 2017.

[19] Lea Steffen, Benedict Hauck, Jacques Kaiser, Jakob Weinland, Stefan Ulbrich, Daniel Reichard, Ame Roennau, and Rüdiger Dillmann. Creating an obstacle memory through event-based stereo vision and robotic proprioception. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 1829–1836. IEEE, 2019.

[20] Lea Steffen, Daniel Reichard, Jakob Weinland, Jacques Kaiser, Arne Roennau, and Rüdiger Dillmann. Neuromorphic stereo vision: A survey of bio-inspired sensors and algorithms. *Frontiers in neurorobotics*, 13:28, 2019.

[21] J. Camilo Vasquez Tieck, Heiko Donat, Jacques Kaiser, Igor Peric, Stefan Ulbrich, Arne Roennau, Marius Zöllner, and Rüdiger Dillmann. Towards grasping with spiking neural networks for anthropomorphic robot hands. In *International Conference on Artificial Neural Networks (ICANN)*, pages 43–51. Springer, 2017.

[22] J. Camilo Vasquez Tieck, Jacques Kaiser, Lea Steffen, Martin Schulze, Axel von Arnim, Daniel Reichard, Arne Roennau, and Rüdiger Dillmann. The neurorobotics platform for teaching – embodiment experiments with spiking neural networks and virtual robots. In *International Conference on Cyborg and Bionic Systems (CBS)*. IEEE, 2019.

[23] J. Camilo Vasquez Tieck, Marin Vlastelica Pogancic, Jacques Kaiser, Arne Roennau, Marc-Oliver Gewaltig, and Rüdiger Dillmann. Learning continuous muscle control for a multi-joint arm by extending proximal policy optimization with a liquid state machine. In *International Conference on Artificial Neural Networks (ICANN)*, pages 211–221. Springer, 2018.

[24] J. Camilo Vasquez Tieck, Lea Steffen, Jacques Kaiser, Roenau Arne, and Rüdiger Dillmann. Multi-modal motion activation for robot control using spiking neurons. In *International Conference on Biomedical Robotics and Biomechatronics (BIOROB)*. IEEE, 2018.

[25] J. Camilo Vasquez Tieck, Lea Steffen, Jacques Kaiser, Daniel Reichard, Arne Roennau, and Ruediger Dillmann. Combining motor primitives for perception driven target reaching with spiking neurons. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 13(1):12, 2019.

[26] J. Camilo Vasquez Tieck, Lea Steffen, Jacques Kaiser, Daniel Reichard, Arne Roennau, and Rüdiger Dillmann. Controlling a robot arm for target reaching without planning using spiking neurons. In *2018 IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 111–116. IEEE, 2018.

Student work

This directory lists student theses that were advertised and supervised by the author of this dissertation as part of his research. This includes the relevant specification of the problem, discussion of the work, as well as marginal specifications for solution, visualization and experimental evaluation.

[27] Alexander Friedrich. Learning spatio-temporal features with event-driven random backpropagation. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2018.

[28] Michael Hoff. Learning closed-loop robot control with spiking neurons and event-based vision. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2017.

[29] Philip Keller. Object motion estimation with spiking neural networks using event-based vision and optical flow. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2018.

[30] Andreas Konle. Reinforcement learning on the neurorobotics lane following benchmark. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2018.

[31] Gerd Lindner. Feature extraction from an event stream using spiking neural networks. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2017.

[32] Svenja Melbaum. Learning movements from visual prediction. Master's thesis, Eberhard Karls Universität Tübingen, Tübingen, 2017.

[33] Fedor Scholz. Evaluation of embodied attention and memory models in the neurorobotics platform. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2018.

[34] Rainer Stal. Liquid state machine from asynchronous motion prediction with spiking neural network. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2016.

[35] Jakob Weinland. Event-based stereo vision with spiking neural networks. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2017.

[36] David Zimmerer. Spiking convolutional deep belief networks for unsupervised high-level feature extraction and pattern reconstruction. Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, 2017.

Bibliography

[37] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[38] Larry F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304, 1999.

[39] Edward H Adelson and James R Bergen. Spatiotemporal energy models for the perception of motion. *Josa a*, 2(2):284–299, 1985.

[40] International Energy Agency. Electricity statistics. https://www.iea.org/statistics/electricity/, accessed March 13, 2020.

[41] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557, 2015.

[42] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[43] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7243–7252, 2017.

[44] Alexander Andreopoulos, Hirak J Kashyap, Tapan K Nayak, Arnon Amir, and Myron D Flickner. A low power, high throughput, fully event-based stereo system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7532–7542, 2018.

[45] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[46] Laura A. Atherton, David Dupret, and Jack R. Mellor. Memory trace replay: The shaping of memory consolidation by neuromodulation. *Trends in Neurosciences*, 38(9):560–570, 2015.

[47] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020.

[48] Praveenram Balachandar and Konstantinos P Michmizos. A spiking neural network emulating the structure of the oculomotor system requires no learning to control a biomimetic robotic head. *arXiv preprint arXiv:2002.07534*, 2020.

[49] Pierre Baldi and Peter Sadowski. A theory of local learning, the learning channel, and the optimality of backpropagation. *Neural Networks*, 83:51–74, 2016.

[50] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pages 9368–9378, 2018.

[51] Felix Christian Bauer, Dylan Richard Muir, and Giacomo Indiveri. Real-time ultra-low power ecg anomaly detection using an event-driven neuromorphic processor. *IEEE transactions on biomedical circuits and systems*, 2019.

[52] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[53] Trevor Bekolay, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence C Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith. Nengo: a python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, 7:48, 2014.

[54] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, pages 787–797, 2018.

[55] Guillaume Bellec, Franz Scherr, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets. *arXiv preprint arXiv:1901.09049*, 2019.

[56] Philipp Bender, Julius Ziegler, and Christoph Stiller. Lanelets: Efficient map representation for autonomous driving. In *Intelligent Vehicles Symposium Proceedings*, pages 420–425. IEEE, 2014.

[57] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

[58] Michael Beyeler, Nikil D Dutt, and Jeffrey L Krichmar. Categorization and decision-making in a neurobiologically plausible spiking network using a stdp-like learning rule. *Neural Networks*, 48:109–124, 2013.

[59] Guo-qiang Bi and Mu-ming Poo. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, 18(24):10464–10472, 1998.

[60] Olivier Bichler, Damien Querlioz, Simon J Thorpe, Jean-Philippe Bourgoin, and Christian Gamrat. Unsupervised features extraction from asynchronous silicon retina through spike-timing-dependent plasticity. In *The 2011 International Joint Conference on Neural Networks*, pages 859–866. IEEE, 2011.

[61] Zhenshan Bing, Claus Meschede, Guang Chen, Alois Knoll, and Kai Huang. Indirect and direct training of spiking neural networks for end-to-end control of a lane-keeping vehicle. *Neural Networks*, 121:21–36, 2020.

[62] Zhenshan Bing, Claus Meschede, Kai Huang, Guang Chen, Florian Rohrbein, Mahmoud Akl, and Alois Knoll. End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[63] Zhenshan Bing, Claus Meschede, Florian Röhrbein, Kai Huang, and Alois C. Knoll. A survey of robotics control based on learning-inspired spiking neural networks. *Frontiers in Neurorobotics*, 12:35, 2018.

[64] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, volume 48, pages 17–37, 2000.

[65] Alban Bornet, Adrien Doerig, Michael Herzog, and Gregory Francis. Crowding asymmetries in a neural model of image segmentation. *Journal of Vision*, 17(10):365–365, 2017.

[66] Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

[67] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240$\times$ 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.

[68] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[69] Lars Buesing, Johannes Bill, Bernhard Nessler, and Wolfgang Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol*, 7(11):e1002211, 2011.

[70] H Burgsteiner, M Kraall, and G Steinbauer. Movement prediction from real-world images using a liquid state machine. *Applied Intelligence*, 26(2):99–109, 2007.

[71] Harald Burgsteiner. Training networks of biological realistic spiking neurons for real-time robot control. In *Proceedings of the 9th international conference on engineering applications of neural networks, Lille, France*, pages 129–136, 2005.

[72] Anthony N Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biological cybernetics*, 95(1):1–19, 2006.

[73] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[74] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

[75] Jan Chorowski, Ron J Weiss, Samy Bengio, and Aäron van den Oord. Unsupervised speech representation learning using wavenet autoencoders. *arXiv preprint arXiv:1901.08810*, 2019.

[76] Ami Citri and Robert C Malenka. Synaptic plasticity: multiple forms, functions, and mechanisms. *Neuropsychopharmacology*, 33(1):18, 2008.

[77] Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Asynchronous event-based corner detection and matching. *Neural Networks*, 66:91–106, 2015.

[78] Claudia Clopath, Lars Büsing, Eleni Vasilaki, and Wulfram Gerstner. Connectivity reflects coding: a model of voltage-based stdp with homeostasis. *Nature neuroscience*, 13(3):344, 2010.

[79] Claudia Clopath and Wulfram Gerstner. Voltage and spike timing interact in stdp–a unified model. *Frontiers in synaptic neuroscience*, 2:25, 2010.

[80] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: Intrinsically motivated multi-task, multi-goal reinforcement learning. *arXiv preprint arXiv:1810.06284*, 2018.

[81] Jorg Conradt, Raphael Berner, Matthew Cook, and Tobi Delbruck. An embedded aer dynamic vision sensor for low-latency pole balancing. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 780–785. IEEE, 2009.

[82] Jörg Conradt, Matthew Cook, Raphael Berner, Patrick Lichtsteiner, Rodney J Douglas, and Tobi Delbruck. A pencil balancing robot using a pair of aer dynamic vision sensors. In *2009 IEEE International Symposium on Circuits and Systems*, pages 781–784. IEEE, 2009.

[83] Sebastien M Crouzet, Holle Kirchner, and Simon J Thorpe. Fast saccades toward faces: face detection in just 100 ms. *Journal of vision*, 10(4):16–16, 2010.

[84] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[85] Andrew P Davison. Pynn: a common interface for neuronal network simulators. *Frontiers in Neuroinformatics*, 2(January):11, 2008.

[86] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor. *Frontiers in neuroscience*, 7:223, 2013.

[87] Guillaume Desjardins et al. Empirical evaluation of convolutional RBMs for vision. Technical Report 1327, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, 2008.

[88] Peter U Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9(August):99, 2015.

[89] Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[90] Georgi Dikov, Firouzi Mohsen, Florian Röhrbein, Jörg Conradt, and Christoph Richter. Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware. *Frontiers in Neuroscience*, 2017.

[91] Mikael Djurfeldt, Johannes Hjorth, Jochen M. Eppler, Niraj Dudani, Moritz Helias, Tobias C. Potjans, Upinder S. Bhalla, Markus Diesmann, Jeanette Hellgren Kotaleski, and Örjan Ekeberg. Run-Time Interoperability Between Neuronal Network Simulators Based on the MUSIC Framework. *Neuroinformatics*, 8(1):43–60, mar 2010.

[92] Bogdan Draganski, Christian Gaser, Volker Busch, Gerhard Schuierer, Ulrich Bogdahn, and Arne May. Neuroplasticity: changes in grey matter induced by training. *Nature*, 427(6972):311, 2004.

[93] Ariane Dröscher. Camillo golgi and the discovery of the golgi apparatus. *Histochemistry and cell biology*, 109(5-6):425–430, 1998.

[94] Michael Eickenberg, Alexandre Gramfort, Gaël Varoquaux, and Bertrand Thirion. Seeing it all: Convolutional network layers map the function of the human visual system. *NeuroImage*, 152:184–194, 2017.

[95] Örjan Ekeberg and Mikael Djurfeldt. MUSIC – Multisimulation Coordinator: Request For Comments. 2008.

[96] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[97] Jochen Martin Eppler. PyNEST: A convenient interface to the NEST simulator. 2(January):1–12.

[98] Steven K Essera, Paul A Merollaa, John V Arthura, Andrew S Cassidya, Rathinakumar Appuswamya, Alexander Andreopoulosa, David J Berga, Jeffrey L McKinstrya, Timothy Melanoa, Davis R Barcha, et al. Convolutional networks for fast energy-efficient neuromorphic computing. *Proc. Nat. Acad. Sci. USA*, 113(41):11441–11446, 2016.

[99] Chrisantha Fernando and Sampsa Sojakka. Pattern recognition in a bucket. In *European Conference on Artificial Life*, pages 588–597. Springer, 2003.

[100] John C Fiala and Kristen M Harris. Dendrite structure. *Dendrites*, 2:1–11, 1999.

[101] Chelsea Finn and Sergey Levine. Deep Visual Foresight for Planning Robot Motion. 2016.

[102] Răzvan V Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.

[103] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[104] Yves Frégnac. A re-examination of hebbian-covariance rules and spike timing-dependent plasticity in cat visual cortex in vivo. *Frontiers in synaptic neuroscience*, 2:147, 2010.

[105] Nicolas Frémaux, Henning Sprekeler, and Wulfram Gerstner. Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS computational biology*, 9(4), 2013.

[106] Uwe Frey, Richard GM Morris, et al. Synaptic tagging and long-term potentiation. *Nature*, 385(6616):533–536, 1997.

[107] Karl Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

[108] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[109] Kenneth Gaarder. Transmission of edge information in the human visual system. *Nature*, 212(5059):321–323, 1966.

[110] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *CoRR*, abs/1904.08405, 2019.

[111] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *CoRR*, abs/1904.08405, 2019.

[112] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. Focus is all you need: Loss functions for event-based vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12280–12289, 2019.

[113] Guillermo Gallego, Jon EA Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-based, 6-dof camera tracking from photometric depth maps. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2402–2412, 2017.

[114] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2018.

[115] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. *arXiv preprint arXiv:1904.08245*, 2019.

[116] Wulfram Gerstner, Richard Kempter, J Leo van Hemmen, and Hermann Wagner. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383(6595):76, 1996.

[117] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[118] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[119] Marc-Oliver Gewaltig and Markus Diesmann. Nest (neural simulation tool). *Scholarpedia*, 2(4):1430, 2007.

[120] Arren Glover and Chiara Bartolozzi. Event-driven ball detection and gaze fixation in clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2203–2208. IEEE, 2016.

[121] Dan FM Goodman and Romain Brette. The brian simulator. *Frontiers in neuroscience*, 3:26, 2009.

[122] X. Guan and L. Mo. Unsupervised conditional reflex learning based on convolutional spiking neural network and reward modulation. *IEEE Access*, 8:17673–17690, 2020.

[123] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *ELife*, 6:e22901, 2017.

[124] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[125] Robert Gütig, Ranit Aharonov, Stefan Rotter, and Haim Sompolinsky. Learning input correlations through nonlinear temporally asymmetric hebbian plasticity. *Journal of Neuroscience*, 23(9):3697–3714, 2003.

[126] Germain Haessig, Andrew Cassidy, Rodrigo Alvarez, Ryad Benosman, and Garrick Orchard. Spiking optical flow for event-based sensors using ibm's truenorth neurosynaptic system. *IEEE transactions on biomedical circuits and systems*, 12(4):860–870, 2018.

[127] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

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

[129] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Wiley Interscience, New York, 1949.

[130] A. Hermann, J. Sun, Z. Xue, S. W. Ruehl, J. Oberlaender, Arne Roennau, J. Marius Zoellner, and Ruediger Dillmann. Hardware and software architecture of the bimanual mobile manipulation robot HoLLiE and its actuated upper body. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics: Mechatronics for Human Wellbeing, AIM 2013*, number July, pages 286–292, 2013.

[131] Michael L Hines and Nicholas T Carnevale. The neuron simulation environment. *Neural computation*, 9(6):1179–1209, 1997.

[132] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[133] Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

[134] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[135] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[136] Yuhuang Hu, Hongjie Liu, Michael Pfeiffer, and Tobi Delbruck. Dvs benchmark datasets for object tracking, action recognition, and object recognition. *Frontiers in Neuroscience*, 10, 2016.

[137] Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 1433–1443, 2018.

[138] Chou P Hung, Gabriel Kreiman, Tomaso Poggio, and James J DiCarlo. Fast readout of object identity from macaque inferior temporal cortex. *Science*, 310(5749):863–866, 2005.

[139] Taras Iakymchuk, Alfredo Rosado-Muñoz, Juan F Guerrero-Martínez, Manuel Bataller-Mompeán, and Jose V Francés-Víllora. Simplified spiking neural network architecture and stdp learning algorithm applied to image classification. *EURASIP Journal on Image and Video Processing*, 2015(1):4, 2015.

[140] Intel. Intel announces neuromorphic research progress. https://newsroom.intel.com/news/intel-announces-neuromorphic-computing-research-collaborators, accessed March 16, 2020.

[141] Laxmi R Iyer, Yansong Chua, and Haizhou Li. Is neuromorphic mnist neuromorphic? analyzing the discriminative power of neuromorphic datasets in the time domain. *arXiv preprint arXiv:1807.01013*, 2018.

[142] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[143] Eugene M Izhikevich. Solving the distal reward problem through linkage of stdp and dopamine signaling. *Cerebral cortex*, 17(10):2443–2452, 2007.

[144] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

[145] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. pages 1–14, 2016.

[146] Herbert Jaeger. Echo state network. *Scholarpedia*, 2(9):2330, 2007.

[147] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[148] Zhuangyi Jiang, Zhenshan Bing, Kai Huang, and Alois C Knoll. Retina-based pipe-like object tracking implemented through spiking neural network on a snake robot. *Frontiers in neurorobotics*, 13:29, 2019.

[149] David Kappel, Stefan Habenschuss, Robert Legenstein, and Wolfgang Maass. Network Plasticity as Bayesian Inference. *PLOS Computational Biology*, 11(11):e1004485, nov 2015.

[150] David Kappel, Robert Legenstein, Stefan Habenschuss, Michael Hsieh, and Wolfgang Maass. A Dynamic Connectome Supports the Emergence of Stable Computational Function of Neural Circuits through Reward-Based Learning. *eneuro*, pages ENEURO.0301–17.2018, apr 2018.

[151] Nikola Kasabov, Kshitij Dhoble, Nuttapod Nuntalid, and Giacomo Indiveri. Dynamic evolving spiking neural networks for on-line spatio-and spectro-temporal pattern recognition. *Neural Networks*, 41:188–201, 2013.

[152] Salman Khan. The membrane potential, 2010.

[153] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, and Timothée Masquelier. Bio-inspired unsupervised learning of visual features leads to robust invariant object recognition. *Neurocomputing*, 205:382–392, 2016.

[154] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[155] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement Learning in Robotics: A Survey. 2009.

[156] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *International Conference on Intelligent Robots and Systems*, volume 3, pages 2149–2154. IEEE, 2004.

[157] Norbert Kruger, Peter Janssen, Sinan Kalkan, Markus Lappe, Ales Leonardis, Justus Piater, Antonio J. Rodriguez-Sanchez, and Laurenz Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? 35(8):1847–1871, 2013.

[158] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[159] Xavier Lagorce, Sio-Hoi Ieng, Xavier Clady, Michael Pfeiffer, and Ryad B Benosman. Spatiotemporal features for asynchronous event-based data. *Frontiers in neuroscience*, 9:46, 2015.

[160] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1346–1359, 2016.

[161] Subhaneil Lahiri and Surya Ganguli. A memory frontier for complex synapses. In *Advances in neural information processing systems*, pages 1034–1042, 2013.

[162] Trevor D. Lamb, Edward N. Pugh, and Shaun P. Collin. The origin of the vertebrate eye. *Evolution: Education and Outreach*, 1(4):415–426, Oct 2008.

[163] Michael F Land. Motion and vision: why animals move their eyes. *Journal of Comparative Physiology A*, 185(4):341–352, 1999.

[164] Michael F Land and Dan-Eric Nilsson. *Animal eyes*. Oxford University Press, 2012.

[165] Louis Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generale*, 9:620–635, 1907.

[166] John Lazzaro, John Wawrzynek, Misha Mahowald, Massimo Sivilotti, and Dave Gillespie. Silicon auditory processors as computer peripherals. In *Advances in Neural Information Processing Systems*, pages 820–827, 1993.

[167] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[168] Honglak Lee et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, pages 609–616, 2009.

[169] Robert Legenstein, Dejan Pecevski, and Wolfgang Maass. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLOS Computational Biology*, 4(10):1–27, 10 2008.

[170] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

[171] WB Levy and O Steward. Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus. *Neuroscience*, 8(4):791–797, 1983.

[172] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11:309, 2017.

[173] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128x128 120 db 15us latency asynchronous temporal contrast vision sensor. *journal of solid-state circuits*, 43(2):566–576, 2008.

[174] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7:13276, 2016.

[175] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. (July):1–14, 2015.

[176] Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 2020.

[177] John Lisman and Nelson Spruston. Questions about stdp as a general model of synaptic plasticity. *Frontiers in synaptic neuroscience*, 2:140, 2010.

[178] Hesheng Liu, Yigal Agam, Joseph R Madsen, and Gabriel Kreiman. Timing, timing, timing: fast decoding of object information from intracranial field potentials in human visual cortex. *Neuron*, 62(2):281–290, 2009.

[179] Margaret Livingstone and David Hubel. Segregation of form, color, movement, and depth: anatomy, physiology, and perception. *Science*, 240(4853):740–749, 1988.

[180] Iulia-Alexandra Lungu, Federico Corradi, and Tobi Delbrück. Live demonstration: Convolutional neural network driven by dynamic vision sensor playing roshambo. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–1. IEEE, 2017.

[181] Wolfgang Maass. Liquid state machines: Motivation, theory, and applications. *Computability in Context: Computation and Logic in the Real World*, pages 275–296, 2010.

[182] Wolfgang Maass. Searching for principles of brain computation. *Current Opinion in Behavioral Sciences*, 11:81–92, 2016.

[183] Wolfgang Maass, Robert Legenstein, and Henry Markram. A new approach towards vision suggested by biologically realistic neural microcircuit models. *Neural Computation*, 2525:282293, 2002.

[184] Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

[185] Stefano Mafrica. *Bio-inspired visual sensors for robotic and automotive applications*. PhD thesis, 2016.

[186] H Markram and B Sakmann. Action potentials propagating back into dendrites trigger changes in efficacy of single-axon synapses between layer v pyramidal neurons. In *Soc. Neurosci. Abstr*, volume 21, page 2007, 1995.

[187] Henry Markram, Wulfram Gerstner, and Per Jesper Sjöström. Spike-timing-dependent plasticity: a comprehensive overview. *Frontiers in synaptic neuroscience*, 4:2, 2012.

[188] Henry Markram, Joachim Lübke, Michael Frotscher, and Bert Sakmann. Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps. *Science*, 275(5297):213–215, 1997.

[189] David Marr. *Vision: a computational investigation into the human representation and processing of visual information*. W.H. Freeman and Company, San Francisco, 1982.

[190] David Marr and Tomaso Poggio. A Theory of Human Stereo Vision. *Proceedings of the Royal Society of London B: Biological Sciences*, 204:301—-328, 1977.

[191] Jacob G Martin, Charles E Davis, Maximilian Riesenhuber, and Simon J Thorpe. Zapping 500 faces in less than 100 seconds: Evidence for extremely fast and sustained continuous visual search. *Scientific reports*, 8(1):12482, 2018.

[192] Susana Martinez-Conde, Stephen L Macknik, and David H Hubel. The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*, 5(3):229–240, 2004.

[193] Timothée Masquelier. *Spike-based computing and learning in brains, machines, and visual systems in particular*. Habilitation à diriger des recherches, Université Toulouse III – Paul Sabatier, October 2017.

[194] Timothée Masquelier and Simon J Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Computational Biology*, 3(2):0247–0257, 2007.

[195] Colleen A McClung and Eric J Nestler. Neuroplasticity mediated by altered gene expression. *Neuropsychopharmacology*, 33(1):3, 2008.

[196] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[197] Carver A Mead and Misha A Mahowald. A silicon model of early visual processing. *Neural networks*, 1(1):91–97, 1988.

[198] Michel Millodot. Dictionary of optometry and visual science. 2009.

[199] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[200] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[201] Saber Moradi, Ning Qiao, Fabio Stefanini, and Giacomo Indiveri. A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). *IEEE transactions on biomedical circuits and systems*, 12(1):106–122, 2017.

[202] Hesham Mostafa, Vishwajith Ramesh, and Gert Cauwenberghs. Deep supervised learning using local errors. *Frontiers in neuroscience*, 12:608, 2018.

[203] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection. In *BMVC*, 2017.

[204] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In *2015 IEEE international conference on Robotics and Automation (ICRA)*, pages 4874–4881. IEEE, 2015.

[205] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440, 2018.

[206] Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based , 6-dof pose tracking for high-speed maneuvers. In *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2014.

[207] Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based, 6-dof pose tracking for high-speed maneuvers. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2761–2768. IEEE, 2014.

[208] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam. *The International Journal of Robotics Research*, 36(2):142–149, 2017.

[209] Farzan Nadim and Dirk Bucher. Neuromodulation of neurons and synapses. *Current opinion in neurobiology*, 29:48–56, 2014.

[210] Jonathan J Nassi and Edward M Callaway. Parallel processing strategies of the primate visual system. *Nature reviews neuroscience*, 10(5):360, 2009.

[211] Emre Neftci, Srinjoy Das, Bruno Pedroni, Kenneth Kreutz-Delgado, and Gert Cauwenberghs. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, 7(January):1–14, 2014.

[212] Emre O. Neftci. Data and Power Efficient Intelligence with Neuromorphic Learning Machines. *iScience*, 5:52–68, 2018.

[213] Emre O Neftci, Charles Augustine, Somnath Paul, and Georgios Detorakis. Event-driven random back-propagation: Enabling neuromorphic deep learning machines. *Frontiers in neuroscience*, 11:324, 2017.

[214] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *arXiv preprint arXiv:1901.09948*, 2019.

[215] Emre O. Neftci, Bruno U. Pedroni, Siddharth Joshi, Maruan Al-Shedivat, and Gert Cauwenberghs. Unsupervised Learning in Synaptic Sampling Machines. pages 1–27, 2015.

[216] Bernhard Nessler, Michael Pfeiffer, Lars Buesing, and Wolfgang Maass. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput Biol*, 9(4):e1003037, 2013.

[217] Danko Nikolić, Stefan Häusler, Wolf Singer, and Wolfgang Maass. Distributed fading memory for stimulus properties in the primary visual cortex. *PLoS biology*, 7(12):e1000260, 2009.

[218] Dan-E Nilsson. Eye evolution and its functional basis. *Visual neuroscience*, 30(1-2):5–20, 2013.

[219] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in neural information processing systems*, pages 1037–1045, 2016.

[220] Mohammad Norouzi, Mani Ranjbar, and Greg Mori. Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2735–2742. IEEE, 2009.

[221] Peter O'Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck, and Michael Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in neuroscience*, 7:178, 2013.

[222] University of Heidelberg. Brainscales. `https://flagship.kip.uni-heidelberg.de/public/BrainScaleS/index.html`, accessed March 16, 2020.

[223] Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[224] OpenAI. Ai and compute. `https://openai.com/blog/ai-and-compute/`, accessed March 16, 2020.

[225] Garrick Orchard, Ryad Benosman, Ralph Etienne-Cummings, and Nitish V Thakor. A spiking neural network architecture for visual motion estimation. In *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 298–301. IEEE, 2013.

[226] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

[227] Marc Osswald, Sio-hoi Ieng, Ryad Benosman, and Giacomo Indiveri. A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems. *Nature Publishing Group*, (January):1–11, 2017.

[228] Marc Osswald, Sio-hoi Ieng, Ryad Benosman, and Giacomo Indiveri. Supplementary Material: A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems. (c):1–14, 2017.

[229] Wei-Xing Pan, Robert Schmidt, Jeffery R Wickens, and Brian I Hyland. Dopamine cells respond to predicted events during classical conditioning: evidence for eligibility traces in the reward-learning network. *Journal of Neuroscience*, 25(26):6235–6242, 2005.

[230] Priyadarshini Panda and Kaushik Roy. Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 299–306. IEEE, 2016.

[231] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*, 6, 2017.

[232] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[233] Diane Pecher and Rolf A Zwaan. *Grounding cognition: The role of perception and action in memory, language, and thinking*. Cambridge University Press, 2005.

[234] José Antonio Pérez-Carrasco, Bo Zhao, Carmen Serrano, Begona Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward convnets. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2706–2719, 2013.

[235] Mihai Alexandru Petrovici. *Form Versus Function: Theory and Models for Neuronal Substrates*. Springer, 2016.

[236] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: opportunities and challenges. *Frontiers in neuroscience*, 12, 2018.

[237] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural computation*, 18(6):1318–1348, 2006.

[238] Ewa Piatkowska, Jurgen Kogler, Nabil Belbachir, and Margrit Gelautz. Improved cooperative stereo matching for dynamic vision sensors with ground truth evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 53–60, 2017.

[239] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. *IEEE Journal of Solid-State Circuits*, 46(1):259–275, 2010.

[240] Dimitri Probst, Wolfgang Maass, Henry Markram, and Marc-Oliver Gewaltig. Liquid computing in a simplified model of cortical layer iv: Learning to balance a ball. pages 209–216, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[241] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[242] Santiago Ramón y Cajal. The croonian lecture.—la fine structure des centres nerveux. *Proceedings of the Royal Society of London*, 55(331-335):444–468, 1894.

[243] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[244] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. Esim: an open event camera simulator. In *Conference on Robot Learning*, pages 969–982, 2018.

[245] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3857–3866, 2019.

[246] Pamela Reinagel. Information theory in the brain. *Current Biology*, 10(15):R542–R544, 2000.

[247] Fred Rieke and David Warland. *Spikes: exploring the neural code*. MIT press, 1999.

[248] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019, 1999.

[249] Martin Rolfs. Microsaccades: small steps on a long way. *Vision research*, 49(20):2415–2441, 2009.

[250] Edmund T Rolls and Gustavo Deco. *The noisy brain: stochastic dynamics as a principle of brain function*, volume 34. Oxford university press Oxford, 2010.

[251] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[252] Bodo Rueckauer and Tobi Delbruck. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers in neuroscience*, 10:176, 2016.

[253] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[254] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[255] João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems*, pages 8721–8732, 2018.

[256] Vieri Giuliano Santucci, Pierre-Yves Oudeyer, Andrew Barto, and Gianluca Baldassarre. Intrinsically motivated open-ended learning in autonomous robots. *Frontiers in Neurorobotics*, 13:115, 2020.

[257] Johannes Schemmel, Daniel Briiderle, Andreas Griibl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1947–1950. IEEE, 2010.

[258] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[259] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[260] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.

[261] Walter Senn and Jean-Pascal Pfister. Spike-timing dependent plasticity, learning rules. *Encyclopedia of Computational Neuroscience*, pages 2824–2832, 2015.

[262] Rafael Serrano-Gotarredona, Matthias Oster, Patrick Lichtsteiner, Alejandro Linares-Barranco, Rafael Paz-Vicente, Francisco Gómez-Rodríguez, Luis Camuñas-Mesa, Raphael Berner, Manuel Rivas-Pérez, Tobi Delbruck, et al. Caviar: A 45k neuron, 5m synapse, 12g connects/s aer hardware sensory–processing–learning–actuating system for high-speed visual object recognition and tracking. *IEEE transactions on neural networks*, 20(9):1417–1438, 2009.

[263] Teresa Serrano-Gotarredona and Bernabé Linares-Barranco. Poker-dvs and mnist-dvs. their history, how they were made, and other details. *Frontiers in neuroscience*, 9:481, 2015.

[264] Harel Z Shouval, Samuel S-H Wang, and Gayle M Wittenberg. Spike timing dependent plasticity: a consequence of more fundamental learning rules. *Frontiers in Computational Neuroscience*, 4:19, 2010.

[265] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, pages 1412–1421, 2018.

[266] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 440–449. ACM, 1992.

[267] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[268] Lawrence C Sincich and Jonathan C Horton. The circuitry of v1 and v2: integration of color, form, and motion. *Annu. Rev. Neurosci.*, 28:303–326, 2005.

[269] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, 2018.

[270] Per Jesper Sjöström, Gina G Turrigiano, and Sacha B Nelson. Rate, timing, and cooperativity jointly determine cortical synaptic plasticity. *Neuron*, 32(6):1149–1164, 2001.

[271] Marc A Sommer and Robert H Wurtz. Influence of the thalamus on spatial visual processing in frontal cortex. *Nature*, 444(7117):374, 2006.

[272] Bongki Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun-Seok Kim, Changwoo Shin, Keunju Park, Kyoobin Lee, Jinman Park, Jooyeon Woo, et al. A 640$\times$ 480 dynamic vision sensor with a 9μm pixel and 300meps address-event representation. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 66–67. IEEE, 2017.

[273] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[274] Terrence Stewart, Feng-Xuan Choo, and Chris Eliasmith. Spaun: A perception-cognition-action model using spiking neurons. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 34, 2012.

[275] Terrence C Stewart. A technical overview of the neural engineering framework. *University of Waterloo*, 2012.

[276] Terrence C Stewart, Ashley Kleinhans, Andrew Mundy, and Jörg Conradt. Serendipitous offline learning in a neuromorphic robot. *Frontiers in neurorobotics*, 10:1, 2016.

[277] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[278] G. Tatai and L. Gulyás. Neural network controlling architectures in autonomous agents. In *Proceedings of the Hungarian National Conference on Agent Based Computing*, 1998.

[279] Simon Thorpe, Denis Fize, and Catherine Marlot. Speed of processing in the human visual system. *nature*, 381(6582):520, 1996.

[280] Simon J Thorpe and Michele Fabre-Thorpe. Seeking categories in the brain. *Science*, 291(5502):260–263, 2001.

[281] Gabriel Urbain, Jonas Degrave, Benonie Carette, Joni Dambre, and Francis Wyffels. Morphological properties of mass–spring networks for optimal locomotion learning. *Frontiers in Neurorobotics*, 11:16, 2017.

[282] J. M. Valeton. Photoreceptor light adaptation models: An evaluation. *Vision Research*, 23(12):1549–1554, 1983.

[283] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based harris corner detection exploiting the advantages of event-driven cameras. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4144–4149. IEEE, 2016.

[284] John Von Neumann. *The computer and the brain*. Yale University Press, 2012.

[285] Philipp Weidel, Mikael Djurfeldt, Renato C. Duarte, and Abigail Morrison. Closed Loop Interactions between Spiking Neural Network and Robotic Simulators Based on MUSIC and ROS. *Frontiers in Neuroinformatics*, 10(31):1–19, aug 2016.

[286] David Weikersdorfer and Jörg Conradt. Event-based particle filtering for robot self-localization. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 866–870. IEEE, 2012.

[287] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[288] P. Wolf, C. Hubschneider, M. Weber, A. Bauer, J. Härtl, F. Dürr, and J. M. Zöllner. Learning how to drive in a real world simulation with deep q-networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 244–250, June 2017.

[289] Daniel M Wolpert, Zoubin Ghahramani, and J Randall Flanagan. Perspectives and problems in motor learning. *Trends in cognitive sciences*, 5(11):487–494, 2001.

[290] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356, 2016.

[291] Gavin C Young. Early evolution of the vertebrate eye-fossil evidence. *Evolution: Education and Outreach*, 1(4):427–438, 2008.

[292] A Yousefzadeh, T Serrano-Gotarredona, and B Linares-Barranco. Mnist-dvs and flash-mnist-dvs databases, 2015.

[293] Zhaofei Yu, David Kappel, Robert Legenstein, Sen Song, Feng Chen, and Wolfgang Maass. CaMKII activation supports reward-based neural network optimization through Hamiltonian sampling. jun 2016.

[294] Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10(1):1–7, 2019.

[295] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *IEEE Transactions on Robotics*, 2020.

[296] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[297] Friedemann Zenke and Wulfram Gerstner. Limits to high-speed simulations of spiking neural networks using general-purpose computers. *Frontiers in neuroinformatics*, 8:76, 2014.

[298] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017.

[299] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Ev-flownet: Self-supervised optical flow estimation for event-based cameras. *arXiv preprint arXiv:1802.06898*, 2018.

[300] Marc Zirnsak, Nicholas A Steinmetz, Behrad Noudoost, Kitty Z Xu, and Tirin Moore. Visual space is compressed in prefrontal cortex before eye movements. *Nature*, 507(7493):504, 2014.

[301] M. R. Zofka, F. Kuhnt, R. Kohlhaas, and J. M. Zöllner. Simulation framework for the development of autonomous small scale vehicles. In *Int. Conf. on Simulation, Modeling, and Programming for Autonomous Robots*, pages 318–324. IEEE, Dec 2016.