Unsupervised Learning of Phase-Change-Based Neuromorphic Systems

THÈSE N^{O} 8129 (2017)

PRÉSENTÉE LE 15 DÉCEMBRE 2017
À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE SYSTÈMES MICROÉLECTRONIQUES
PROGRAMME DOCTORAL EN MICROSYSTÈMES ET MICROÉLECTRONIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Stanislaw Andrzej WOZNIAK

acceptée sur proposition du jury:

Prof. C. Guiducci, présidente du jury
Prof. Y. Leblebici, Dr A. Pantazi, directeurs de thèse
Prof. G. Indiveri, rapporteur
Dr E. Eleftheriou, rapporteur
Prof. P. Ienne, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2017

Dedicated to my parents and my wife.

Abstract

Neuromorphic systems provide brain-inspired methods of computing. In a neuromorphic architecture, inputs are processed by a network of neurons receiving operands through synaptic interconnections, tuned in the process of learning. Neurons act simultaneously as asynchronous computational and memory units, which leads to a high degree of parallelism. Furthermore, owing to developments in novel materials, memristive devices were proposed for area- and energy-efficient mixed digital-analog implementation of neurons and synapses. In this dissertation, we propose neuromorphic architectures based on phase-change memristors combined with biologically-inspired synaptic learning rules, and we experimentally demonstrate their pattern- and feature-learning capabilities.

Firstly, by exploiting the physical properties of phase-change devices, we propose neuromorphic building blocks comprising phase-change-based neurons and synapses operating according to an unsupervised local learning rule. At the same time, we introduce multiple enhancements for pattern learning: an integration threshold for the phase-change soma to ensure noise-robust operation; selective synaptic depression mechanism to limit negative impact of asymmetric conductance response of phase-change synapses during the learning; WTA (Winner-Take-All) mechanism with level-tuned neurons that decreases power consumption in comparison to the classic lateral inhibition WTA; and learning WTA that enhances the quality of pattern visualization. Experimental results demonstrate the capabilities of the proposed architectures. In particular, a neuron with phase-change synapses was shown to learn and re-learn patterns of correlated activity. Furthermore, an all-phase-change neuron with a record number of 1M synapses successfully detected and visualized weakly-correlated patterns. Lastly, a network of all-phase-change neurons operating with level-tuned neurons accurately learned multiple patterns.

Secondly, to scale-up the proposed architectures, we identify the need to improve the knowledge representation to learn features rather than patterns. We determine the key role of the feedback links for controlling the learning process, and combine intraneuronal with interneuronal feedback. Intraneuronal feedback determines what each neuron learns, whereas interneuronal feedback determines how information is distributed between the neurons. We propose two feature-learning architectures: an architecture with interneuronal feedback to the learning rule, and an architecture inspired by the biological observation of synaptic competition for learning-related proteins. Furthermore, we introduce a model of synaptic

competition that guides the learning as well as detects novelty in the input, which is then used to dynamically adjust the size of the network. In a series of benchmarks for different feature types, synaptic competition outperformed other common methods, simultaneously adjusting the network to the optimal size. Finally, it was the only method that succeeded for a challenging dataset that violates the common machine learning assumption on the independent and identically-distributed input presentation.

To conclude, we proposed phase-change-based neuromorphic architectures and we realized them in a large-scale prototype platform. The experimental results demonstrate pattern- and feature-learning capabilities and constitute an important step towards designing unsupervised online learning neuromorphic systems.

Keywords: neuromorphic systems, phase-change memristors, spiking neural networks, WTA, STDP, correlation detection, unsupervised online learning, feature extraction, independent components, synaptic competition.

Zusammenfassung

Neuromorphe Systeme stellen ein vom Gehirn inspiriertes Berechnungsmodell dar. In einer neuromorphen Architektur werden Eingaben durch ein Netzwerk von Neuronen bearbeitet und die Verarbeitungslogik wird durch gelernte synaptische Verbindungen repräsentiert. Neuronen dienen gleichzeitig als asynchrone Rechnen- und Speichereinheiten, was zu einem hohen Grad an Parallelität führt. Durch die Entwicklung neuartiger Materialien konnten memristive Flächen- und Energie-effiziente digital-analoge Implementierungen von Neuronen und Synapsen entwickelt werden. In dieser Dissertation, schlagen wir Phasenwechsel-Memristoren-basierte neuromorphe Architekturen mit biologisch inspirierten Lernregeln vor, und wir zeigen in Experimenten deren Fähigkeiten, Muster und Merkmale zu lernen.

Unter Ausnutzung der physikalischen Eigenschaften von Phasenwechselzellen schlagen wir zunächst neuromorphe Bausteine vor, die aus Phasenwechsel-Neuronen und Synapsen bestehen, die mit einer lokalen unbeaufsichtigten Lernregel betrieben werden. Gleichzeitig stellen wir mehrere Verbesserungen für das Musterlernen vor: ein Phasenwechsel-Soma mit Integrationsschwelle, das robust gegen Rauschen ist; ein selektiv synaptisches Depressionsverfahren zur Begrenzung der negativen Auswirkungen der asymmetrischen Leitfähigkeitsreaktion von Phasenwechsel-Synapsen; ein Winner-Take-All (WTA)-Mechanismus mit Ebenen-abgestimmten Neuronen, das den Energieverbrauch verringert; und Lernen-WTA, welches die Qualität der Mustervisualisierung verbessert. Experimentelle Ergebnisse zeigen die Fähigkeiten der vorgeschlagenen Architekturen. Insbesondere wurde ein Neuron mit Phasenwechsel-Synapsen verwendet, um Muster korrelierter Aktivität zu lernen. Weiterhin wurde ein Phasenwechsel-Neuron mit einer Rekordzahl von 1M Synapsen gezeigt, das erfolgreich schwach korrelierte Muster erkennen und visualisieren konnte. Schließlich lernte ein Netzwerk von vollständig Phasenwechsel- und Ebenen-abgestimmten Neuronen mehrere Muster.

Zweitens zeigen wir im Hinblick auf die Skalierung der vorgeschlagenen Architekturen, dass es notwendig ist, die Wissensrepräsentation zu verbessern, um Merkmale anstatt Muster zu lernen. Wir stellen die Schlüsselrolle von Rückkopplungen zur Steuerung des Lernprozesses heraus, und wir kombinieren Intra- mit Inter-neuronaler Rückkopplung. Interneuronale Rückkopplung bestimmt, was jedes Neuron lernt, während intraneuronale Rückkopplung bestimmt, wie die Information zwischen den Neuronen verteilt ist. Wir stellen zwei Architekturen für das Merkmals-Lernen vor: eine mit interneuronaler Rückkopplung zur Lernregel

und eine, die durch biologische Beobachtung der synaptischen Konkurrenz lernbezogener Proteine inspiriert ist. Weiterhin stellen wir ein Modell der synaptischen Konkurrenz für das Lernen vor, das basierend auf Neuheiten in den Eingaben die Netzwerkgröße dynamisch anpasst. In einer Reihe von Benchmarks zeigen wir, dass die synaptische Konkurrenz andere Methoden übertrifft, wobei gleichzeitig das Netzwerk auf die optimale Größe angepasst wurde. Schließlich war sie die einzige Methode, die für einen anspruchsvollen Datensatz erfolgreich war, der nicht unabhängige und gleich verteilte Eingaben hat.

In Zusammenfassung, schlagen wir Phasenwechsel-basierte neuromorphe Architekturen vor und implementierten sie als Prototyp. Die experimentellen Resultate zeigen Muster- und Merkmals-Lernen und sind ein wichtiger Schritt im Entwurf von unüberwachtem Online-Lernen in neuromorphen Systemen.

Stichwörter: neuromorphe Systeme, Phasenwechsel-Memristoren, gepulste neuronale Netze, WTA, STDP, Korrelationserkennung, unüberwachtes Online-Lernen, Merkmal-Extraktion, unabhängige Komponente, synaptische Konkurrenz.

Contents

List of Figures

List of Tables

1 Introduction

Neuromorphic systems are computing systems whose design takes inspiration from the brain [Mead, 1990]. The aim of neuromorphic design is to provide an alternative to the conventional von Neumann computing architecture. Although von Neumann-based machines excel at fast and accurate solving of well-defined mathematical problems and algorithms, they remain in stark contrast to humans in terms of cognitive capabilities. A human brain is able to perform complex cognitive tasks, while consuming $\approx 20\,\mathrm{W}$ and occupying $\approx 1500\,\mathrm{cm}^3$. With the advance of technology, von Neumann computers matched human capabilities in areas traditionally associated with higher intelligence such as Chess (IBM DeepBlue in 1996), Jeopardy (IBM Watson in 2011) or Go (DeepMind AlphaGo in 2017). Nevertheless, this trend applies to a limited group of tasks that require specially-crafted algorithms and often run on supercomputers consuming significant amount of energy: IBM Watson used 80 kW for playing Jeopardy. Therefore, it is appealing to explore the operating principles of the brain to develop neuromorphic systems that could complement the existing von Neumann computers by efficiently processing cognitive workloads.

In this chapter, we first provide a condensed top-level overview of the current understanding of the brain and describe it in relation to neural network modeling and learning. In Sec. 1.2, we discuss different principles behind the neuromorphic systems design and its impact on other research fields. Then, in Sec. 1.3, we state the research objectives, the scope of the thesis, and we list the publications stemming from this work.

1.1 Brain-inspired computing

In neurobiology, the brain is studied at various levels of abstraction, from the dynamics of chemical processes to the emergence of cognitive functions [Woźniak et al., 2015], as schematically illustrated on five abstraction levels [Markram, 2012] in Fig. 1.1. On the molecular level, the brain is a complex assembly of components. From computational perspective, *synapses* are of particular importance as they provide means for electrical and chemical transmission of information. On the cellular level, the basic information processing units are nervous cells

called *neurons*. A neuron may be divided into the following parts: dendrites, cell soma and an axon, that respectively collect, process, and emit pieces of information. The information is transmitted in form of voltage spikes conveyed between the neurons through the synapses. Groups of interconnected neurons form neural circuits, called *neural networks*, which provide more complex computational capabilities. Large neural networks, connected to other networks or nerves from the senses, form brain regions that specialize in different cognitive functions, such as vision (visual cortex), memory (hippocampus) or emotions (amygdala). Interactions between the regions on the level of the whole organ give rise to higher cognitive functions, such as behavior planning, abstract thinking, or general intelligence.

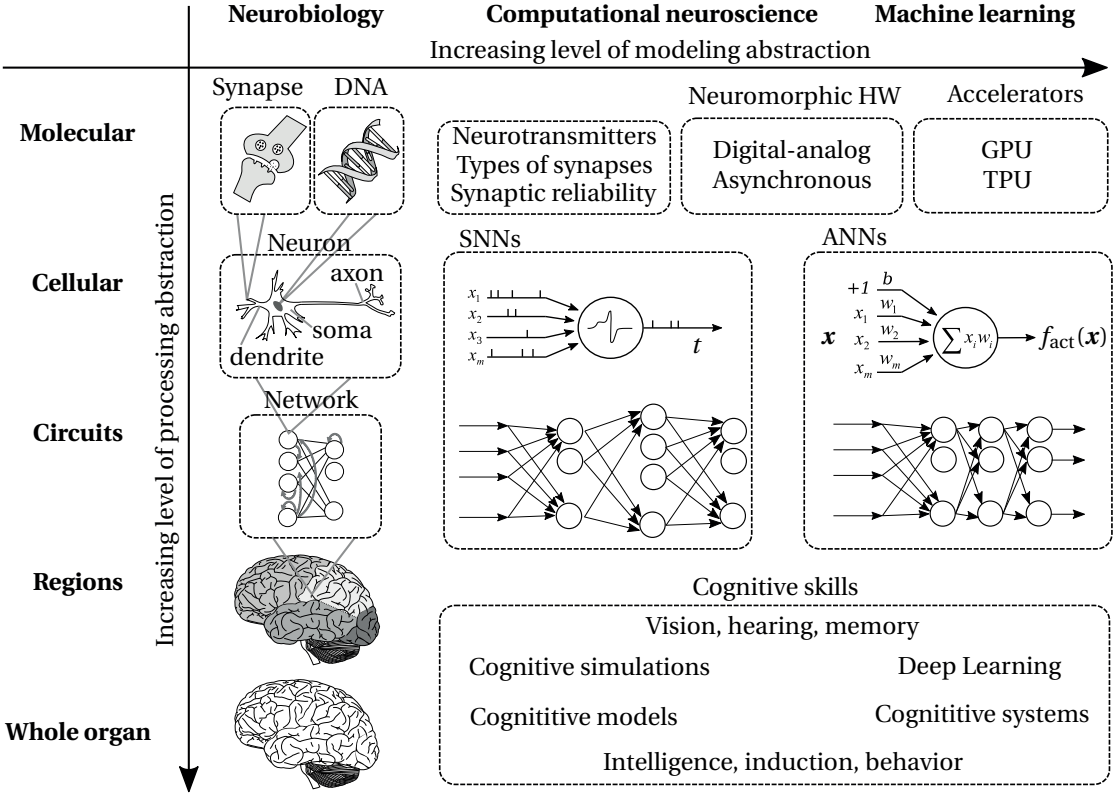


Figure 1.1: **Abstraction levels for studying the brain** Vertical direction: the operation of the brain is analyzed at various scales that correspond to different levels of processing abstractions in the brain. Horizontal direction: brain-inspired models are developed with a varying level of modeling abstraction.

Computational capabilities of the brain are modeled in different disciplines at a varying level of detail. In computational neuroscience, the focus is to understand the information processing in the brain. Research-oriented models are developed on each level of abstraction to search for important factors that could improve our understanding of how the brain operates. For instance, on the molecular level, researchers analyze the impact of neurotransmitters on the learning [Izhikevich, 2006], or the reliability of the synapses [Harris et al., 2012]. On the circuits and the cellular level, the activity of the neurons and the evolution of their synaptic

weights is studied using biologically-feasible abstraction of a Spiking Neural Network (SNN) model [Gerstner et al., 2014]. On higher levels, basic cognitive skills are modeled using top-down mathematical models of cognition, which may be also mapped to the SNN framework [Eliasmith et al., 2012].

Contrary to the brain-oriented research in computational neuroscience, in machine learning the focus is on applications. The main model is an Artificial Neural Network (ANN), which has some degree of structural resemblance to the brain structure on the circuits and the cellular level, but limited functional resemblance in terms of the execution model and the learning mechanisms. Biological details are of little importance: rather than taking inspiration from molecular details to enhance the models, the effort in ANNs is put into speeding up the execution by matching their structure to general-purpose (GPUs) or special-purpose (TPUs [Jouppi et al., 2017]) accelerators. Nevertheless, the machine learning approach to neural networks modeling developed in the field of deep learning led to many important and practically applicable models. Furthermore, recently it was shown to implement basic cognitive functions, such as generating textual captions for images [Vinyals et al., 2015], or solving question and answering tasks [Weston et al., 2014]. Owing to this rapid progress, the notion of *neural networks* in computer science and engineering is nowadays often identified with the ANN model.

Lastly, neuromorphic computing is a research field that is in-between computational neuroscience and machine learning in terms of the modeling abstraction. Its focus is on applications, but the significance of the computational aspects in the brain is not underestimated. In comparison to the brain, ANNs still require significantly more power, in particular when it comes to learning that is often executed on supercomputers. Neuromorphic computing postulates to go beyond the accelerators and to re-think the computing architectures based on the insights from computational neuroscience. It also focuses on exploring more efficient and flexible learning approaches by using SNNs coupled with biologically-inspired learning mechanisms.

1.1.1 Designing neural networks

The SNN and ANN classes comprise a large variety of models, architectures and techniques, which may often be reused for both classes. Therefore, we propose to visualize them similarly to a network protocol stack in a *technology stack of neural network design* [Woźniak et al., 2015] presented in Fig. 1.2. We discuss now briefly the design choices on each layer of the stack.

Initially, the neuron type is chosen from the ANN or the SNN class. Within the ANN class, a basic binary neuron was first introduced in 1943 [McCulloch and Pitts, 1943] in form $y = f_{\text{act}}(\boldsymbol{wx})$, where f_{act} is a logical threshold function. The threshold value is often called a *bias* and denoted with b. Nowadays, an explicit formulation with the bias is typically used: $y = f_{\text{act}}(\boldsymbol{wx} + b)$, and the activation function is often a sigmoid function: $f_{\text{act}}(z) = \frac{1}{1+e^{-z}}$, or a piece-wise linear (rectified linear) function: $f_{\text{act}}(z) = (z)^+$, where $(z)^+ = \max(0, z)$. Within

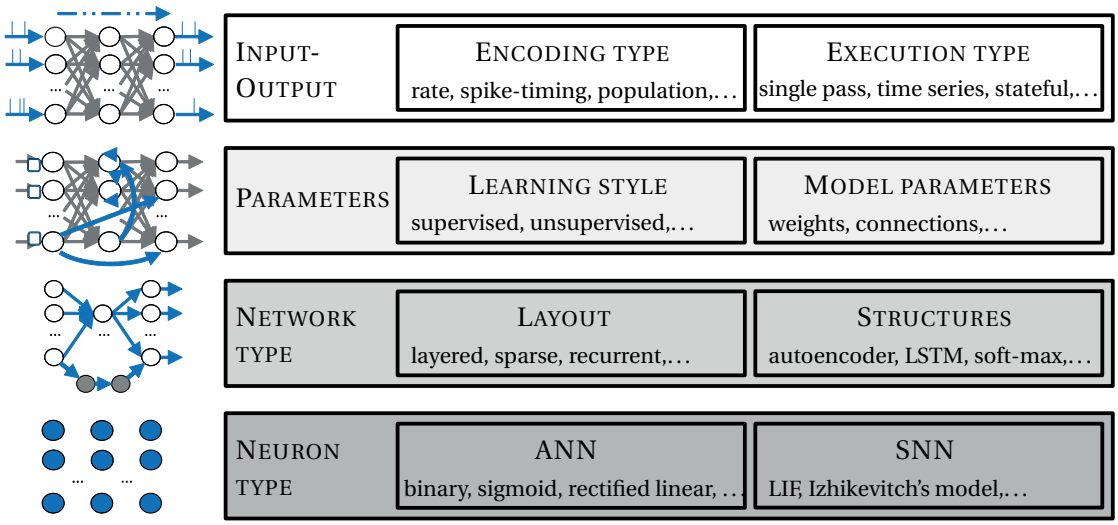


Figure 1.2: **Technology stack of neural network design** There are many neural network modeling techniques and design choices that build on top of both ANNs and SNNs. Adapted from [Woźniak et al., 2015]. Reprinted with permission of Springer.

the SNN class, many models were proposed, inspired by the description of the neuronal membrane dynamics provided in 1952 [Hodgkin and Huxley, 1952]. These models may be classified into various classes incorporating different biological phenomena [Gerstner et al., 2012]. Nevertheless, the most popular implementation is the Leaky Integrate-and-Fire (LIF) neuron. It is a basic approximation of the behavior of a biological neuron, in which when information collected from the dendrites surpasses a given threshold, a spike of voltage is emitted on the axon. This simple approach, extended with a moving threshold, won the INCF competition by predicting 60% of the neuronal activity recorded from real neurons [Gerstner and Naud, 2009]. It was also shown to be a more general model than the ANN neuron, as the ANN model may be considered as an averaged stateless approximation of the SNN model under certain additional assumptions [Dayan and Abbott, 2005]. Although the SNN neurons are more expressive, the ANN neurons are more popular, because they are simple to simulate and provide computational capabilities sufficient for many practical applications.

The next design choice is to determine the aspects of the network. The most common approach is to have a layered feed-forward neural network, with full connectivity between adjacent layers, forming a directed acyclic graph. If a network exhibits loops, the model becomes a recurrent neural network (RNN). An RNN can be considered equivalent to a feed-forward network, if we unroll it in time into a network that has one layer for each time step [Minsky and Papert, 1969] [Rumelhart et al., 1985]. Next, additional structural motifs are often introduced to precondition the network for particular tasks. For instance, spatial invariance may be provided by convolutions, which are groups of weights forming filters that are moved around the input; or soft-max structures may be used to provide class assignment probability estimates

for classification problems by arranging neurons into groups and normalizing their activities using the soft-max function.

Then, the parameters of the network should be defined. The most commonly tuned parameters are the synaptic weights and the biases of the neurons, but in general all aspects of the design may be treated as parameters [Almási et al., 2016]. This includes the network structure and the types of the neurons. The neurons may be added or removed, and the computational properties of a neuron may change, all of which may be tuned for instance using evolutionary algorithms [Ferreira, 2006] or other learning approaches that will be discussed in Sec. 1.1.2.

Lastly, input-output signals should be structured according to the particular design and the characteristics of the task. In ANNs, the inputs are vectors of floating-point values. In SNNs, they are streams of dirac pulses representing voltage spikes. A temporal signal may be passed to a stateless ANN as a series of input snapshots if the state in not relevant for the task, or otherwise as an input snapshot accompanied by a moving window of the input history. Alternatively, a stateful SNN model may be applied for such task. The inputs in SNNs may convey information in various forms, such as the rate or the timing [Borst and Theunissen, 1999], discussed in detail in Chapter 2.

1.1.2 Learning

A neural network model is just a meta-model for a particular task it is designed for. It is not useful until the model of that particular task is reflected in the values of the network parameters. The appropriate values of the parameters may be inferred from a set of examples, called the training set, in the process of learning. Therefore, learning may be perceived as a data-driven process of delivering feedback to the model parameters. The learning results depend on the quality of the training data, but also on how well-directed is the learning feedback:

- **undirected** – learning through evolution. In this case, random changes happen anywhere in the model in any direction, and their fitness is assessed through the global performance of the entire model. Such learning occurs on the molecular level of the organisms in the DNA, and the global feedback is the survival of the organism. Undirected feedback delivers typically low amount of feedback (survival) at a low rate (lifetime).
- **directed** – following gradual steps along a particular direction. The ability to determine the direction in which each parameter should be changed when feedback is received enables much more rapid learning. For instance, the ANN backpropagation algorithm [Hinton, 1986] estimates the error-minimizing gradients for all the parameters each time feedback is received. For this task the algorithm relies firstly on a differentiable network model, and secondly on neurons with monotonous activation functions. Consequently, all the model parameters are adjusted proportionally to their contribution to the error multiplied by a learning rate α. In practice, learning using backpropagation is faster than learning through evolution by a factor of the number of the model parameters.

- **exact** – providing values from an analytic solution. Closed-form solutions providing the optimal model parameters may be used for some particular neural models proposed within the framework of reservoir computing [Lukoševičius and Jaeger, 2009] and Extreme Learning Machines (ELMs) [Almási et al., 2016]. In ELMs the intermediate network layers comprise non-linear neurons with random weights and the last layer comprises linear output neurons with parameters calculated using a closed-form solution. Furthermore, instant exact learning is also an active area of research in deep learning, known under the term of *one-shot learning*. It may be implemented through combination of a gradually learning network with an instantly learning external memory [Graves et al., 2016].

Based on the source of the feedback, learning approaches are commonly classified as [Almási et al., 2016]:

- **supervised** – extrinsic error calculation. The direction of the parameter changes is determined based on the error stemming from feedback delivered from outside of the model. The training dataset $D = \{(\boldsymbol{x}^{(k)}, y^{(k)})\}$ comprises an explicit split into the input vectors $\boldsymbol{x}^{(k)}$, annotated with respective target values $y^{(k)}$, called the *labels*. The labels are compared to the activations of the system $f(\boldsymbol{x}^{(k)})$ to minimize a supervised loss function $L_s(f(\boldsymbol{x}^{(k)}), y^{(k)})$. The amount of feedback depends on the size of the dataset D. The rate of feedback is proportional to the number of different labels that typically determines the dimensionality of the output layer of the network.
- **unsupervised** – intrinsic error calculation. Labeling is usually a manual and time-consuming process, so most of the generally available data is not labeled. For unlabeled datasets, the direction of parameter changes may be directed through the error calculated by the network itself based on the input vector $\boldsymbol{x}^{(k)}$ only. For instance, a network may minimize a reconstruction loss $L_u(f(\boldsymbol{x}^{(k)}), g(f(\boldsymbol{x}^{(k)})))$, which may be understood as learning an association or correlation between the input attributes. The amount of feedback depends on the size of the dataset D. The rate of feedback is proportional to the variability of the input vectors $\boldsymbol{x}^{(k)}$ that is typically larger than of the labels $y^{(k)}$. On the other hand, the feedback from the input is not "distilled" to the most meaningful signal for solving a particular task.

  Supervised learning may be formulated as a special case of unsupervised learning. Assuming that the unsupervised input is $\boldsymbol{x}_u^{(k)} := (\boldsymbol{x}_s^{(k)}, y_s^{(k)})$, let the $y_s^{(k)}$ be passed through f_u in an unchanged form: $f_u((\boldsymbol{x}_s^{(k)}, y_s^{(k)})) = (f(\boldsymbol{x}_s^{(k)}), y_s^{(k)})$, and let g_u return only the labels $g(f(\boldsymbol{x}_s^{(k)}), y_s^{(k)}) = y_s^{(k)}$. Then, it is possible to define an unsupervised loss: $L_u((a,b),c) := L(a,c)$, so that the unsupervised learning loss becomes equivalent to the supervised learning loss: $L_u(f_u(\boldsymbol{x}_u^{(k)}), g_u(f_u(\boldsymbol{x}_u^{(k)}))) = L_s(f(\boldsymbol{x}_s^{(k)}), y_s^{(k)})$.
- **reinforcement** – delayed feedback from the environment. Reinforcement learning is based on the idea of having an agent interact with its environment. The agent chooses an action that yields delayed probabilistic feedback in the form of rewards from the environment. The goal of the learning is to maximize the total reward function. Using

these principles, an interesting result was obtained in [Mnih et al., 2013], where a deep network learned to play Atari2600 games using as input what is visible on the video screen, along with the reward information on the points scored in the game.

Based on the spatial extent of the feedback, learning approaches may also be classified as [Almási et al., 2016]:

- **global** – parameter changes require deliberate invocation of external routines, as in backpropagation or genetic algorithms. Typically such algorithms take as an input a complete stable state of the network, which cannot be executed until the routines have finished processing the feedback and adjusting the model parameters.
- **local** – neurons and synapses modify themselves. Routines adjusting the parameters are built into each part of the network, which enables them to change independently from a global state. Such an approach is suspected to be the base of learning in the brain, although it might also include a global signaling mechanism through neuromodulators. For instance, dopamine alters the learning in the entire brain within an opportunity window of 15-25 seconds after its release [Izhikevich, 2006].

Lastly, we may consider the temporal extent of the feedback stemming from the learning [Almási et al., 2016]:

- **offline** – the training phase is separate from the model execution. Parameters are determined so as to comply with the problem knowledge available at a certain point in time, and then they stay fixed during the use of the model. Most ANNs trained with the backpropagation algorithm use this approach.
- **online** – there is no clear boundary between the training phase and the model execution. This is common to biologically inspired local learning algorithms. It is also related to unsupervised learning as the data observed during execution is typically unlabeled.

Offline learning is suited for applications, in which the ground truth does not change. For instance, in character recognition, the characters are expected to remain the same over time, even if new words appear. However, in many problems, such as spam classification, or recommender systems, accuracy will be reduced over time, as the messages, or user preferences, evolve and concepts drift away from their initial definitions.

1.2 Towards neuromorphic systems

Taking inspiration from the brain may provide capabilities beyond the conventional von Neumann architecture that can be classified into two categories: structural and functional. The structural aspects include the type of the architecture and its area and energy efficiency. The functional aspects include the ability to autonomously solve complex cognitive tasks that may include executing loosely-formulated tasks, processing unstructured data, and learning from scarce unpreprocessed examples.

From the structural point of view, the basic blocks of the von Neumann architecture include a computational unit connected to the memory. This architecture is increasingly challenged by the limited throughput between the physically separated memory and computing unit. In contrast, neurons in the brain perform operations asynchronously, and synapses directly receive relevant operands from a network of synaptic interconnections, as schematically illustrated in Fig. 1.3a. Furthermore, chips are designed as planes, whereas the brain is a dense 3D structure. In consequence, simulating 5 s of brain activity would take around 500 s and need 1.4 MW of power if the state-of-the-art supercomputers are used [Modha et al., 2011]. Focusing on the structural aspects leads to the development of architectures such as massively parallel accelerators or computational memory [Gallo et al., 2017]. The idea of computational memory is to enrich the memory with computational capabilities, so that the basic operations are directly executed within it. The computational unit may send only the control signals to reduce the load on the bus. This can enhance the existing functionality of the von Neumann architecture, while maintaining compatibility with current computing systems.

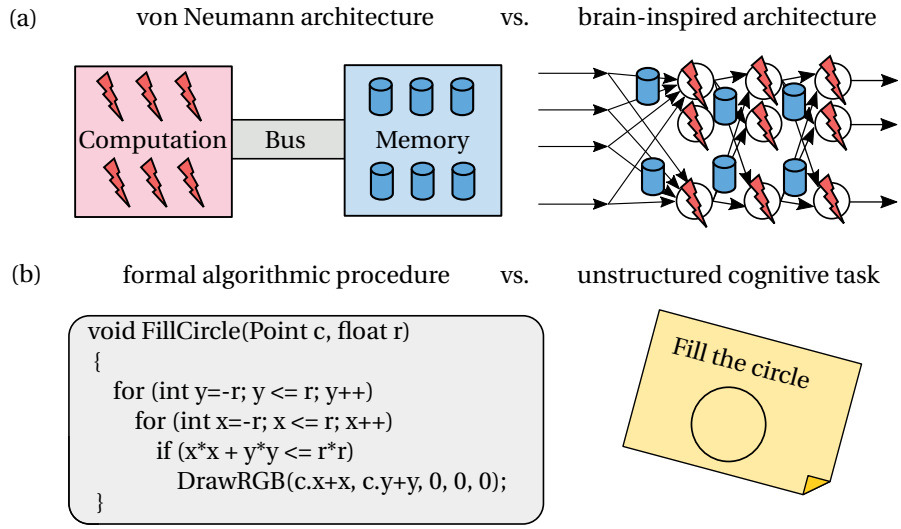


Figure 1.3: **Taking inspiration from the brain** (a) Structural aspects impact the area and energy efficiency. (b) Functional aspects determine the kind of tasks that may be handled.

From the functional point of view, von Neumann computers are suited to the formulation of the problems as mathematical and numerical procedures that involve quite long sequences of base operations, schematically illustrated in Fig. 1.3b. These base operations, such as multiplication or division, have a complex logical character which is *"obscured by our long and almost instinctive familiarly with them"*, as noted in [Von Neumann, 1958]. Because the errors are superimposed during each operation, digital systems with high precision are required to execute such programs. In contrast, the brain operates in a mixed digital-analog manner that provides a variety of base operations. Solving cognitive tasks often involves processing ambiguous inputs and autonomously seeking a context-dependent interpretation that does not have to rely on the strict logic and high-precision arithmetic. John von Neumann argued that: *"When we talk mathematics, we may be discussing a secondary language, built on the*

primary language truly used by the central nervous system." [Von Neumann, 1958] Although the exact understanding of the computational logic of the brain is still an open research topic, it is commonly reduced to applying the SNN or the ANN model. Owing to the algorithmic advances, robust functional models of ANNs may obtain higher accuracy than the traditional mathematical models. For instance, in case of speech recognition, recurrent ANNs [Graves et al., 2013] surpassed the state-of-the-art hidden Markov models.

Lastly, neuromorphic systems consider both the structural and the functional aspects of the design. Several approaches have been explored to develop the structures for neuromorphic architectures using mixed analog-digital technologies [Benjamin et al., 2014] [Qiao et al., 2015], microprocessor based systems [Schemmel et al., 2010] [Furber et al., 2013], large-scale networks of neurosynaptic core chips [Akopyan et al., 2015], and memristor-based neuromorphic circuits [Kim et al., 2015] [Wang et al., 2016]. These structures are often combined with the functional model of an SNN, which is particularly predestined for an efficient mixed digital-analog implementation. The asynchronous communication using all-or-none spikes enables noise-robust implementation in form of voltage spikes that may be send reliably over long distances between the neurons. Simultaneously, an analog implementation of the synapses and the neuron soma is much more efficient than digital design using transistors [Kuzum et al., 2012]. Moreover, owing to developments in the field of nanomaterials, further benefits in area efficiency and power dissipation can be gained by using memristive devices for the neuro-synaptic realization [Serrano-Gotarredona et al., 2013]. Specifically, neurons and synapses can be implemented using memristive circuit elements in which the device conductance represents the potential in the neuron soma [Tuma et al., 2016b] [Pantazi et al., 2016] or the synaptic weight [Kuzum et al., 2012] [Burr et al., 2014] [Garbin et al., 2015] [Woźniak et al., 2016] [Sidler et al., 2017] [Eryilmaz et al., 2014] [Kim et al., 2015].

1.2.1 The impact of neuromorphic research

Neuromorphic systems research is a multidisciplinary field that involves designing hardware architectures, advancing learning algorithms' development and understanding the aspects of computational neuroscience. Therefore, intermediate results from the neuromorphic systems research may contribute to these disciplines.

From the hardware design perspective, the technological challenges with further scaling of the clock frequencies and reducing the sizes of transistors require to follow alternative paths. Taking inspiration from the structural aspects of the brain is one of such paths, that may provide additional computational capabilities while increasing the energy-efficiency. Advancements in the neuromorphic systems may lead to incorporation of neuromorphic co-processors in the common computing architectures, or even fully replace them for particular cognitive applications.

From the learning algorithms perspective, development of unsupervised learning algorithms, such as the ones in the brain, is essential for the progress of machine learning. Currently,

most of the learning systems are trained using offline supervised methods, whereas humans primarily learn in an online unsupervised manner. Prof. Yann LeCun provides a following comparison: *"If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we don't know how to make the cake."* Taking inspiration from the functioning of the brain may advance the unsupervised learning capabilities, and enable novel use cases that could utilize the massive amounts of unstructured and unlabeled data all around us.

From the computational neuroscience perspective, building neuromorphic systems provides validation of the neuroscientific models. Information about what works in neuromorphic systems suggests what may be the next important neuroscientific research direction. Building neuromorphic models provides a feedback loop between the two disciplines: it enables to verify the utility of neuroscientific hypotheses, which then contribute to development of novel neuromorphic architectures.

1.3 Overview of the thesis

In this section, we discuss the focus of the thesis, present its outline and list the resulting publications. Lastly, brief remarks on the notation are provided.

1.3.1 Research objective and strategy

The research objective of the thesis is to propose mixed analog-digital neuromorphic architectures based on the SNN model. In particular, the work aims at providing an efficient hardware implementation of neuromorphic systems for which we utilize phase-change memristors. Simultaneously, it aims at learning meaningful knowledge from the inputs in an online unsupervised manner. Here, the focus is set on developing biologically-inspired local learning rules to implement pattern- and feature-learning capabilities.

The research strategy followed in the thesis involves multiple iterations between the structural and the functional aspects of the neuromorphic design. Therefore, we call our approach an *algorithmic and hardware co-design*. We start from well-established solutions: a single-layered SNN model on the algorithmic front, and phase-change technology on the hardware front. Then, we combine the two to obtain building blocks of a mixed digital-analog neuromorphic system. We follow a pragmatic approach, in which we are willing to accept trade-offs between the hardware design and the faithfulness of the SNN model implementation, in order to limit the hardware complexity. To develop online unsupervised learning with local implementation, we seek inspiration in well-established approaches: biologically-inspired local learning rules, and commercially successful deep learning ANNs. Based on obtained insights, we explore possibilities to enrich the functionality of the state-of-the-art SNNs to reach and go beyond un-

supervised learning capabilities of ANNs. Simultaneously, we always experimentally validate proposed designs.

1.3.2 Outline

The main content of the thesis is organized in five chapters, grouped into two themes, each of which begins with theory and ends with experimental hardware demonstrations.

The first theme, comprising three chapters, focuses on implementing conventional pattern learning approaches in a neuromorphic system. In Chapter 2, we introduce the details of the SNN model, which is used as the logical model in our neuromorphic implementation. In Chapter 3, we introduce phase-change memristors. The chapter is primarily hardware-oriented and involves device characterization. In Chapter 4, we build phase-change-based neuromorphic architectures that combine the elements from the previous chapters. The chapter focuses on hardware experiments and algorithmic improvements that tailor the SNN model for a phase-change implementation.

The second theme, comprising two chapters, focuses on extending the algorithmic models of SNNs with more versatile learning capabilities. In Chapter 5, we discuss the theory of knowledge representation in SNNs in comparison to deep learning ANNs. We identify the need to improve the knowledge representation in SNNs. In Chapter 6, we propose algorithmic advancements for SNNs that improve their knowledge representation. We co-design the computational aspects together with the prototype hardware, so as to utilize the properties of the phase-change technology for efficient implementation. We demonstrate experimental results for each proposed design.

Chapter 7 summarizes both parts of the thesis, and presents future research directions.

1.3.3 Publications

The contents of this thesis reflects a body of scientific work that was presented at multiple conferences and published in many journals. Chapter 1 is based on a conference paper [C1] and its extended journal version [J1]. Chapter 2 partially refers also to these papers, and to a journal paper [J3]. Neuromorphic building blocks proposed in papers [C2], [C4], [J2], [J3] are described together in Chapter 3, whereas architectures and results from these papers are described in Chapter 4. Chapter 5 extends the conference paper [C3]. Chapter 6 extends the contributions from two conference papers [C3] and [C5]. Furthermore, the work resulted in patent applications [P1], [P2], [P3] and [P4].

Conference papers

[C1] S. Woźniak, A.-D. Almási, V. Cristea, Y. Leblebici, and T. Engbersen, "Review of advances in neural networks: Neural design technology stack," in *Proceedings of ELM-2014 Volume 1*. Springer, 2015, pp. 367–376.

[C2] S. Woźniak, T. Tuma, A. Pantazi, and E. Eleftheriou, "Learning spatio-temporal patterns in the presence of input noise using phase-change memristors," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 365–368.

[C3] S. Woźniak, A. Pantazi, Y. Leblebici, and E. Eleftheriou, "Neuromorphic system with phase-change synapses for pattern learning and feature extraction," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017.

[C4] S. Sidler, A. Pantazi, S. Woźniak, Y. Leblebici, and E. Eleftheriou, "Unsupervised learning using phase-change synapses and complementary patterns," in *2017 ENNS International Conference on Artificial Neural Networks (ICANN)*. 2017.

[C5] S. Woźniak, A. Pantazi, Y. Leblebici, and E. Eleftheriou, "Feature learning using synaptic competition in a dynamically-sized neuromorphic architecture," in *International Conference on Rebooting Computing (ICRC)*. (Accepted), IEEE, 2017.

Journal papers

[J1] A.-D. Almási, S. Woźniak, V. Cristea, Y. Leblebici, and T. Engbersen, "Review of advances in neural networks: Neural design technology stack," *Neurocomputing*, vol. 174 A, pp. 31–41, 2016.

[J2] A. Pantazi, S. Woźniak, T. Tuma, and E. Eleftheriou, "All-memristive neuromorphic computing with level-tuned neurons," *Nanotechnology*, vol. 27, no. 35, p. 355205, 2016.

[J3] S. Woźniak, A. Pantazi, S. Sidler, N. Papandreou, Y. Leblebici, and E. Eleftheriou, "Neuromorphic architecture with 1M memristive synapses for detection of weakly correlated inputs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2017.

Patent applications

[P1] A. Pantazi, S. Wozniak, and T. Tuma, "Neuromorphic architecture with multiple coupled neurons using internal state neuron information," U.S. Patent App. 15/189 449, Jun. 22, 2016.

[P2] S. Wozniak and A. Pantazi, "Neuromorphic architecture for unsupervised pattern detection and feature learning," U.S. Patent App. 15/264 081, Sep. 13, 2016.

[P3] S. Sidler, S. Wozniak, and A. Pantazi, "Neuromorphic architecture for unsupervised classification tasks using information from complementary patterns," U.S. Patent App. 15/680 140, Aug. 17, 2017.

[P4] S. Wozniak and A. Pantazi, "Neuromorphic architecture for feature learning using a spiking neural network," U.S. Patent App. 15/725 320, Oct. 5, 2017.

1.3.4 Notation

The notation used in the thesis is a compromise between different conventions. For instance, a weight w associated with a directed connection from neuron i to neuron j is denoted as w_{ij} in ANNs and machine learning literature, but we convert it to w_{ji} notation from SNNs and control theory literature. In many places we simplify the neuroscientific notation, omitting commonly used subscripts to make place for indices used for identifying variables of neurons operating in a network. Whenever possible, we retain common conventions. For instance, to denote physical quantities we use the common identifiers in their capitalized version, such as voltage V, resistance R, and conductance G.

We use the remaining italic capital letters to denote sets, e.g. S or X. With X^Q we denote a subset of X induced by a set of indicies Q: $X^Q = \{x_i \in X : i \in Q\}$. By X^Q_k we denote a k-th element of this set, assuming that its elements are sorted using an ordering obvious from the context, such as the index-based natural ordering of the x_i elements.

We use small italic letters, e.g. f, g, x, w, to denote functions and variables. The parameters of a function are often skipped if they are obvious from the context, for instance: $f(t)$ might be written as f.

We use roman letters to denote operators, such as differentiation $\frac{\mathrm{d}x}{\mathrm{d}t}$, and selected functions, such as variance $\mathrm{Var}(g)$, or probability density function $\Pr(y = 1)$. For binary random variables, we skip "$= 1$" in the notation, so that it becomes: $\Pr(y)$.

Lastly, to indicate that data is a vector or a matrix of scalars, we use bold italic letters. For instance, $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ is an m-dimensional vector of scalar values, where i is the index identifying it within a dataset D of scalar vectors $D = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots\}$. A matrix of scalar values is denoted as $\boldsymbol{W}$ and its elements are denoted as $\boldsymbol{W}_{ji}$. Depending from the context, we denote the weights of neurons as functions $w_{ji}(t)$, values of scalar vectors $\boldsymbol{w}_i$, or values of scalar matrices $\boldsymbol{W}_{ji}$. In many cases these notations can be used interchangeably, but in selected fragments a particular notation may be preferred for improved understanding.

2 Spiking neural networks

This chapter provides background information on the SNN model, which is the logical model of operation of the neuromorphic systems proposed in the thesis. In Sec. 2.1 we present a common SNN architecture with a layer of spiking neurons, which we explain step by step throughout the rest of this chapter. We begin by discussing the input to the system in Sec. 2.2. Next, we describe in detail a single spiking neuron in Sec. 2.3, listing its parts, their functions, and their operating principles. Then, we focus on possible applications of a single neuron in Sec. 2.4. Lastly, we discuss the operation of a layer of neurons interconnected with a feedback mechanism.

2.1 Spiking neural network architecture

A general diagram of a single layer of spiking neurons is presented in Fig. 2.1. It illustrates a common architecture in which n neurons $N_1, \ldots, N_n$ share the same m inputs $x_1, \ldots, x_m$, that receive input information in form of the input spikes x_i, and emit computation results in form of the output spikes y_j. To differentiate computation in a group of neurons receiving exactly the same input information, feedback links are added between the neurons.

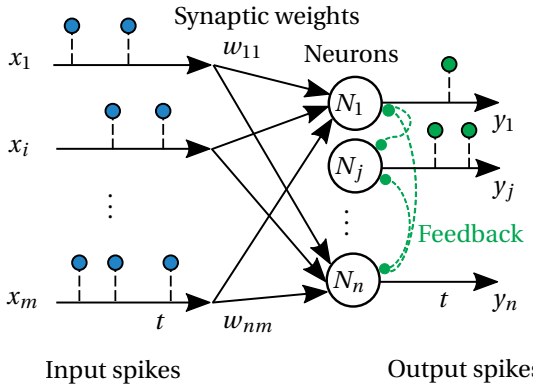


Figure 2.1: **A layer of spiking neurons** Input spikes from m inputs arrive at synapses w_{ji} of n neurons emitting output spikes on n outputs.

A set of input spikes observed over some period of time on input x_i is called a spike train and denoted as a function of time $x_i(t)$. The variables of the model at a specific time instance t_k, such as an input spike $x_1(t_k) = 1$ or lack of an output spike $y_1(t_k) = 0$, are often written with the time omitted in the notation, because in most cases the network activity is analyzed at a particular snapshot t_k.

2.2 Information coded in the spikes

The input and the output spikes of the SNN model are assumed to be all-or-none binary events. These spikes are used for interneuron communication and carry information through the shear fact of their appearance, whereas their amplitude and width are neglected. Interneuron spikes are often considered as diracs δ and drawn as vertical segments or dots. We primarily use a convention of drawing a dot with a supporting vertical segment below. The dot in this convention makes it easy to notice a spike, while the height of the supporting segment may be used to visualize an analog value. Although interneuron spikes are assumed to be binary, intraneuron spikes of potentials, which appear later in the text, may convey information in their amplitude. For simplicity, we assume that all spike-like signals or potentials observed in the model are called spikes and visualized using aforementioned convention.

In the next subsections we discuss means of coding information for interneuron communication with binary spikes. From a variety of coding schemes [Gerstner and Kistler, 2002] [Eliasmith, 2013], we focus on two most commonly used: rate coding and temporal coding.

2.2.1 Rate coding

A single spike train may code information in the frequency of spike appearance, called the spiking rate. This insight is inspired by biological experiments, in which the muscle contraction was determined to be proportional to the spiking frequency at the nerves controlling the muscles [Adrian and Zotterman, 1926]. Rate coding assumes that all the information is contained in the rate and the exact timing of the spikes is neglected. Sample spike trains are illustrated in Fig. 2.2a and their values decoded using rate coding are provided in Fig. 2.2b. The rate is typically calculated as the number of spikes observed in a given sampling window T_s, and then 0-1 normalized w.r.t. to the maximum possible spiking rate. The choice of the sampling window affects the accuracy of the calculated rate. A small sampling window is sensitive to the alignment of spikes that occur at the boundaries of the window, as illustrated for the input x_8, where the rate is underestimated in the first window, and is overestimated in the second window. Large sampling window enables more accurate rate estimation, but it also increases the response time of a system, because longer spike trains are required for communicating a single value.

A group of spike trains may independently code the same information in form of regular periodic spikes: for instance the inputs x_2 and x_3 code the value 0.5. Such temporal relation

of the spikes may impact the operation of a system. Despite the assumption that timing is neglected, the system may learn this regularity rather than the rate. Therefore, in rate-based systems it is common to explicitly reduce the regularity of input spikes [Querlioz et al., 2011]. A viable approach involves adding jitter – small timing deviations, to the timing of the spikes, as illustrated for the inputs x_4 and x_5. Another popular approach is to generate the spikes using Poisson processes with their mean rate equal to the desired spiking rate [Song et al., 2000]. This results in a random arrival time of the spikes, as illustrated for the input x_6.

Besides encoding information independently in each input, a group of spike trains may jointly code a single piece of information. In population coding, multiple neurons may provide information about a single value with a higher accuracy than a single neuron, or detect signals of higher frequency than their individual maximum spiking frequency. In the latter case, the population coding provides means for neuronal processing of signals going beyond the Nyquist-Shannon limit of a single neuron [Tuma et al., 2016b].

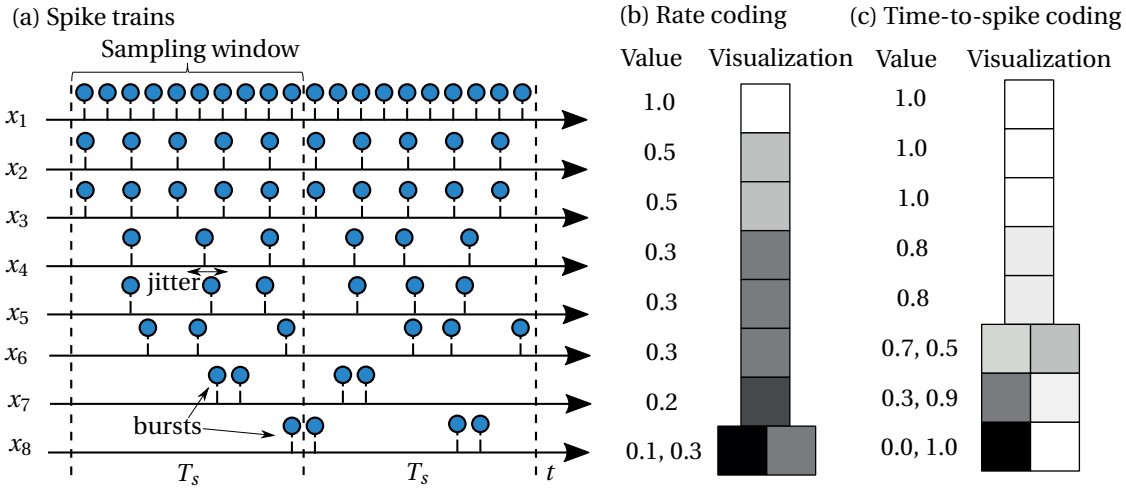


Figure 2.2: **Sample spike trains and their interpretation using different coding schemes** (a) Spike trains $x_1, \ldots, x_8$ plotted for a period of two sampling windows T_s. Maximum 10 spikes may appear within T_s. (b) Values decoded assuming rate coding, accompanied by a visualization in form of gray-scale pixels. Two values are provided if the decoded value is not consistent across the sampling windows. (c) Values decoded assuming time-to-spike coding. The beginning of each sampling window was treated as the reference time for the calculation.

2.2.2 Temporal coding

A single spike train may also code information in the timing of the spike appearance. One of the arguments behind this insight is that the response time of the nervous system is relatively short in comparison to the time required for collecting enough spikes to calculate an accurate spiking rate estimate [Van Rullen et al., 1998]. Therefore, the time-to-spike, also called the rank order of the inputs, may convey the complete information. For instance, the strongest activated neuron may respond first, thus representing the highest value, as illustrated in

Fig. 2.2c. An important aspect of time-to-spike coding is to determine the reference time, because the values are calculated relatively to it. It is common to define it as an on-set of a stimulus [Van Rullen et al., 1998]. A point of reference may also be provided by a burst of spikes, such as one in x_7 or x_8, or an on-set of neural oscillations.

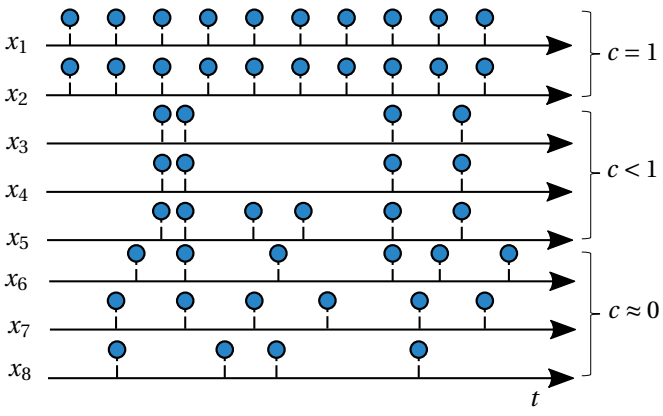
Information contained in the input correlations

Information may be also contained in the relative co-occurrence of the spikes within a population of neurons, which avoids the need to determine the point of reference. To quantify the co-occurrence, a Pearson cross-correlation coefficient c may be calculated in a pair-wise manner for any two spike trains $x_i(t)$ and $x_j(t)$ [Gütig et al., 2003]:

$$c = \mathrm{corr}(x_i(t), x_j(t)) = \frac{\mathrm{Cov}(x_i(t), x_j(t))}{\sqrt{\mathrm{Var}(x_i(t))\mathrm{Var}(x_j(t))}} \tag{2.1}$$

Examples of temporally-correlated inputs are illustrated in Fig. 2.3a and their respective pair-wise cross-correlation coefficients are visualized in form of a cross-correlation matrix in Fig. 2.3b. Inputs x_1 and x_2 are correlated with $c = 1$ and form a perfectly correlated pair of inputs. In general, a subset of inputs of any size may form a correlated group characterized by high intra-group correlation coefficients and low inter-group correlation coefficients. For instance, assuming an inter-group c threshold larger than 0.75, the inputs $\{x_3, x_4, x_5\}$ may form a single weakly correlated group with $c < 1$. Visually this corresponds to a dark square cluster in the cross-correlation matrix. The group formed by the remaining inputs with low pair-wise correlation coefficients is referred to as uncorrelated inputs with $c \approx 0$, and visually corresponds to a cluster with a dark diagonal in the cross-correlation matrix.

(a) Spike trains



(b) Cross-correlation matrix

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
x_1	1.00	1.00	0.00	0.00	0.22	-0.22	0.00	-0.25
x_2	1.00	1.00	0.00	0.00	0.22	-0.22	0.00	-0.25
x_3	0.00	0.00	1.00	1.00	0.76	0.22	-0.05	-0.25
x_4	0.00	0.00	1.00	1.00	0.76	0.22	-0.05	-0.25
x_5	0.22	0.22	0.76	0.76	1.00	0.05	0.05	-0.33
x_6	-0.22	-0.22	0.22	0.22	0.05	1.00	-0.19	-0.05
x_7	0.00	0.00	-0.05	-0.05	0.05	-0.19	1.00	0.22
x_8	-0.25	-0.25	-0.25	-0.25	-0.33	-0.05	0.22	1.00

Figure 2.3: **Examples of input correlations** (a) Presented spike trains $x_1, \ldots, x_8$ may be clustered into three groups: perfectly correlated ($c = 1$), weakly correlated ($c < 1$) and uncorrelated ($c \approx 0$). (b) A cross-correlation matrix for inputs $x_1, \ldots, x_8$ with c values visualized using the brightness. The three groups of inputs are marked with red squares.

Generating a dataset with a correlated pattern

In a system operating in a universe of input attributes $U = \{1, \ldots, m\}$, a pattern of correlated attributes Q, visualized as a binary image with a set of on-pixels $Q \subset U$ and off-pixels $N \subset U$, illustrated in Fig. 2.4a, may be represented at the inputs x_i through correlated activity. The attributes corresponding to Q form a correlated group of inputs $X^Q = \{x_i : i \in Q\}$, called the *pattern inputs*, whereas the inputs $X^N = \{x_i : i \in N\}$, called the *noise inputs*, remain uncorrelated. To generate the corresponding spike trains $x_i(t)$, assuming discrete time slots ΔT for the spikes, Bernoulli trials may be performed using the following probabilities [Gütig et al., 2003]:

$$\Pr(y^Q) = r_Q \Delta T \tag{2.2}$$

$$\Pr(X_i^Q | y^Q) = \sqrt{c} + \Pr(X_i^Q | \neg y^Q) \tag{2.3}$$

$$\Pr(X_i^Q | \neg y^Q) = r_N \Delta T (1 - \sqrt{c}) \tag{2.4}$$

$$\Pr(X_i^N) = r_N \Delta T \tag{2.5}$$

where r_Q is the rate of correlated activity at the pattern inputs, r_N is the rate of noise, each input $x_i \in X$ is treated as a random variable representing a spike appearance at a particular input, and y^Q is a random variable representing pattern appearance, which corresponds to the arrival of correlated spikes at the pattern inputs. If the pattern appears (a Bernoulli trial returns $y^Q = 1$ for Eq. 2.2), the inputs belonging to the correlated group spike (Eq. 2.3). However, depending from c, these inputs can also spike in uncorrelated manner (Eq. 2.4), as illustrated in Fig. 2.4b. Generating spikes using conditional Bernoulli trials in Eq. 2.3 and Eq. 2.4 was shown to produce a group of inputs with the pairwise correlation coefficients equal to c [Gütig et al., 2003]. Simultaneously, independent Bernoulli trials for X_i^N in Eq. 2.5 result in generation of uncorrelated Poissonian noise on the uncorrelated inputs.

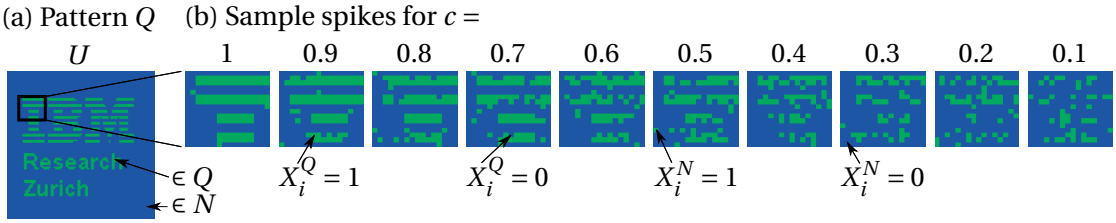


Figure 2.4: **Correlated inputs for varying correlation coefficient** c (a) Pattern Q determines which inputs form a correlated group of pattern inputs X^Q and which inputs are uncorrelated noise inputs X^N. (b) Sample spikes generated using Bernoulli trials based on Eq. 2.3 and Eq. 2.5, assuming an appearance of the pattern $y^Q = 1$.

The information contained in the correlated arrival of the pattern inputs may become dominated by the information contained in the rates [Moraitis et al., 2017], analogously to the issue of the timing information impacting the information in the rate coding. To ensure that the system learns only from the timing, the same mean rate may be used for all the inputs, which

implies $r_Q = r_N$ [Woźniak et al., 2016]. Alternatively, it is possible to have the information coded simultaneously in both the rate and the timing, but learn either from the rate or the timing [Moraitis et al., 2017].

2.3 Operation of a spiking neuron

A spiking neuron [Woźniak et al., 2017b], illustrated in Fig. 2.5, receives spikes from the inputs $x_1, \ldots, x_m$. An input spike x_i that arrives at a synapse with synaptic weight w_{ji} is weighted by the value of the synaptic weight and forms a post-synaptic potential $q_{ji} = x_i w_{ji}$. The post-synaptic potentials travel to the neuron soma which aggregates them into a total post-synaptic potential $\text{TPSP}_j = \sum_i q_{ji}$ and integrates into a membrane potential V_j. When the membrane potential crosses a threshold value V_{th}, an output spike is emitted by the neuron on its output y_j, and the membrane potential is reset. The membrane potential might also be reset by a feedback mechanism, connecting N_j with other neurons, and preventing emission of output spikes. Simultaneously, the relative timing of the output and the input spikes, $\Delta t_{ji} = t_j^{\text{post}} - t_i^{\text{pre}}$, is used by a Spike-Timing-Dependent Plasticity (STDP) learning mechanism to adjust the synaptic weights [Gütig et al., 2003]. In the next subsections, we discuss in detail the parts of the neuron, following the travel of information from the input spikes through the synapses, the neuronal soma, the output spikes, and then back to the synapses in form of STDP weight adjustments.

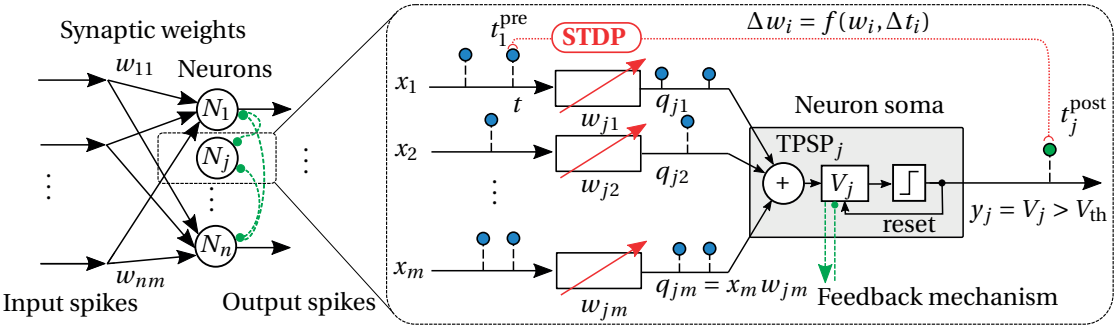


Figure 2.5: **A spiking neuron** Input spikes from m inputs arrive at the synapses w_{ji} that learn by adjusting their weights using STDP plasticity rule.

2.3.1 Synapses

A synapse processes the input spikes to provide post-synaptic potential q_{ji}, and the magnitude of this potential depends on the synaptic conductance, called the weight w_{ji}. Contrary to unbounded floating-point weights in an ANN model, a weight of an SNN model is assumed to be a uni-polar bounded value. These assumptions stem from the observations of biological neurons. Firstly, a synapse is a physical part of a neuron that may not exceed its physical limits. Therefore, it is common to assume that its value is bounded between the minimum and the

maximum conductance. Secondly, there are two categories of synapses: excitatory, which tend to provide positive potential q from a pre-synaptic excitatory neuron to the post-synaptic neuron soma, and inhibitory, which tend to provide negative potential q from a pre-synaptic inhibitory neuron to the post-synaptic neuron soma [Parisien et al., 2008]. Each synapse is uni-polar, because its type is determined by the type of the pre-synaptic neuron. Uni-polar operation of synapses and neurons is not limiting the computational capabilities of SNNs, as is it possible to transform between the bi-polar and the uni-polar models [Parisien et al., 2008].

In this work, similarly to many others [Querlioz et al., 2011] [Diehl and Cook, 2015], we assume that the synapses corresponding to the inputs $x_1, \ldots, x_m$ are excitatory only and we assume that their weights are in 0-1 normalized range. The inhibitory output synapses of inhibitory neurons typically have a special modulatory effect on the activity of the excitatory neurons. We rely on this effect for the operation of the feedback mechanism, and we discuss them later in Sec 2.5.

2.3.2 Neuron soma

Neuron soma stores information accumulated from the synapses in the membrane potential V_j and emits output spikes if V_j crosses the spiking threshold V_{th}. A biologically-realistic operation of the neuron soma in a non-linear Leaky Integrate-and-Fire (LIF) model considers the in- and out-flow of the charges, and is formulated using a differential equation [Song et al., 2000]:

$$\tau \frac{\mathrm{d}V_j}{\mathrm{d}t} = V_{\text{rest}} - V_j + G_{\text{ex}\,j}(t)(E_{\text{ex}\,j} - V_j) + G_{\text{in}\,j}(t)(E_{\text{in}\,j} - V_j) \tag{2.6}$$

where $V_{\text{rest}} = -70\,\text{mV}$ is the resting potential, towards which all biological neurons tend to converge when there is no input; $E_{\text{ex}\,j}$, $E_{\text{in}\,j}$ are the electric potentials at the excitatory and the inhibitory synaptic inputs; and $G_{\text{ex}\,j}$, $G_{\text{in}\,j}$ are time-dependent conductances of the exicitatory and the inhibitory synaptic inputs, that determine the change in V_j proportionally to the difference of the potentials in the soma and at the synapses. Alternatively, it is possible to perform a series of variable substitutions and to rescale the membrane potential, so that the resting potential vanishes and the integrate-and-fire model is reduced to a standard form [Gerstner et al., 2014]:

$$\frac{\mathrm{d}V_j}{\mathrm{d}t} = d(V_j) + I(t) \tag{2.7}$$

where d is a non-linear function of the membrane potential that incorporates the effects of the leak, and $I(t)$ is a time-dependent function corresponding to the aggregated synaptic current from all the inputs at time t.

In this work, we use the standard form equation in discrete simulations with a time step ΔT:

$$V_j := V_j + \Delta V_j, \quad \Delta V_j = d(V_j)\Delta T + I(t)\Delta T \tag{2.8}$$

We define the aggregated input over ΔT as $\text{TPSP}_j = I(t)\Delta T$ and we assume that V_j decays with a time constant τ towards 0: $d(V_j) = -V_j/\tau$. The final equation used in simulations is then:

$$\Delta V_j = -\frac{\Delta T}{\tau} V_j + \text{TPSP}_j \tag{2.9}$$

This formulation reflects the basic principles of the operation of an LIF neuron: storing information in a leaky membrane potential and accumulating information aggregated from the inputs. On the other hand, the transformation in Eq. 2.7 combined with our assumptions does not provide biologically realistic values. Therefore, we use auxiliary units for the variables TPSP_j and V_j.

2.3.3 Spiking threshold

When the membrane potential V_j crosses the spiking threshold V_{th}, a neuron emits an output spike and the membrane potential V_j is reset to its resting potential V_{rest}. For a neuron associated to a particular pattern of interest, the spiking threshold V_{th} should be low enough for the neuron to spike when that pattern appears at the input, but high enough to avoid spiking for input noise.

The choice of the spiking threshold involves understanding the interplay between the membrane leakage τ and the temporal characteristic of the input. Fig. 2.6 illustrates the evolution of the membrane potential V_j in a function of the membrane leakage τ for two different temporal characteristics of an input pattern consisting 100 spikes. If the arrival of the pattern is distributed over a short period of time, the impact of the leaky integration is limited, as illustrated in Fig. 2.6a. There are many acceptable choices for the spiking threshold, such as $V_{\text{th}} = 40$, and even a system that does not implement integration would detect the pattern, because the $\text{TPSP}(3 \times \Delta T) = 44$ crosses this threshold. Such threshold is likely to be high enough for the neuron to spike only for this pattern and not for the noise. If the arrival of the pattern is distributed over a long period of time, as in Fig. 2.6b, the choice of τ becomes a critical prerequisite for the choice of the spiking threshold. For the low values of τ, spiking threshold would have to be set to $V_{\text{th}} < 10$, which might result in false positive spiking even after a few non-pattern noise inputs. For $\tau > 1.5^6$, the same threshold of 40 works consistently for both the case in Fig. 2.6a and in Fig. 2.6b. The correct choice of τ enables to use a single spiking threshold value that is robust to the temporal distribution of the input pattern. Fig. 2.7 illustrates the maximum possible V_{th} for a system with leaky integration, versus a system that only checks the momentary value of the TPSP. Properly tuned leaky integration provides means for consistent choice of the spiking threshold.

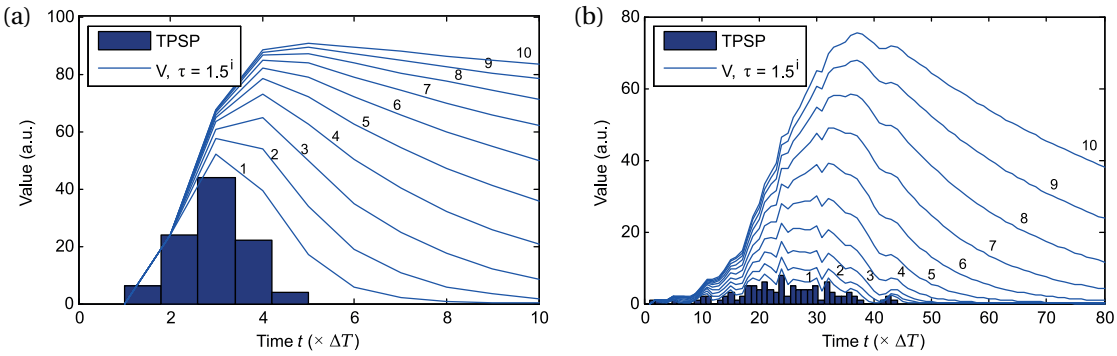


Figure 2.6: **The membrane leakage vs. temporal characteristics of the input** The pattern for which the neuron should spike comprises 100 spikes, normally distributed over time t. Membrane potential plots V are calculated for membrane leakage $\tau = 1.5^i$, where i is indicated on each curve. (a) For a pattern distributed over a short period of time ($5 \times \Delta T$) the value of the membrane potential is relatively high ($V = 52$) already for the lowest τ. (b) For a pattern distributed over a long period of time ($44 \times \Delta T$) the value of the membrane potential reaches higher V values only for the larger τ.

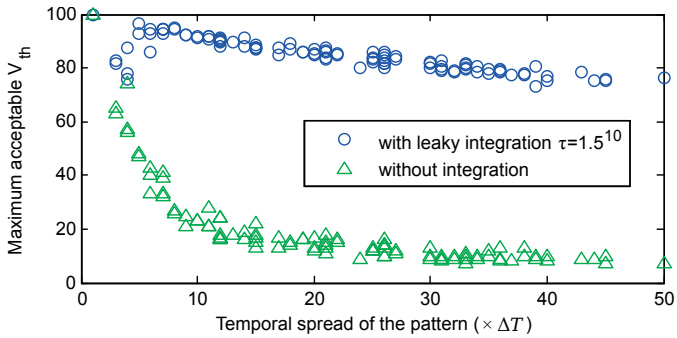


Figure 2.7: **Spiking thresholds for operation with and without integration** Maximum V_{th} values, corresponding to the maximum V observed during a series of trials, in which 100 input spikes formed a pattern that was temporally spread over a given period of time. For integration with a properly adjusted τ, there are many possible levels of V_{th} robust to the temporal spread of the pattern. Without integration, the V_{th} required to spike for a temporally-spread pattern quickly decreases to low levels, that makes it impossible to choose a noise-robust V_{th}.

Spiking threshold may be automatically adjusted following a biologically-plausible homeostatic plasticity mechanism [Marder and Goaillard, 2006]. In this approach, a mean firing output rate of the neuron is calculated over a given period of time and compared to a predefined target output firing rate. If the observed rate is above/below the target rate, the threshold is decreased/increased [Querlioz et al., 2011]. However, the important choice of the target output firing rate might not be obvious in the unsupervised learning setting.

2.3.4 STDP learning mechanism

Crossing the spiking threshold V_{th} results in a neuronal firing at time t_j^{post} that provides feedback to the synapses, which may adjust their weights w_{ji} using STDP, as illustrated in Fig. 2.5. Based on experiments with biological neurons [Markram et al., 1997], it was discovered that the relative difference of the timing of the pre- and post-synaptic spikes $\Delta t_{ji} = t_j^{\text{post}} - t_i^{\text{pre}}$ determines the direction and the magnitude of the synaptic conductance change Δw_{ji}. The plot of this conductance change, sometimes called the *STDP curve*, is often approximated using two exponentially decaying functions f_+ and f_- with parameters A_+, τ_+ and A_-, τ_-: $\Delta w = A_\pm \, e^{-|\Delta t|/\tau_\pm}$ [Song et al., 2000], plotted in Fig. 2.8a.

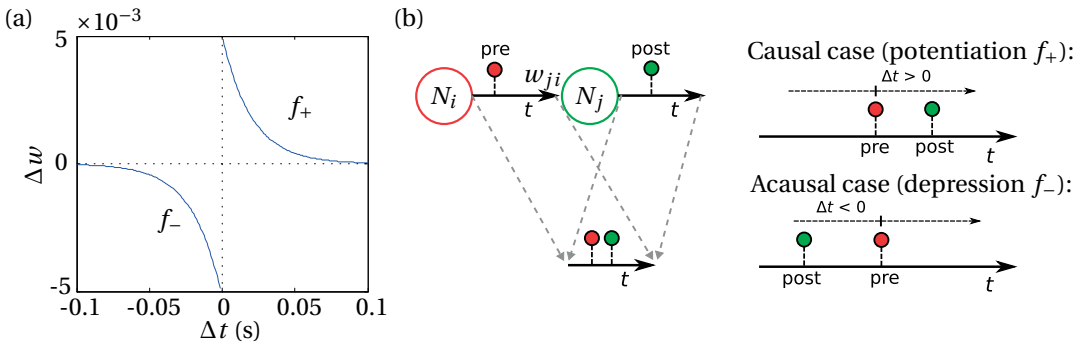


Figure 2.8: **Learning using Spike-Timing-Dependent Plasticity** (a) A common approximation of the biological STDP curve using two exponentially decaying functions. The direction and the magnitude of the synaptic weight change Δw depends from the relative timing of the pre- and post-synaptic spikes observed at the synapse. (b) When a pre-synaptic spike is followed by a post-synaptic spike, STDP reflects this causal relationship through potentiation. When the order of spikes is reversed, the acausal appearance of spikes leads to depression.

The shape of the STDP curve may be interpreted as capturing the causality of the inputs, as illustrated in Fig. 2.8b. If a synapse w_{ji} of a neuron N_j observes an input spike, followed by an output spike of the neuron N_j, there might be a causal relationship between the information coming from N_i and the activation of N_j. The evidence from this observation is captured in the synaptic weight w_{ji} through weight increase f_+, called Long-Term Potentiation (LTP). In the acausal case, in which the order of the spikes is reversed, the input spike at a synapse w_{ji} arrives after the output spike of N_j, so the information from N_i might be irrelevant for the activation of N_j. The evidence from this observation is captured through weight decrease f_-, called Long-Term Depression (LTD). In the process of learning, the synapses are typically increased and decreased multiple times before they collect enough evidence and reach a steady state. Lastly, the exponentially-decaying shape of LTP and LTD might be interpreted as a temporal coherence prior, which states that the closer in time the events occur, the stronger is the evidence that they might be associated [Bengio et al., 2013], thus the larger the weight change.

Weight change Δw_{ji} depends also from the value of the weight itself, because the synaptic conductance is bounded, as discussed in Sec. 2.3.1. Weight dependence may be encompassed in the STDP mechanism either implicitly by clipping the weights to their bounds after an application of a weight-independent $f_{\pm}$ [Song et al., 2000], or explicitly through weight-dependent updates [Gütig et al., 2003]. In this work, we will use an explicit weight-dependent STDP formulation in form:

$$\Delta w_{ji} = f(w_{ji}, \Delta t_{ji}) = \begin{cases} f_+(w_{ji}, \Delta t_{ji}) & \text{if } \Delta t_{ji} \geq 0 \\ f_-(w_{ji}, \Delta t_{ji}) & \text{if } \Delta t_{ji} < 0 \end{cases} \tag{2.10}$$

There are multiple approaches to the algorithmic execution of the weight updates Δw_{ji}: on the pre- and the post-synaptic spikes [Song et al., 2000], on the pre-synaptic spikes only [Brader et al., 2007] or on the post-synaptic spikes only [Querlioz et al., 2011]. In most of this work, we use the pre- and the post-synaptic convention for the STDP logic, formulated in Alg. 1 [Woźniak et al., 2017b], which is executed on each pre- and post-synaptic spike. In a neuron N_j, S denotes the set of inputs that receive an input spike at each time instance t. The variable t_j^{post} stores the last time of a post-synaptic spike of the neuron N_j, and t_i^{pre} stores the last time of a pre-synaptic spike arriving at each synapse w_{ji}. In the presence of a post-synaptic spike, potentiation f_+ is applied to all synapses that contributed to the neuron spiking (line 5). If there is no post-synaptic spike, arriving pre-synaptic spikes may lead to depression if they occur closely after a post-synaptic spike (line 8).

Algorithm 1 STDP

1: $S = \{i : x_i = 1\}$
2: **if** $y_j = 1$ **then**
3: $\quad t_j^{\text{post}} \leftarrow t$
4: $\quad \forall_{i \in S} \;\; t_i^{\text{pre}} \leftarrow t$
5: $\quad \forall_i \;\; w_{ji} \leftarrow w_{ji} + f_+(w_{ji}, t_j^{\text{post}} - t_i^{\text{pre}})$
6: **else**
7: $\quad \forall_{i \in S} \;\; t_i^{\text{pre}} \leftarrow t$
8: $\quad \forall_{i \in S} \;\; w_{ji} \leftarrow w_{ji} + f_-(w_{ji}, t_j^{\text{post}} - t_i^{\text{pre}})$

Alg. 1 captures the main principle of STDP operation, but it does not account for multiple temporally close pre-synaptic spike events. To solve this issue and to collect the evidence from a series of temporally close pre-synaptic spikes, we use an auxiliary variable for each synapse which modulates the final weight change based on all the observed spikes [Song et al., 2000].

2.3.5 Simplified STDP

Other STDP-like learning mechanisms were proposed for SNN models, such as so-called *simplified STDP* learning rule [Bichler et al., 2012a]. This rule, illustrated in Fig. 2.9a, is executed on a post-spike only: LTP with a fixed magnitude α_+ is performed for the synapses

that observed input spikes within a time period T_{LTP} prior to the post-spike, or otherwise LTD with a fixed magnitude α_- is performed. Therefore, the choice between LTP and LTD is a function of the past history before a post-spike. Sometimes this rule is schematically illustrated in an STDP-like Δw_{ji} plot in Fig. 2.9b, in which LTD stretches to $\pm\infty$. This learning mechanism has different semantics than the pre- and post-spike STDP. The key difference is that it treats lack of input preceding a post-spike as an acausal evidence that should result in depression. Its simplicity enables straightforward implementation as a learning mechanism in neuromorphic systems [Querlioz et al., 2011] [Bichler et al., 2012a] [Bichler et al., 2012b] and we also use it in selected experiments.

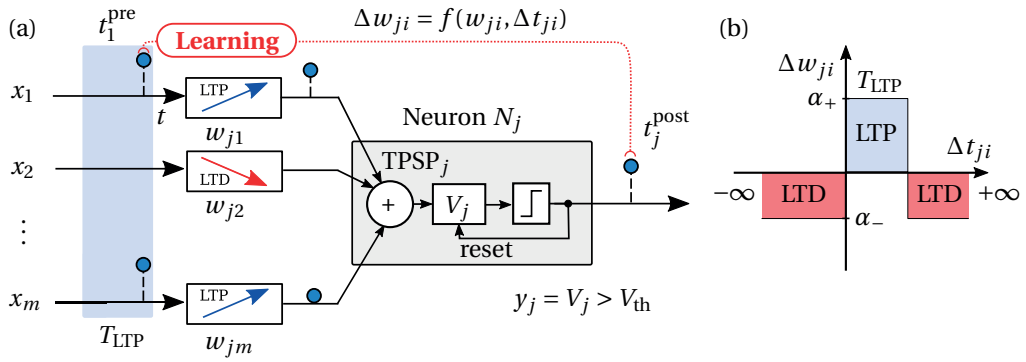


Figure 2.9: **Simplified STDP rule** (a) The rule is executed on post-spikes and considers only the past history before a post-spike. Lack of input is treated as an acausal evidence and results in LTD. (a) The synaptic weight w_{ji} is potentiated by a fixed magnitude α_+ when $\Delta t_{ji} \in [0, T_{\text{LTP}}]$, or otherwise it is depressed by a fixed magnitude α_-. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

2.4 Spiking neuron as a computational primitive

The learning mechanism enables a spiking neuron to learn a certain representation of a pattern Q, whose appearance might be detected afterwards through observation of the output spikes. Therefore, a single spiking neuron, illustrated in Fig. 2.10, is a computational primitive that provides a memory w_{ji} of a pattern Q that can be used for pattern visualization, and emits output spikes y_j that can be used for pattern detection. We discuss now both applications in more detail.

Pattern visualization

In the pattern visualization task, illustrated in Fig. 2.10a, the neuron N_j is expected to provide a reconstruction of the pattern Q in its weights w_{ji}. The pattern Q may correspond to an association between any categorical attributes of the input, because an input x_i might represent any piece of information, such as color, sound, text, or an abstract concept. However, image patterns are intuitively well-understood and can be concisely presented in the plots, so we use them throughout the rest of this thesis. Simultaneously, we abstract from the aspects of visual

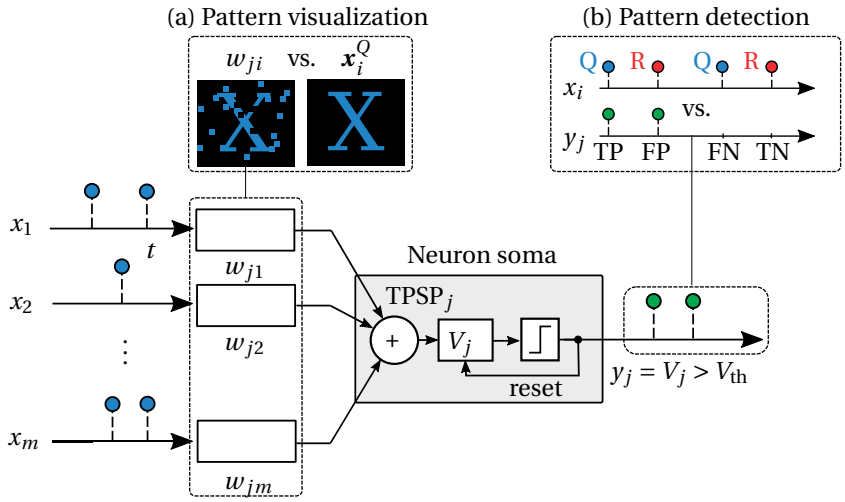


Figure 2.10: **Applications of a spiking neuron** (a) In pattern visualization, the neuron is expected to reconstruct the pattern in its synaptic weights w_{ji}. (b) In pattern detection, the neuron is expected to detect the presence of the pattern in the input by emitting output spikes.

sensory information processing, such as invariance to translation or rotation, keeping in mind the original abstract categorical nature of the inputs.

Depending from how information about the pattern Q is encoded in the inputs, learning an accurate reconstruction of Q might be a challenging task. Therefore, a reconstruction error may be calculated between the synaptic weights w_{ji} and the scalar vector $\boldsymbol{x}_i^Q$ of the reference values for the pattern Q, which for a binary pattern Q correspond to 1 when $i \in Q$, or to 0 otherwise. The most common approach is to calculate a Mean Squared Error:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(w_{ji} - \boldsymbol{x}_i^Q)^2 \tag{2.11}$$

Additionally, we define an MSE for a subset of input attributes S:

$$\text{MSE}(S) = \frac{1}{|S|}\sum_{i\in S}(w_{ji} - \boldsymbol{x}_i^Q)^2 \tag{2.12}$$

For the universe of input attributes $U = \{1,\ldots,m\}$, $\text{MSE}(U) = \text{MSE}$. Lastly, because we use 0-1 normalized weights, we may use MSE to define visualization accuracy as: $\text{A}(S) = 1 - \text{MSE}(S)$, which is reported in percentage points.

Using a neuron for pattern visualization is limited to particular synthetic cases only. In a highly-integrated dedicated neuromorphic hardware, or in a biological system, it might be difficult and inefficient to retrieve the values of all the weights. Moreover, the information read from a single neuron may provide only partial information on the pattern, as discussed further in Chapter 5. Lastly, the low MSE of a stored pattern may not be the right criterion for

the assessment. The corresponding neuron may still fail to detect the pattern, for instance if its spiking threshold is high and the input contains only partial information about the pattern.

Pattern detection

In the pattern detection task, illustrated in Fig. 2.10b, the neuron N_j is expected to detect the presence of the pattern in the input by spiking if the pattern Q appears at the inputs x_i, and refrain from spiking if a different pattern R appears. There are multiple approaches of assessing the spiking behavior of the neuron, which are often defined based on the following categories of output spikes, marked in Fig. 2.10b:

- **TP** (True Positive) – a neuron correctly spikes for the pattern Q,
- **FP** (False Positive) – a neuron incorrectly spikes for a different pattern R,
- **FN** (False Negative) – a neuron incorrectly does not spike for the pattern Q,
- **TN** (True Negative) – a neuron correctly does not spike for a different pattern R.

Spiking accuracy is defined as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{2.13}$$

which can be intuitively interpreted as the number of "the correct responses" over the total number of "the decisions that had to be taken". Depending on the distribution of the input patterns, the calculated accuracy might be misleading. Assume that the input comprises 1 appearance of the pattern Q and 99 appearances of an irrelevant pattern. If the neuron does not spike at all, the accuracy is: Accuracy $= \frac{0+99}{1+99} = 99\%$.

F-score measure provides in such case a more accurate assessment. It is a normalized harmonic mean of precision, defined as TP/(TP+FP), and of recall, defined as TP/(TP+FN):

$$\text{F-score} = \frac{2}{(\frac{\text{TP}}{\text{TP+FP}})^{-1} + (\frac{\text{TP}}{\text{TP+FN}})^{-1}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{2.14}$$

The calculation of the accuracy or the F-score requires prior knowledge on the exact timing of the pattern appearances and their assignment to particular classes, which might be difficult to define for patterns encoded in jittered or temporally-spread spikes. A more general approach involves calculation of Pearson cross-correlation from Eq. 2.1 between the output spike train $y_j(t)$ and each of the input spike trains $x_i(t)$ [Pantazi et al., 2016]. The correct behavior of the neuron corresponds to a high input-output cross-correlation for the inputs corresponding to the pattern Q.

In cognitive applications, assessing output spikes is much more natural than inspecting the synaptic weights. For instance, observation of a moving animal is in fact an indirect observation of the output spikes emitted by the neurons connected to the muscles, whereas the

information in the weights remains hidden to the observer. Because interneuron communication is realized using spikes, the contents of the weights may significantly differ from a low-MSE representation of any pattern Q, given that the neuron emits the proper spikes.

2.5 Operation of a spiking neural network

A spiking neural network comprises multiple spiking neurons communicating with each other through the synapses. We constrain our analysis to a single layer of neurons, for which we are able to provide concrete insights, as the computation properties of a multi-layer spiking neural network are not well understood. In particular, we consider a single layer network of excitatory spiking neurons $N_1, \ldots, N_n$ interconnected through feedback links, illustrated in Fig. 2.11a. The neurons are connected to the same inputs $x_1, \ldots, x_m$ and operate using the same spiking neuron model, described in Sec. 2.3. Therefore, without additional interneuronal feedback, they would learn the same information.

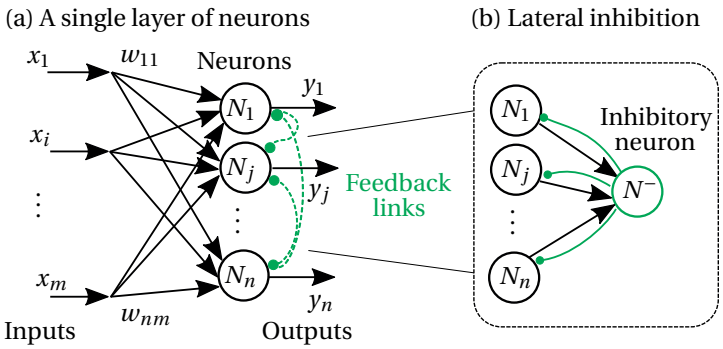


Figure 2.11: **Single layer spiking neural network with feedback links** (a) Excitatory neurons $N_1, \ldots, N_n$ are interconnected with feedback links, which foster divergent activity in the neurons. (b) Feedback links often implement the lateral inhibition mechanism, in which an additional inhibitory neuron N^- inhibits the excitatory neurons.

A common biologically-feasible interneuronal feedback scheme is lateral inhibition, illustrated in Fig. 2.11b. In this scheme, an additional inhibitory neuron N^- receives input from the excitatory neurons $N_1, \ldots, N_n$ and when a particular neuron N_j becomes active, N^- sends inhibitory feedback through inhibitory synapses to all the neurons. Because learning is triggered by spikes, as discussed in Sec. 2.3.4, and neuron N_j is able to spike, whereas the activity of the other neurons is attenuated, the neurons learn different information despite using the same learning mechanism. Finally, the scheme is consistent with the statement that neurons communicate through synapses, as all the communication is realized through excitatory and inhibitory synapses.

In practical realizations of SNNs, it is common to implement lateral inhibition in a simplified implicit form of a Winner-Take-All (WTA) architecture [Maass, 2000] [Nessler et al., 2009] [Querlioz et al., 2011]. The dynamics of lateral inhibition is simplified to instantly choosing the first neuron that spikes as the winner and resetting the membrane potentials of the others.

The simplification assumes that from neurons $N_1 \dots N_n$ with membrane voltages $V_1, \dots, V_n$ that crossed the spiking thresholds of these neurons, only one neuron is allowed to spike ($y_j = 1$). This is formulated [Maass, 2000] as:

$$(y_1, \dots, y_n) = \text{WTA}(V_1, \dots, V_n) \tag{2.15}$$

$$\text{if } V_j = \max(V_1, \dots, V_n) \text{ then } y_j = 1, \text{ else } y_j = 0. \tag{2.16}$$

If multiple neurons have the same membrane potential, a tie resolution strategy needs to be applied. A possible solution is to choose the winner with the lowest index.

A single layer of neurons in the WTA architecture may provide many important computational capabilities. Sometimes it is generalized to a k-WTA architecture, in which up to k winning neurons are allowed to spike simultaneously [Maass, 2000]. The classic WTA corresponds then to 1-WTA. Furthermore, a soft version of k-WTA, in which the output of a spiking neuron is a floating-point value representing its rank in the neural competition, was theoretically proven to approximate any arbitrary continuous function [Maass, 2000], even if all the neurons operate using a linear neuron soma model. The discussed computational properties and the conceptual simplicity of WTA makes it a popular choice for designing SNN architectures.

3 Phase-change-based spiking neurons

The information on the SNNs in the previous chapter provides a theoretical model for the operation of a spiking neuromorphic system. In this chapter, we develop a mixed digital-analog hardware implementation of a neuron tailored for phase-change technology. We begin with an introduction to phase-change technology in Sec. 3.1, where we discuss challenges stemming from the properties of phase-change devices. We describe a neuromorphic platform used to experimentally validate our designs, and we perform its characterization. Next, we proceed in steps to implement a synapse, a neuron, and an entire network using phase-change devices, providing simulation and experimental results at each step. In Sec. 3.2, we propose a phase-change-based synapse combined with a variation of the STDP learning rule. Lastly, in Sec. 3.3, we discuss a phase-change-based implementation of a neuronal soma. We analyze its behavior during operation with noisy inputs, and we propose a noise-robust phase-change neuron soma. All elements combined provide a complete basic functionality of a spiking neuron in an efficient neuromorphic implementation.

3.1 Phase-change technology

Phase-change technology explores the physical properties of phase-change materials to build memristive devices. These devices have been extensively studied in the context of memory applications, therefore a single phase-change memristive device is often called a *phase-change memory cell* (PCM cell) or simply a *cell*. Advantages of PCM cells include high scalability down to nm-scale, CMOS compatibility, low latency on the order of ns, and non-volatile analog storage capability [Burr et al., 2010]. With proper programming [Papandreou et al., 2011], a single nanodevice provides multiple intermediate resistive states that are used to build multi-level memory cells. This increases the storage density in the context of digital memories for von Neumann architectures. However, it is possible to directly use the analog properties of phase-change devices to design power- and area-efficient elements of a neuromorphic system, such as nm-scale synapses, orders of magnitude smaller than their µm-scale biological counterparts. To use the analog properties of PCM, we need to understand in detail its operation.

3.1.1 Operation of a phase-change cell

In a phase-change cell, information is stored in the structural configuration of the phase-change material. The phase-change material can be in the amorphous phase that has low electrical conductance, or in the crystalline phase, which has high electrical conductance. The conductance of the entire cell depends on the volume, the relative configuration of the two phases, and the cell design. In this work we focus on mushroom-type cells [Close et al., 2010], although other more sophisticated designs are possible, such as projected cells [Koelmans et al., 2015].

A mushroom type phase-change cell is schematically illustrated in Fig. 3.1a. The phase-change material is situated between a wide top electrode and a narrow bottom electrode. The material in the amorphous phase forms a dome-shaped structure of height u_a at the bottom electrode. The conductance of the cell is typically inversely proportional to u_a: in a low conductance state, a cell will have a large volume of the amorphous phase measured by high values of u_a, and vice versa.

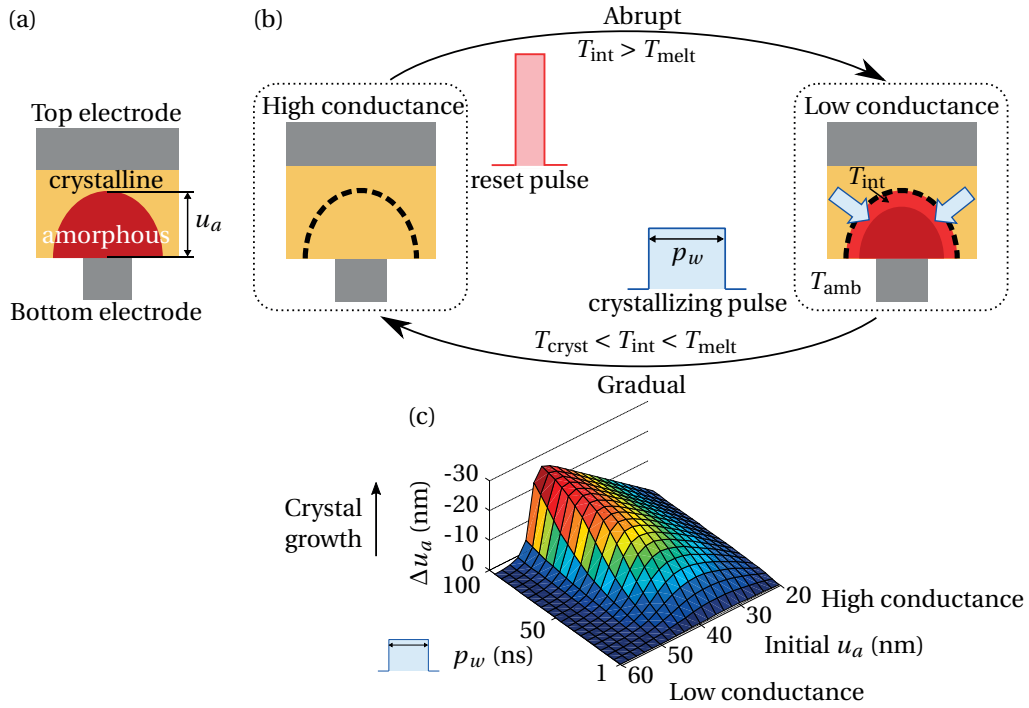


Figure 3.1: **A phase-change mushroom cell** (a) The conductance of the cell depends on the relative configuration of the crystalline and the amorphous phase. The amorphous phase forms a dome-shaped structure of height u_a situated at the bottom electrode. (b) Phase transition is induced through application of pulses that heat up the phase-change material. A crystallizing pulse gradually increases the crystalline phase and shrinks the amorphous dome. A reset pulse abruptly recreates the amorphous dome. (c) Simulation of crystal growth as a function of initial dome height u_a and applied pulse width p_w. The vertical axis was deliberately inverted to show the crystal growth, which corresponds to negative Δu_a. Subfigure adapted from [Woźniak et al., 2016], © 2016 IEEE.

A change of the phase-change configuration is induced through a voltage pulse applied to the electrodes of the cell, as illustrated in Fig. 3.1b. A pulse with high power that heats up the phase-change material above its melting temperature T_{melt} is called a *reset pulse*. An abrupt cut-off of the reset pulse quenches the material into an amorphous phase, and the cell is switched into a low-conductance state. The resulting conductance is independent of the number of applied reset pulses and is only a function of the applied current. A pulse with an amplitude that heats the phase-change material below the melting temperature T_{melt}, but above the crystallization temperature T_{cryst} is called a *crystallizing pulse*. A crystallizing pulse causes progressive crystallization that affects the entire interior of the dome, but primarily the region near the interface of the two phases of the material, gradually reducing the height u_a of the dome. In consequence, the cell can be in a range of analog states, depending on the u_a.

For interfacial crystal growth, the dynamics of the crystallization process is modeled using following equations [Sebastian et al., 2014]:

$$\begin{aligned} \frac{\mathrm{d}u_a}{\mathrm{d}t} &= -v_g(T_{\text{int}}(u_a)),\ u_a(0) = u_{a0} \\ T_{\text{int}}(u_a) &= R_{\text{th}}(u_a)P_{\text{inp}} + T_{\text{amb}} \end{aligned} \tag{3.1}$$

The change of u_a is a function of the crystal growth velocity v_g, which depends on the material and on T_{int}, the temperature at the interface between the two phases. Simultaneously, the temperature at the interface depends recursively from u_a. As u_a is decreased, the current flowing through the device increases and raises the temperature T_{int}. R_{th} denotes the thermal resistance across the cell, P_{inp} is the power of the applied pulse, and T_{amb} is the ambient temperature.

Results from a numerical simulation of Eq. 3.1 depicted in Fig. 3.1c illustrate how u_a evolves upon the application of a single crystallizing pulse of a given width [Woźniak et al., 2016]. Focusing on the initial u_a, for a large amorphous dome height, the crystal growth velocity at the interface is relatively low owing to low temperature. It increases rapidly once u_a is in the vicinity of the cell hotspot. Then, as the dome becomes smaller and the resistance of the cell decreases, the temperature at the interface is reduced and the crystal growth speed is also decreased. Focusing on the pulse width, longer pulses result in a larger change of the dome height. The concrete relationship depends on the type of the phase-change material and the particular cell design.

3.1.2 Crossbars of phase-change cells

To scale up the number of cells, phase-change devices can be aligned into a crossbar structure, illustrated in Fig. 3.2a. The rows and the columns of the crossbar form a grid in which the cells are located at the cross-points, as depicted in Fig. 3.2b. A cell is addressed by issuing a pulse on the proper row and column, corresponding to the cell location in the grid.

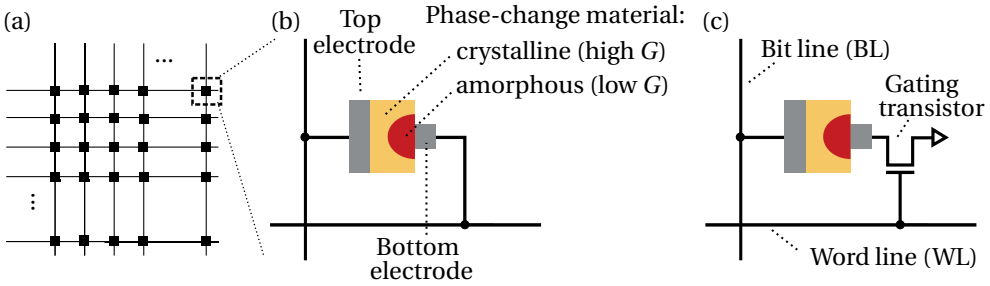


Figure 3.2: **Memristive crossbar** (a) Crossbar structure. (b) Phase-change memory cells are situated at the intersections of the rows and the columns of the crossbar. (c) In 1T1R design each cell is gated with a transistor. To access a cell, voltage is applied on a respective word line, and the pulses are issued afterwards on a respective bit line. Figure adapted from [Woźniak et al., 2017c], © 2017 IEEE.

The advantages of aligning phase-change cells into a crossbar include:

- **High area-efficiency** – phase-change cells are densely aligned in the crossbar. Combined with the capability to store analog values in the cells, such crossbars provide efficient means of storing information in comparison to transistor-based registers.
- **Programmable non-volatile storage** – the stored values are non-volatile and can be reprogrammed by modulating the pulses flowing through the cells. The programming and reading logic can be implemented outside of the crossbar and shared between multiple cells.
- **CMOS-compatibilty** – the cells can be manufactured and coupled with a standard CMOS design.
- **Computational capabilities** – the crossbar structure can be used to efficiently perform certain operations using Kirchhoff's laws. In particular, if a row with cell conductances $[G_1, \ldots, G_m] = \boldsymbol{g}$ is enabled and pulses $[x_1, \ldots, x_m] = \boldsymbol{x}$ are issued to all the columns in that row, the crossbar approximates $\boldsymbol{g} \cdot \boldsymbol{x}$ computation. Further computational capabilities of crossbars are explored in the field of computational memory [Gallo et al., 2017].

Depending on the conductance of the other cells in the crossbar, the current may flow through the other cells of the crossbar, leading to a phenomena called sneak current [Burr et al., 2017]. To prevent the current from following the sneak paths, the cells are often gated with a transistor, as illustrated in Fig. 3.2c and referred to as 1T1R design. To access a cell, voltage is applied to the transistor connected to the particular word line, and the pulses are issued afterwards to the top electrode connected to a respective bit line.

3.1.3 An experimental platform with GST phase-change cells

We use an experimental setup to validate our results in the rest of this work. The setup comprises mushroom-type cells with doped-chalcogenide alloy, $Ge_2Sb_2Te_5$ (GST), as the

phase-change material. A cross-sectional tunneling electron microscopy (TEM) image of the phase-change cell is shown in Fig. 3.3a. The cells are aligned into two 2 Mcells memristive arrays in 1T1R design using FET transistors, and integrated in a prototype chip [Close et al., 2010], illustrated in Fig. 3.3b. The chip is fabricated in 90 nm CMOS technology and contains cell addressing and programming circuitry, as well as an on-chip 8-bit analog-to-digital converter (ADC) for reading out the conductance. The phase-change chip is mounted on an FPGA-based platform [Papandreou et al., 2011], shown in Fig. 3.3c. The platform comprises pulse generators and two FPGAs to control the operation of the entire setup.

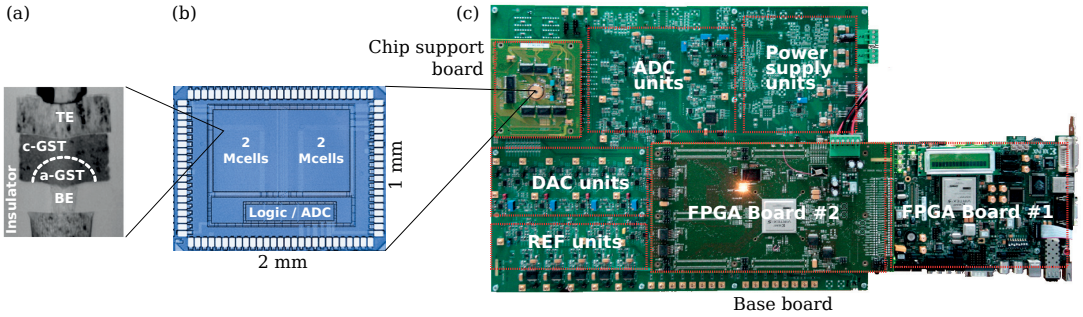


Figure 3.3: **Experimental platform** (a) TEM image of a GST phase-change mushroom cell. (b) Cells are integrated into a 2×2 Mcells prototype chip in 90 nm CMOS technology. (c) Base board with FPGAs for controlling the pulses applied to the cells. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

Characterization

A current-voltage (I-V) characteristic of a phase-change cell is illustrated in Fig. 3.4 [Pantazi et al., 2016]. Initially, the cell is in an OFF state, in which the current increases with the applied voltage. A cell in the amorphous state follows the least conductive path, whereas a cell in the

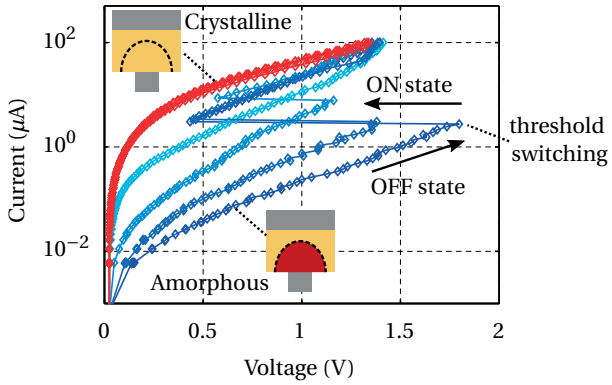


Figure 3.4: **I-V characteristic of a typical phase-change cell** In an OFF state, voltage increases along a path dependent on the phase-change configuration. After threshold switching, conductance of a cell in an ON state rapidly increases, which is used for programming. Adapted from [Pantazi et al., 2016], © IOP Publishing. Reproduced with permission.

crystalline configuration follows the highly conductive path. If the phase-change configuration is in an intermediate state, the cell follows and intermediate path. This property enables to read an analog value stored in the phase-change configuration. Changing the phase-change configuration relies on the threshold switching property: the voltage is increased until the cell switches from the OFF state to an ON state, in which the cell conductance rapidly increases. The phase-change material heats up and the cell is programmed as described in Sec. 3.1.1.

We analyzed the impact of various programming pulses by characterizing the conductance response for a randomly selected subset of 400 cells from the prototype chip [Pantazi et al., 2016]. We applied crystallizing pulses with constant current $I = 130\,\mu$A, followed by read pulses with constant voltage $V_{\text{read}} = 200\,$mV. Fig. 3.5a illustrates the cell conductance as a function of the width of the crystallizing pulse plotted for different initial conditions and averaged over all the cells. We observe gradual conductance increase in function of the applied pulse duration. The effective change in conductance depends on the actual cell state and the more the cell is crystallized, the smaller the conductance increase. Fig. 3.5b illustrates the crystal growth in the experimental setup, presented as a conductance change ΔG. Its non-linear response follows the shape predicted by the simulations: similar non-linearity is observed for cross-sections of Fig. 3.1c, with u_a corresponding to the initial conductance G_0.

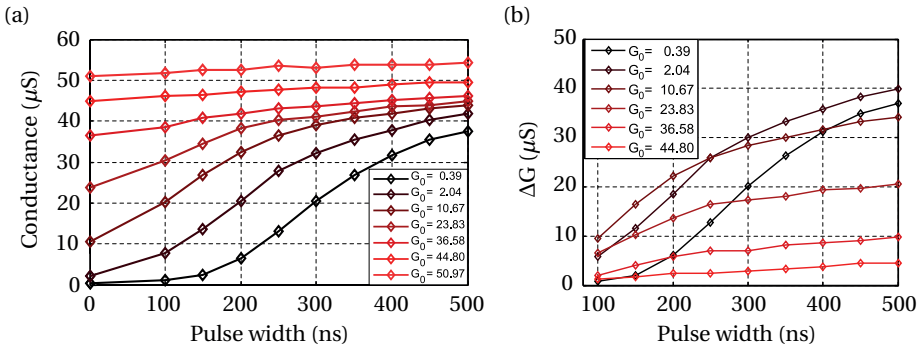


Figure 3.5: **Characterization of GST phase-change cells** (a) Cell conductance response in function of the crystallizing pulse width p_w. Subfigure from [Pantazi et al., 2016], © IOP Publishing. Reproduced with permission. (b) Crystal growth dynamics illustrated using conductance change ΔG plot. Subfigure from [Woźniak et al., 2016], © 2016 IEEE.

We analyzed the cell conductance in terms of the conductance variability by consecutive application of fixed-width pulses to 1000 cells [Woźniak et al., 2017c]. Fig. 3.6a shows the conductance response for crystallizing pulses with a width ranging from 50 ns to 500 ns. The response is non-linear and the variability is illustrated with the standard deviation error bars. Fig. 3.6b presents a histogram of device conductances after a given number n_p of crystallizing pulses with constant pulse width $p_w = 50\,$ns.

The results of the characterization were used to build a model of a phase-change cell. The model is based on a look-up table of the averaged values of conductance response for a given pulse width. In this work we relied on this model for rapid development of the algorithms and for the selection of the appropriate pulse widths for the experiments.

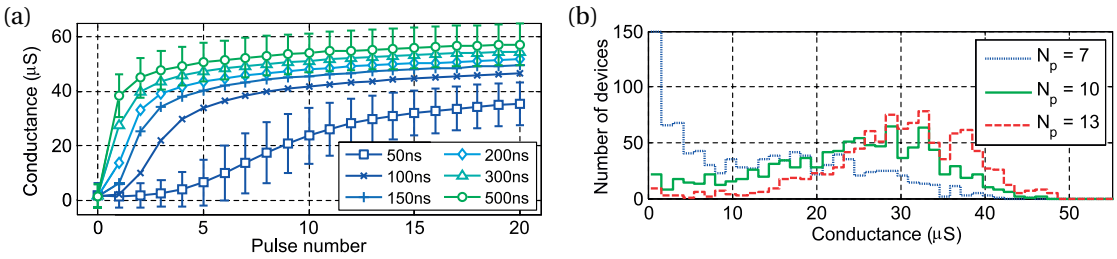


Figure 3.6: **Conductance response after application of multiple pulses** (a) Conductance response $G = f(n_p)$ for consecutive applications of constant-width pulses. (b) Inter-device variability after application of n_p pulses with $p_w = 50\,\text{ns}$. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

3.2 Phase-change-based synapses

Synapses are the basic memory element in a spiking neuron model. Their functionality involves storing a value from a predefined range and adjusting this value in the process of learning. These requirements can be fulfilled in many ways, for instance by using digital logic with transistor-based registers. However, memristive devices, such as the phase-change cells, offer the capability to implement a synapse using a single analog nanodevice at a lower complexity than using a pure transistor-based approach [Kuzum et al., 2012].

On the other hand, phase-change-based synaptic implementation involves multiple challenges. Firstly, the general operation mode of the phase-change with gradual conductance increase and abrupt conductance decrease, discussed in Sec. 3.1.1, needs to be considered while designing the architecture of the system. Secondly, the design has to address the properties of the hardware that include non-linear response of the cells, stochastic operation and production variability, characterized for our experimental platform in Sec. 3.1.3. In the next section, we discuss different types of synapse designs that address these challenges.

3.2.1 Types of phase-change synapse designs

Different designs were proposed to address the challenges of a hardware neuromorphic synapse implementation. First, we describe an approach using one PCM device per synapse. Next, we discuss the trend to use multiple PCM devices per synapse in 2-PCM and multi-PCM designs. Finally, we compare the advantages and disadvantages of particular PCM synapse types, and explain our synapse type choice for use in the further experiments.

1-PCM synapse

In 1-PCM synapse [Kuzum et al., 2012] a single device is used to represent the synaptic weight, as illustrated in Fig. 3.7. The weight is stored in the phase configuration of the device. Weight

increase is realized through application of crystallizing pulses that gradually crystallize the cell during synaptic potentiation, whereas weight decrease is realized through application of reset pulses that abruptly reamorphize the phase-change material at each synaptic depression.

The dynamics of the 1-PCM synapse follows the dynamics of a phase-change cell, described in Sec. 3.1.1. Therefore, the disadvantages of the 1-PCM design follow from the limitations of the phase-change technology and include asymmetric and non-linear conductance response, stochastic operation and high variability.

On the other hand, 1-PCM synapse is the most compact and straightforward design. Its advantages include using a small number of devices for a given number of synapses, low synapse complexity, and direct implementation in form of a memristive crossbar. In consequence, 1-PCM design directly inherits the advantages of the crossbar, such as area-efficiency and additional computational capabilities, discussed in Sec. 3.1.2. In particular, the vector multiplication capability may be used to accelerate the computation of the weighted sum of the synaptic inputs x_i and the synaptic weights w_i for the computation of TPSP in a spiking neuron model illustrated in Fig. 2.1 (page 15).

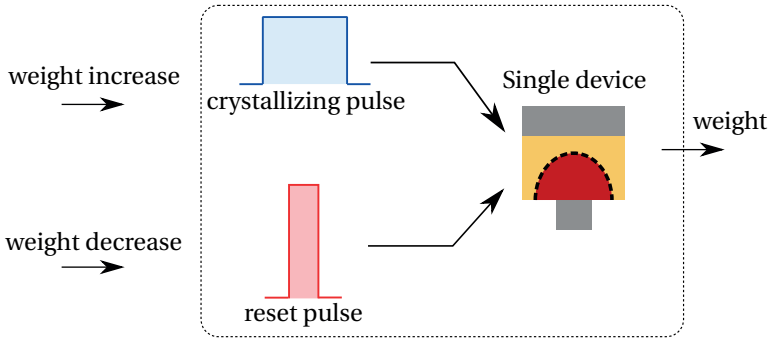


Figure 3.7: **1-PCM synapse** Crystallizing pulses gradually increase the weight during synaptic potentiation, whereas a reset pulse reamorphizes the cell at each synaptic depression event.

2-PCM synapse

A 2-PCM synapse [Suri et al., 2011] addresses the asymmetric conductance response challenge by using two devices per synapse. The devices, denoted as positive and negative, are connected in a differential configuration illustrated in Fig. 3.8. A gradual synaptic weight increase is realized through application of a crystallizing pulse to the positive device with conductance G^+. Similarly, a gradual synaptic weight decrease is realized through application of a crystallizing pulse to the negative device with conductance G^-. The effective weight of the synapse is calculated as the difference between the two conductances G^+ and G^-. Therefore, the synapse avoids the abrupt characteristics of the reset pulses during most of its operation.

However, if a series of weight increases is interleaved with a series of weight decreases, a positive or negative cell may reach the state of the maximum crystallization, and the synapse

will become *saturated*. A saturated synapse changes its weight in a single direction if a single device is saturated (e.g. $G^+ = G^{\max} \implies \Delta G < 0$), or does not change its weight at all if both devices become saturated ($G^+ = G^- = G^{\max} \implies \Delta G = 0$). Therefore, an additional mechanism of conductance rebalancing [Burr et al., 2014] needs to be triggered when saturation occurs. It involves reading the weight of the synapse, resetting both cells, and crystallizing only one of the cells to correspond to the given weight.

The differential configuration results in a more complicated non-linear behavior of the synapse. From the operation principles of phase-change cells, described in Sec. 3.1.1, we know that the conductance response is a non-linear function of the state of the cell. With two cells per synapse, the conductance response of a synapse with value G is a non-linear function of the state of both G^+ and G^- cell, which may store different pairs of values as long as $G^+ - G^- = G$. Still, this design is successfully applied for hardware realizations of SNNs [Sidler et al., 2017] [Tuma et al., 2016a] and ANNs [Burr et al., 2014].

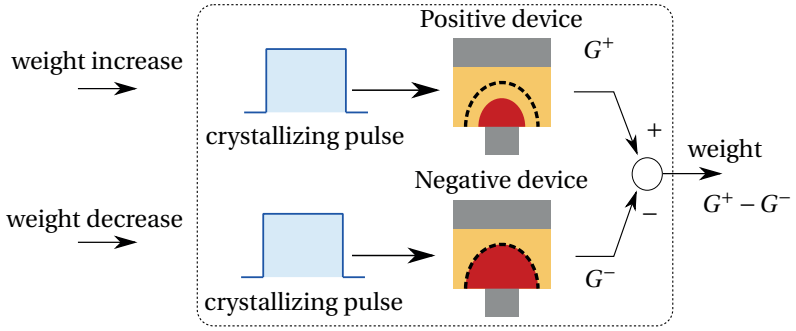


Figure 3.8: **2-PCM synapse** Two devices are combined in a differential configuration. Weight increase and decrease are gradual owing to application of crystallizing pulses only.

Multi-PCM synapse

Using multiple memristive devices per synapse provides more capabilities to address various challenges of the hardware implementation at the expense of increased area and complexity. A schematic illustration is shown in Fig. 3.9, in which the particular programming logic and the weight read-out depends on the design.

A multi-PCM design with cell selection based on modular arithmetic implements cumulative synaptic depression without the need for conductance rebalancing [Boybat et al., 2017]. Furthermore, it may provide a close to linear synaptic response and reduce the magnitude of weight uncertainty stemming from inherent phase-change cell's stochasticity. Lastly, it increases the dynamic range of the synapse beyond the range of a single cell.

Even though the study is performed for phase-change devices, some concepts are applicable to other technologies. The capability to increase the dynamic range is particularly important for bi-stable memristive devices, such as HfO_x/TiO_x-based cells. A multi-memristive synapse

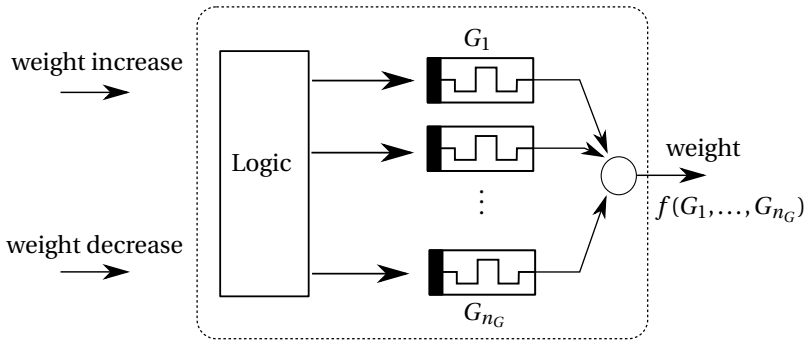


Figure 3.9: **Multi-memristive synapse** Additional capabilities are available at the expense of increased area and complexity.

comprising n_G parallel bi-stable memristive devices provides n_G+1 synaptic conductance levels [Bill and Legenstein, 2014]. Stochasticity of these devices becomes a positive factor, which is exploited for the learning. During application of programming pulses, they increasingly switch their state with probability dependent on the magnitude of the applied current.

Comparison of PCM synapse types

A summary of the advantages and disadvantages of the particular synapse types is provided in Tab. 3.1. The operation of 1-PCM synapse follows the stochastic non-linear phase-change dynamics with asymmetric conductance response. Multi-PCM synapses aim to reduce the impact of these properties. In 2-PCM design, the asymmetric conductance response is addressed. In designs with more cells, it is possible to improve additional characteristics, such as to linearize the conductance response or to reduce the stochasticity of a synapse.

Synapse type	1-PCM	2-PCM	Multi-PCM
Low synapse stochasticity	×	×	✓
Linear synapse response	×	×	✓
High dynamic range	×	×	✓
Cumulative synaptic depression	×	✓	✓
Low synapse complexity	✓	×	×
High synapse area-efficiency	✓	×	×
Direct TPSP calculation using a crossbar	✓	~	~

Table 3.1: **Comparison of PCM synapse types** A checkmark ✓ indicates that a design provides the particular feature. A crossmark × indicates lack of the particular feature. A tilde ~ in the last row indicates that a design may to some extent benefit from the use of a crossbar.

The motivation behind the multi-PCM designs is a bottom-up approach with an assumption that reliable synapses lead to a reliable neuromorphic system. However, improving the synthetic specification of a synapse significantly increases its complexity and may have limited practical impact on the operation of the entire system. The biological synapses are well-known

for their unreliable operation. In many cases the transmission reliability might be as low as between 10% and 30% [Harris et al., 2012] [Gerstner and Kistler, 2002], which implies also limited learning reliability. Therefore, in this work we postulate to use imperfect 1-PCM synapses and compensate for the phase-change imperfections on the system level, for instance in the learning rule.

3.2.2 STDP for PCM synapses

The synaptic weight modifications in the SNN model are determined by the plasticity learning rule, such as STDP, discussed in Sec. 2.3.4. These weight modifications are relative to the current weight, and the two basic operations are weight incrementation and decrementation. There are several ways of implementing these operations in neuromorphic systems.

For certain types of memristive devices, such as oxide-based memristors, the programming pulse reaching a memristive device may be a superposition of specially-designed waveforms [Querlioz et al., 2011]. The shape of these waveforms determines the learning curve and may be tuned to a desired characteristic, such as the STDP curve [Serrano-Gotarredona et al., 2013].

In realizations using phase-change cells, it is possible to approximate the reference STDP curve from Fig. 2.8 by translating the Δt to the crystallizing pulse width in 2-PCM synapse [Tuma et al., 2016a]. In 1-PCM synapse, an abrupt reset characteristics discards the initial state during the weight decrease. A solution that applies reset pulses with varying amplitude was proposed to reproduce the gradual shape of the STDP curve for weight decrease [Kuzum et al., 2012]. However, the amplitude of these pulses depends on the initial state of the synapse. Therefore, a single synaptic weight decrement involves reading synaptic state and applying a reset pulse with an amplitude calculated as a function of that state and the magnitude of the desired decrementation. This depression logic needs to be then implemented for each synapse.

3.2.3 Asymmetric STDP for 1-PCM synapses

We propose an STDP learning rule suitable for use with 1-PCM synapses. The potentation of a synapse is realized through application of variable-width crystallizing pulses and relies on the accumulative characteristic of the phase-change devices, whereas depression is realized through an application of a reset pulse. The synaptic weight adjustment is expressed in a general form:

$$\Delta w_{ji} = f(w_{ji}, \Delta t_{ji}) = \begin{cases} f_+(w_{ji}, \Delta t_{ji}) & \text{if } \Delta t_{ji} \in [0, T_{\text{LTP}}] \\ -w_{ji} & \text{if } \Delta t_{ji} \in [-T_{\text{LTD}}, 0) \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where T_{LTP} is the long-term potentiation period and T_{LTD} is the long-term depression period. Weight adjustments occur only during *a learning window* in the temporal vicinity of post-

synaptic spikes, delimited by T_{LTD} and T_{LTP}, highlighted in Fig. 3.10a. In the potentiation part, the relative timing of pulses Δt_{ji} is mapped to the pulse width of the crystallizing pulses, so that the weight adjustment f_+ corresponds to the shape of the classic STDP in Fig. 2.8. In the depression part, the f_- is a straight line, rather than a symmetric exponential curve of the classic STDP. Therefore, we call our approach *asymmetric STDP (A-STDP)*. A-STDP complies with the mode of operation of a phase-change device, in which the conductance is gradually increased in an accumulative fashion and decreased by the reset pulse to a fixed value.

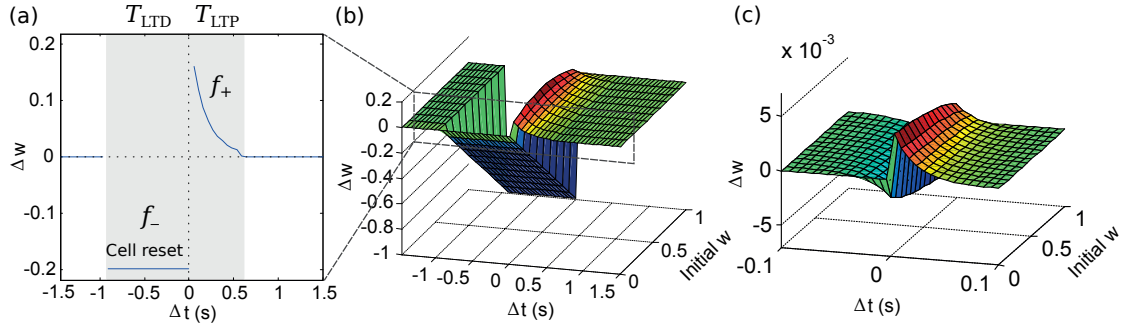


Figure 3.10: **Asymmetric STDP for 1-PCM synapses** (a) The crystallizing pulse widths are mapped to Δt to reproduce the shape of the STDP curve for potentiation f_+. Reset pulse is applied for depression f_-. (b) A 3D plot of A-STDP weight adjustments calculated using crystal growth simulation for different initial weights. Subfigure from [Woźniak et al., 2016], © 2016 IEEE. (c) A 3D plot of the classic STDP. Weight adjustments are bounded by the dynamic range of the synapses.

The effect of STDP weight adjustment f is a function of the current weight value w_{ji}, which is typically omitted in the STDP plots. A 3D plot in Fig. 3.10b is calculated based on the crystal-growth dynamics simulation introduced in Sec. 3.1.1 and visualizes the impact of the cell reset on synaptic weights operating with A-STDP in function of an initial weight. The common 2D STDP plot in Fig. 3.10a is a cross-section for a given initial condition, here for $w_0 = 0.2$. In general, for any implementation of STDP with constrained weights, the magnitude of the weight changes will be constrained near their minimum and maximum values to maintain the weights within their bounds. This is illustrated for 0-1 normalized weights and the classic STDP in Fig. 3.10c.

An experimental realization of the A-STDP is presented in Fig. 3.11a. Multiple curves are plotted for different initial conductance G_0, where each curve is obtained by averaging weight adjustments of 400 cells in the experimental platform described in Sec. 3.1.3. In Fig. 3.11b, a simplified version of A-STDP is presented, in which the duration of the crystallizing pulse during potentiation is fixed to $p_w = 100\,\mathrm{ns}$. In neuromorphic systems, it is common to reduce the complexity of the design by simplifying the shape of the STDP curve [Bichler et al., 2012a].

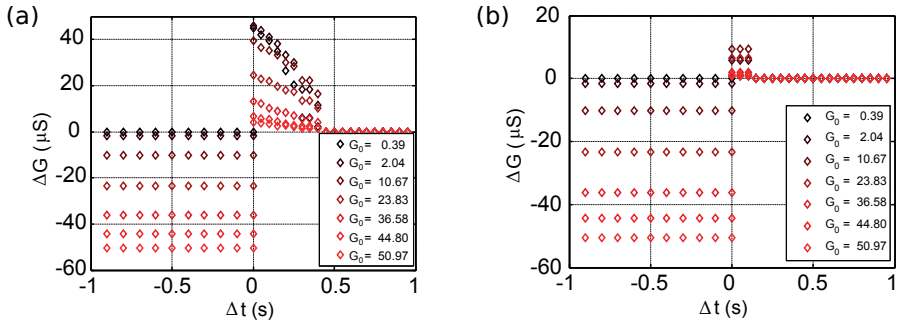


Figure 3.11: **Experimental realization of A-STDP** (a) A-STDP weight change plot based on the experimental data with weights mapped to phase-change cell conductances. Multiple curves are plotted for different initial conductance G_0. (b) Simplified version of A-STDP, in which a constant-width pulse is applied during potentiation f_+. Figure from [Woźniak et al., 2016], © 2016 IEEE.

3.3 Phase-change-based neurons

The synapses need to be combined with a neuronal soma to form a neuron. Different implementations are possible for the neuronal soma, including CMOS [Wu et al., 2015] and FPGA designs [Shimada and Torikai, 2015]. The classic approach for LIF implementation [Gütig et al., 2003] involves an RC circuit, illustrated in Fig. 3.12a. The capacitor C performs the integration and corresponds to the membrane potential V. The resistance R provides the leak term τ, which is crucial for limiting false positive neuronal spiking for the noise, as discussed in Sec. 2.3. The current I coming from the synapses is charging the capacitor, thus increasing

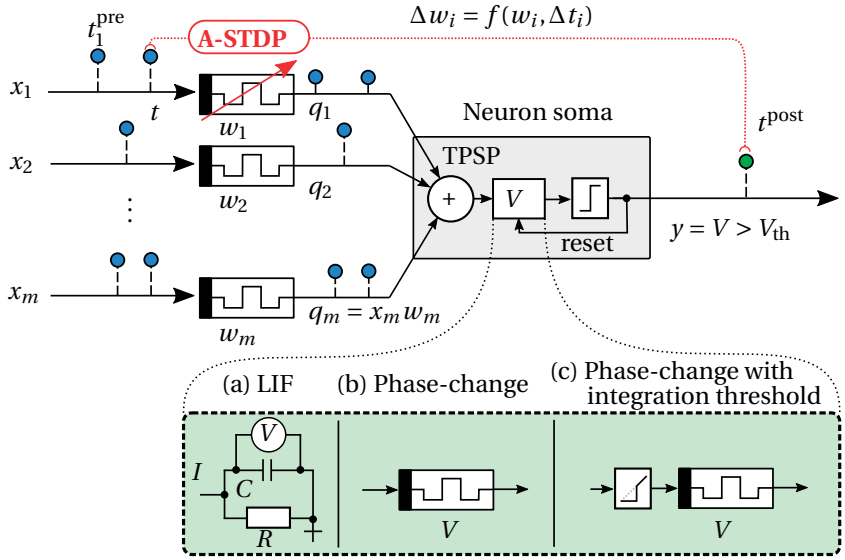


Figure 3.12: **Neuron soma hardware implementations** (a) LIF implementation using a capacitor C and a resistance R. (b) A phase-change neuron soma implementation. (c) A phase-change neuron implementation with an integration threshold.

the voltage V. The LIF neuron is the most commonly used model, as it is easily simulated in software using differential equations as well as implemented in hardware.

Having implemented the synapses using 1-PCM, it is appealing to explore the possibility to use phase-change devices also for the neuronal soma. A basic idea for a phase-change (PC) neuron is to use the phase configuration of a phase-change device to represent the membrane potential V [Tuma et al., 2016b], as illustrated in Fig. 3.12b. The current coming from the synapses is mapped proportionally to the width of the crystallizing pulses, thus modifying the conductance of the phase-change device. The conductance of the phase-change device is denoted as V, because its magnitude corresponds to the membrane voltage V of the spiking neuron model. In consequence, the membrane potential V for the phase-change neuron is measured in Siemens rather than in Volts, which might seem confusing. A different nomenclature could be introduced, but we keep using the term membrane potential V, as it is commonly used to denote the value accumulated in the soma.

An experimental characterization of the membrane potential V is plotted in Fig. 3.13a [Pantazi et al., 2016]. Next, assuming a spiking threshold of 40 µS we determine the number of pulses required to reach the spiking threshold in Fig. 3.13b [Pantazi et al., 2016]. When the neuron reaches the threshold, an output spike is emitted and the phase-change cell of the neuron soma is reset. The neuron continues to integrate and spike, with a frequency that is an approximately linear function of the crystallizing pulse width.

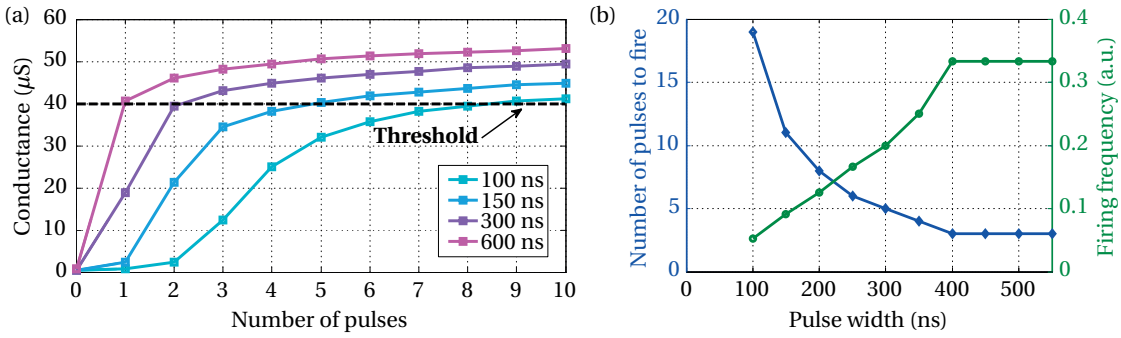


Figure 3.13: **Operation of a phase-change neuron** (a) Conductance characterization of a phase-change neuronal soma after application of a given number of crystallizing pulses with $I = 130\,\mu A$. Values averaged over 400 cells. (b) Average firing rate of a phase-change neuron as a function of repeating application of constant-width crystallizing pulses. The firing rate is an approximately linear function of the crystallizing pulse width. From [Pantazi et al., 2016], © IOP Publishing. Reproduced with permission.

The phase-change neurons have multiple unique advantages. Firstly, the non-volatile character of the membrane potential allows for low frequency and low power operation of a neuron, which might become important for long-term operation, e.g. in an IoT device. Secondly, phase-change devices exhibit intrinsic stochasticity [Tuma et al., 2016b], which might be used in particular applications relying on the stochastic activation of the neurons. Lastly, it might

be advantageous to have a uniform design utilizing a phase-change crossbar for both the synapses and the neurons.

On the other hand, the phase-change neuron implementation inherits the characteristics of a phase-change device. Owing to the asymmetric conductance response, the membrane potential in Fig. 3.13a cannot be gradually decreased. In consequence, the PC neuron lacks the gradual leak term τ and may emit false positive spikes for noisy inputs. The asymmetry issue can be addressed in an analogous way as described in Sec. 3.2 for the phase-change synapses – through the use of two or more devices. However, similarly to the 1-PCM approach we followed for the synapses, we use a 1-PCM neuronal soma and compensate for the asymmetry at a different stage to achieve noise-robust operation.

3.3.1 Noise-robust phase-change neurons

We propose a noise-robust phase-change neuron with integration threshold (PCth) [Woźniak et al., 2017c], illustrated in Fig. 3.12c, in which a thresholding unit is added before the integration stage. We explain the functionality of the PCth neuron in comparison to the LIF neuron.

An LIF neuron, introduced in Sec. 2.3, does not fire for a low magnitude TPSP owing to the decay of the V, following Eq. 2.9 (page 22). The low magnitude TPSP typically corresponds to activation of non-pattern inputs with low weights that result in low q_i values, so we denote it as noise TPSP_N. Continuous contribution from TPSP_N over a period of time ΔT is compensated by the leak term and stabilizes the membrane potential V at an equilibrium level V_N. This prevents the emission of false positive spikes for the noise, as $V_N < V_{\mathrm{th}}$.

Maintaining a proper level of V_N for a given application requires tuning the leak time constant τ. Repeated arrival of TPSP_N each ΔT stabilizes the membrane potential at a voltage denoted as V_N, which implies that $\Delta V_N = 0$ over ΔT. Solving Eq. 2.9 for TPSP_N using $\Delta V_N = 0$, we get:

$$\mathrm{TPSP}_N = V_N \frac{\Delta T}{\tau} \tag{3.3}$$

Let us denote the highest possible magnitude of the TPSP stemming from the noise as $\mathrm{TPSP}_N^{\max}$. The maximum membrane potential V_N should not cross the spiking threshold V_{th}. Therefore, the border condition for $\mathrm{TPSP}_N^{\max}$ is:

$$\mathrm{TPSP}_N^{\max} = V_{\mathrm{th}} \frac{\Delta T}{\tau} \tag{3.4}$$

The firing frequency of an LIF neuron in function of the input magnitude for different values of the leak term τ is plotted in Fig. 3.14a [Woźniak et al., 2017c]. The $\mathrm{TPSP}_N^{\max}$ of the input above which an LIF neuron with $\tau = 0.1, V_{\mathrm{th}} = 13, \Delta T = 0.05$ will fire is for equal to $\mathrm{TPSP}_N^{\max} = 6.5$.

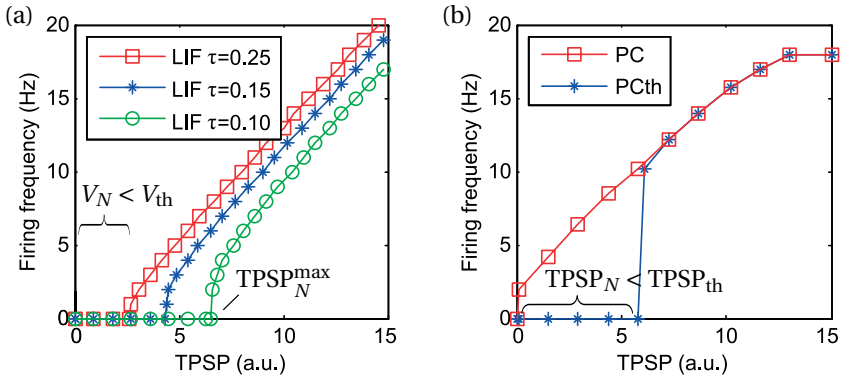


Figure 3.14: **The firing frequency of an LIF and a phase-change neuron** (a) An LIF neuron with $V_{th} = 13$ for varying τ. (b) A phase-change (PC) neuron with $V_{th} = 30.8\mu S$, and a phase-change neuron with integration threshold (PCth) with $TPSP_{th} = 6.1$. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

The frequency response of a standard phase-change (PC) neuron is illustrated in Fig. 3.14b [Woźniak et al., 2017c], where the TPSP range of 1 to 13 is mapped linearly to a p_w range of 50 ns to 500 ns. The leak term of the LIF neuron offsets the slope of the firing frequency to higher input magnitudes, as depicted in Fig. 3.14a. In a phase-change neuron, to achieve a similar effect on the shape of the slope, we introduce an integration threshold $TPSP_{th}$ in PCth, illustrated in Fig. 3.14b.

The PCth neuron is noise-robust, as it eliminates noise by integrating only values of TPSP > $TPSP_{th}$. The firing frequency of a phase-change neuron with integration threshold in Fig. 3.14b is similar to the response of a neuron with a leak, but instead of tuning τ, $TPSP_{th}$ is tuned to be above $TPSP_N^{max}$. For input encoding, in which the true positive signals cluster into correlated patterns, such as in the correlated group encoding described in Sec. 2.2.2, PCth performs as well as an LIF neuron for the noisy inputs. The neuron operation in an SNN architecture will be demonstrated in the next section.

3.4 Conclusions

An implementation of the basic theoretical primitives of the spiking model was discussed in the context of neuromorphic hardware. In particular, phase-change memristors were considered for an analog design of synapses and neurons. A 1-PCM synapse with an A-STDP learning mechanism was proposed along with a noise-robust phase-change neuron with integration threshold, which provide the building blocks for an all-phase-change-based SNN. Owing to technological constraints, these blocks differ to some extent from the theoretical SNN model, described in Chapter 2. In the remaining parts of the thesis, we utilize these phase-change-based building blocks for concrete applications to experimentally validate their operation.

4 Architectures for pattern learning

In this chapter, we propose phase-change-based neuromorphic architectures for pattern learning and experimentally demonstrate their operation. In Sec. 4.1, we show how to learn patterns from correlated inputs using a single LIF neuron with phase-change-based synapses operating according to the A-STDP rule. Next, in Sec. 4.2, we extend this approach to an all-phase-change implementation of a neuron, with a record number of 1M phase-change synapses, learning from a more challenging weakly-correlated input. Lastly, in Sec. 4.3, we implement a network of all-phase-change spiking neurons, for which we propose a WTA feedback scheme with level-tuned neurons as an alternative to the lateral inhibition WTA. We use it to experimentally demonstrate simultaneous learning of multiple patterns.

4.1 Learning a correlated pattern

We propose a neuromorphic phase-change-based architecture for learning a correlated pattern from parallel input spike trains [Woźniak et al., 2016]. The architecture, illustrated in Fig. 4.1, comprises an array of m synapses of 1-PCM type, discussed in Sec. 3.2, coupled with an LIF neuron soma, implemented as described in Sec. 2.3.2. The synapses learn based on the relative timing of the pre- and post-synaptic spikes using the A-STDP introduced in Sec. 3.2.3.

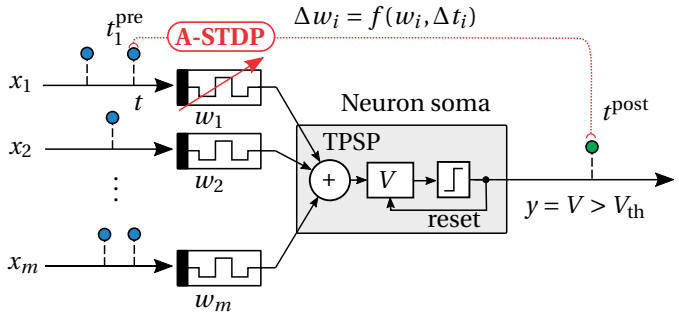


Figure 4.1: **An architecture for learning correlated patterns** 1-PCM phase-change synapses operating using A-STDP are coupled with an LIF neuron. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

We apply the proposed architecture for the task of learning a correlated pattern Q from temporally-coded input. In particular, we use three 20×20-pixel binary patterns Q_1, Q_2, Q_3 corresponding to the images of letters 'I', 'B', and 'M'. The information about each pattern is present in the input for a 60 s time interval, as schematically illustrated in Fig. 4.2. Following the dataset generation scheme from Sec. 2.2.2, the synapses corresponding to the pattern pixels receive mutually correlated spikes with $c = 1$ at time instances drawn from a Poisson distribution with a rate $r_Q = 1$ Hz. The remaining synapses receive uncorrelated spikes representing Poissonian noise with a rate $r_N = 1$ Hz.

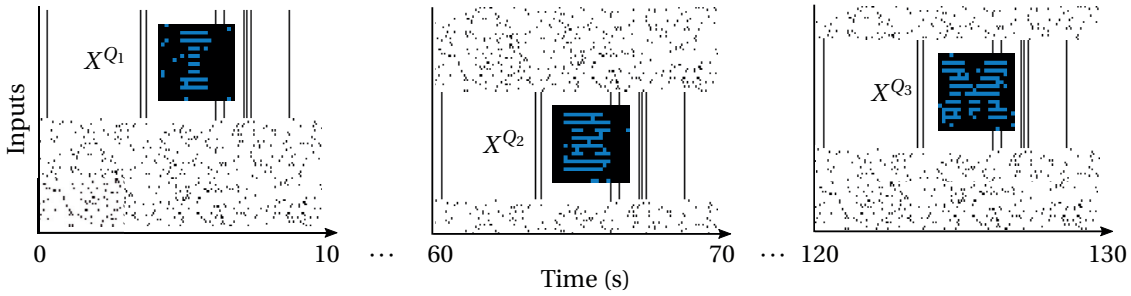


Figure 4.2: **Inputs with correlation coded information about the patterns** The correlated group with $c = 1$, corresponding to the pattern inputs X^Q changes each 60 s. The pattern and the noise inputs have equal rates of 1 Hz. For each pattern, sample 50 correlated inputs X^Q and 50 noise inputs X^N are plotted and displayed in an arbitrarily sorted order to visualize the correlated nature of the inputs forming vertical lines and the changes of the correlated group.

We provided the inputs consecutively row by row to 400 phase-change synapses of the proposed architecture. The operation of synapses was simulated using the model from Sec. 3.1.1 as well as realized experimentally using the platform described in Sec. 3.1.3. In the experiment, we mapped the synaptic weights to a 10 µS – 60 µS range of conductance, and used crystallizing pulses with $I = 70$ µA and width of 100 ns – 200 ns.

4.1.1 Learning results

The inputs appearing during the first 60 s increasingly lead to emission of post-synaptic spikes during the pattern appearance. The arrival of the correlated pattern concurrently activates a large number of synapses that provide a significant contribution in increasing the membrane potential above the threshold V_{th} and trigger a neuron-firing event. Therefore, the temporally correlated inputs increasingly gain control of the neuron firing, and the corresponding synapses increase their weight owing to the feedback mechanism. This results in a progressive increase in the crystallization of the correlated group of phase-change synapses for each appearance of the image pattern at inputs $x_i \in X^Q$. Conversely, the uncorrelated synapses are subject to negative feedback as acausal input-firing events at $x_i \in X^N$ trigger depression [Woźniak et al., 2016].

The visualization of the weights from the experimental realization at $t = 60\,\text{s}$, shown in Fig. 4.3a, demonstrates that the neuron captured the pattern from the inputs. Fig. 4.3b illustrates the distribution of the normalized synaptic conductances of the 1-PCM synapses. The bi-modal distribution correctly corresponds to the group of uncorrelated noise inputs X^N and correlated pattern inputs X^{Q_1}. The shape of the modes is impacted by the variability of the phase-change devices, most of which saturate at conductance $\approx 39\,\mu\text{S}$ that corresponds to $w_i \approx 0.84$.

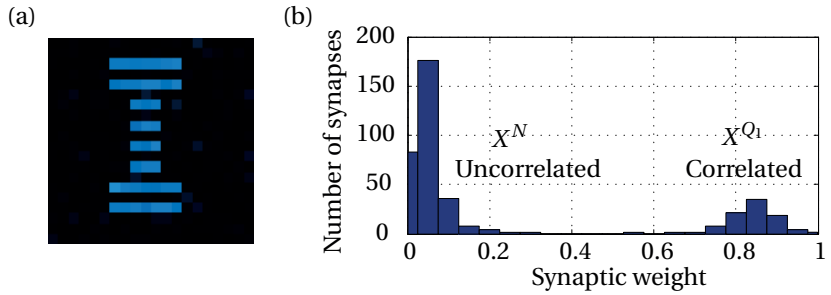


Figure 4.3: **Weights of a learned neuron** (a) Visualization of the weights at time $t = 60\,\text{s}$. (b) A histogram illustrating the bi-modal distribution of the weights of 1-PCM synapses using A-STDP. The shape of the modes is impacted by the variability of the phase-change devices. Figure adapted from [Woźniak et al., 2016], © 2016 IEEE.

4.1.2 Relearning results

Starting from $t = 60\,\text{s}$ and then from $t = 120\,\text{s}$, the input distribution changes and different inputs receive the correlated spikes. Uncorrelated spike arrival at inputs that used to belong to the correlated group $x_i \in X^{Q_1} \setminus X^{Q_2}$ leads to depression of the respective synapses, whereas correlated spike arrival at inputs that used to receive noise $x_i \in X^{N_1} \cap X^{Q_2}$ leads to potentiation. In consequence, the neuron constantly re-learns the statistics of the input distribution.

The re-learning progress is illustrated in Fig. 4.4 [Woźniak et al., 2016]. Fig. 4.4a shows the time-lapse snapshots. Specifically, the first row shows the synaptic input signals; the second row shows the weight of the corresponding synapses from the software simulation that is based on the effective amorphous thickness u_a, and the last row shows the experimental realization in which the synaptic weight is encoded in the phase-change cell conductance G. Fig. 4.4b shows the evolution of the average conductance of the synapses corresponding to the pattern pixels in comparison to the average conductance of the synapses corresponding to noise inputs. The set of the synapses for which the mean is calculated is changed every 60 s-period to reflect the ground truth distribution. Because some inputs remain correlated $X^{Q_k} \cap X^{Q_{k+1}} \neq \emptyset$, re-learning patterns Q_2 and Q_3 proceeds faster than learning the initial pattern Q_1.

We assess the quality of the learned patterns by calculating visualization MSE, defined in Eq. 2.11. The values are plotted in Fig. 4.4c and the averaged MSE over each pattern's second half

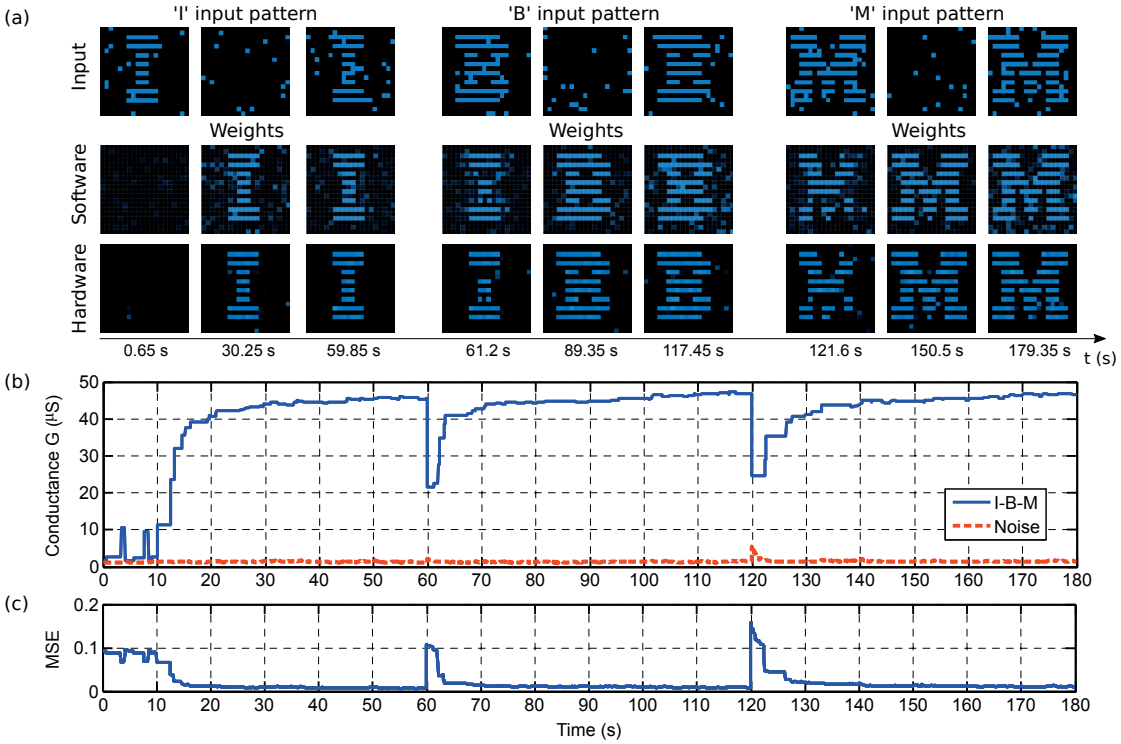


Figure 4.4: **A single neuron relearning correlated patterns using A-STDP** (a) Snapshots of the input signals and synaptic weights in the software simulation and in the hardware experiment at times indicated on the t-axis. (b) Average conductance of the synapses corresponding to the input images in comparison to the average conductance of the synapses corresponding to noise inputs. (c) MSE of synaptic weights versus expected input pattern at each moment in time. Figure from [Woźniak et al., 2016], © 2016 IEEE.

of the exposition time for the three patterns is 0.96%. The results show that the architecture is capable of learning and re-learning temporally-correlated patterns through the feedback provided by neuronal activation and operation of A-STDP.

4.2 Learning a weakly correlated pattern

In a more realistic setting, the information about the pattern may not appear in the form of perfectly correlated spikes. Therefore, in this section we consider inputs containing information in a form of weakly correlated spikes with correlation coefficient $c < 1$. The uncorrelated noise appears then also on the pattern inputs X^Q, as illustrated for $c = 0.2$ in Fig. 4.5a for 50 pattern inputs and 50 noise inputs.

Learning from weakly correlated inputs containing information about a pattern Q, such as one illustrated in Fig. 4.5b, is challenging both for the neuron soma and the synapses. In the perfectly correlated case, the synapses at inputs X^Q receive complete information about the

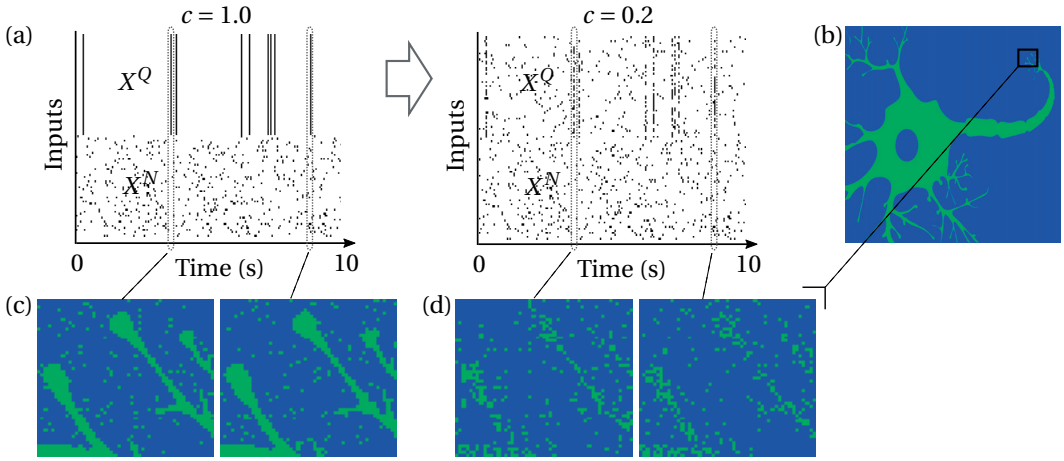


Figure 4.5: **Weakly correlated inputs** (a) The pattern inputs X^Q receive uncorrelated noise for $c < 1$. Subfigure adapted from [Woźniak et al., 2017c], © 2017 IEEE. (b) The reference pattern Q. (c) For $c = 1$, correlated inputs contain the complete information about the pattern. (d) For $c = 0.2$, correlated inputs contain incomplete and inconsistent information about the pattern.

pattern during correlated activity, as illustrated in Fig. 4.5c. In the weakly correlated case, the information about the pattern is incomplete and inconsistent over time, as illustrated in Fig. 4.5d. In consequence, when the pattern appears at the input, fewer pattern synapses are activated and the TPSP observed by the neuron soma is lower than in the highly correlated case. Simultaneously, an arrival of an uncorrelated spike on a highly potentiated pattern input $x_i \in X^Q$ leads to a high TPSP contribution to the membrane voltage V, even though there is no pattern appearing at this time instance. Therefore, to provide high spiking accuracy, a noise-robust neuron soma implementation becomes essential.

In the rest of this section, we consider learning from weakly correlated inputs using two noise-robust neuron soma implementations. We assess them on the tasks of pattern detection and pattern visualization, introduced in Sec. 2.4. Next, we discuss how to improve the visualization accuracy for weakly correlated inputs. We present an architecture for high-accuracy weakly correlated pattern visualization and compare the experimental results.

4.2.1 Results for an all-phase-change neuron

We execute an experiment to compare weakly correlated pattern learning capabilities of two noise-robust neuron soma implementations: an LIF and a phase-change neuron with integration threshold (PCth), introduced in Sec. 3.3.1. The experimental setup consists of a single neuron with 10^6 phase-change synapses operating according to A-STDP, as in the architecture from Fig. 4.1 (page 47). The LIF neuron soma is emulated in software, as discussed in Sec. 2.3.2. The PCth neuron soma is implemented using one additional phase-change device, so that the neuron is an all-phase-change neuron using $10^6 + 1$ cells from the prototype platform described in Sec. 3.1.3.

The inputs comprise weakly correlated information about a 1000×1000-pixel image from Fig. 4.5b. The size of the correlated group corresponding to the pattern is $|X^Q| = 203,143$. The pattern and the noise appear with equal rates: $r_Q = r_N = 1\,\text{Hz}$, $\Delta T = 50\,\text{ms}$. Consecutive steps of operation of the neuromorphic architecture for $c = 0.2$ are shown in Fig. 4.6. The spikes appearing at the inputs of a neuron are modulated by the synaptic weights (Fig. 4.6a), and contribute to the synaptic output (Fig. 4.6b), which is integrated into the membrane potential (Fig. 4.6c). We may formally interpret the values using notation from Sec. 2.2.2. When no pattern is appearing ($\neg Y$), the system observes activity generated by the noise and receives on average $|X^Q|\Pr(X_i^Q|\neg y^Q) + |X^N|\Pr(X_i^N)$ input spikes. This results in a noise component of TPSP, denoted by TPSP_N in Fig. 4.6b, which should not cause a spike in a properly tuned noise-robust neuron soma. For the LIF we use $V_{\text{th}} = 13$, $\tau = 0.15$, and for the PCth we use $V_{\text{th}} = 30.8\,\mu\text{S}$, $\text{TPSP}_{\text{th}} = 6.1$. Lastly, Fig. 4.6d demonstrates that both neuron soma implementations become selective to the weakly correlated pattern.

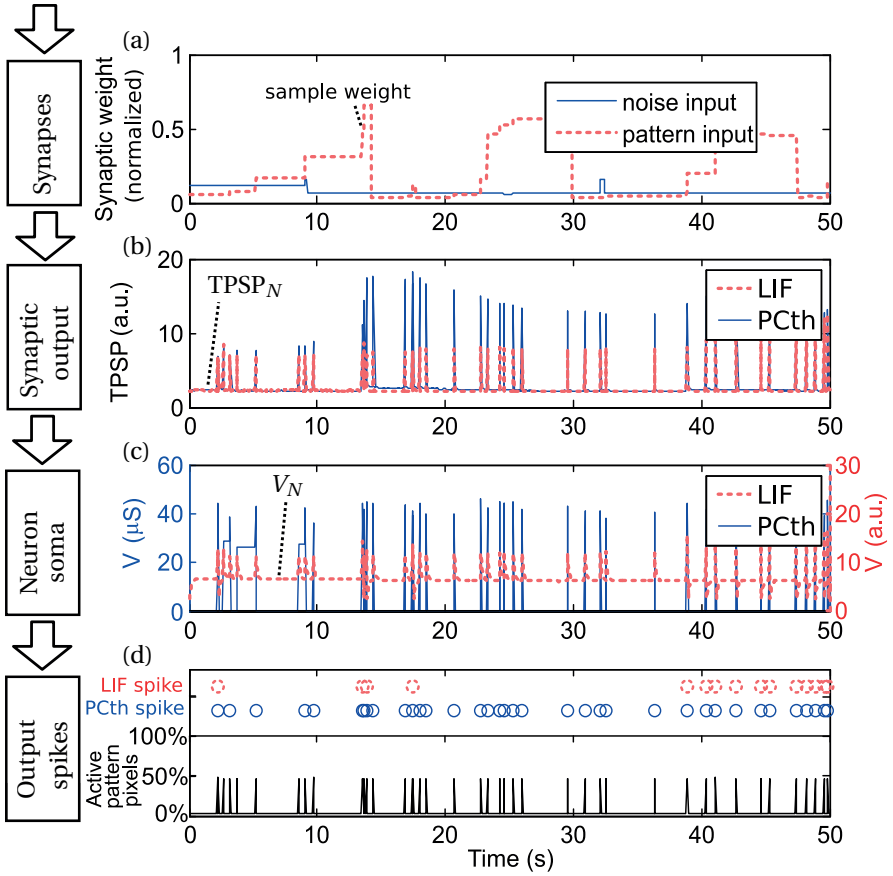


Figure 4.6: **Comparison of LIF and PCth soma applied for pattern detection** $c = 0.2$, $|X^Q| = 203,143$, $|U| = 1\text{M}$ (a) The inputs are weighted by synaptic weights, which evolve over time according to A-STDP. (b) TPSP observed by both types of neurons is very similar. (c) V differs between the neurons, because the noise is handled differently. (d) After some initial time, the pattern appearance consistently drives the neuronal activity of both types of neurons. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

To assess the pattern detection capabilities, we calculate the cross-correlation between input spiking and neuron firing for the correlation coefficients $c = 1$ and $c = 0.2$. The cross-correlations plotted in Fig. 4.7 are the same for both implementations, and demonstrate that even for $c = 0.2$ the neuronal firing is driven exclusively by the pattern inputs. The correlation to the noise inputs is near zero. Therefore, we conclude that both LIF and PCth provide a noise-robust neuron soma implementation that enables reliable weakly correlated pattern detection with A-STDP.

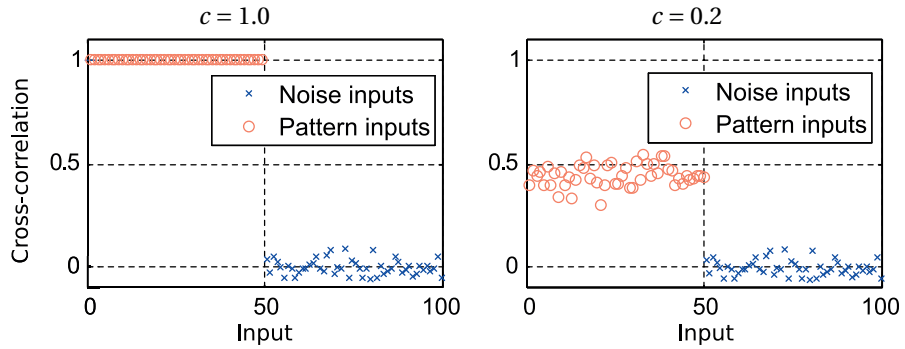


Figure 4.7: **Pattern detection assessment using cross-correlation** After an initial period of learning, the LIF and the PCth fire at exactly the same time instances, correlated with the activation of the pattern inputs. The cross-correlation calculated for 400 s of operation is the same for both neurons. For improved visualization, the values are presented for the first 50 X^Q inputs and the first 50 X^N inputs. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

To assess pattern visualization capabilities, we plot the synaptic weight evolution of the PCth neuron for $c = 0.2$ in Fig. 4.8a. The pattern weights reach high values, but are frequently depressed, similarly to the sample weight in Fig. 4.6a. The neuronal firing remains unaffected after such weight resets because on the average (bold lines) the synaptic weights corresponding to the pattern have a higher value than those corresponding to the noise. Nevertheless, a complete pattern reconstruction is difficult, and the final weights, visualized in Fig. 4.8b, store a rudimentary image of the reference pattern Q from Fig. 4.5b.

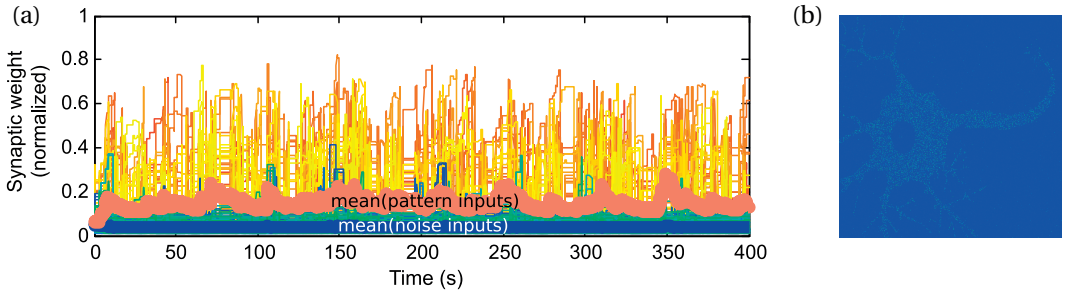


Figure 4.8: **Weakly correlated pattern visualization** $c = 0.2$ (a) Weight evolution plot. Thin lines: 100 sample weights corresponding to pattern inputs (red to yellow) and noise inputs (blue to green). Bold lines: means. (b) Final weights visualization. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

4.2.2 Towards accurate weakly correlated pattern visualization

An accurate visualization of a weakly correlated pattern might be essential in applications that require a deeper insight than just detecting the appearance of the pattern. In this section, we demonstrate that an accurate information about the entire pattern is indeed collected throughout the course of learning in the synaptic weights operating using A-STDP. For fast computation, we convey the analysis using a 100×100-pixel image presented in Fig. 4.9a.

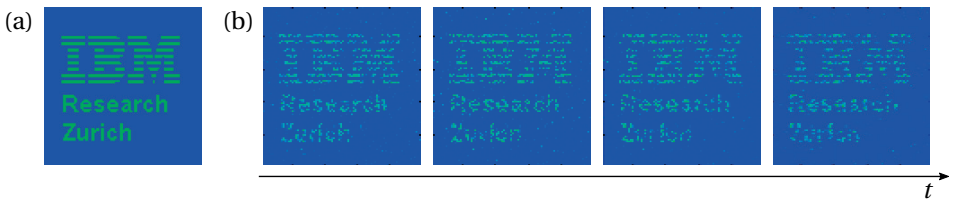


Figure 4.9: **Weight snapshots for weakly correlated inputs** (a) The reference pattern Q. (b) Weights' snapshots from a simulation of an LIF neuron operating with A-STDP for $c = 0.2$.

We analyze the weight evolution in a simulation using LIF and A-STDP operating for weakly correlated inputs with $c = 0.2$ for a period of 100 s. Fig. 4.9b presents obtained weights' snapshots that contain distorted memories of the pattern Q. We notice that the highly-potentiated synapses contain subsets of the correct pattern Q, and these subsets vary between the snapshots. Therefore, the information about the entire pattern is a function of the history of the weights, and it may be retrieved through post-processing the weights' evolution.

Pattern visualizations calculated using various post-processing approaches are presented in Fig. 4.10. In particular, visualizations in Fig. 4.10a were obtained by averaging the values of

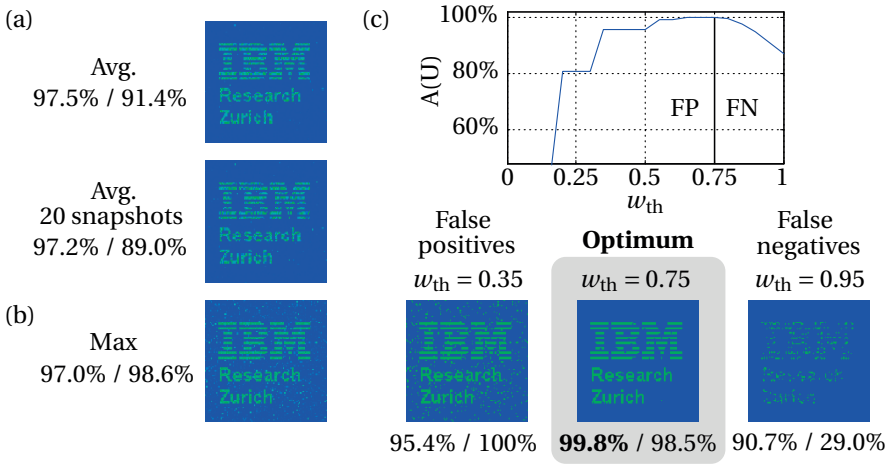


Figure 4.10: **A-STDP weights post-processing for** $c = 0.2$ In each case two visualization accuracy values are reported: A(U) / A(Q). (a) Averaging the weights enables to reconstruct the pattern. (b) Maximum operation reconstructs the pattern, but results in high background noise levels. (c) Thresholding applied to the results of the maximum operation obtains the highest A(U) accuracy. The choice of the threshold w_{th} is an optimization problem.

the synaptic weights over the entire experiment, or just over a given number of snapshots, and performing 0-1 normalization. The drawback of the calculation of an average is that a high dynamic range is required to accurately sum multiple snapshots, i.e. 20 snapshots with 8-bit resolution require a 13-bit accumulator. A simple alternative is to save the maximum of the weights, which provides a high-accuracy reconstruction of the pattern, but also captures the background noise, as illustrated in Fig. 4.10b. To remove the noise, thresholding can be applied, as depicted in Fig. 4.10c. The threshold w_{th} impacts the final accuracy and its choice is an optimization problem: lower thresholds provide a reconstruction of the entire pattern at the expense of allowing noisy pixels (false positive pixels); whereas higher thresholds provide partial reconstruction of the pattern (false negative pixels) with no noise.

The results demonstrate that with weights post-processing a high-accuracy pattern visualization is possible for a neuron operating with A-STDP. On the other hand, the discussed approaches introduce additional complexity to the system and require to double the number of memory elements to store intermediate post-processing results, which limits their practical applicability.

4.2.3 An architecture using A-STDP with selective depression

We propose a variant of A-STDP directly capable of accurately visualizing weakly correlated patterns using 1-PCM synapses [Woźniak et al., 2017c]. We introduce a selective depression mechanism that uses a threshold q_{th} on the synaptic output q_i to prevent depression of highly potentiated inputs. Because learning is performed on spike arrival, the value of q_i is available as an intermediate result behind a synapse receiving a spike. The implementation becomes straightforward if we move the A-STDP execution behind the synapses, as illustrated in Fig. 4.11. Then, in the learning logic, we prevent the application of reset pulses to the synapses corresponding to the inputs with $q_i \geq q_{\text{th}}$. These inputs typically correspond to the pattern owing to their spiking probabilities. When a neuron spikes for a pattern appearance Y, the probability of a pattern input spiking, and thus potentiation, is higher than for a noise

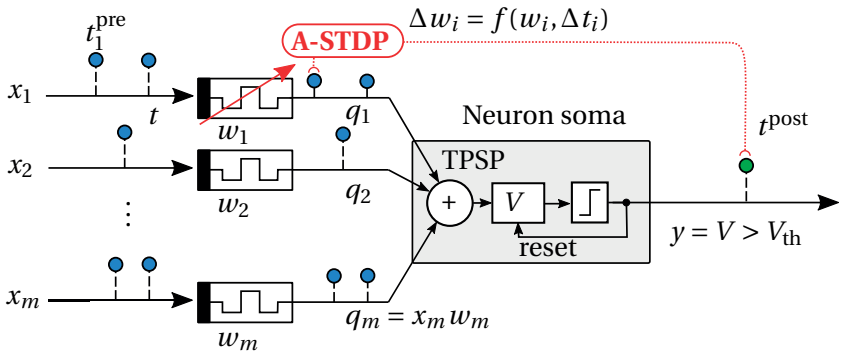


Figure 4.11: **Neuron architecture for A-STDP with selective depression** A-STDP execution is moved behind the synapses to enable thresholding whilst avoiding additional explicit read of the weights.

input: $\Pr(X_i^Q|y^Q) > \Pr(X_i^N)$, whereas during depression it is lower: $\Pr(X_i^Q|\neg y^Q) < \Pr(X_i^N)$. In consequence, the probability that a synapse is potentiated multiple consecutive times prior to a depression is higher for pattern inputs than for noise inputs.

The number of consecutive potentiations of the synapses corresponding to the pattern inputs depends mainly on the timing parameters of the learning rule and the correlation coefficient. For instance, in Fig. 4.12a we plot the distribution of consecutive potentiations obtained in the pattern detection experiment in Sec. 4.2.1. The distribution is bi-modal, with median number of potentiations equal to 2 for noise inputs, and equal to 8 for pattern inputs. The pattern inputs can be clearly distinguished as receiving 5 or more potentiations. The higher number of potentiations leads to a separation of the synaptic weights' values of the correlated inputs from the noise. This separation enables the selection of an appropriate synaptic output threshold q_{th} for the selective depression mechanism.

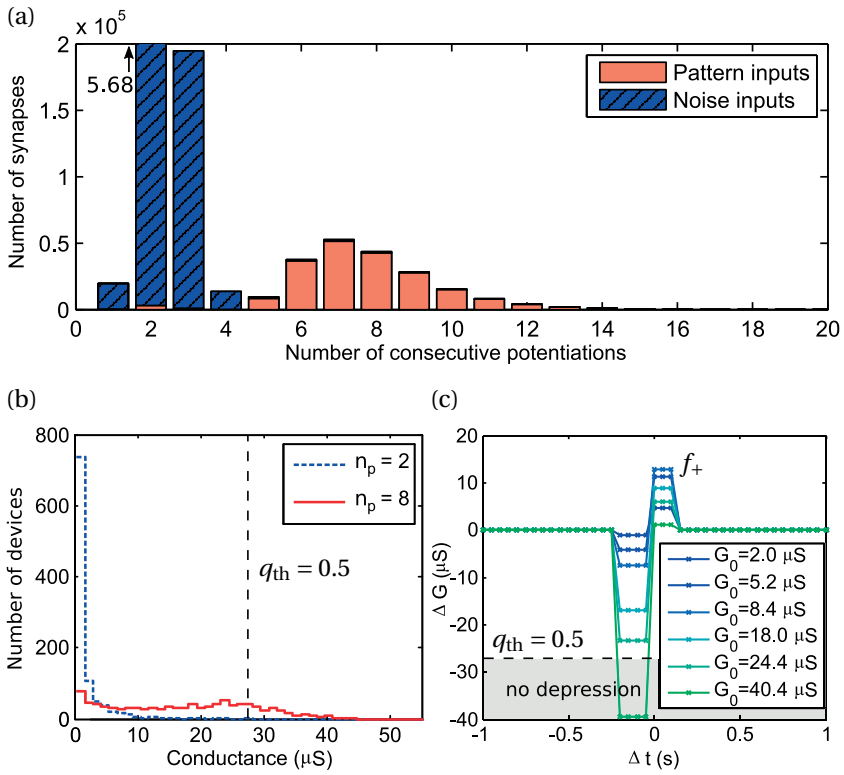


Figure 4.12: **Determining q_{th} for selective potentiation** (a) The maximum number of consecutive potentiations is higher for the synapses corresponding to the pattern inputs than for the synapses corresponding to the noise inputs. (b) Distributions of synaptic conductances after application of the median number of potentiating pulses for the noise inputs $n_p = 2$ and for the pattern inputs $n_p = 8$. (c) Asymmetric STDP implementation for phase-change synapses. For the synaptic potentiation, we use crystallizing pulses with $I = 130\,\mu$A and $p_w = 50$ ns, whereas for the depression we use reset pulses with $I = 450\,\mu$A. Selective depression with $q_{\text{th}} = 0.5$ corresponds to limiting the depression ΔG to 27.5 µS. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

However, in a phase-change synaptic implementation, the potentiation f_+ also involves the inter-device variability. In Fig. 4.12b, we plot the distribution of phase-change synaptic conductances for a subset of 1000 devices after a given number of potentiations n_p. Based on the characterization results, an appropriate value for the selective depression threshold q_{th} is 0.5 (normalized). The noise synapses do not reach this threshold, whereas the pattern synapses will eventually cross it and remain potentiated. Fig. 4.12c illustrates the region in the A-STDP response in which the selective depression threshold prevents the depression of highly potentiated synapses.

A-STDP with selective depression is similar to the threshold-based post-processing in form $(\max_t w_i(t)) > w_{\text{th}}$, that finds the weights that were potentiated above w_{th} at some point in time. Performing that post-processing corresponds to reverting the application of the depression events for the synaptic weights that were potentiated to a high value in the past, whereas A-STDP with selective depression prevents such depression events from occurring.

4.2.4 Results for A-STDP with selective depression

We experimentally implement A-STDP with selective depression using 10^6 phase-change synapses in a single neuron [Woźniak et al., 2017c]. We extend the algorithm described in Sec. 4.2.1 to introduce the selective depression parameter q_{th}, and to analyze the impact of selective depression for inputs with correlation coefficients $c = 0.2$ and $c = 1$. Specifically, the regular A-STDP with LIF neuron soma is compared with an A-STDP extended with selective depression for $q_{\text{th}} = 0.5$ coupled with a PCth neuron soma. We use the same setup and the 1000×1000-pixel image as in the previous experiment from Sec. 4.2.1.

The synaptic weight evolution for $c = 0.2$ with selective depression is plotted in Fig. 4.13. If a synapse collects sufficient evidence that an input belongs to the correlated group, measured by crossing q_{th}, it converges to a high value. As a result, pattern reconstruction improves in comparison to regular A-STDP operation, although because of the device variability also a few synapses corresponding to the noise converge to high values.

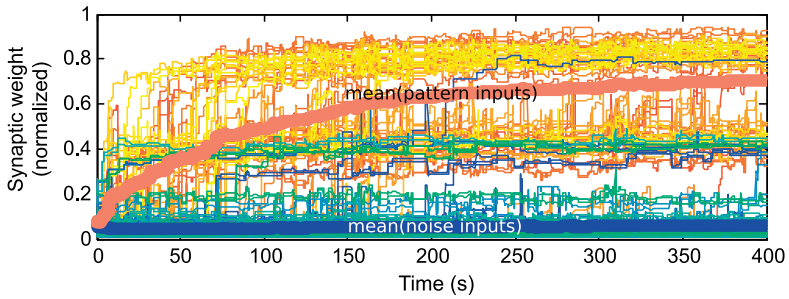


Figure 4.13: **Weakly correlated pattern visualization with selective depression** Weight evolution plot for $c = 0.2$ with $q_{\text{th}} = 0.5$. Thin lines: 100 sample weights corresponding to pattern inputs (red to yellow) and noise inputs (blue to green). Bold lines: means. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

The pattern visualizations and weight histograms at $t = 400\,\mathrm{s}$ are compared for $c = 0.2$ and $c = 1$ using A-STDP with and without selective depression, as illustrated in Fig. 4.14. For weakly correlated inputs, regular A-STDP (see Fig. 4.14a) cannot visualize the pattern well. A-STDP with selective depression (see Fig. 4.14b) performs better and provides a clearly separated distribution of the final weights that yields a high-contrast visualization of the pattern Q. For highly correlated inputs, we analyze whether the use of selective depression impacts performance. As shown in Fig. 4.14c-d, both approaches reconstruct the pattern, but in the case of selective depression a few noise synapses cross the q_{th} threshold. To quantitatively assess the performance, we calculate the overall visualization accuracy $A(U)$ and the pattern accuracy $A(Q)$, defined in Sec. 2.4. The values are presented in Table 4.1. A-STDP with selective depression outperforms regular A-STDP by 62.7 percentage points for $c = 0.2$ pattern reconstruction, while maintaining a similar overall and the same pattern accuracy for $c = 1$.

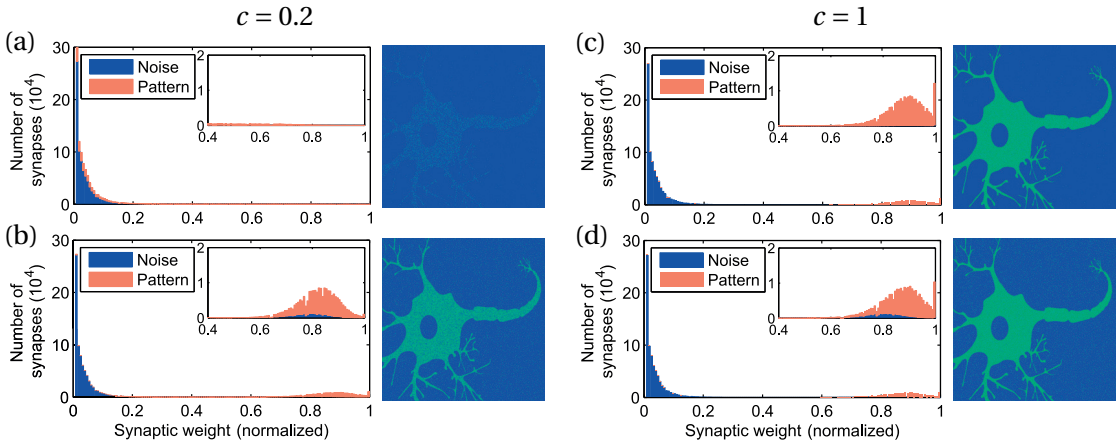


Figure 4.14: **Comparison of pattern visualization for $c = 0.2$ and $c = 1.0$** Pattern visualization for $c = 0.2$: (a) regular A-STDP (using an LIF neuron soma), (b) A-STDP with selective depression (using a PCth neuron soma). Pattern visualization for $c = 1.0$: Both (c) A-STDP, and (d) A-STDP with selective depression, learn the entire pattern. Figure from [Woźniak et al., 2017c], © 2017 IEEE.

Table 4.1: **Accuracy for the pattern visualization task** Overall ($A(U)$) and for the correlated pattern inputs only ($A(Q)$). Table from [Woźniak et al., 2017c], © 2017 IEEE.

	$c = 0.2$		$c = 1$	
	$A(U)$	$A(Q)$	$A(U)$	$A(Q)$
A-STDP	83.6%	20.8%	98.5%	95.5%
with sel. depr.	94.9%	83.5%	97.0%	95.5%

With 10^6 phase-change synapses, our experimental realization of a neuron is the largest to date in terms of the number of synapses. It has two orders of magnitude more synapses than the average for a biological neuron equal to 10^4, and the nanoscale phase-change synapses

are over an order of magnitude smaller than the microscale biological synapses. This is a demonstration that neuromorphic systems provide capabilities to go beyond what is observed in the nature in terms of the size of a neuron.

4.3 Learning multiple correlated patterns

A single neuron provides capabilities for pattern learning and re-learning. However, they are practically constrained to handling one pattern at-a-time, because a neuron has a single binary output. A natural extension for handling multiple patterns at-a-time is to use multiple outputs from a network of spiking neurons in a WTA architecture, described in Sec. 2.5. In this section, we discuss how to implement such network using phase-change neurons operating with phase-change synapses. Firstly, we introduce a power-efficient WTA architecture with level-tuned neurons as an alternative to the classic lateral inhibition WTA, and demonstrate results from an all-phase-change implementation of a neuromorphic SNN. Secondly, we propose a learning mechanism that incorporates the WTA results to ensure consistent high quality pattern visualization during operation of A-STDP in a WTA architecture.

4.3.1 WTA with level-tuned neurons

WTA with level-tuned neurons aims to improve the efficiency of a phase-change-based SNN for operation with temporally coded inputs in comparison to lateral inhibition WTA, discussed in Sec. 2.5. The feedback evoked within a neuron is moved from resetting the membrane potential V_j in lateral inhibition to gating the neuronal inputs of level-tuned neurons.

A neuromorphic architecture implementing lateral inhibition WTA, presented in Fig. 4.15a, may operate as follows: on an arrival of a correlated group of inputs, signals pass through

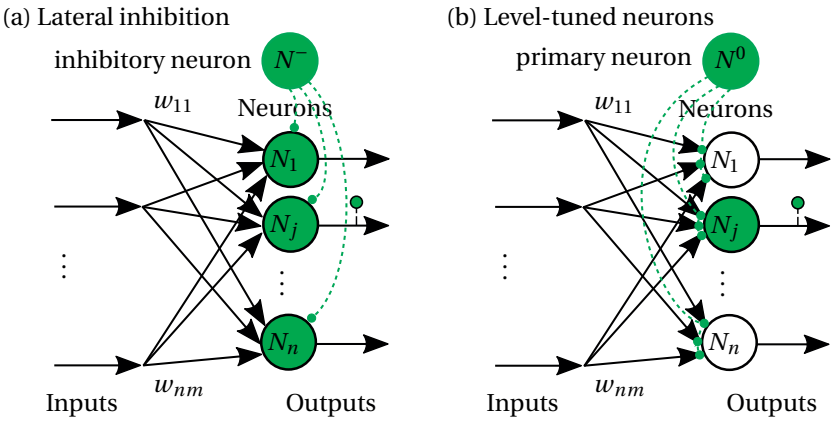


Figure 4.15: **WTA operation in a network** (a) In lateral inhibition, all neurons are enabled by default and then a disabled, except from the winning neuron. (b) All level-tuned neurons are disabled by default and then the winning neuron is enabled by the primary neuron.

all the m synapses of all the n neurons (nm read pulses). Then, n neurons increase their membrane potential V_j (n crystallzing pulses), which is compared with spiking threshold V_{th} (n read pulses). Next, assuming that at least one neuron crossed its threshold, the lateral inhibition mechanism picks a single winner and resets the membrane potentials of all neurons (n amorphizing pulses). The neurons activated in the network prior to the inhibition are highlighted in Fig. 4.15a, and the number of pulses required during operation is $(nm+n)p_R + np_C + np_A$, where p_R, p_C, p_A denotes a read, crystallizing and amorphizing (reset) pulse respectively.

A neuromorphic architecture implementing WTA with level-tuned neurons operates in an opposite manner to lateral inhibition: the key idea is to have all the neurons disabled by default and enable them based on the activation of an additional primary neuron. On an arrival of a correlated group of inputs, signals pass through m synapses of the primary neuron (m read pulses). Based on the level of its TPSP^0, particular neurons in the network are activated. Assuming a 1-WTA mode (a single winner), one neuron will be activated: the signal will travel through its synapses (m read pulses), increase the membrane potential (1 crystallizing pulse), which compared with the spiking threshold (1 read pulse) may lead to an output spike and consecutive membrane potential reset (1 amorphizing pulse). The neurons activated in the network are highlighted in Fig. 4.15b, and the number of required pulses is $(2m+1)p_R + 1p_C + 1p_A$. This is less than for the lateral inhibition, and the advantage of the level-tuned WTA increases with the size of the network.

The wining neuron N_j is activated at time t_X based on the level $L^0(t_X) = \text{TPSP}^0(t_X)$ of the primary neuron, provided that it satisfies $L^0(t_X) \in < l_j^{\min}, l_j^{\max} >$, as illustrated in Fig. 4.16. The level L^0 is typically characteristic of a given pattern. However, it may happen that different

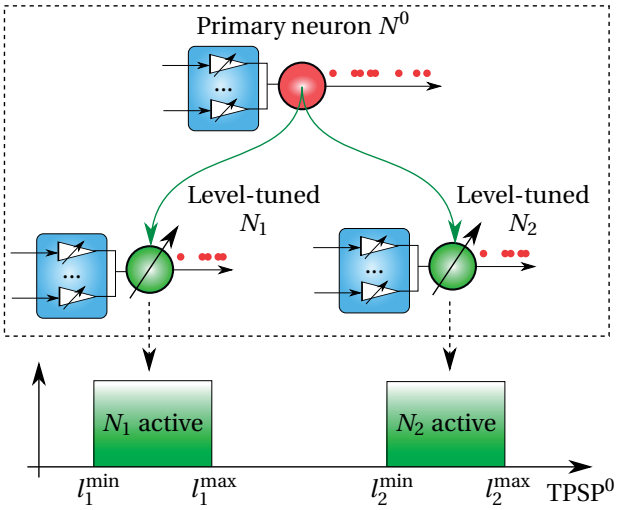


Figure 4.16: **Enabling level-tuned neurons** The level of TPSP^0 of a primary neuron is used to enable the regular neurons. Adapted from [Pantazi et al., 2016], © IOP Publishing. Reproduced with permission.

patterns X and Y will evoke the same level of the primary neuron: $L^0(t_X) = L^0(t_Y)$. If we treat L^0 as a hashing function, this situation resembles the problem of hashing collision. The solution might be to change the function L from the calculation of TPSP to a different one, which again corresponds to the challenge of determining the optimal hashing function.

4.3.2 Multiple pattern learning results

We experimentally demonstrate operation of an architecture comprising two level-tuned neurons and one primary neuron [Pantazi et al., 2016]. The neurons are all-phase-change with 1-PCM synapses and a phase-change-based soma, implemented in the experimental platform described in Sec. 3.1.3. Constant-width 100 ns crystallizing pulses are used for A-STDP and variable-width 100 ns – 1000 ns crystallizing pulses are used for the phase-change soma. The inputs receive spikes with temporally-coded information about two patterns, where the patterns appear in form of correlated groups with a mean rate of $r_Q = 1$ Hz, and the remaining inputs receive Poissonian noise with a mean rate of $r_N = 1$ Hz.

Firstly, we consider a system with 400 inputs receiving two synthetic patterns comprising correlated groups of 50 inputs and 30 inputs with $c = 1$, illustrated in Fig. 4.17a. For improved readability, only 150 inputs are visualized. After an initial period of learning, the level TPSP^0 of the primary neuron becomes different for each pattern, and enables either the first or the second level-tuned neuron. The neurons learn the respective patterns and correctly spike during their appearance. The system maintains the performance also for weakly correlated inputs. Fig. 4.17b illustrates operation for two patterns of 50 inputs with $c = 1$ and 50 inputs with $c = 0.6$. The neurons correctly detect all respective pattern appearances.

Secondly, we consider a system with 10000 inputs receiving two correlated groups corresponding to two overlapping image patterns, visible in the input snapshots in Fig. 4.18a. The primary neuron learns the overall input statistics that comprises both patterns, as visualized in the snapshots in Fig. 4.18b(1) and in the mean weights plot in Fig. 4.18c(1). Then, it selectively enables the level-tuned neurons, which learn only particular patterns, as visualized in the snapshots in Fig. 4.18b(2-3) and in the mean weights plots in Fig. 4.18c(2-3). At $t \approx 95$ s, the level-tuned neurons contain an accurate representations of the patterns. However, occasional drops in the conductance corresponding to the common part of the patterns may occur owing to the asymmetric conductance response of 1-PCM synapses. The level-tuned neurons continue to spike accurately for each of the patterns, and quickly relearn the entire patterns. Nevertheless, in the next section we propose to enhance the learning mechanism in order to consistently maintain the accurate representation.

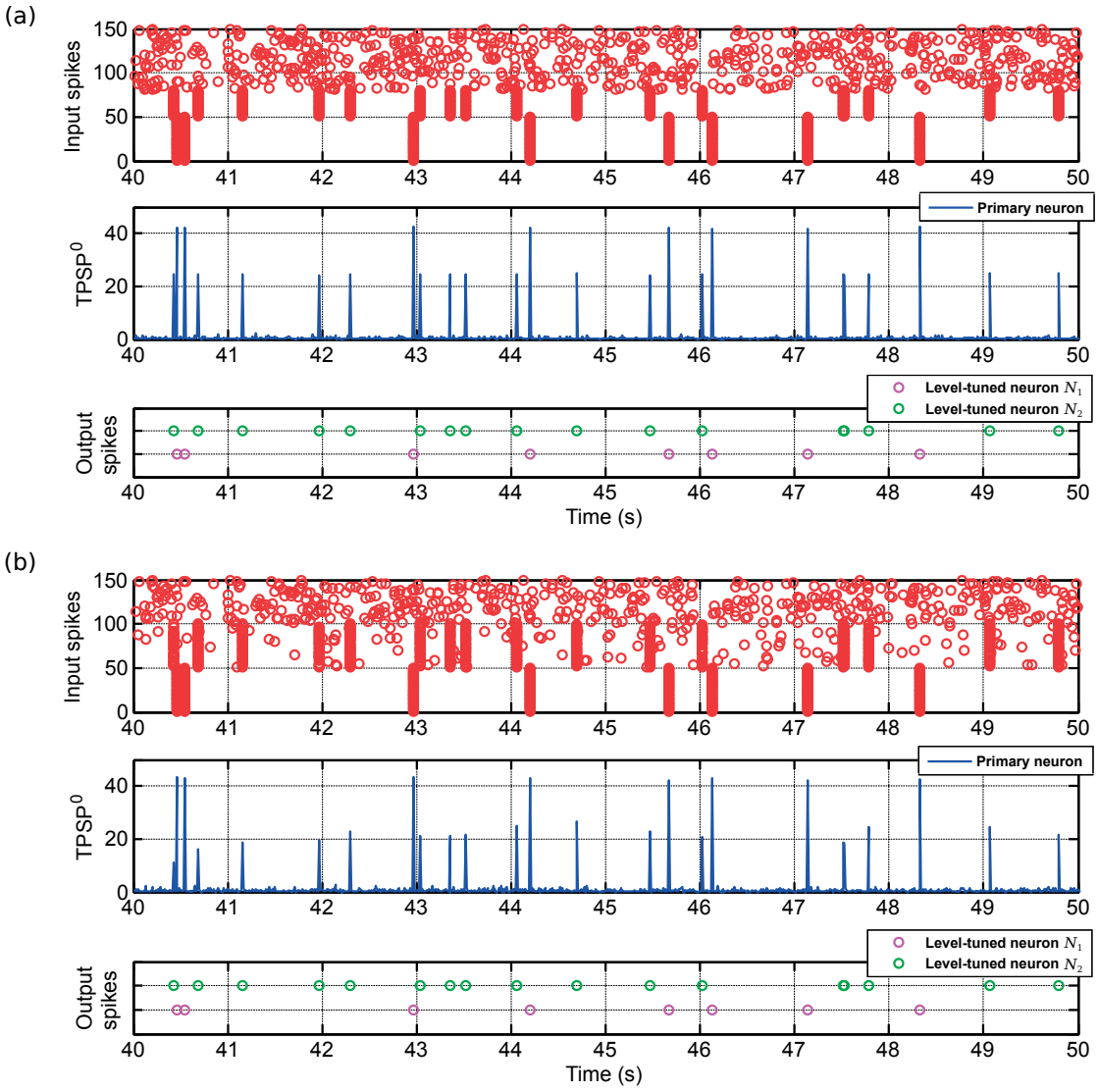


Figure 4.17: **An all-phase-change neural network detecting two correlated groups** (a) After an initial period of learning (not shown in the figure), the level-tuned neurons correctly spike when respective correlated groups appear at the input. (b) The system maintains the performance for weakly correlated inputs, where one of the groups is correlated with $c = 0.6$. Adapted from [Pantazi et al., 2016], © IOP Publishing. Reproduced with permission.

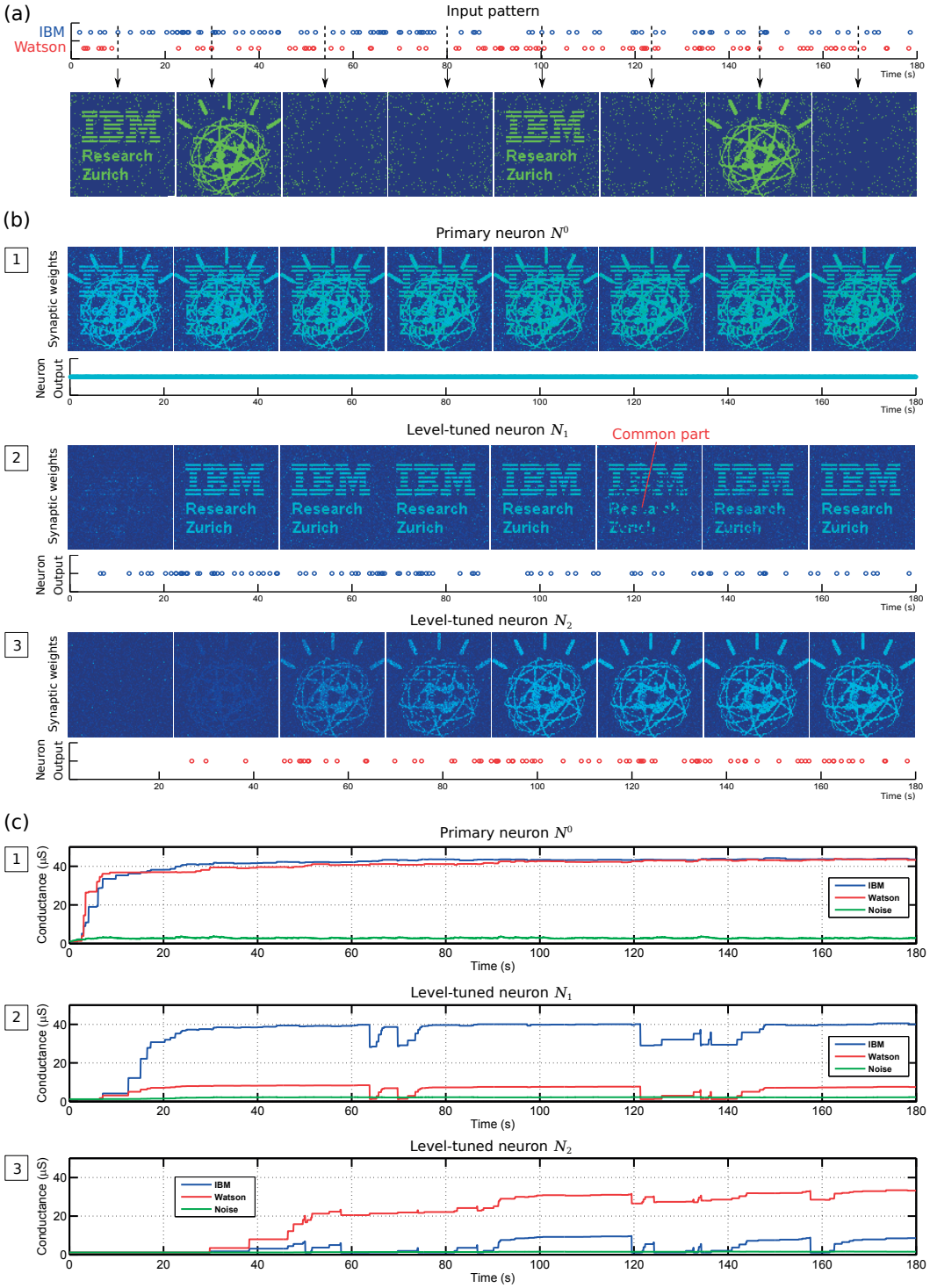


Figure 4.18: **An all-phase-change neural network learning two patterns** After an initial period of learning, each neuron correctly spikes on appearance of the respective pattern. The weights provide high-quality visualization of each pattern, although there are occasional drops in the conductance of the common part of the patterns (marked in a snapshot). Adapted from [Pantazi et al., 2016], © IOP Publishing. Reproduced with permission.

4.3.3 Enhanced multiple overlapping pattern learning

Independently from the WTA scheme used, depression artifacts may arise in a network of neurons operating with A-STDP owing to the asymmetric conductance response of 1-PCM synapses. Assume that a neuron N_j spikes on an observation of a pattern X. If a different overlapping pattern Y appears shortly afterwards within the depression window T_{LTD} of the A-STDP, the 1-PCM synapses will undergo an abrupt depression. Y might be a correlated pattern with a significant overlap with X: $X \cap Y \neq \emptyset$. The neuron N_j would normally spike for such pattern and potentiate its synapses. However, if the WTA mechanism blocks the spiking of neuron N_j, a depression of the common part is executed instead, as marked in Fig. 4.18b(2).

Algorithm 2 [Woźniak et al., 2017b] provides a solution tailored for learning multiple spatially overlapping patterns with phase-change synapses and A-STDP. Specifically, to avoid depression of the parts overlapping with other patterns, the depression (line 8) is only executed if none of the other neurons are spiking. In consequence, at a post-synaptic spike event only the winning neuron ($y_j = 1$) will learn, leading to a post-spike *learning WTA*.

Algorithm 2 Enhanced multiple pattern learning

1: $S = \{i : x_i = 1\}$
2: **if** $y_j = 1$ **then**
3: $\quad t_j^{\mathrm{post}} \leftarrow t$
4: $\quad \forall_{i \in S}\ t_i^{\mathrm{pre}} \leftarrow t$
5: $\quad \forall_i\ w_{ji} \leftarrow w_{ji} + f_+(w_{ji}, t_j^{\mathrm{post}} - t_i^{\mathrm{pre}})$
6: **else**
7: $\quad \forall_{i \in S}\ t_i^{\mathrm{pre}} \leftarrow t$
8: $\quad$ **if** $\forall_k y_k = 0$ **then** $\forall_{i \in S}\ w_{ji} \leftarrow w_{ji} + f_-(w_{ji}, t_j^{\mathrm{post}} - t_i^{\mathrm{pre}})$

We simulate and experimentally implement a level-tuned WTA architecture with 3 neurons with 1-PCM synapses using A-STDP with learning WTA from Alg. 2. Synaptic weights are initialized to low values, which correspond to the amorphous state of the cells. The initialization is performed by applying a reset pulse with a current of $I = 450\,\mu$A followed by a single accumulation pulse with $I = 130\,\mu$A applied for a duration of $p_w = 100\,$ns.

The input spikes presented to the network contain information about three patterns of letters 'I', 'B', 'M', used also in Sec. 4.1. The patterns appear in a random order at time instances drawn from Possion distribution with rate of 1 Hz. The high value pixels of the patterns correspond to correlated groups of spikes, and the remaining non-pattern inputs receive Poissonian noise with an average rate of 1 Hz. After a 600 s period of network activity, the synaptic weights are visualized in Fig. 4.19. Throughout the learning, no common part depression events occurred and the final results show that the neurons learn high-quality pattern reconstructions for overlapping patterns. Specifically, Fig. 4.19a presents the synaptic weights obtained using the look-up table simulation and Fig. 4.19b presents the synaptic weights as experimentally measured from the cells' conductance. Fig. 4.19c shows the distribution of the final weights

for neuron N_1 for the simulation and the experimental implementation. Due to variability, the distribution of the weights in the experimental cells is not perfectly aligned with the simulation, but this does not affect the result as it remains bi-modal.

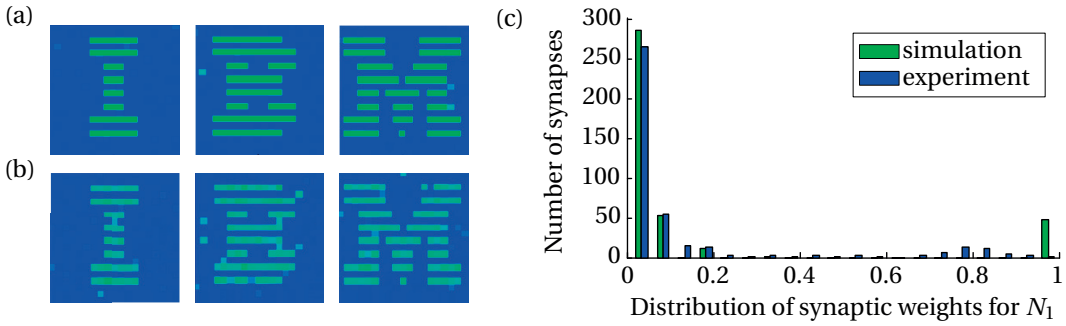


Figure 4.19: **Learning three overlapping patterns** The final weights: (a) simulation, (b) experiment. (c) A histogram with comparison of the final weights of the neuron N_1 for the simulation and the experiment.

4.4 Conclusions

In this chapter we presented how to combine the neuromorphic phase-change-based building blocks, introduced in Chapter 3, to construct neuromorphic architectures for pattern learning. We demonstrated one-at-a-time pattern learning and re-learning in a single neuron with 1-PCM synapses using A-STDP. The system successfully learned the patterns purely from the temporally-coded information. Next, we scaled up the neuron to a record number of 1M phase-change synapses and applied it for weakly correlated pattern learning. The neuron was implemented using both LIF and phase-change-based neuron soma and achieved high pattern detection accuracy. Based on the analysis of the weights evolution, we extended A-STDP with the selective depression mechanism, that provided high quality pattern visualization of weakly correlated patterns.

By combining multiple all-phase-change neurons, we constructed a single layer phase-change-based network to support learning multiple patterns in parallel. In particular, we proposed an energy-efficient WTA scheme with level-tuned neurons that uses an additional primary neuron to control the activation of the regular neurons. We demonstrated learning of highly and weakly correlated patterns. Lastly, we addressed the negative impact of the 1-PCM asymmetric conductance response on the quality of the pattern reconstructions, by introducing a modified A-STDP logic with learning WTA.

All the contributions were validated in a set of experiments executed using physical cells in an experimental hardware platform. The proposed architectures demonstrate the applicability of phase-change neuronomorphic hardware for pattern learning.

5 Knowledge representation

The experimental results presented in the previous chapter validate phase-change technology as a candidate for implementing classic architectures of SNN neuromorphic systems. However, the basic SNN architectures of today have limited capabilities for tackling complex cognitive problems. In this chapter, we first discuss in Sec. 5.1 the challenges of scaling up the single-layered SNN architecture, and compare it to the ANN deep learning architectures. We identify the need to improve the internal representation learned in SNNs to involve learning features rather than memorizing patterns. Next, in Sec. 5.2, we provide intuitive understanding of features and assumptions behind different feature types. In Sec. 5.3, we demonstrate how these assumptions lead to explicit optimization constraints in a mathematical formulation of feature learning as matrix factorization. Lastly, in Sec. 5.4, we discuss how similar feature learning functionality arises implicitly in the ANN architectures.

5.1 Scaling up neural network architectures

Practical applications usually require to scale up the size of a neural network to provide high accuracy pattern detection. For instance, in case of hand-written digit recognition, at least 10 neurons are required to distinguish the digits from '0' to '9'. However, owing to the variation of the writing style, typically more neurons are required for reliable detection. Fig. 5.1 illustrates hand-written digits test set pattern detection accuracy for a popular MNIST dataset of 28×28-pixel patterns, comprising 60000 training patterns and 10000 test patterns.

In SNNs [Querlioz et al., 2011] [Diehl and Cook, 2015], it is common to scale up the network size while maintaining the same single-layered WTA architecture. Adding new neurons results in an increase of the *width* of the network, as illustrated in Fig. 5.2a. In consequence, a network of n neurons is able to memorize n templates of different writing styles of various digits. Because multiple neurons correspond to a particular digit, the test phase involves counting how many neurons from each class respond to a given input and choosing the class with the largest response. With an increasing number of neurons, the system is able to store more templates and better capture the variability of the input patterns.

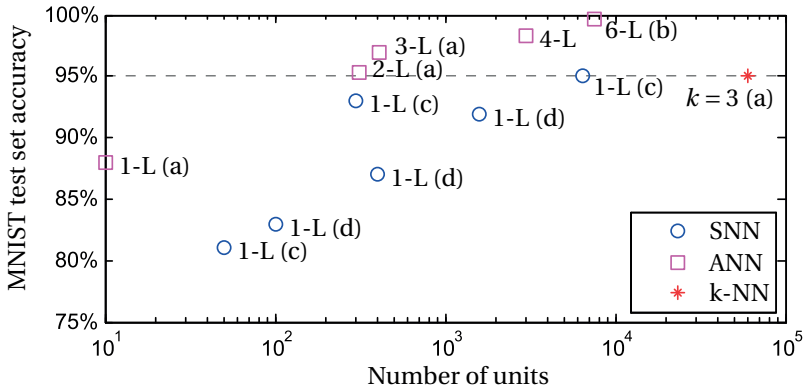


Figure 5.1: **Scaling up improves accuracy** In SNNs, typically the number of neurons is scaled up while maintaining the single layer (1-L) architecture. Such networks typically do not surpass the pattern detection accuracy of a k-NN approach. In ANNs, additional neurons are often arranged in multiple layers (n-L). Reported values originate from: (a) [LeCun et al., 1998], (b) [Cireşan et al., 2010], (c) [Querlioz et al., 2011], (d) [Diehl and Cook, 2015].

5.1.1 Relationship to k-NN statistical model

For a layer size of n equal to the training set size, a 1-L SNN can theoretically memorize the entire training set. This resembles the operation of a k-nearest-neighbor (k-NN) statistical model [LeCun et al., 1998], in which all the training patterns are stored in the model without any learning. Evaluating the model for a new input involves assigning it to the class with the largest response among k best-matching known patterns. The matching criterion is typically an Euclidean distance between the inputs x_i and the weights w_{ji} of the units j of the model:

$$\arg\min_j \sum_i (x_i - w_{ji})^2 \tag{5.1}$$

We rewrite the sum as:

$$\sum_i (x_i^2 - 2x_i w_{ji} + w_{ji}^2) = 2(\frac{1}{2}\sum_i x_i^2 - \sum_i x_i w_{ji} + \frac{1}{2}\sum_i w_{ji}^2) \tag{5.2}$$

We invert the direction of the optimization and ignore the x_i^2 term, which is independent of j and does not change the optimization result:

$$\arg\max_j \sum_i x_i w_{ji} - \frac{1}{2}\sum_i w_{ji}^2 \tag{5.3}$$

Lastly, we notice resemblance to an SNN model discussed in Sec. 2.3. The first term corresponds to the variable contribution of the input – TPSP, and the second term is a fixed offset of a unit j, which may be incorporated into a spiking threshold $V_{\text{th},j} = \frac{1}{2}\sum_i w_{ji}^2$. Using this analogy, the maximization becomes:

$$\arg\max_j \text{TPSP}_j - V_{\text{th},j} \tag{5.4}$$

Finding k units with the highest TPSP_j in comparison to their spiking thresholds is similar to operation of a k-WTA SNN architecture. Therefore, an analysis of k-NN accuracy, such as 3-NN plotted in Fig. 5.1, may bring insights into the accuracy limits of single-layered SNNs. None of the 1-layered SNNs in Fig. 5.1 surpasses the accuracy of 3-NN, which is surpassed by all of the 2 or more layered ANNs.

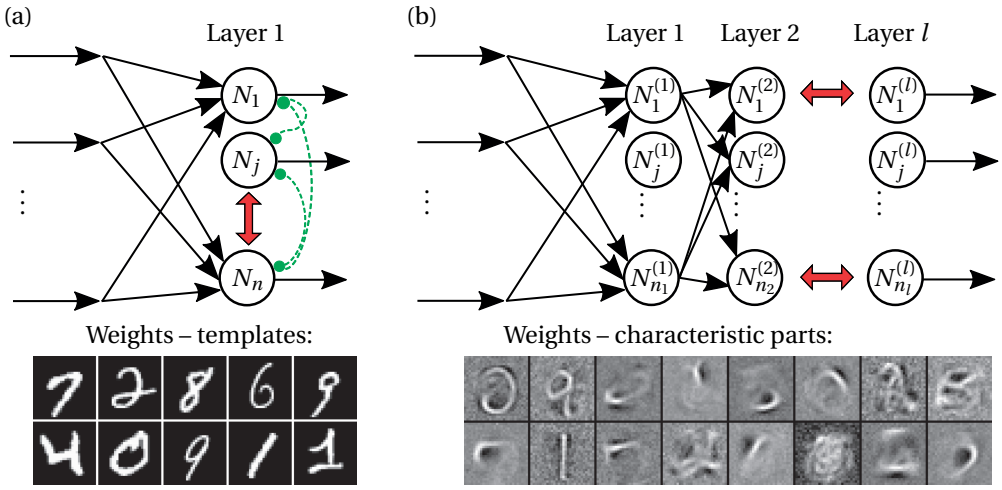


Figure 5.2: **Scaling up approaches in SNNs and ANNs** (a) In single-layered SNNs, the width of the network is increased: n neurons learn n templates. (b) In ANNs, the depth of the network is increased: n_k neurons in layer k learn n_k parts, which can be combined to 2^{n_k} patterns.

5.1.2 Knowledge representation in deep networks

In ANNs [LeCun et al., 1998] [Cireşan et al., 2010], it is common to scale up the network by increasing the number of layers that increases the *depth* of the network, which is reflected in the name of deep learning. Already a 2-layer ANN with 310 neurons achieves 95.3% accuracy and surpasses the 95.0% accuracy of a 1-layer SNN with 6400 neurons. This is possible because of a different internal knowledge representation. Rather than memorizing a subset of the patterns in form of templates, ANNs learn abstractions of the data that comprise characteristic parts of the patterns, called *features*. This leads to a feature-based representation [Bengio et al., 2013], illustrated in Fig. 5.2b, in which n_k neurons in layer k learn n_k parts, which can be combined in 2^{n_k} ways to form the patterns. Beside an exponentially higher storage efficiency of the parts-based implementation, it also enables generalization of the knowledge: new inputs may be interpreted as a combination of the parts that was never observed during the training.

The knowledge from an ANN can be translated into an SNN by porting the weights [O'Connor et al., 2013], or the algorithms [Neftci et al., 2013], which leads to improved accuracy of SNNs. However, it is interesting to explore the properties of native spike-timing-dependent learning in SNNs for feature extraction, which is the subject of Chapter 6. Before this, we discuss in detail the nature of the features and the mathematical foundations of feature learning.

5.2 Feature types

To provide intuitive understanding of the features, we begin by analyzing a dataset comprising three simplistic binary patterns illustrated in Fig. 5.3a. We call it the *Weather* dataset as the patterns symbolically represent weather conditions. We colorize the patterns and superimpose them in Fig. 5.3b. In consequence, we distinguish five characteristic regions, that may form sets of input attributes corresponding to the features illustrated in Fig. 5.3c. Sets F_1 and F_5 capture the unique characteristics of the patterns and F_2 captures the common part. Each pattern from Fig. 5.3a can be represented as a combination of these features. On an abstract level, the decomposition into the features F_1-F_5 improves the understanding of the weather conditions that may involve clouds F_2, bright clouds F_1, light rain F_3, sun F_4 and thunderstorm F_5. Based on such representation, the system is able to "understand" previously unseen conditions involving e.g. sun during a thunderstorm or a thunderstorm accompanied only by a light rain. We will use this simple dataset to illustrate the operation of various algorithms, although quantitative assessments require larger datasets.

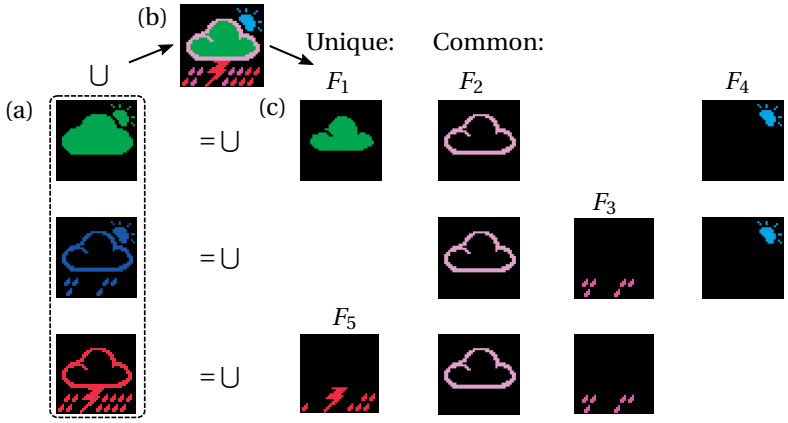


Figure 5.3: **Weather dataset** (a) Three patterns representing weather conditions. (b) Superposition of all the patterns indicates possible features. (c) Manual decomposition of each pattern into parts-based features. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

The Swimmer [Donoho and Stodden, 2004] is a larger dataset of a type similar to the Weather dataset. It contains 32×32-pixel patterns that consist of a simplistic body and four limbs in four possible positions, as in samples presented in Fig. 5.4a. It is used to illustrate parts-based feature extraction, because the ground truth in Fig. 5.4b consists non-overlapping parts, that can be combined to form any of the input patterns. Mathematically, a non-overlapping

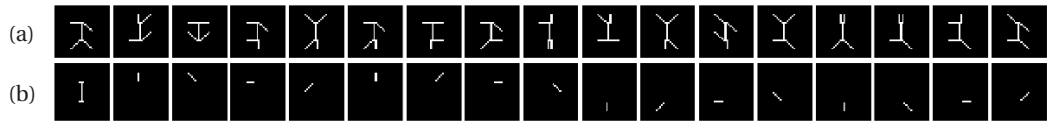


Figure 5.4: **Swimmer dataset** (a) Samples. (b) Ground truth decomposition into 17 orthogonal features.

parts-based representation corresponds to finding orthogonal components of the inputs, because each pair of vectors representing the features satisfies $\boldsymbol{x}^{F_i} \cdot \boldsymbol{x}^{F_j} = 0$. A basic statistical technique extracting orthogonal components from a dataset is Principal Components Analysis (PCA) [Bengio et al., 2013]; therefore we will call these features *PCA-like*.

However, other types of features may be appropriate for different data. The Bars dataset [Földiak, 1990] contains 8×8-pixel patterns with horizontal and vertical bars, appearing with probability $\frac{1}{8}$ at 8 horizontal and 8 vertical positions, as in samples presented in Fig. 5.5a. It is used to illustrate independent component extraction: the ground truth in Fig. 5.5b contains all unique bars, which are statistically independent components of the input. The features formed by independent components are not orthogonal, as the bars can overlap. A basic statistical technique extracting independent components, without orthogonality constraint, is Independent Component Analysis (ICA) [Bengio et al., 2013], so we will call these features *ICA-like*.

It is also possible to extract PCA-like features for the ICA-like Bars dataset, illustrated in Fig. 5.5c. Such features comprise 64 individual pixels corresponding to all the possible cross-points between the bars. Theoretically, these are valid features, but we call them *degenerated features*, because they only permute the inputs.

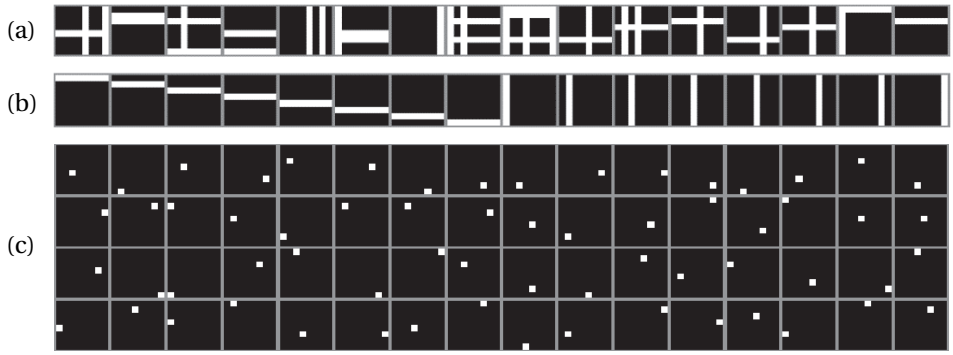


Figure 5.5: **Bars dataset** (a) Samples. (b) Ground truth decomposition into independent components: 8 horizontal bars and 8 vertical bars. (c) Decomposition into 64 orthogonal features.

Feature-based representations may significantly reduce the number of neurons in a network in comparison to a pattern-based representation. Only 17 neurons are required to represent all $4^4 = 256$ patterns from the Swimmer dataset using PCA-like features and 16 neurons are required to represent $\approx 2^{16} = 65536$ patterns from the Bars dataset using ICA-like features. Lastly, more feature types are analyzed in detail in the field of representation learning [Bengio et al., 2013]. A particular representation can be obtained using various methods, but more interestingly a particular method with additional constraints can obtain various representations, as discussed in the next section.

5.3 Explicit feature learning using matrix factorization

To interpret how representation constraints shape the type of the features, we formulate the task of feature learning as a constrained matrix factorization problem from linear algebra. Matrix factorization provides a mathematical framework, in which we can explicitly state all the assumptions behind a feature-based representation, and observe their consequences for the type of representation, factoring out the impact of neuronal dynamics on the learning process. On the other hand, we keep in mind that this analysis is constrained to the outputs being a linear superposition of the inputs whereas in neural networks the neural activations are a non-linear function of the inputs [Bengio et al., 2013].

Feature learning is formulated in the matrix factorization framework as follows [Lee and Seung, 1999]: a dataset in a matrix $\boldsymbol{D}$ is approximated as $\boldsymbol{D} \approx \boldsymbol{W}\boldsymbol{H}$, where $\boldsymbol{W}$ is a matrix of basis vectors that correspond to the features, multiplied by encodings $\boldsymbol{H}$, as illustrated in Fig. 5.6. A typical matrix factorization implementation adjusts $\boldsymbol{W}$ and $\boldsymbol{H}$ with the number of features n given as an input parameter. Optimization is performed to obtain the best approximation $\boldsymbol{D} \approx \boldsymbol{W}\boldsymbol{H}$, measured by a divergence $\mathscr{D}(\boldsymbol{D}||\boldsymbol{W}\boldsymbol{H})$ between the dataset and the feature-based reconstruction. Constraints are introduced to achieve particular properties of the features. Throughout the rest of this section, we discuss a few popular sets of constraints and their impact on the representation.

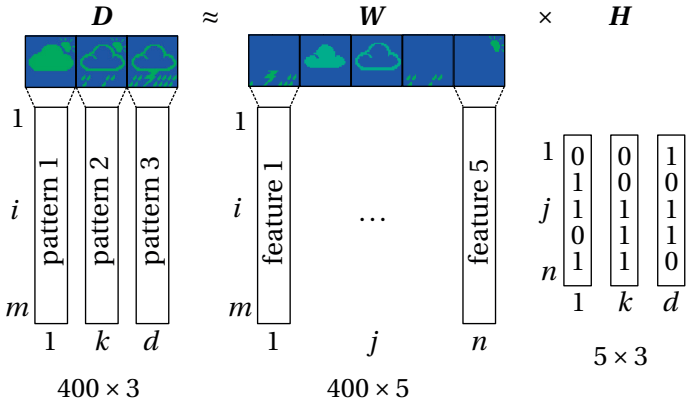


Figure 5.6: **An example of matrix factorization** Three 20×20-pixel patterns from the Weather dataset factorized into 5 features. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

5.3.1 Vector Quantization

Vector Quantization (VQ) is an approach to lossy compression, which may be formulated as a matrix factorization [Lee and Seung, 1999]. Each pattern in $\boldsymbol{D}$ is represented by a single best-matching prototype in $\boldsymbol{W}$. In consequence, $\boldsymbol{H}$ is a binary matrix with a single non-zero value per column, so that:

$$\boldsymbol{W}_{ji} \in \mathbb{R}, \;\; \boldsymbol{H}_{kj} \in \{0,1\}, \;\; \sum_{j} \boldsymbol{H}_{kj} = 1 \tag{5.5}$$

Assuming $n = 3$, a matrix $\boldsymbol{W}$ obtained for the Weather dataset using this approach is illustrated in Fig. 5.7a. It contains the original patterns. Such representation using prototypes is analogous to a 1-WTA operation of a single-layered SNN: $\boldsymbol{H}_{kj} \in \{0, 1\}$ corresponds to using a spiking neuron model, and $\sum_j \boldsymbol{H}_{kj} = 1$ corresponds to the 1-WTA mechanism. To obtain a true feature-based representation multiple features should be activated, which corresponds to relaxing the $\sum_j \boldsymbol{H}_{kj} = 1$ constraint.

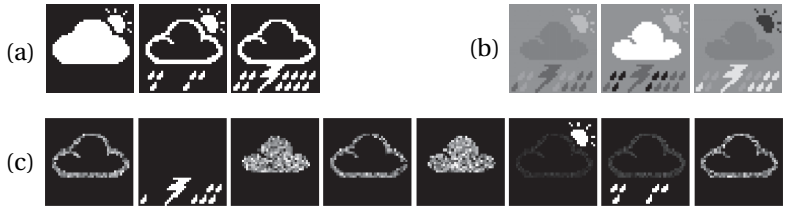


Figure 5.7: **Different matrix factorizations of the Weather dataset** (a) Vector Quantization with $n = 3$. (b) Principal Components Analysis with $n = 3$. (c) Non-negative Matrix Factorization with $n = 8$.

5.3.2 Principal Components Analysis

Principal Components Analysis (PCA) is a statistical approach providing an alternative orthogonal basis for representation of the dataset $\boldsymbol{D}$, in which the basis vectors are oriented towards the highest directions of data variation [Lee and Seung, 1999]. Ignoring the order of the basis vectors, a result of PCA is a matrix factorization with orthogonality constraint on the column vectors $\boldsymbol{w}_{j*} = [\boldsymbol{W}_{j1}, \ldots, \boldsymbol{W}_{jm}]$ of $\boldsymbol{W}$:

$$\boldsymbol{W}_{ji} \in \mathbb{R}, \ \boldsymbol{H}_{kj} \in \mathbb{R}, \ \forall_{j_1 \neq j_2} \boldsymbol{w}_{j_1 *} \cdot \boldsymbol{w}_{j_2 *} = 0 \tag{5.6}$$

The disadvantage of PCA is that the features involve negative values and lack direct intuitive interpretability, as illustrated for the Weather dataset in Fig. 5.7b for $n = 3$. However, in comparison to storing prototypes in VQ, PCA indeed provides the features with many non-zero values in the encoding $\boldsymbol{H}$.

The advantage of orthogonal features is an easy calculation of the encoding $\boldsymbol{H}$. For a factorization $\boldsymbol{D} \approx \boldsymbol{W}\boldsymbol{H}$, the encoding $\boldsymbol{H} = \boldsymbol{P}\boldsymbol{D}$ is typically calculated using a projection matrix $\boldsymbol{P} = \boldsymbol{W}^{-1}$, which in general requires an inversion of matrix $\boldsymbol{W}$. In the particular case of orthogonal features, based on the canonical definition of an orthogonal matrix $\boldsymbol{I} = \boldsymbol{W}\boldsymbol{W}^\top$, left multiplying by $\boldsymbol{W}^{-1}$ we obtain $\boldsymbol{W}^{-1} = \boldsymbol{W}^\top$. In consequence, the encoding $\boldsymbol{H}$ may be calculated using a transpose $\boldsymbol{P} = \boldsymbol{W}^\top$ of the features, as illustrated in Fig. 5.8. The calculation becomes similar to a feed-forward operation of a neural network for a sequence of input patterns, where each column of $\boldsymbol{H}$ corresponds to a single response of the network for a particular input pattern from $\boldsymbol{D}$. Although the depicted features are PCA-like, they are different from the typical PCA features in Fig. 5.3b. Therefore, we continue searching for additional constraints that would result in interpretable features.

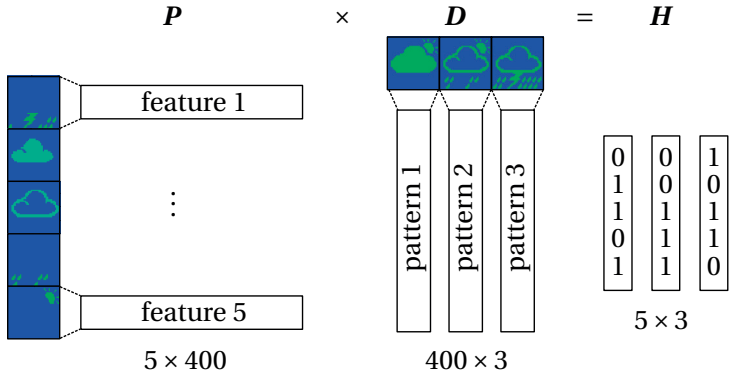


Figure 5.8: **Projection matrix** Calculation of the encoding $\boldsymbol{H}$ using a projection matrix $\boldsymbol{P}$ is similar to the feed-forward operation of a neural network. For the depicted features of the Weather dataset, the projection matrix $\boldsymbol{P}$ is a transpose of the features matrix $\boldsymbol{W}$.

5.3.3 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) [Lee and Seung, 1999] is a matrix factorization that aims to provide features with high interpretability. It takes inspiration from neural networks and introduces non-negativity constraint for the features and the encodings. It assumes that $\boldsymbol{W}$ corresponds to synaptic weights of a neural network and $\boldsymbol{H}$ corresponds to neural activations. In biology, the synapses do not change the sign of their synaptic weights. Simultaneously, the neural activations have no physical sense for negative values, because they correspond to the firing frequency in Hz. Therefore, the features are found by minimizing $\mathscr{D}(\boldsymbol{D}||\boldsymbol{W}\boldsymbol{H})$ subject to:

$$\boldsymbol{W}_{ji} \geq 0,\ \boldsymbol{H}_{kj} \geq 0 \tag{5.7}$$

Local NMF (LNMF) [Feng et al., 2002] is a variant of NMF, which additionally maximizes feature orthogonality, as well as encoding sparsity and information content of the features. Constraints are formulated in terms of $\boldsymbol{U} = \boldsymbol{W}^\top \boldsymbol{W}$ and $\boldsymbol{V} = \boldsymbol{H}^\top \boldsymbol{H}$. Orthogonality is maximized by $\min \sum_{i \neq j} \boldsymbol{U}_{ij}$, which is equivalent to minimization of the dot product $\boldsymbol{w}_{a*} \cdot \boldsymbol{w}_{b*}$ for any two feature learning neurons a and b. Encoding sparsity is maximized indirectly by $\min \sum_i \boldsymbol{U}_{ii}$, which can be interpreted as minimizing $||\boldsymbol{w}_{a*}||_2$ to evenly distribute the weights' values. This avoids solutions consisting of multiple degenerated features having few non-zero weights, more of which would be required to represent an input, thus decreasing the encoding sparsity. Lastly, information content of the features is maximized by $\max \sum_i \boldsymbol{V}_{ii}$, which corresponds to maximizing squared activities $\sum_{ij} (\boldsymbol{H}_{ij})^2$.

We apply LNMF optimization for the Weather dataset using an iterative algorithm proposed in [Feng et al., 2002] with a divergence $\mathscr{D}(\boldsymbol{A}||\boldsymbol{B}) = \sum_{i,j} (\boldsymbol{A}_{ij} \log \frac{\boldsymbol{A}_{ij}}{\boldsymbol{B}_{ij}} - \boldsymbol{A}_{ij} + \boldsymbol{B}_{ij})$ [Lee and Seung, 2001] for $n = 8$. The obtained features are illustrated in Fig. 5.7c, and they finally correspond to the features from the manual decomposition in Sec. 5.2. Key constraints that led to these easily-interpretable parts-based features include orthogonality and non-negativity.

5.4 Implicit feature learning in neural networks

The representation constraints are rarely stated in an explicit form in neural networks applied for feature learning. Some of them may be hidden behind the neuron model, the structure of the network, the operation of the learning rule and the feedback mechanism. In case of ANNs trained with backpropagation, additional constraints in form of regularization terms may be added to the learning optimization problem [Bengio et al., 2013], but in our opinion this provides a false impression of control over the constraints. Therefore, we refer to the neural networks as implicit feature learning models. In the rest of this section, we present examples of feature learning ANN architectures.

5.4.1 Autoencoder

An ANN Autoencoder (AE) [Bengio et al., 2013] poses the feature learning task as minimizing the reconstruction error in an architecture comprising an encoder f_θ that encodes the input into code $\boldsymbol{H}$, and a respective decoder $g_{\theta'}$ that reconstructs the input from the code, as depicted in Fig. 5.9. Encoder's parameters θ and decoder's parameters θ' are typically found by solving the following optimization problem using the backpropagation algorithm:

$$\theta^*, \theta'^* = \arg\min_{\theta,\theta'} \frac{1}{d} \sum_{k=1}^{d} L(\boldsymbol{x}^{(k)}, g_{\theta'}(f_\theta(\boldsymbol{x}^{(k)}))) \tag{5.8}$$

where $\boldsymbol{x}^{(k)}$ are the input vectors corresponding to the d patterns from the dataset D, $g_{\theta'}(f_\theta(\boldsymbol{x}^{(k)})$ is an input reconstruction function and L is a loss function between an input vector and its reconstruction.

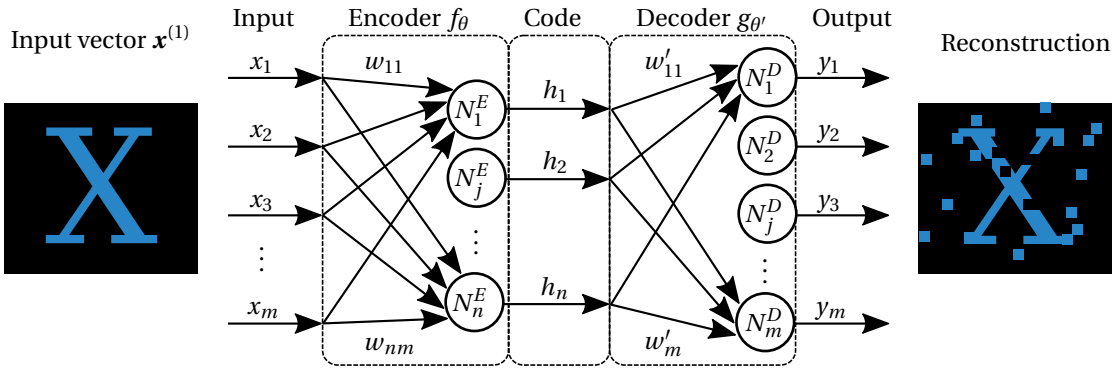


Figure 5.9: **Autoencoder** Encoder f_θ encodes the input into code $\boldsymbol{H}$, and a respective decoder $g_{\theta'}$ reconstructs the input from the code.

Encoder and decoder may use a multi-layered ANN [Hinton and Salakhutdinov, 2006], but we focus on a case with a single layer using a squared loss function. The parameters $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ and $\theta' = \{\boldsymbol{W}', \boldsymbol{b}'\}$ contain the weights of the layers and the biases of the ANN neurons, corresponding to the spiking thresholds of the SNN neurons. Such autoencoders are known to

produce PCA-like features [Bengio et al., 2013], because the reconstruction error is minimized if the forward pass of the decoder operates similarly to the projection in PCA, discussed in Sec. 5.3.2, so that $\boldsymbol{W}' = \boldsymbol{S}\boldsymbol{W}^\top$ up to a scaling matrix $\boldsymbol{S}$ [Bourlard, 2000].

In ANN community it is common to use so called *tied weights* autoencoders, in which $\boldsymbol{W}' = \boldsymbol{W}^\top$. The exact equality is supposed to force the autoencoder to develop non-PCA-like features, because the decoder loses the scaling ability [Bengio et al., 2013]. Still, in some cases autoencoders with tied weights may learn PCA-like features [Vincent et al., 2010]. In Fig. 5.10a we illustrate the features obtained from a tied weights autoencoder for the Weather dataset. Except for a few neurons that remain inactive, the features correspond to the ones obtained using PCA in Fig. 5.3b. In Fig. 5.10b we illustrate the features obtained for the Bars dataset. The maxima of the features lie at the intersections of the bars, which bears some resemblance to the PCA-like features for the Bars dataset in Fig. 5.5c. Nevertheless, the features in both 5.10a and 5.10b lack intuitive interpretablility.

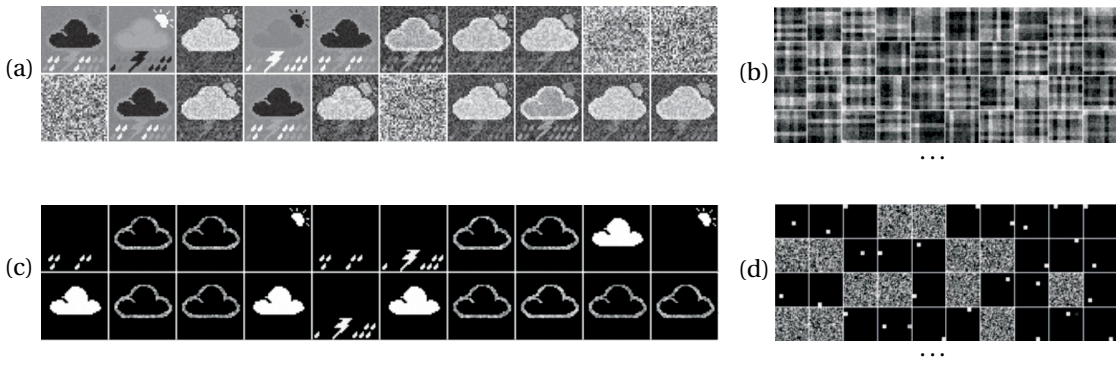


Figure 5.10: **Features learned using an ANN autoencoder** (a) Weather dataset for $n = 20$. (b) Bars dataset for $n = 100$. (c) Weather dataset for $n = 20$, $\lambda = 0.05$. (d) Bars dataset for $n = 400$, $\lambda = 0.05$.

To improve the interpretability of the features, we apply the insight from NMF to constrain the weights to non-negative values. We introduce a regularization term into the optimization Eq. 5.8 to penalize the negative weights:

$$\theta^*, \theta'^* = \underset{\theta,\theta'}{\arg\min} \frac{1}{d} \sum_{k=1}^{d} L(\boldsymbol{x}^{(k)}, g_{\theta'}(f_\theta(\boldsymbol{x}^{(k)}))) + \lambda \sum_{j,i} (w_{ji})^- \tag{5.9}$$

where λ is the magnitude of the penalty, and $(z)^- = -\min(z, 0)$ is the negative part of a number. The features obtained using this approach for $\lambda = 0.05$ are illustrated in Fig. 5.10c for the Weather dataset and in Fig. 5.10d for the Bars dataset. Those features correspond to the PCA-like features in Fig. 5.3c and Fig. 5.5c.

5.4.2 Restricted Boltzmann Machine

A Restricted Boltzmann Machine (RBM) [Hinton and Salakhutdinov, 2006] is usually depicted as in Fig. 5.11a. It is said to be an ANN with two layers that form a bipartite graph of visible input neurons with activations v_i in the input layer and hidden feature detectors with activations h_j in the hidden layer. However, to some degree it resembles an SNN, because it uses stochastic binary neurons that spike with a probability determined by their activation values. Moreover, if we explicitly visualize the computations performed during learning of an RBM using a single step of Contrastive Divergence (CD-1), it operates like a three-layer feed-forward network, presented in Fig. 5.11b.

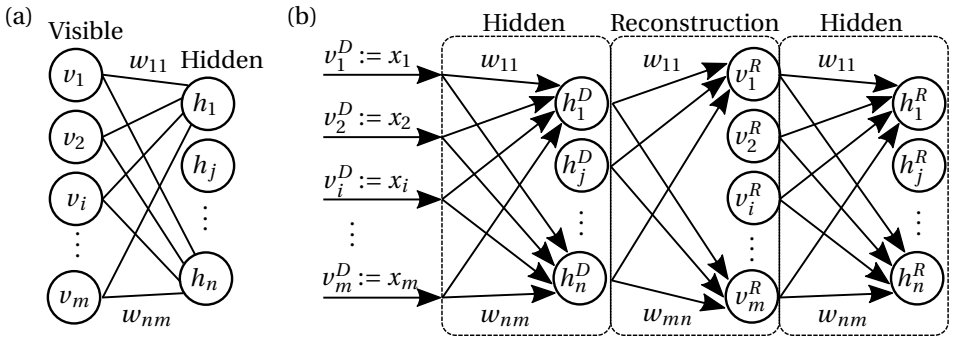


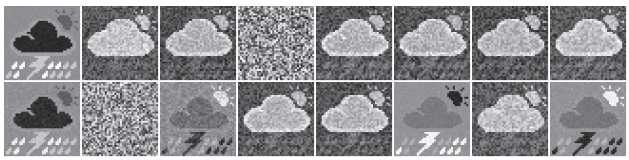
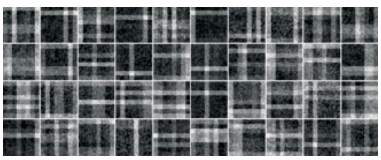
Figure 5.11: **Restricted Boltzmann Machine** (a) The classic depiction as a bipartite graph of visible and hidden units. (b) An alternative depiction that unrolls the bi-directional computations of the CD-1 learning to a feed-forward architecture.

Learning using CD-1 operates as follows: binarized input values x_i from a dataset D are considered to be the initial visible activations v_i^D. In RBM nomenclature, the visible units are said to be clamped to the inputs. Then, activations of the hidden neurons h_j^D are calculated and binarized in a stochastic manner, so that $h_j^D \in \{0,1\}$. Based on the hidden activations, a reconstruction of the input v_i^R is calculated using the same set of weights. This is a consequence of the bidirectional operation of the connections in an RBM, indicated by the lack of arrow heads in Fig. 5.11a. Using the weights in the opposite direction is equivalent to using a transposition of the weight matrix $\boldsymbol{W}$, which resembles the tied weights of AE in Sec. 5.4.1, or PCA projection matrix in Sec. 5.3.2. Lastly, the activations of the hidden neurons h_j^R are calculated based on the reconstruction v_i^R and the weight adjustments are performed according to:

$$\Delta w_{ji} = \alpha(v_i^D h_j^D - v_i^R h_j^R) \tag{5.10}$$

where α is the learning rate. The procedure is repeated for all the patterns from the dataset.

The features learned by an RBM for the Weather and the Bars dataset are illustrated in Fig. 5.12. They closely resemble the PCA-like features learned by the ANN autoencoder in Fig. 5.10a-b.

(a)  (b) 

...

Figure 5.12: **Features learned using an RBM** (a) Weather dataset for $n = 20$. (b) Bars dataset for $n = 100$.

5.4.3 Dendritic inhibition

Dendritic inhibition [Spratling, 2006] is an ANN architecture developed to learn independent components. A single layer of rate neurons is coupled with a dendritic inhibition mechanism, in which the activity of the neurons inhibits the strength of the inputs. The architecture may operate using bi-polar or uni-polar weights, but we focus on the uni-polar implementation. The operation of the model involves an iterative calculation of a steady-state competition solution for each presentation of an input pattern to the network, based on the following system of equations:

$$\begin{cases} y_j = & \sum_{i=1}^{m} w_{ji} x'_{ji} \\ x'_{ji} = & x_i (1 - \alpha \max_{r=1, r \neq j}^{n} \frac{w_{ri}}{\max_{q=1}^{m} w_{rq}} \frac{y_r}{\max_{q=1}^{n} y_q})^+ \end{cases} \tag{5.11}$$

where $(v)^+$ is a positive part of v. The input pattern is kept fixed for 25 iterations, in which α is adjusted from 0 to 6 in 0.25 steps. The final y_j values are considered to be a stable state, and the weights are adjusted according to:

$$\Delta w_{ji} = \frac{x_i - \overline{x}}{\sum_{p=1}^{m} x_p} (y_j - \overline{y})^+ \tag{5.12}$$

Lastly, the weights are clipped to non-negative values and normalized within each neuron:

$$\forall_{j,i} \; w_{ji} := (w_{ji} + \Delta w_{ji})^+ \tag{5.13}$$

$$\forall_j \sum_i w_{ji} \leq 1 \tag{5.14}$$

To resolve competition ties and cause bifurcation of the y_j values, random weight initialization is used. Dendritic inhibition applied to the Bars dataset learns the features illustrated in Fig. 5.13, that correctly correspond to the ground-truth decomposition from Fig. 5.5b.

Figure 5.13: **Features learned using dendritic inhibition** Correct ICA-like features of the Bars.

6 Architectures for feature learning

In the previous chapter we motivated the need for learning the features rather than the patterns. We developed an intuitive understanding of the features and of the mathematical constraints that lead to their development. Deep learning ANNs that operate using feature-based representations currently achieve the highest accuracies for the most challenging datasets. Therefore, learning feature-based representations should be beneficial also for the SNN architectures, and recently there have been some advancements in unsupervised feature learning in SNNs [Neftci et al., 2013] [Masquelier and Thorpe, 2010] [Jonke et al., 2017].

In this chapter, we propose SNN feature learning architectures compatible with phase-change technology. Firstly, in Sec. 6.1, we discuss the key role of feedback for shaping the direction of learning in SNNs. In Sec. 6.2 we provide a more explicit theoretical analysis of the learning using feedback within a single neuron. In Sec. 6.3 we propose to introduce 1-bit – a minimum amount, of additional feedback to the STDP learning rule, and experimentally demonstrate feature learning capabilities of such architecture using phase-change synapses. In Sec. 6.4 we discuss a biologically-feasible mechanism of synaptic competition that provides feedback between the synapses of different neurons. We use the result of this feedback to build a dynamically-sized network architecture and experimentally demonstrate its feature learning capabilities. Lastly, in Sec. 6.5, we demonstrate that the proposed SNN architecture with synaptic competition enables to learn the features from datasets with input presentation distribution that is challenging for most of the machine learning architectures.

6.1 Feedback in spiking neural networks

Feedback links are a critical element for controlling the learning in neuromorphic SNNs. Because a neuromorphic system is a physical design rather than an abstract mathematical optimization problem, all representation constraints must be physically implementable in form of feedback links. In Fig. 6.1 we present various types of feedback links that can be used in a single layer of spiking neurons.

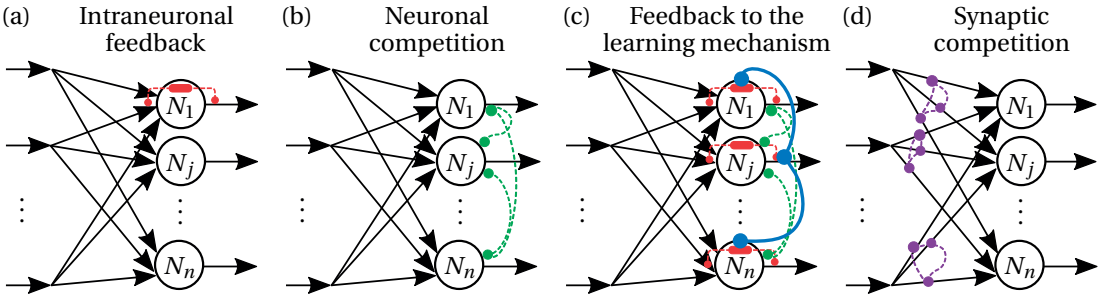


Figure 6.1: **Feedback schemes in SNNs** (a) Intraneuronal feedback marked in N_1 is typically not drawn in network diagrams. It enables learning in neurons. (b) Neuronal competition is depicted as links between the neurons. It enables divergent learning in a network of neurons. (c) We propose a scheme that enriches the computational capabilities using 1-bit feedback links from outputs of neurons to the learning mechanism of other neurons. (d) We also propose a scheme with feedback between the synapses that leads to synaptic competition.

The most ubiquitous type of feedback is a link from the output of a neuron to its synapses, marked for N_1 in Fig. 6.1a. Because this is an intraneuronal feedback link, it is commonly concealed inside the depiction of a neuron N_j. It may be drawn only in the figures explaining a single neuron, such as in Fig. 2.5 (page 20), or even entirely omitted. This intraneuronal feedback provides means for implementing STDP learning mechanism, which in case of a network with the same type of neurons, leads to learning the same pattern by all of them.

To foster learning different patterns, mechanisms such as WTA or k-WTA are introduced. They are typically depicted as feedback links between the neurons in Fig. 6.1b, although the implementation details are usually more intricate, as described for lateral inhibition and level-tuned neurons in Sec. 4.3.1. These mechanisms lead to neuronal competition for the patterns, so that the neurons communicate within the network to decide how to split the inputs. Then, because they operate using regular learning mechanisms, the winning neurons tend to simply memorize the corresponding inputs.

To advance the learning beyond memorization, we enrich the learning capabilities of the neurons. We propose to use additional binary feedback links from the outputs of the neurons to the learning rules of other neurons [Woźniak et al., 2017b], illustrated in Fig. 6.1c. In consequence, the neurons not only decide how to split the patterns among themselves, but also how to split the content of these patterns to learn the features, as explained in detail in Sec. 6.3.

Lastly, we propose an architecture, in which competition is moved from the neurons to the synapses [Woźniak et al., 2017a], as illustrated in Fig. 6.1d. Synaptic competition arises through feedback between the synapses of different neurons connected to the same input. It provides important computational capabilities at the synapses, which enable feature learning using standard learning mechanisms and without the need for neuronal competition, as explained in detail in Sec. 6.4.

6.2 Analytic interpretation of intraneuronal feedback

To explore the computational capabilities of interneuronal feedback, a more explicit understanding of the learning through intraneuronal feedback is required. In this section, we analytically interpret learning in a single neuron, assuming that the inputs are temporally-coded and the information is conveyed through correlated activity at the inputs. Firstly, we analyze the operation of A-STDP. Next, we analyze the simplified STDP, described in Sec. 2.3.5, and the implications of its different learning semantics. Lastly, we consider the conditions under which the A-STDP can perform similar computation to the simplified STDP.

6.2.1 A-STDP

We provide an interpretation of the operation performed by the intraneuronal feedback in a single neuron with synapses using A-STDP [Woźniak et al., 2017b]. For improved tractability, we constrain the analysis to a single potentiation and a single depression, illustrated in Fig. 6.2. Let X be a set of correlated inputs that appear at time t_X and causes the neuron to emit a post-synaptic spike, and let Y be a set of correlated inputs that appear at later time t_Y and does not result in a post-synaptic spike. Assume that there are no other spikes present in the learning window, so that we can apply the STDP logic from Alg. 1 (page 25). Furthermore, consider all possible spatial relationships between the sets X and Y in the universe $U = \{1, \ldots, m\}$ of input attributes, as presented in a Venn diagram in Fig. 6.2b. In this context, we formulate the weight adjustments evoked at t_X and t_Y separately for each subset of synapses in Tab. 6.1.

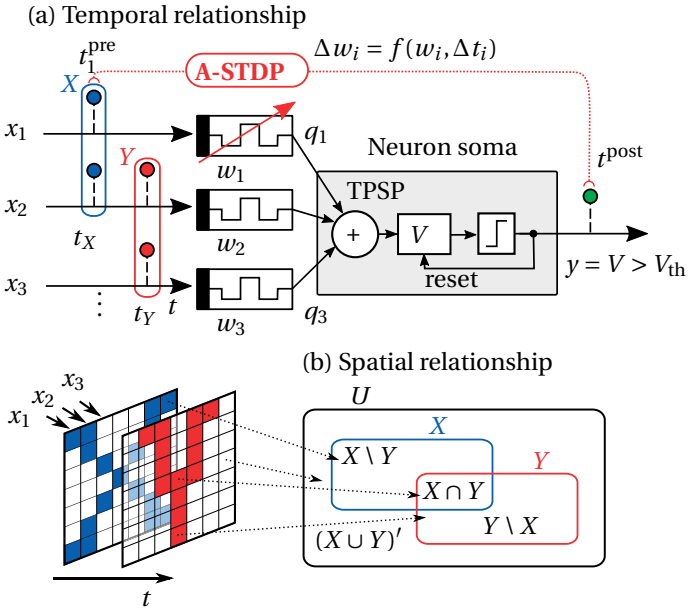


Figure 6.2: **Spatio-temporal patterns arriving at a neuron** (a) The neuron spikes for a pattern X, and does not spike for a consecutive pattern Y. (b) A Venn diagram of all possible spatial relationships between the correlated patterns X and Y. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

$i \in$	w_{ji} **at time** t_X	w_{ji} **at time** t_Y
$(X \cup Y)'$	$w_{ji} := w_{ji}(t_0) + f_+(w_{ji}(t_0), t_X - t_i^{\text{pre}})$	no spikes: $w_{ji} := w_{ji}(t_0)$
$Y \setminus X$	$= w_{ji}(t_0)$ as: $t_X - t_i^{\text{pre}} \notin [-T_{\text{LTD}}, T_{\text{LTP}}]$	$w_{ji} := w_{ji}(t_X) + f_-(w_{ji}(t_X), t_X - t_Y)$
$X \cap Y$	$w_{ji} := w_{ji}(t_0) + f_+(w_{ji}(t_0), t_X - t_X)$	$= w_{ji}(t_X) - w_{ji}(t_X) = \mathbf{0}$
$X \setminus Y$	$= \boldsymbol{w_{ji}(t_0) + f_+(w_{ji}(t_0), 0)}$	no spikes: $w_{ji} := \boldsymbol{w_{ji}(t_X)}$

Table 6.1: **Analytic interpretation of A-STDP** The effective weight adjustments for all possible spatial relationships between X and Y. Table from [Woźniak et al., 2017b], © 2017 IEEE.

Let us define a set of inputs that correspond to the synapses that were potentiated as a result of two patterns, X and Y, appearing within an STDP learning window as $P_j(\boxed{XY}) = \{i : \Delta w_{ji} > 0\}$. Then, based on the potentiation and depression events marked in bold in Tab. 6.1, we conclude that a spiking neuron firing for pattern X and using A-STDP performs a set minus operation $P_j(\boxed{XY}) = X \setminus Y$ when the pattern Y appears in the depression part of the learning window [Woźniak et al., 2017b].

This analytic interpretation complies with the pattern learning results. In case of learning a perfectly correlated pattern in Sec. 4.1.1, the neuron spikes for inputs containing a pattern, and remains inactive for the inputs with noise. Executing \ operation corresponds to removing the noise from the representation stored in the weights, and leads to correctly learning a denoised pattern. In case of learning multiple patterns in Sec. 4.3, the common part depression artifacts are a direct consequence of executing $X \setminus Y$ operation, which depresses the weights at $X \cap Y$.

6.2.2 Simplified STDP

Simplified STDP is an STDP-like learning rule that has different semantics than the classic STDP, as discussed in Sec. 2.3.5. To analyze its operation, similarly to the previous section, we consider an arrival of a pattern X at t_X followed by an arrival of a pattern Y at t_Y. We assume that $t_Y - t_X > T_{\text{LTP}}$. Because the learning is performed on the post-spikes only, we also assume that the neuron emits post-spikes for both patterns.

We formulate the respective weight adjustments at t_X and t_Y in Tab. 6.2. The arrival of the pattern X triggers LTP with a fixed magnitude α_+ at the synapses corresponding to the inputs $i \in X = (X \cap Y) \cup (X \setminus Y)$, and LTD with a fixed magnitude α_- at all other synapses. Similarly, the arrival of the pattern Y triggers LTP at the synapses corresponding to the inputs $i \in Y = (Y \setminus X) \cup (X \cap Y)$, and LTD at all other synapses.

We compare the final weights with respect to the initial weights. The weights that do not belong to any of the patterns are strongly depressed. The weights corresponding to the unique parts of patterns X and Y remain around their initial values, depending from the ratio of the parameters α_- and α_+. The set of the synapses that increases their weights the most is: $\arg\max_{\{i\}} w_{ji}(t_Y) - w_{ji}(t_0) = X \cap Y$. Therefore, we conclude that the simplified STDP rule has a propensity to preform $X \cap Y$ operation.

$i \in$	w_{ji} **at time** t_X	w_{ji} **at time** t_Y
$(X \cup Y)'$	$w_{ji} := w_{ji}(t_0) - \alpha_-$	$w_{ji} := w_{ji}(t_0) - 2\alpha_-$
$Y \setminus X$	as: $t_X - t_i^{\text{pre}} \notin [-T_{\text{LTP}}, 0]$	$w_{ji} := w_{ji}(t_0) - \alpha_- + \alpha_+$
$X \cap Y$	$w_{ji} := w_{ji}(t_0) + \alpha_+$	$w_{ji} := \boldsymbol{w_{ji}(t_0) + 2\alpha_+}$
$X \setminus Y$	as: $t_X - t_i^{\text{pre}} \in [-T_{\text{LTP}}, 0]$	$w_{ji} := w_{ji}(t_0) + \alpha_+ - \alpha_-$

Table 6.2: **Analytic interpretation of simplified STDP** The effective weight adjustments for all possible spatial relationships between X and Y.

6.2.3 Inverted A-STDP

Here, we consider the conditions under which the A-STDP may have similar semantics to the simplified STDP and perform the $X \cap Y$ operation. We propose to invert the logic of the LTD in the pre- and post-spike STDP Alg. 1 (page 25) from operating on inputs $i \in S$ that received spikes to operating on the complement of the inputs $i \notin S$, as illustrated in Fig. 6.3. In consequence, the realized operation becomes $P_j(\boxed{XY'}) = X \setminus Y' = X \cap Y$. We refer to this mode of A-STDP operation as *inverted A-STDP* [Woźniak et al., 2017b].

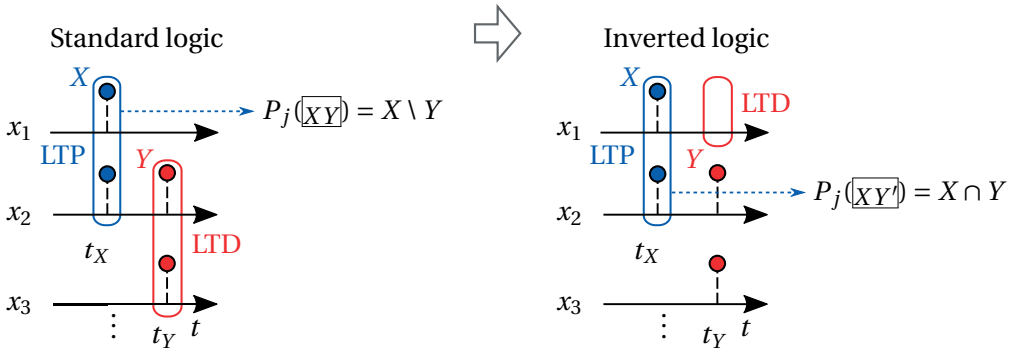


Figure 6.3: **Operation of inverted A-STDP** In comparison to the standard logic, the inverted logic applies depression to the complement of the pattern Y, which results in $X \cap Y$ operation.

6.3 Feedback to the learning mechanism

As discussed above, the choice of a particular variant of the STDP learning mechanism determines the type of the operation performed by a neuron. In this section, we propose to enrich the computational capabilities of a neuron by using different modes of STDP operation, controlled by additional feedback to the learning mechanism [Woźniak et al., 2017b]. We use the least amount of feedback possible – 1 bit, which may be implemented similarly to communicating 1-bit inhibitory signal of lateral inhibition to the soma of that neuron. That single bit of feedback provides possibility to switch between two modes of STDP learning.

To determine which two modes of learning would be useful for feature extraction, we analyze the manual decomposition of the Weather features from Fig. 5.3 (page 70). Firstly, in Fig. 6.4a

we formulate the computation of each feature as a sequence of two base operations: \ and ∩. Secondly, in Fig. 6.4b we propose a network architecture with WTA applied to neuronal modules M_k, that enables multiple neurons in the module to become active for a single pattern. We use modules with four feature-learning neurons capable of operating using two operations, assigned to extract the PCA-like features matching the manual decomposition.

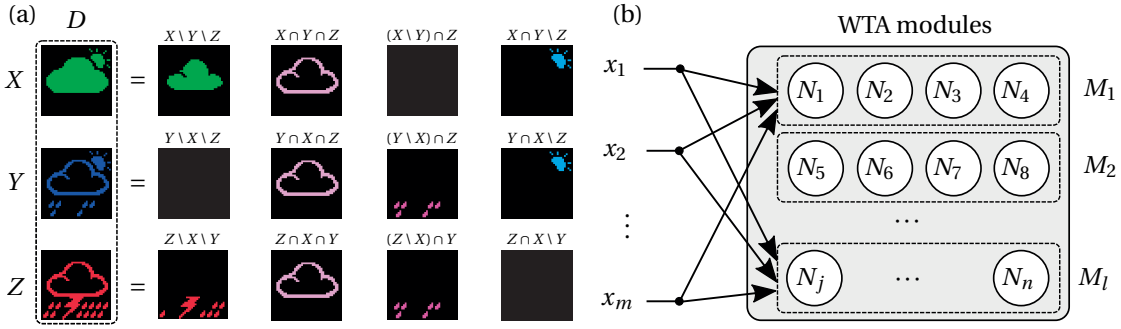


Figure 6.4: **Patterns are decomposed into features defined by sequences of operations** (a) Each PCA-like feature of the Weather dataset is obtained using two operations: \ and ∩. (b) WTA mechanism applied to neuronal modules consisting of multiple neurons; here modules with four neurons are depicted. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

6.3.1 Feature learning A-STDP

We implement a learning mechanism capable of switching between \ and ∩ operation through the proposed concept of feedback to the learning. We base it on the standard A-STDP logic that provides the \ operation, discussed in Sec. 6.2.1, and the inverted A-STDP logic that provides the ∩ operation, discussed in Sec. 6.2.3. Feature learning A-STDP that combines both \ and ∩ operation is formulated in Alg. 3 [Woźniak et al., 2017b]. The \ operation is implemented according to the depression characteristics of the A-STDP (lines 10 and 11). The inverted A-STDP logic (lines 7 and 8) is enabled if any neuron N_k connected to the neuron N_j with a feedback link activates that link (line 6).

The structure of the feedback connectivity is defined by the set invert_j, and determines the type of operation executed by each feature learning neuron. If $k \in \text{invert}_j$, then an activation z_k of a neuron N_k provides feedback to the learning rule of the feature learning neuron N_j. For instance, for the Weather dataset we use an architecture from Fig. 6.4b with three neuronal modules M_1, M_2 and M_3, that are sensitive to patterns X, Y and Z, respectively. We assume that the values $z_k, k = 1 \ldots l$, correspond to the outputs of l pattern-learning neurons using Alg. 2, so that $z_k = 1$ provides information about an appearance a pattern X, Y or Z. For module M_1 the feedback connectivity is structured as in Tab. 6.3. If inverted A-STDP is never enabled, as for N_1, then such a neuron will keep executing the \ operation in the A-STDP learning window, extracting the unique part of a pattern. If inverted A-STDP is always enabled, as for

Algorithm 3 Feature learning A-STDP

1: $S = \{i : x_i = 1\}$
2: **if** $y_j = 1$ **then**
3: $\quad t_j^{\text{post}} \leftarrow t$
4: $\quad \forall_{i \in S}\ t_i^{\text{pre}} \leftarrow t$
5: $\quad$ **if** not learned$_j$ **then** $\forall_i\ w_{ji} \leftarrow w_{ji} + f_+(w_{ji}, t_j^{\text{post}} - t_i^{\text{pre}})$
6: **else if** $\exists_{k \in \text{invert}_j} z_k = 1$ **then**
7: $\quad \forall_{i \notin S}\ t_i^{\text{pre}} \leftarrow t$
8: $\quad \forall_{i \notin S}\ w_{ji} \leftarrow w_{ji} + f_-(w_{ji}, t_j^{\text{post}} - t_i^{\text{pre}})$
9: **else**
10: $\quad \forall_{i \in S}\ t_i^{\text{pre}} \leftarrow t$
11: $\quad \forall_{i \in S}\ w_{ji} \leftarrow w_{ji} + f_-(w_{ji}, t_j^{\text{post}} - t_i^{\text{pre}})$

N_2, then such a neuron will eventually extract the common part. Other features are extracted as combinations of the operations. The effective operation of the neurons within the module learning X for patterns Y and Z appearing in the depression window is presented in the last column of Tab. 6.3, and is consistent with the feature definitions in Fig. 6.4a. Similar feedback structure is defined for modules M_2 and M_3.

Neurons $N_j \in M_1$	invert$_j$	**Effective operation**
N_1	$\emptyset$	$X \setminus Y \setminus Z$
N_2	$\{2,3\}$	$X \cap Y \cap Z$
N_3	$\{3\}$	$(X \setminus Y) \cap Z$
N_4	$\{2\}$	$X \cap Y \setminus Z$

Table 6.3: **Effective operation of a neuron in a neuronal module** Inverted STDP is enabled in different cases for different neurons. This table contains inversions for neurons in module M_1 learning X, based on the spiking of neurons responsive to Y and Z. Inversions lead to execution of a different sequence of operations by each neuron. Table from [Woźniak et al., 2017b], © 2017 IEEE.

However, even if we successfully extract a particular feature, A-STDP potentiation in line 5 of the algorithm will gradually re-potentiate the entire pattern. For example, re-appearance of the pattern X at the input results in a gradual re-learning of the entire pattern X and forgetting the insights from the cross-pattern feedback between X and Y. Therefore, after the neuron has learned a certain pattern, we disable the potentiation in line 5 of the algorithm. Allowing only depression enables the extraction of the features through the feedback mechanism. The learned$_j$ criterion can be determined using various approaches, such as elapsed training time or convergence of the values of the weights to their upper bounds. Alternatively, we can track the maximum values of TPSP$_j$ to indirectly reason about the state of the weights. If the neuron has learned, TPSP$_j$ exhibits high instantaneous values, and then we disable the potentiation.

6.3.2 Orthogonal feature learning results

The feature learning architecture with feedback to the learning mechanism was both simulated and realized experimentally [Woźniak et al., 2017b] using the platform described in Sec. 3.1.3. Learning was executed for 600 s with 12 feature learning neurons having 1600 synapses of 1-PCM type per neuron. The synapses utilized the feature learning A-STDP from Alg. 3. As illustrated in Fig. 6.5, all features of the Weather dataset were discovered in both cases. Even though the experimental results exhibit variability due to the intrinsic hardware cell characteristics, the features can be clearly distinguished from the synaptic weights.

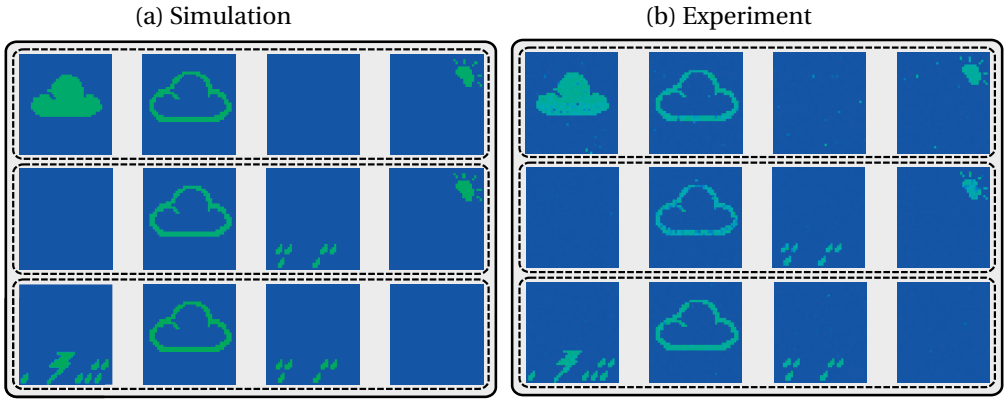


Figure 6.5: **Features learned using feedback to the learning** Synaptic weights obtained using feature extracting neurons in: (a) simulation using a model of the phase-change synapse, (b) experiment using phase-change devices. Figure from [Woźniak et al., 2017b], © 2017 IEEE.

The features are of PCA-like type, so the functionality of the proposed feature learning algorithm involves application of an orthogonality constraint. The orthogonality constraint is realized through the $\setminus$ operation in the sequences of operations defined in Tab. 6.3. When a pattern Y appears at the input and the neuron receiving feedback to the learning rule calculates the feature $X \cap Y$, other neurons perform $X \setminus Y$ operation. The $\setminus$ operation removes any potential overlap between what is learned by the neurons operating in either of the two A-STDP modes: $(X \cap Y) \cap (X \setminus Y) = \emptyset$, thus ensuring pairwise orthogonality of the weights. Based on this insight, it might be interesting to explore alternative versions of Alg. 3 to extract other types of features using different operations than $\{\setminus, \cap\}$, as discussed in the next section.

6.3.3 Generalization of the feature learning architecture

In this section, we generalize the proposed feature learning architecture with feedback to the learning mechanism for application to different types of features and to larger datasets. We demonstrate operation of a scaled-up version of this architecture in a simulation using Swimmer and Bars datasets, introduced in Sec. 5.2. The results are compared to the ones obtained from NFM and LNMF matrix factorization, described in Sec. 5.3.3, and ANN autoencoder, described in Sec. 5.4.1.

We propose a scaled-up version of the architecture with feedback to the learning mechanism, as its direct application to a larger dataset would require many neurons. For a dataset with d patterns, the architecture from Fig. 6.4b grows vertically to d neurons. Then, each pattern can be combined with $d-1$ remaining patterns in 2 ways ($\setminus$ or $\cap$), that corresponds to growing the size of the architecture horizontally. The total number of neurons would be $d \cdot 2^{(d-1)}$. However, we notice that many combinations yield empty features, as already visible for 3 patterns in Fig. 6.4a. Therefore we constrain the size of the network vertically and horizontally to n_v and n_h neurons respectively. The architecture extracts features not from the entire set, but rather from n_v random patterns. Then, in horizontal dimension, we use random values for invert_j, which corresponds to having random feedback connectivity between the neurons. To further avoid empty features, we stop the algorithm when the smallest feature was found.

To obtain the benchmark results, we use 150 basis vectors in case of NMF and LNMF, and 150 neurons in case of feature extracting SNN and ANN autoencoder. We apply our architecture in $\{\setminus, \cap\}$ version with $n_v = 15$ and $n_h = 10$, LNMF and ANN autoencoder, trained for 10^4 epochs with $\alpha = 0.1$, $\lambda = 0.01$ and squared loss, to 100 samples from the Swimmer dataset. To test the functionality of the architectures in a comparable manner, we execute them in software without incorporating weights' variability. For each algorithm we find features that best match the ground truth in terms of L_2 norm. We report ground-truth feature visualization accuracy using MSE. All methods find the correct features, presented in Fig. 6.6. Our approach exactly matches the ground truth (MSE = 0), because of how the algorithm works: it either discovers the correct sequence of set operations that leads to the proper feature, or does not, in which case a very sharp suboptimal feature is found. LNMF achieves MSE = 0.12 and autoencoder MSE = 0.02.

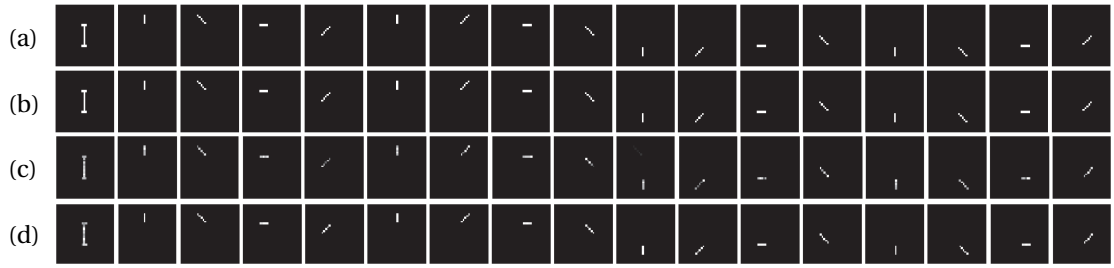


Figure 6.6: **Feature learning results for the Swimmer dataset** (a) Ground truth decomposition. Best matching features: (b) $\{\setminus, \cap\}$ feature learning SNN, (c) LNMF, (d) autoencoder. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

For the Bars dataset we use our algorithm, LNMF and NMF, and present the results in Fig. 6.7. LNMF method fails as it attempts to extract features with orthogonality constraint. NMF, which does not use this constraint, successfully extracts all the features with MSE = 0.19. The $\{\setminus, \cap\}$ version of our algorithm fails similarly to LNMF, as it also extracts PCA-like features. Therefore, we consider another variant of our algorithm, by replacing the orthogonalizing $\setminus$ operation with _ representing no operation. The $\{\_, \cap\}$ version of the algorithm can obtain

features without the orthogonality constraint, and for $n_v = 75$, $n_h = 2$ successfully extracts with no error all the ICA-like features. The capability to extract ICA-like features using only the $\cap$ operation suggests that the simplified STDP might be used for independent feature learning, which will be explored in the next section.

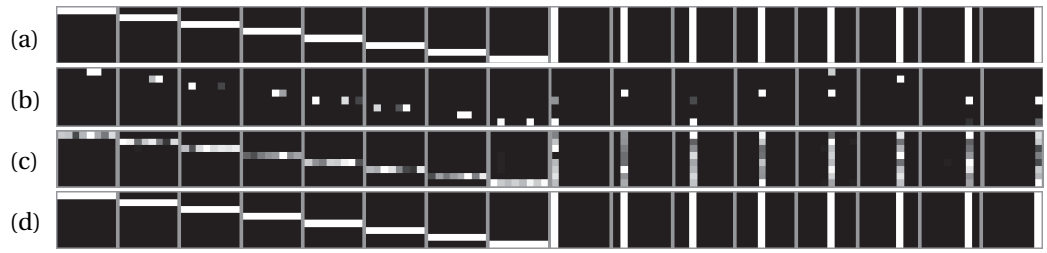


Figure 6.7: **Feature learning results for the Bars dataset** (a) Ground truth decomposition. Best matching features: (b) LNMF, (c) NMF, (d) $\{\_, \cap\}$ feature learning SNN. Figure adapted from [Woźniak et al., 2017b], © 2017 IEEE.

The presented results illustrate the applicability of the architecture with feedback to the learning mechanism for extraction of PCA-like and ICA-like features. The architecture can be implemented using efficient phase-change-based neuromorphic hardware, and its computational properties may be adjusted through the choice of the base operations and the feedback connectivity structure. This could enable extracting other feature types, beyond the ones demonstrated in this section. Nevertheless, the architecture uses more neurons than the number of features it extracts, so it is appealing to explore alternative forms of feedback that could lead to more compact architectures with feature-learning capabilities.

6.4 Feedback between the synapses

Most of the feedback schemes, such as those illustrated in Fig. 6.1a-c, consider feedback links inside and between the neurons, which lead to neuronal competition. However, it is possible to explore competition schemes at earlier stages. Biological insights suggest that interactions between synapses of different neurons play a role in the learning [Sajikumar et al., 2014]. Moreover, we can use the results of these interactions to guide the forming of connections to new neurons, and develop more flexible network architectures that do not require an *a priori* specified network size [Woźniak et al., 2017a]. Throughout this section, we introduce the concept of synaptic competition, incorporate it into the learning rule, and propose a dynamically-sized network architecture. Lastly, we describe the implementation and present experimental results of learning PCA-like and ICA-like features.

6.4.1 Synaptic competition

We introduce a model of synaptic competition [Woźniak et al., 2017a] that is schematically illustrated in Fig. 6.8a in analogy to the neuronal competition using lateral inhibition at the

neurons. In biology, synaptic competition is a consequence of the dynamics at the synapses, which compete for learning-related proteins [Sajikumar et al., 2014]. Owing to limited resources around an axon, not all of the synapses connected to the same pre-synaptic axon will be able to increase their weights during learning.

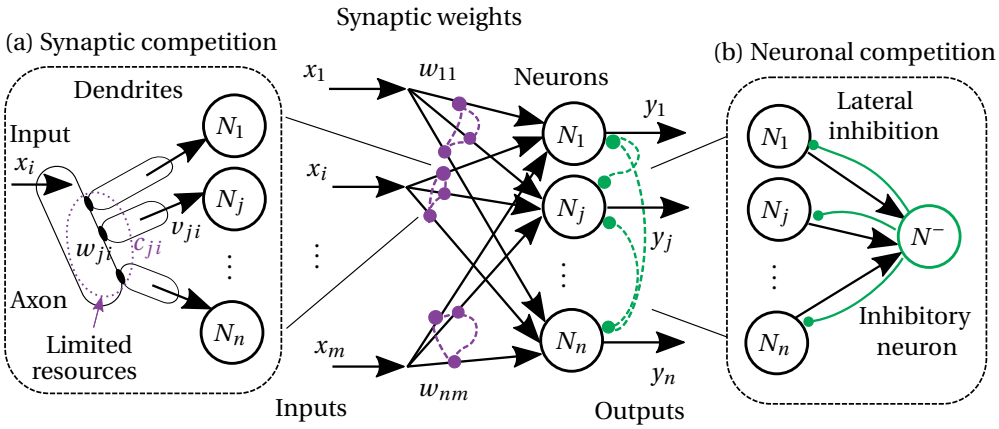


Figure 6.8: **Competition mechanisms in front and behind the neurons** (a) Synaptic competition is a consequence of limited resources in front of the neurons. (b) Neuronal competition is a consequence of inhibitory activity behind the neurons. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

In comparison to neuronal competition in Fig. 6.8b that leads to neuronal WTA dynamics, the competition occurs at an earlier stage: before the neurons. We take inspiration from this, and introduce a simplified synaptic competition model with instantaneous effect that leads to synaptic WTA dynamics. Our model assumes that from all the synapses connected to the same input, only one is allowed to increase its weight. We determine this synapse, called the winning synapse, as the one having the highest post-synaptic potential v_{ji} at the dendrite. The competition result c_{ji} is determined as:

$$(c_{1i},\ldots,c_{mi}) = \mathrm{WTA}(v_{1i},\ldots,v_{mi}) \tag{6.1}$$

$$\text{if } v_{ji} = \max(v_{1i},\ldots,v_{mi}) \text{ then } c_{ji} = 1, \text{ else } c_{ji} = 0. \tag{6.2}$$

The semantics of the WTA function in synaptic competition is the same as in neuronal competition: an index corresponding to the maximum value is chosen as the winner. In case of a tie, the synapse with the lowest index becomes the winner.

In biological neurons, the potentials at the different parts of the neuron: the synapses, particular segments of the dendrites and the soma, may be analyzed separately owing to their capacitive properties [Stuart et al., 2016, p. 443]. In our simplified model, we assume that the resting potential at a dendrite is 0 and then we propose to use two approaches to determine the potential v_{ji}. First, using only the charge induced by an axonal spike x_i in the pre-synaptic neuron, denoted with 'A': $v_{ji}^A = w_{ji}x_i$. Second, considering both an axonal spike 'A' in the pre-synaptic neuron, and a back propagating action potential (bAP) [Sjöström and Häusser,

2006] in the post-synaptic neuron, denoted with 'B'. We assume that the contribution of a bAP corresponds to the binary value of the post-synaptic neuronal activation y_j; therefore, $v_{ji}^{AB} = w_{ji} x_i y_j$. This does not aim to directly correspond to in vivo observations: analyzing the dynamics of bAPs on the voltages across the compartments of a neuron is a wide research topic [Sjöström and Häusser, 2006] [Sjöström et al., 2008].

We illustrate the results of synaptic competition for three sample neurons by visualizing c_{ji} values in Fig. 6.9. The input pattern consists of three bars, similarly to the Bars dataset. With neurons having the weights presented in the figure, the pattern activates neurons N_1 and N_3. In the process of synaptic competition, the synapses with the highest weights in each neuron will become the winners. In the c_{ji}^{AB} case, there is one tie between the synapses of neurons N_1 and N_3 competing for an input in the middle of the pattern (marked by a red circle), but it is resolved based on the lowest-index principle and the synapse from N_1 wins. In the c_{ji}^{A} case, there is additionally one tie between the synapses of the neurons N_2 and N_3, and the synapse from N_2 wins. The results of synaptic competition c_{ji}^{A} or c_{ji}^{AB} constitute then the inputs to the respective neurons N_j.

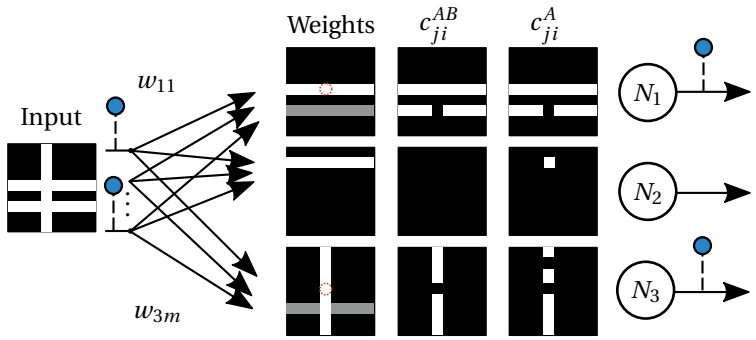


Figure 6.9: **Synaptic competition example** The results of synaptic competitions are visualized in form of c_{ji} matrices for each neuron. Values are proportional to their brightness. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

Lastly, we notice that there is a conceptual similarity between the proposed synaptic competition mechanism and the dendritic inhibition from Sec. 5.4.3, as both approaches use feedback that affects the synaptic inputs. In dendritic inhibition, a system of equations is optimized to find a stable state of inhibited input values x'_{ji} and neural activations y_j. The inhibited input values x'_{ji} are then discarded, because they just provide means for obtaining activations y_i. These activations are then used by a dedicated learning rule, combined with weights normalization. In contrast to this approach, synaptic competition is simple to compute and directly returns the competition result c_{ji} using the WTA function. The neuronal activations y_j are calculated in the classic way, without a need to optimize a system of equations. Finally, the competition results c_{ji} are further utilized by the learning mechanism.

6.4.2 Incorporating synaptic competition into the learning

In the architecture with synaptic competition, we incorporate the results of competition c_{ji} into the learning using simplified STDP, described in Sec. 2.3.5. LTP is performed only for the weights of the winning synapses ($c_{ji} = 1$), whereas the weights that lose in the synaptic competition ($c_{ji} = 0$) remain unchanged. LTD operation is the same as in simplified STDP: synapses that do not observe input spikes in T_{LTP} undergo LTD.

For simplicity of the analysis, we assume that the periods between appearances of different patterns are larger than T_{LTP}, and that the patterns' spikes arrive all at once, so that we can visualize the weights' adjustments solely based on the current input. In Fig. 6.10, we illustrate the operation of the simplified STDP rule in a network of three neurons with synaptic competition. Next to the initial weights, the results of synaptic competition are presented and denoted with c_{ji}, as in this particular case they are identical for both v_{ji}^{A} and v_{ji}^{AB} formulation. The synaptic weight adjustments are visualized for neuron N_3 for the simplified STDP and for the simplified STDP with synaptic competition. The simplified STDP performs either LTP or LTD, which leads to gradual overriding of the neuron's knowledge with the current input, whereas with synaptic competition the non-winning synapses remain unchanged.

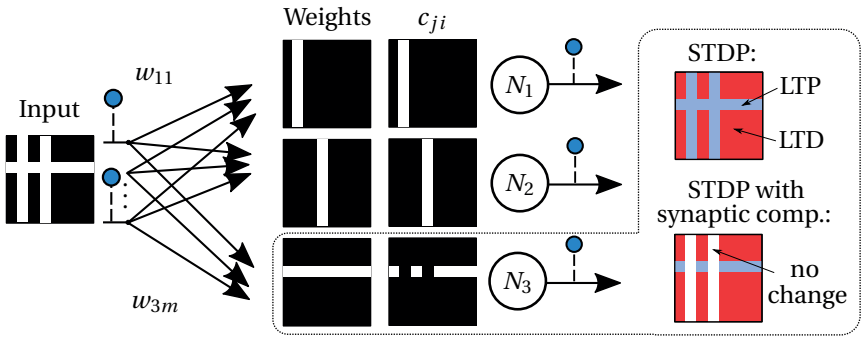


Figure 6.10: **Synaptic competition incorporated into the learning** Comparison of synaptic weights' adjustments for N_3 using simplified STDP with and without synaptic competition. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

6.4.3 Dynamically-sized network

The synaptic weights of all neurons in the network form a representation of the knowledge that together with the input information guides the learning. For example, in ANN autoencoders discussed in Sec. 5.4.1, the information how well the network matches the inputs is quantified by the reconstruction error, which is then used by backpropagation to adjust the weights. With synaptic competition, it is possible to obtain some insight into how well the representation in an SNN matches the input by observing the course of the competition. If a particular synapse clearly wins a competition with a high v_{ji}, this suggests that the specific input is well represented in the network. However, if none of the synapses is able to clearly win in the competition, this suggests that the input is poorly represented. Contrary to ANN autoencoders,

we do not treat this information as error, but rather consider it as a novelty that cannot be represented by the network. We call this novelty *representation overflow* [Woźniak et al., 2017a].

We propose to capture the representation overflow by introducing the concept of an overflow neuron and a dynamically-sized network [Woźniak et al., 2017a]. At each point in time, one additional neuron is denoted as a special overflow neuron N_j^*. Its synaptic weights are fixed to small, non-zero values (we use 0.05), and its spiking threshold is low and equal to a fraction of the desired smallest feature size. A low threshold will make the overflow neuron spike for almost any input and participate in the synaptic competition. In consequence, inputs that cannot be explained using the current knowledge of the network are diverted to this additional neuron that focuses on novelty-based learning. Fig. 6.11 illustrates the operation of a network with regular neurons and an overflow neuron. The overflow neuron N_3^* participates in synaptic competition with a regular neuron N_2, and captures the representation overflow. The overflow neuron becomes a regular neuron N_3, and another neuron is designated as an overflow neuron. Finally, to ensure that the representation in the neurons is meaningful, we perform pruning of the network in a similar way. We disable the neurons that, owing to depression, represent features smaller than the minimum feature size.

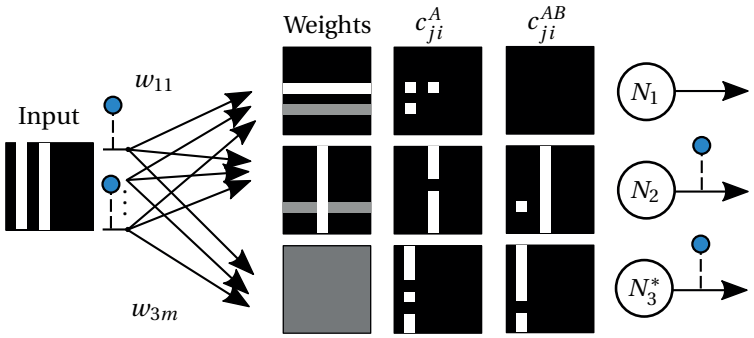


Figure 6.11: **An example of representation overflow** for regular neurons N_1, N_2 and overflow neuron N_3^*. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

6.4.4 Orthogonal feature learning results

In this section, we demonstrate the operation of the introduced feature learning architecture with synaptic competition in a dynamically-sized network. Firstly, we provide a manual step-by-step analysis of learning PCA-like orthogonal features from the Weather dataset based on the v_{ji}^{A} synaptic potential formulation. Then, we use the Swimmer dataset to execute a larger benchmark between various approaches and we implement the proposed architecture in the prototype platform.

An analysis for the Weather dataset

To develop a better understanding of the operation of the proposed architecture with synaptic competition in a dynamically-sized network, we provide a complete example of learning the

features from the Weather dataset in Fig. 6.12. A series of six patterns is presented to the network in the order illustrated in Fig. 6.12a. We assume that the inputs are noise-free and the temporal distance between the patterns is larger than T_{LTP}. The operation of the network is illustrated in Fig. 6.12b. For each input, the first column depicts the competition results c_{ji}^{A}, and the top right corner is used to indicate neuronal spiking. The color of the c_{ji}^{A} indicates LTP (blue) or LTD (red). We assume full LTP and LTD: $\alpha_{+} = 1$ and $\alpha_{-} = 1$. The second column depicts the final weights of a neuron N_j. The last neuron N_n^* is always an overflow neuron.

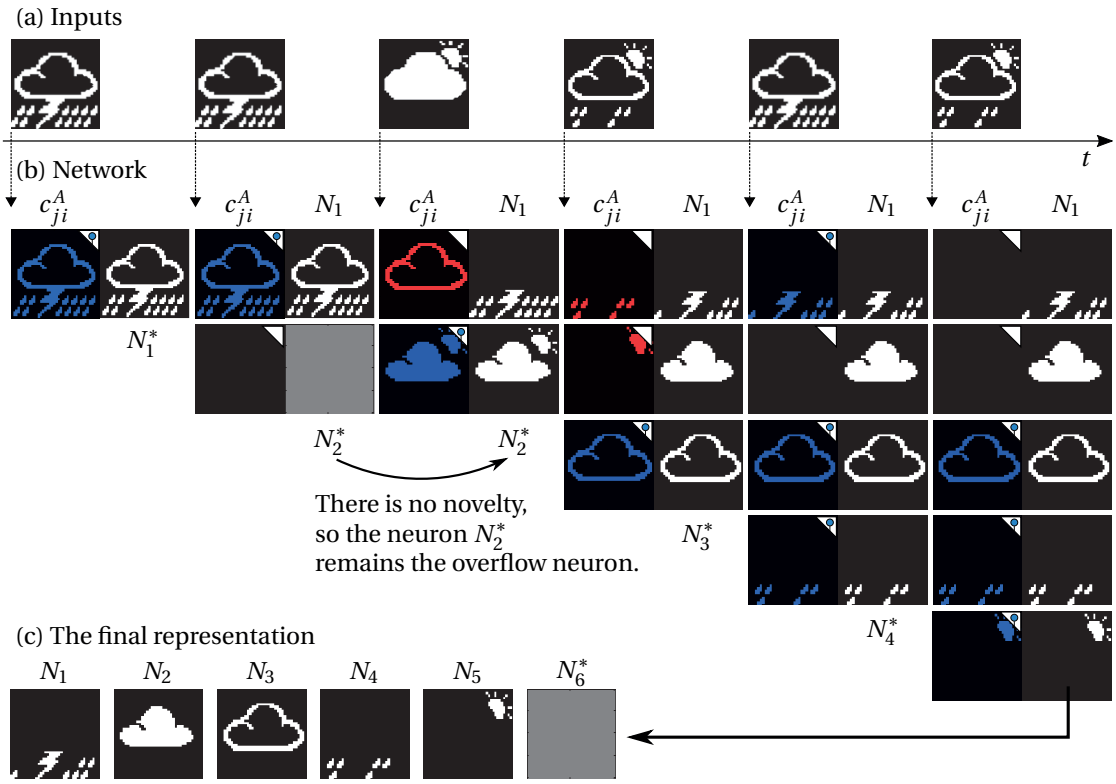


Figure 6.12: **Synaptic competition operation in a dynamically-sized network** (a) Inputs presented to the system. (b) Operation of a network for each input. The last neuron is always an overflow neuron. (c) The final representation comprises five regular neurons $N_1, \ldots, N_5$, and an overflow neuron added after the final step. The overflow neuron will remain inactive, as there is no further novelty in the input patterns.

We explain step-by-step the operation of the network in Fig. 6.12b. The network initially has only one neuron N_1^* that is an overflow neuron. When the first pattern appears, the synapses of the overflow neuron win all the inputs and undergo LTP that stores that pattern. The neuron becomes a regular neuron N_1 and an overflow neuron N_2^* is added to the network. If the same pattern reappears, the synapses of N_1 again win all the inputs and neuron N_1 spikes, whereas the overflow neuron N_2^* remains an overflow neuron. Next, when a different pattern appears, the synapses of the regular neuron N_1 win a significant part of the input. However, because the pattern is different, the total number of active inputs in N_1 do not result in neuronal

spiking and LTD is applied to the winning inputs. Simultaneously, novel parts of the pattern are captured by the overflow neuron N_2^*. It becomes a regular neuron N_2 and another overflow neuron N_3^* is added. The learning continues until all the patterns are decomposed into the features and no further novelty is present in the inputs. The final representation is illustrated in Fig. 6.12c. It corresponds to the ground-truth decomposition of the Weather dataset into PCA-like features from Fig. 5.3 with a difference that it comprises non-redundant features only. Moreover, owing to lack of neuronal WTA mechanism, all respective feature neurons correctly become active when the feature is present in the input.

Experiments for the Swimmer dataset

To validate the operation of synaptic competition on a larger scale, we use 400 patterns from the Swimmer dataset and compare the performance of orthogonal feature learning between an SNN with synaptic competition, an SNN with neuronal competition using simplified STDP, and a feature learning ANN with dendritic inhibition from Sec. 5.4.3.

We implement the v_{ji}^A version of synaptic competition both in software and in hardware (HW) using the neuromorphic platform described in Sec. 3.1.3. Synaptic weights are mapped to a $2\,\mu S - 55\,\mu S$ conductance range of 2048 phase-change cells. For LTD we use reset pulses with $I = 450\,\mu A$ and for LTP we use crystallizing pulses with $I = 130\,\mu A$ applied for a duration of 200 ns. The hardware results are obtained from a single experiment, whereas the software results for all the methods are averaged over 10 trials with different random-number generator seeds. For software simulations of synaptic competition and neuronal competition we use the simplified STDP with LTP $\alpha_+ = 0.1$ and LTD $\alpha_- = 0.2$. In dendritic inhibition, the magnitude of the weight adjustment is defined by the appropriate formulas in Sec. 5.4.3. For neuronal competition and dendritic inhibition, the network size is set to 18 neurons.

For synaptic competition, we use an adaptive spiking threshold to control the neuronal firing. Specifically, the spiking threshold linearly increases over time: starting from an initial value $V_{\text{th}}(t_0)$ (we use 3) at the initial time t_0, it reaches a final value $V_{\text{th}}(t_e)$ (we use 3.4) at a predefined time t_e. The motivation behind this approach is that the lower initial thresholds enable the neurons to learn by more frequent spiking. However, at a later stage, the neurons should become more selective and spike only for the inputs that closely correspond to their weights.

For lateral inhibition, we choose a trivial spiking threshold of 0, as the WTA mechanism ensures that at most one neuron will spike. This choice is possible owing to our simplifying assumptions that each time there is an input, it corresponds to a pattern. In other implementations, especially with noisy inputs and irregular input pattern presentation, it is important to determine the correct threshold, as discussed in Sec. 2.3.3.

Firstly, we assess the results based on the number of features found and the feature visualization MSE per neuron, plotted in Fig. 6.13. The number of features found is calculated by assigning each neuron to a specific feature using the approach presented in [Spratling,

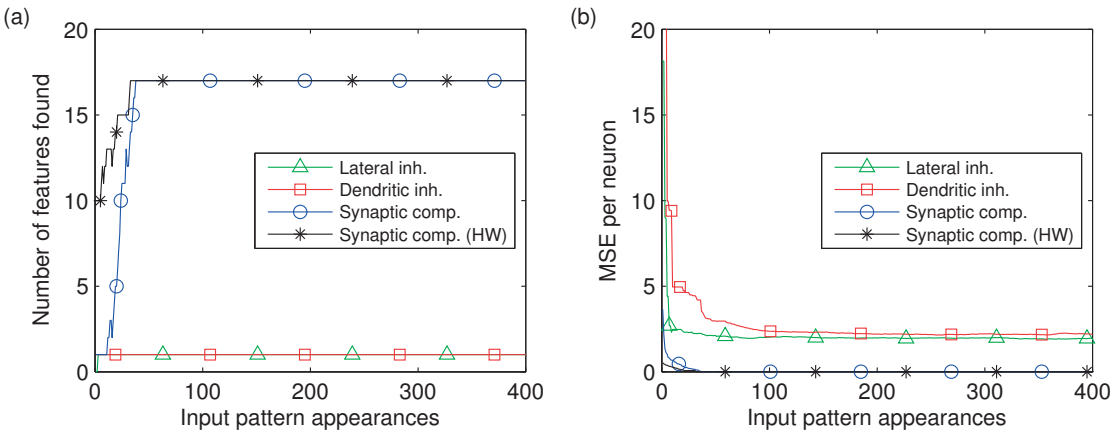


Figure 6.13: **Assessment of the features learned from the Swimmer dataset** (a) Synaptic competition learns all 17 features, whereas the other approaches learn only 1. (b) The mean square error of the learned representation vs. the ground truth.

2006]. Specifically, a neuron is considered to have learned a feature if the sum of the weights corresponding to the feature is at least twice that of the sum of weights corresponding to any other feature and the minimum weight corresponding to the learned feature is larger than the average of all neuron's weights. The MSE per neuron is calculated as an average of the MSE of all neurons in a network, where individual MSE for neuron j is the MSE of the best-matching feature $\boldsymbol{x}^{(k)}$: $\arg\min_k \sum_i (w_{ji} - \boldsymbol{x}_i^{(k)})^2$, where $\boldsymbol{x}^{(k)}$ are the target ground-truth features. As plotted in Fig. 6.13a, synaptic competition with v_{ji}^A correctly learns all 17 ground-truth features of the Swimmer dataset, whereas the remaining approaches learn only 1 feature: the swimmer's body. The MSE per neuron in Fig. 6.13b decreases throughout the learning for all the methods. For synaptic competition it reaches ≈ 0. For the other methods it oscillates around 2, because they learn to correctly capture the swimmer's body, but not to distinguish different limb positions. This is illustrated in the visualization of the final weights in Fig. 6.14a-b, in which beside the sharp body, shadows of multiple limbs are captured. Synaptic competition learns the correct ground truth features both in software and in hardware, as illustrated in Fig. 6.14c-d.

Secondly, in Fig. 6.15, we report the feature detection results based on the spiking accuracy and F-score. We do this in an online manner using a moving window of the last 100 input pattern appearances. The accuracy is calculated as the percentage of the neurons that are activated correctly for the observed input, defined in Eq. 2.13. However, because the sizes of TP and TN classes are imbalanced (a few features appear simultaneously), a system that never spikes achieves a relatively high accuracy. Therefore, we also calculate the F-score, defined in Eq. 2.14. This provides a more accurate assessment than the accuracy, as high TN values do not skew the F-score, as discussed in Sec. 2.4. The values reported for lateral inhibition and dendritic inhibition are a direct consequence of the learned representations: because there are no neurons fulfilling the assignment criteria to any of the 16 limb positions, each time

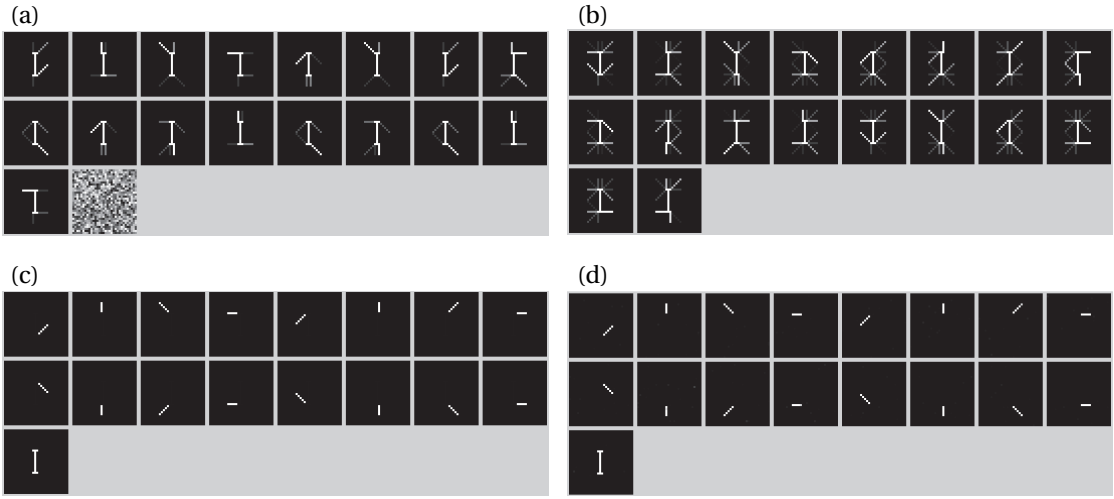


Figure 6.14: **Features learned from the Swimmer dataset** (a) lateral inhibition, (b) dendritic inhibition, (c) synaptic competition, (d) synaptic competition (HW).

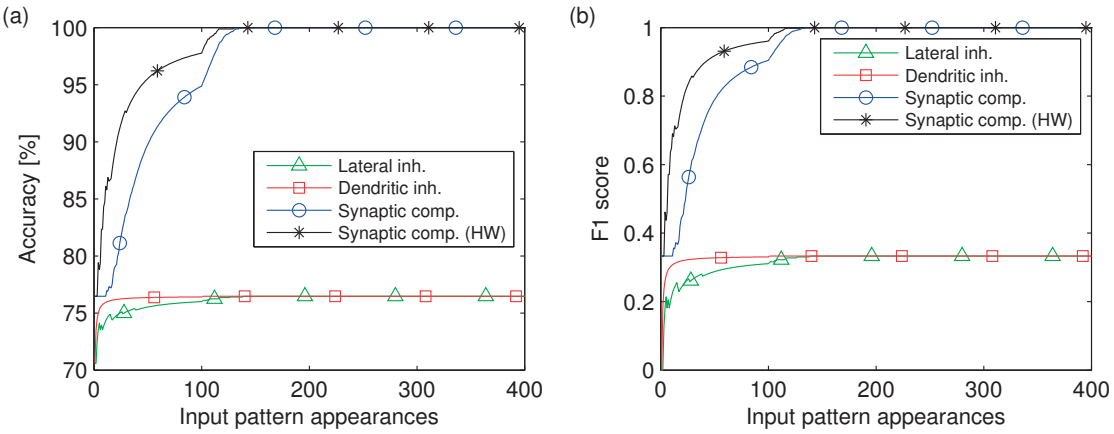


Figure 6.15: **Feature detection for the Swimmer dataset** Assessment of the activity of the networks: (a) accuracy, (b) F-score.

a pattern appears none of the 4 limbs will be properly recognized (FN = 4). Simultaneously, there will be no activity for the remaining limb positions, yielding a high true negative value (TN = 12). Lastly, the activation of any neuron is treated as a true positive detection of the swimmer's body (TP = 1). The accuracy calculated using Eq. 2.13 is: $\frac{1+12}{1+4+12} \approx 76.5\%$, and the F-score from Eq. 2.14 is: $\frac{2}{2+4} \approx 0.333$. On the other hand, synaptic competition achieves the maximum possible feature detection accuracy and F-score, as also summarized in Tab. 6.4.

Table 6.4: **Spiking accuracy and F-scores for the Swimmer dataset**

	Accuracy	F-score
Lateral inh.	76.5%	0.333
Dendritic inh.	76.5%	0.333
Synaptic comp.	100%	1.0
Synaptic comp. (HW)	100%	1.0

6.4.5 Independent feature learning results

The comparison provided in the previous section was performed for PCA-like orthogonal features, which are not commonly supported by lateral inhibition and dendritic inhibition. Therefore, it would be appealing to compare the performance of synaptic competition on the ICA-like Bars dataset with the other methods executed with optimally-tuned parameters. Therefore, the rest of this section is structured as follows: firstly, we discuss the possibility to learn ICA-like features with synaptic competition. Secondly, we determine the optimal network sizes for the other methods. Lastly, we present the benchmark results and visualize the learned features.

Learning different types of features with synaptic competition

The computational capabilities of synaptic competition may be adjusted through the choice of the potential v_{ji}. Changing the form of the v_{ji} leads to introduction of additional constraints that provide different learning functionality. In the previous section, we used $v_{ji}^{A} = w_{ji}x_i$ formulation to learn PCA-like features from the Weather and the Swimmer dataset. For the Bars dataset, the same formulation leads to learning only either 8 horizontal or 8 vertical bars. If we skip the minimum feature constraint, synaptic competition yields a result presented in Fig. 6.16. These 64 pixels form the orthogonal PCA-like features of the Bars dataset, corresponding to Fig. 5.5c. Therefore, we conclude that synaptic competition with v_{ji}^{A} leads to PCA-like orthogonalization of the representation. However, if we use $v_{ji}^{AB} = w_{ji}x_i y_j$ formulation, the computation performed by the network changes. We use this form to extract the Bars throughout the rest of this section.

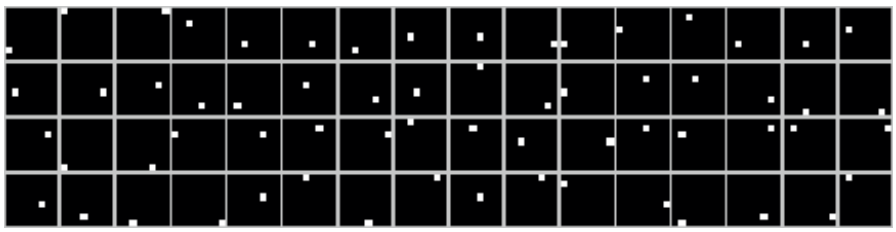


Figure 6.16: **Orthogonal features for the Bars** Synaptic competition with dendritic potential defined as v_{ji}^{A} learns PCA-like features from the Bars.

An optimal network size

In synaptic competition, the representation overflow mechanism dynamically adjusts the network size to the complexity of the dataset. Fig. 6.17a illustrates the evolution of the network size for the Bars dataset, averaged over 10 trials, as a function of input appearances. In all cases, the network size converges to 16 neurons corresponding to the target number of features.

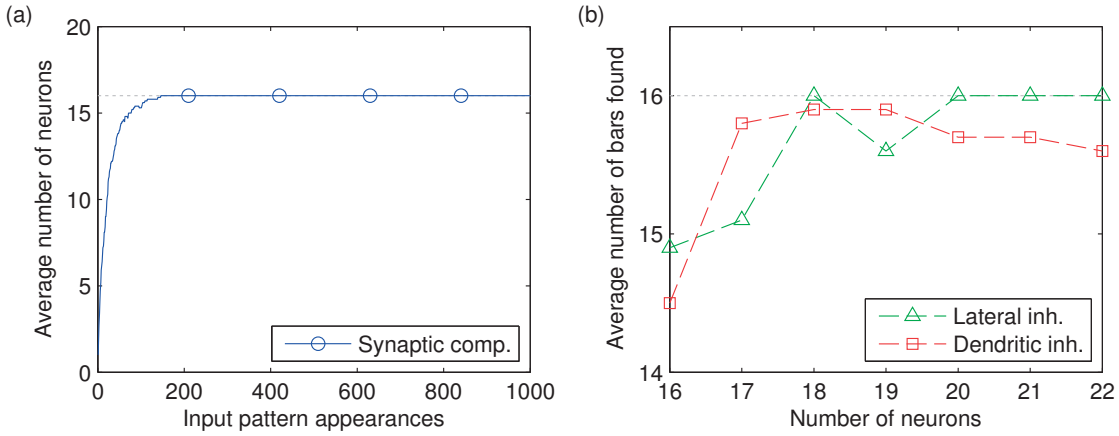


Figure 6.17: **Network size for the Bars dataset** (a) The architecture used with synaptic competition and $V_{\text{th}}(t) \in [5,7]$ dynamically adjusts the size of the network to 16 neurons. (b) The average number of bars found after 10 trials with 1000 input appearances as a function of network size for lateral and dendritic inhibition. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

For the other methods with fixed network size, we determine the smallest optimal number of neurons. A necessary condition to learn k features for a network of size n is that $n \geq k$. However, n should also be as small as possible. Therefore, in Fig. 6.17b, we plot the number of correctly extracted features by lateral inhibition and dendritic inhibition after 1000 input pattern presentations as a function of the network size, starting from $n = 16$. Results were averaged over 10 trials. With an increasing size of the network, performance of lateral inhibition improves, whereas dendritic inhibition performs best for 18 and 19 neurons. The smallest number of neurons for which both methods perform well is 18 neurons, and we use it in the subsequent experiments.

Benchmark results

We execute the benchmark for 1000 samples from the Bars dataset using the experimental setup previously described in Sec. 6.4.4. All approaches find all 16 bars, as illustrated in Fig. 6.18a. In consequence, the MSE per neuron in Fig. 6.18b in all cases decreases throughout the learning. Lateral inhibition learns slower than the other methods, as the neuronal WTA always limits learning to a single neuron. It limits also the fluctuations of the learned weights, in contrast to dendritic inhibition, for which the weights fluctuate at higher MSE values.

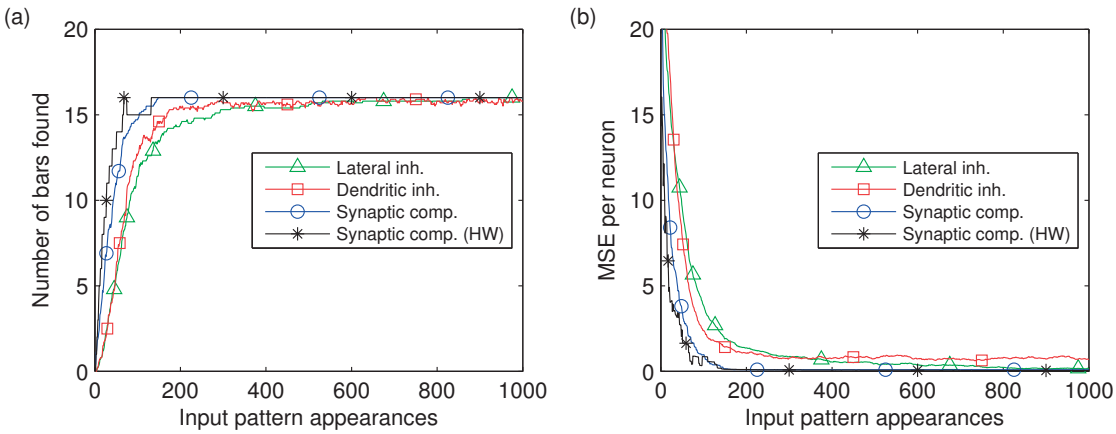


Figure 6.18: **Assessment of the features learned from the Bars dataset** (a) Number of bars found: all methods find all bars. (b) The mean square error of the learned representation vs. the ground truth decreases throughout the learning. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

The sample visualization of the final weights is presented in Fig. 6.19, where neurons are sorted to correspond to the ground truth decomposition of the Bars dataset from Fig. 5.5b (page 71). In Fig. 6.19a-b, the remaining neurons are shown below the 16 best matches for lateral inhibition and dendritic inhibition. Synaptic competition adjusts the size of the network to exactly 16 neurons, as shown in Fig. 6.19c. Hardware version of the synaptic competition also correctly adjusts the network to 16 neurons, as shown in Fig. 6.19d.

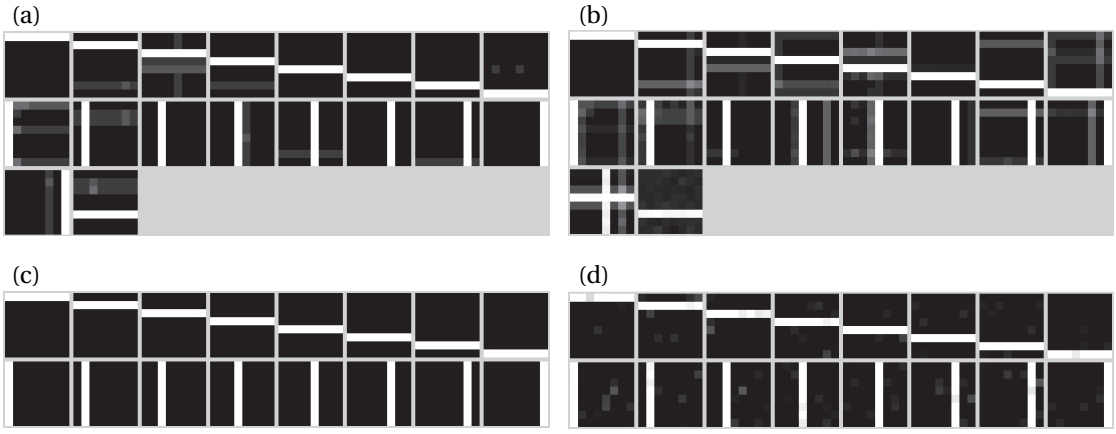


Figure 6.19: **Features learned from the Bars dataset** (a) lateral inhibition, (b) dendritic inhibition, (c) synaptic competition, (d) synaptic competition (HW). Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

The accuracy and F-score are plotted in Fig. 6.20. Synaptic competition performs well and quickly learns to detect the entire dataset, both in software and in hardware (HW). Despite learning all the bars, classic WTA lateral inhibition performs slightly worse in terms of accuracy

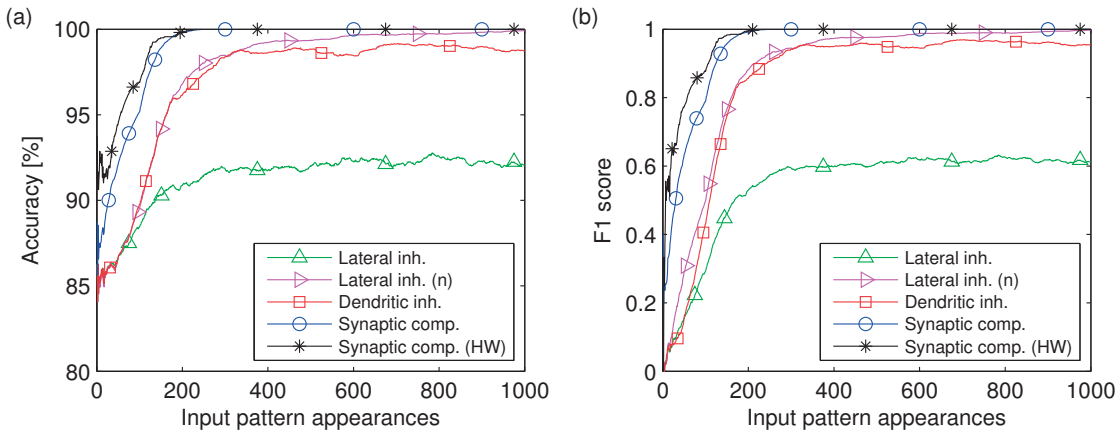


Figure 6.20: **Feature detection for the Bars dataset** Assessment of the activity of the networks: (a) accuracy, (b) F-score. Figure adapted from [Woźniak et al., 2017a], © 2017 IEEE.

and notably worse in terms of F-score. This is a consequence of the operation of the WTA scheme for the neurons, in which only one neuron is allowed to fire and at most one bar can be detected. To go beyond this limitation, it is common to introduce a separate testing phase, where the accuracy is calculated by disabling the lateral inhibition and running the network without the WTA mechanism [Querlioz et al., 2011]. Here, we propose a different approach to provide a fair comparison. We compare the membrane potentials V_j before the application of lateral inhibition and the neuronal thresholds used for synaptic competition. Then, we generate hypothetical spikes that do not affect the WTA mechanism, and plot performance curves denoted with (n) – as in this case up to n neurons can potentially spike. This substantially improves the results for lateral inhibition.

The results are summarized in Tab. 6.5. Synaptic competition consistently provides the highest feature detection accuracy, both in software and in hardware. Its benefits include simple computation of the competition results in comparison to dendritic inhibition, and inherent detection of multiple features in contrast to lateral inhibition.

Table 6.5: **Spiking accuracy and F-scores for the Bars dataset** Data from [Woźniak et al., 2017a].

	Accuracy	**F-score**
Lateral inh.	92.0%	0.611
Lateral inh. (n)	99.9%	0.997
Dendritic inh.	98.7%	0.954
Synaptic comp.	100%	1.0
Synaptic comp. (HW)	100%	1.0

6.5 Learning features from non-IID datasets

In this section, we discuss in detail the assumptions behind input presentation in the common datasets. We propose a more challenging version of the Bars dataset, which aims to resemble inputs observed in a more realistic online learning setting. The proposed dataset is then used to compare the performance of ICA-like feature learning using lateral inhibition, dendritic inhibition and synaptic competition.

6.5.1 Dataset presentation assumptions

In the classic formulation of the Bars dataset, the probability of an appearance of each bar is independent (I) from the appearance of the other bars in the current and in the previously observed patterns. Furthermore, the probability remains constant, so the patterns observed by the learning system are identically-distributed (ID). The dataset presentation therefore satisfies an independent and identically-distributed (IID) assumption. This assumption is ubiquitous in the machine learning community, although it is rarely stated in an explicit form. It follows from the usual training setup, in which the dataset is fixed and the patterns are drawn from it randomly, which is critical for the correct training of ANNs learned using backpropagation.

However, the IID assumption might not hold in a setting in which a learning system learns online through the observation of the environment. For instance, varying the speed of a robot traveling through exactly the same environment may lead to a different series of observations, as illustrated in Fig. 6.21. Firstly, in a slowly moving robot, the same or significantly overlapping viewpoint might be consecutively captured multiple times. Secondly, in a robot traveling at a variable speed, the change of speed impacts the distribution of the observations.

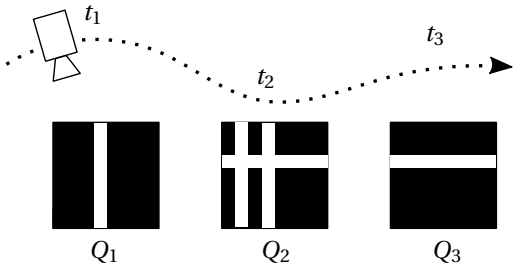


Figure 6.21: **More realistic dataset presentation** A robot may move at various speeds in the same environment, with (t_1, t_2, t_3) being respectively equal to $(1,2,3) \times \Delta T$ as well as $(1,3,6) \times \Delta T$. The first case corresponds to a typical sequential cycling through a dataset: (Q_1, Q_2, Q_3). However, in the second case, the observed inputs might be: $(Q_1, Q_1, Q_2, Q_2, Q_2, Q_3)$. Such change of the training input violates the common IID assumption.

We propose a more challenging version of the Bars dataset that violates the IID assumption. In particular, we introduce dependence of the input patterns by repeating consecutively each pattern r times, as illustrated in Fig. 6.22. In terms of the robot example, this corresponds to

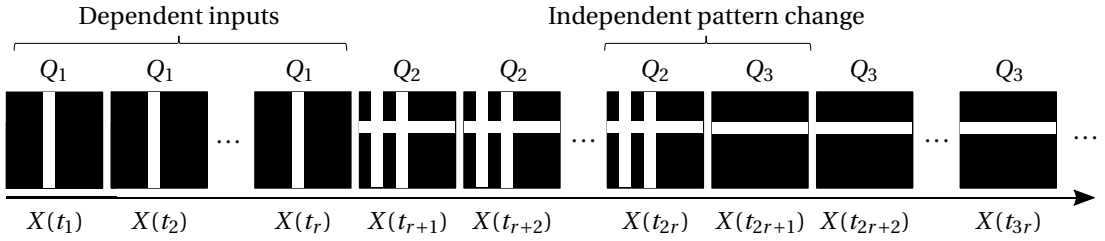


Figure 6.22: **Non-IID Bars dataset** The patterns repeat r times, which introduces dependence between the inputs and violates the IID dataset presentation assumption. The patterns remain independent during a pattern change after r input presentations.

moving slowly at a constant speed through the environment, and the time period required to "travel" from pattern Q_i to another pattern Q_j is $r \times \Delta T$. Therefore, inputs X become dependent for each time period $[t_{nr+1}, t_{nr+r}]$, $n \in \mathbb{N}$, which may be formulated as:

$$\Pr(X(t_{nr+1}) = \boldsymbol{x}^{Q_i}, X(t_{nr+2}) = \boldsymbol{x}^{Q_i}, \ldots, X(t_{nr+r}) = \boldsymbol{x}^{Q_i}) = 1 \tag{6.3}$$

After r input presentations, the patterns change in a statistically-independent manner:

$$\Pr(X(t_{nr}) = \boldsymbol{x}^{Q_i}, X(t_{nr+1}) = \boldsymbol{x}^{Q_j}) = \Pr(X(t_{nr}) = \boldsymbol{x}^{Q_i}) \Pr(X(t_{nr+1}) = \boldsymbol{x}^{Q_j}) \tag{6.4}$$

Because the bars within the patterns are independent, and patterns change in an independent manner after r presentations, we still refer to learning from this dataset as extracting independent features.

6.5.2 Non-IID independent feature learning results

We execute a benchmark for the non-IID Bars dataset with $r = 10$ using the same setup as for the classic Bars dataset in the previous section. We increase the number of input pattern presentations by r, so as to provide the same number of different patterns into the system as in the classic Bars dataset with 1000 input pattern presentations. All algorithms are executed in software only. Assessment of the learned representation is presented in Fig. 6.23. As illustrated in Fig. 6.23a, only synaptic competition finds all the bars. Other methods find more than half of the bars at any point in time, but the representation fluctuates – the neurons constantly change their learned representations. Similarly, in Fig. 6.23b, the MSE for these methods initially drops and then fluctuates. The snapshots of the final weights after 10000 input presentation are illustrated in Fig. 6.24. Lateral inhibition tends to completely forget some of the bars, and learn duplicate ones, as in Fig. 6.24a. Dendritic inhibition tends to maintain the shadows of the correct bars together with some other bars, as in Fig. 6.24b. Synaptic competition accurately captures all the bars in Fig. 6.24c.

Accuracy and F-score are plotted in Fig. 6.25. In comparison to the results for the classic Bars dataset in Fig. 6.20, where dendritic inhibition performed better than lateral inhibition, for the

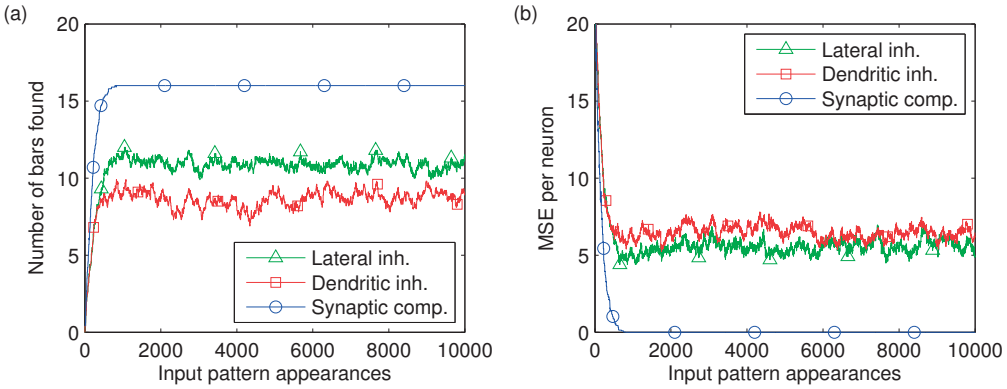


Figure 6.23: **Assessment of the features learned from the non-IID Bars dataset** (a) Number of bars found – only synaptic competition finds all the bars. (b) MSE of the learned representation initially decreases, but then fluctuates for lateral and dendritic inhibition.

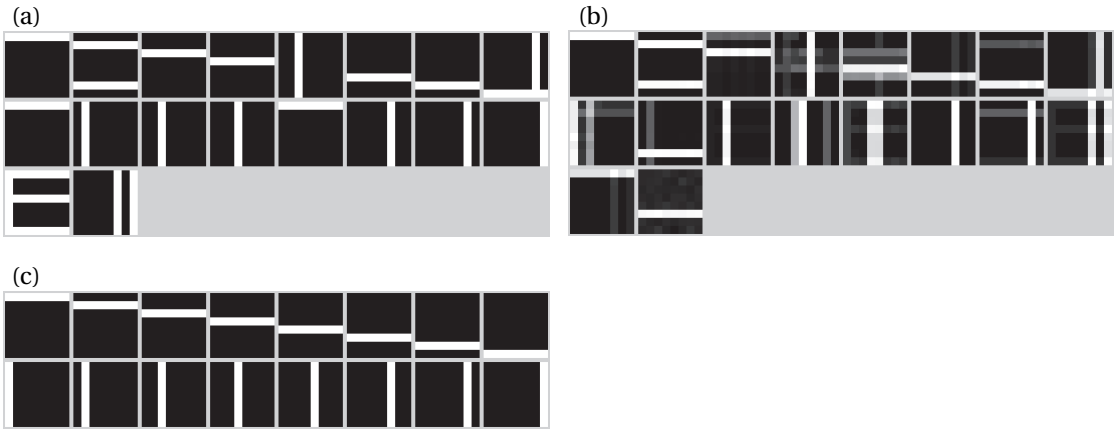


Figure 6.24: **Features learned from the non-IID Bars dataset** (a) lateral inhibition, (b) dendritic inhibition, (c) synaptic competition.

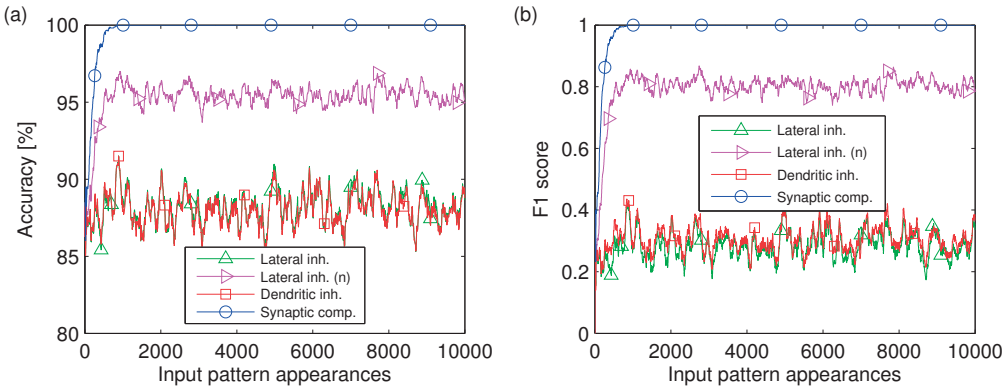


Figure 6.25: **Feature detection for the non-IID Bars dataset** Assessment of the activity of the networks: (a) accuracy, (b) F-score.

repeating Bars both methods achieve similar results. Again, synaptic competition performs better than the other methods. The results are summarized in Tab. 6.6.

Table 6.6: **Spiking accuracy and F-scores for the non-IID Bars dataset**

	Accuracy	**F-score**
Lateral inh.	88.7%	0.335
Lateral inh. (n)	95.2%	0.793
Dendritic inh.	88.8%	0.366
Synaptic comp.	100%	1.0

6.5.3 Discussion on the stability of representation

The reason why synaptic competition succeeds to learn all the non-IID bars where other methods fail can be explained by the difference in the operation of STDP. In synaptic competition, potentiation is applied only to a subset of particular neuron's synapses – winning synapses, illustrated in Fig. 6.10 (page 91). When the network correctly learns particular bars, reappearance of these bars becomes idempotent: no further weight increases are performed, as the learned neurons have weights at their maximum, and no other synapses are potentiated. In consequence, after discovering the correct features, weights obtained from synaptic competition are stable, which is not the case for the other methods.

In lateral inhibition and in dendritic inhibition potentiation is performed for the entire pattern. If a correctly learned network receives a series of repeating patterns with multiple bars, all neurons corresponding to that pattern will tend to lose their specificity and learn the entire pattern. This process is gradual, because of the incremental steps of LTP equal to $\alpha_+ = 0.1$. However, using the Bars dataset with r repeats, where $r\alpha_+$ is large enough to potentiate weights to high values, e.g. $r\alpha_+ = 10 \times 0.1 = 1$, leads to significant learning artifacts. In the classic Bars dataset, synapses nonspecific to a component learned by a particular neuron are rarely potentiated multiple times in a row. Each time the specific component appears at the input, the remaining components are changing, which leads to depression of any previously potentiated nonspecific synapses. Therefore, lateral inhibition and dendritic inhibition rely on the "canceling-out" effects arising from the IID dataset presentation assumption to maintain a stable representation.

6.6 Conclusions

In this chapter we addressed the challenge of improving internal representation of SNNs to learn features rather than patterns. We identified the importance of intraneuronal and interneuronal feedback for shaping what is learned by the network. First, we analytically demonstrated that neurons have propensity to execute $X \cap Y$ or $X \setminus Y$ operation, depending

from the learning rule implementation. Then, we demonstrated that performing these two operations in different neurons at different point in time leads to development of feature-based representation in a neural network. Based on this insight, we developed in a bottom-up manner an architecture that uses 1-bit of additional feedback to the learning rule. We demonstrated its feature learning capabilities for PCA-like and ICA-like features in simulations and experiments.

Next, we also proposed a different architecture that takes inspiration from a biologically-feasible mechanism of synaptic competition. We proposed to simplify its dynamics to a synaptic competition WTA scheme, similarly to the simplification of lateral inhibition to a neuronal competition WTA scheme. We introduced a dynamically-sized network architecture that adjusts its size based on the results of synaptic competition. We demonstrated feature learning capabilities of this architecture in simulations and experiments, and we compared it with other feature learning methods. Synaptic competition quickly learned both PCA-like and ICA-like features, while maintaining optimal network size.

Lastly, we considered a more realistic dataset presentation scheme, in which inputs violate the common IID assumption. We executed a benchmark for non-IID version of the Bars dataset. Synaptic competition successfully learned all the features, in contrast to other approaches that implicitly rely on the IID assumption.

The proposed architectures advance the understanding of the functional aspects of SNNs, and simultaneously provide means of implementing feature learning capabilities in phase-change-based neuromorphic systems.

7 Conclusions and future work

The research objective of this thesis was to advance the design of mixed analog-digital neuromorphic architectures capable of online unsupervised learning. The importance of this endeavor was discussed in the context of hardware design, machine learning and computational neuroscience. Neuromorphic designs arise from the confluence of these fields, and may bring qualitative advancements in the cognitive capabilities of computing systems and simultaneously improve our understanding of the brain.

To execute the research objective, we built on top of the logical model of spiking neural networks. We analyzed the common components of an SNN and proposed to implement them using the physical properties of phase-change memristors. Based on the in-depth understanding of the phase-change dynamics and the characterization results obtained from a prototype chip, we demonstrated basic building blocks for phase-change-based neuromorphic systems. We used the conductance of a single phase-change device to realize a 1-PCM design of a synapse, and proposed an A-STDP learning rule tailored for the phase-change characteristics. We also extended a phase-change neuron soma with an integration threshold to obtain a noise-robust phase-change neuron. In a series of simulations and experiments using a prototype phase-change chip, we demonstrated the operation of online unsupervised learning in phase-change-based SNNs. Firstly, we used a neuron with 1-PCM synapses to learn and re-learn patterns of correlated activity at the inputs. Next, we demonstrated an all-phase-change experimental realization of a neuron capable of detecting patterns of weakly correlated activity with pair-wise input cross-correlation coefficient as low as $c = 0.2$. To provide accurate weakly correlated pattern visualization for 1-PCM synapses with A-STDP, we proposed the selective depression mechanism, and we obtained in the experiments over 60 percentage points accuracy improvement over the regular A-STDP. With 1M phase-change synapses, our prototype system is the largest phase-change experimental realization of a spiking neuron to date in terms of the number of synapses. Lastly, we proposed an all-phase-change neural network capable of learning multiple patterns, and we introduced a power-efficient WTA scheme with level-tuned neurons as an alternative to the lateral inhibition WTA scheme. To enhance the quality of the patterns learned using A-STDP, we proposed to apply WTA scheme

also to the learning mechanism. We demonstrated its operation in a multiple pattern learning experiment.

Furthermore, we analyzed how to advance the learning capabilities of SNNs. We identified the need to improve the internal knowledge representation by learning features rather than patterns. We discussed the advantages of feature-based representation and introduced different kinds of features: orthogonal PCA-like features and ICA-like independent components. Then, we developed a more rigorous understanding of feature learning using a mathematical formulation in form of matrix factorization. We discussed how optimization constraints impact the type of the features, and we provided intuition on feature learning in neural networks by analyzing how such constraints are reflected in the ANNs' design. In the context of SNNs, we identified the key role of feedback links for shaping the learning process. Firstly, we provided an analytic interpretation of the intraneuronal feedback stemming from the operation of STDP rules. Secondly, we proposed a new form of interneuronal feedback to the learning rule. Through enhancements of A-STDP, we incorporated feature-extraction functionality into a multi-neuron configuration. We demonstrated the capability to extract PCA- and ICA-like features, depending from constraints introduced into the algorithm. Thirdly, we proposed an SNN architecture inspired by the synaptic competition for plasticity-related proteins. In this architecture, the dynamics of synaptic competition was simplified to a synaptic WTA scheme, similarly to the simplification of lateral inhibition to a neuronal WTA scheme. The results of synaptic competition were used to guide the learning as well as to provide means for detecting novelty, which was used to dynamically adjust the size of the network. In comparison with lateral inhibition and dendritic inhibition, synaptic competition achieved better performance on the Bars and the Swimmer dataset. Lastly, for a more challenging non-IID version of the Bars dataset, only synaptic competition successfully learned the proper features. The proposed model of synaptic competition provides an interesting alternative to the commonly used neuronal competition. Finally, it opens up new opportunities for further exploration of the involvement of synapses in the learning.

Throughout the thesis we focused on biologically-inspired learning mechanisms that use local information for the learning. In consequence, the proposed architectures are suited for compact hardware implementation, and experimental results using an array of phase-change-based synapses confirmed their functionality. Simultaneously, the learning was executed in an online unsupervised manner. Advancing unsupervised learning is critical for making use of the vast amounts of data around us, most of which is unlabeled. The proposed methodology of explicit analysis of the relationship between feedback and representation constrains, along with the experimental realization, constitute an important step towards incorporating unsupervised feature-extraction functionality in spiking neural networks.

7.1 Future work

The contributions of the thesis advanced the state-of-the-art in the phase-change-based neuromorphic systems and in the learning architectures for SNNs. Still, many open questions need to be addressed before neuromorphic systems are deployed for practical applications. In particular, this thesis provides ground for future work on the following topics:

Exploring other feature types

With the Swimmer and the Bars datasets we demonstrated that different types of features can be extracted from the data, depending on the assumptions in the algorithm. They may either take a form of explicit optimization constraints or particular design choices, such as the introduction of feedback to the learning or different forms of feedback in synaptic competition determined by the definition of v_{ji}. Moreover, these assumptions can be conflicting, as in the case of PCA-like and ICA-like features, so that the algorithms performing well on one dataset do not perform well on another, as observed in Sec. 6.4.4. In consequence, generalization of feature extraction to more complicated applications with diverse datasets becomes challenging. It poses an open question on the advantages of a particular type of features or, equivalently, on what the set of the correct assumptions for a particular task is. One straightforward approach is to continue exploring different types of features, including a spectrum of "soft" assumptions. Such assumptions may be formulated by defining the synaptic competition potential as: $v_{ji} = \gamma v_{ji}^{A} + (1-\gamma) v_{ji}^{AB}$, where γ is a weighting factor.

Multi-layer unsupervised learning

In Sec. 5.1 we discussed that using multiple layers of neurons learning feature-based representations is critical for improving the accuracy of SNN-based neuromorphic systems. In this thesis, we analyzed in depth the operation of single-layered neural networks, and addressed the challenge of feature learning in single-layer SNNs. In the future work, we envision to construct unsupervised multi-layer feature-learning SNNs. A possible approach is to explore the concept of stacking unsupervised SNN layers on top of each other, similarly to the concept of ANN stacking proposed in Deep Belief Networks (DBNs) [Hinton and Salakhutdinov, 2006].

Developing a complete neuromorphic system

In our research, we used an FPGA-based prototype hardware platform, hosting the phase-change chip. Through combination of hardware and software elements, it enabled to perform the characterization of the nanodevices and to realize the functionality of the proposed neuromorphic architectures. Nevertheless, it provided limited opportunity to benefit from the area- and power-efficient structure of neuromorphic systems, because proposed architectures were mapped to the constraints of the platform. Therefore, developing an integrated hardware

solution with a complete neuromorphic system would provide the final validation of its capabilities and enable comprehensive system-level power consumption assessment. However, owing to limited possibility to frequently change the hardware design, this step should be executed only after the requirements of a multi-layer architecture become clear.

Bibliography

[Adrian and Zotterman, 1926] Adrian, E. D. and Zotterman, Y. (1926). The impulses produced by sensory nerve-endings. *The Journal of Physiology*, 61(2):151–171.

[Akopyan et al., 2015] Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., Taba, B., Beakes, M., Brezzo, B., Kuang, J. B., Manohar, R., Risk, W. P., Jackson, B., and Modha, D. S. (2015). 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.

[Almási et al., 2016] Almási, A.-D., Woźniak, S., Cristea, V., Leblebici, Y., and Engbersen, T. (2016). Review of advances in neural networks: Neural design technology stack. *Neurocomputing*, 174 A:31–41.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

[Benjamin et al., 2014] Benjamin, B. V., Peiran Gao, McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J.-M., Alvarez-Icaza, R., Arthur, J. V., Merolla, P. A., and Boahen, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716.

[Bichler et al., 2012a] Bichler, O., Querlioz, D., Thorpe, S. J., Bourgoin, J.-P., and Gamrat, C. (2012a). Extraction of temporally correlated features from dynamic vision sensors with spike-timing-dependent plasticity. *Neural Networks*, 32:339–348.

[Bichler et al., 2012b] Bichler, O., Suri, M., Querlioz, D., Vuillaume, D., DeSalvo, B., and Gamrat, C. (2012b). Visual pattern extraction using energy-efficient "2-PCM synapse" neuromorphic architecture. *IEEE Transactions on Electron Devices*, 59(8):2206–2214.

[Bill and Legenstein, 2014] Bill, J. and Legenstein, R. (2014). A compound memristive synapse model for statistical learning through STDP in spiking neural networks. *Frontiers in Neuroscience*, 8.

[Borst and Theunissen, 1999] Borst, A. and Theunissen, F. E. (1999). Information theory and neural coding. *Nature Neuroscience*, 2(11).

[Bourlard, 2000] Bourlard, H. (2000). Auto-association by multilayer perceptrons and singular value decomposition. Technical Report EPFL-REPORT-82601, IDIAP.

[Boybat et al., 2017] Boybat, I., Le Gallo, M., Nandakumar, S. R., Moraitis, T., Tuma, T., Rajendran, B., Leblebici, Y., Sebastian, A., and Eleftheriou, E. (2017). An efficient synaptic architecture for artificial neural networks. In *2017 17th Non-Volatile Memory Technology Symposium (NVMTS)*. (Submitted), IEEE.

[Brader et al., 2007] Brader, J. M., Senn, W., and Fusi, S. (2007). Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Computation*, 19(11):2881–2912.

[Burr et al., 2014] Burr, G., Shelby, R., di Nolfo, C., Jang, J., Shenoy, R., Narayanan, P., Virwani, K., Giacometti, E., Kurdi, B., and Hwang, H. (2014). Experimental demonstration and tolerancing of a large-scale neural network (165,000 synapses), using phase-change memory as the synaptic weight element. In *2014 IEEE International Electron Devices Meeting (IEDM)*, pages 29.5.1–29.5.4. IEEE.

[Burr et al., 2010] Burr, G. W., Breitwisch, M. J., Franceschini, M., Garetto, D., Gopalakrishnan, K., Jackson, B., Kurdi, B., Lam, C., Lastras, L. A., Padilla, A., Rajendran, B., Raoux, S., and Shenoy, R. S. (2010). Phase change memory technology. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures*, 28(2):223.

[Burr et al., 2017] Burr, G. W., Shelby, R. M., Sebastian, A., Kim, S., Kim, S., Sidler, S., Virwani, K., Ishii, M., Narayanan, P., Fumarola, A., Sanches, L. L., Boybat, I., Le Gallo, M., Moon, K., Woo, J., Hwang, H., and Leblebici, Y. (2017). Neuromorphic computing using non-volatile memory. *Advances in Physics: X*, 2(1):89–124.

[Cireşan et al., 2010] Cireşan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Comput.*, 22(12):3207–3220.

[Close et al., 2010] Close, G., Frey, U., Breitwisch, M., Lung, H., Lam, C., Hagleitner, C., and Eleftheriou, E. (2010). Device, circuit and system-level analysis of noise in multi-bit phase-change memory. In *2010 IEEE International Electron Devices Meeting (IEDM)*, pages 29.5.1–29.5.4. IEEE.

[Dayan and Abbott, 2005] Dayan, P. and Abbott, L. F. (2005). *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. The MIT Press.

[Diehl and Cook, 2015] Diehl, P. U. and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9.

[Donoho and Stodden, 2004] Donoho, D. and Stodden, V. (2004). When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, pages 1141–1148. MIT Press.

[Eliasmith, 2013] Eliasmith, C. (2013). *How to Build a Brain: A Neural Architecture for Biological Cognition*. Oxford University Press.

[Eliasmith et al., 2012] Eliasmith, C., Stewart, T. C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., and Rasmussen, D. (2012). A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205.

[Eryilmaz et al., 2014] Eryilmaz, S. B., Kuzum, D., Jeyasingh, R., Kim, S., BrightSky, M., Lam, C., and Wong, H.-S. P. (2014). Brain-like associative learning using a nanoscale non-volatile phase change synaptic device array. *Frontiers in Neuroscience*, 8.

[Feng et al., 2002] Feng, T., Li, S. Z., Shum, H.-Y., and Zhang, H. (2002). Local non-negative matrix factorization as a visual representation. In *The 2nd International Conference on Development and Learning*, pages 178–183. IEEE.

[Ferreira, 2006] Ferreira, C. (2006). Designing neural networks using gene expression programming. *Applied Soft Computing Technologies: The Challenge of Complexity*, pages 517–535.

[Földiak, 1990] Földiak, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64(2):165–170.

[Furber et al., 2013] Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., and Brown, A. D. (2013). Overview of the SpiNNaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467.

[Gallo et al., 2017] Gallo, M. L., Sebastian, A., Mathis, R., Manica, M., Tuma, T., Bekas, C., Curioni, A., and Eleftheriou, E. (2017). Mixed-precision memcomputing. *CoRR*, abs/1701.04279.

[Garbin et al., 2015] Garbin, D., Vianello, E., Bichler, O., Azzaz, M., Rafhay, Q., Candelier, P., Gamrat, C., Ghibaudo, G., DeSalvo, B., and Perniola, L. (2015). On the impact of OxRAM-based synapses variability on convolutional neural networks performance. In *Proceedings of the 2015 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH 15)*, pages 193–198. IEEE.

[Gerstner and Kistler, 2002] Gerstner, W. and Kistler, W. M. (2002). *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press.

[Gerstner et al., 2014] Gerstner, W., Kistler, W. M., Naud, R., and Paninski, L. (2014). *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press.

[Gerstner and Naud, 2009] Gerstner, W. and Naud, R. (2009). How good are neuron models? *Science*, 326(5951):379–380.

[Gerstner et al., 2012] Gerstner, W., Sprekeler, H., and Deco, G. (2012). Theory and simulation in neuroscience. *Science*, 338(6103):60–65.

[Graves et al., 2013] Graves, A., Mohamed, A. R., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE.

[Graves et al., 2016] Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., Badia, A. P., Hermann, K. M., Zwols, Y., Ostrovski, G., Cain, A., King, H., Summerfield, C., Blunsom, P., Kavukcuoglu, K., and Hassabis, D. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*.

[Gütig et al., 2003] Gütig, R., Aharonov, R., Rotter, S., and Sompolinsky, H. (2003). Learning input correlations through nonlinear temporally asymmetric Hebbian plasticity. *The Journal of Neuroscience*, 23(9):3697–3714.

[Harris et al., 2012] Harris, J., Jolivet, R., and Attwell, D. (2012). Synaptic energy use and supply. *Neuron*, 75(5):762–777.

[Hinton, 1986] Hinton, G. E. (1986). Learning distributed representations of concepts. volume 1, page 12. Amherst, MA.

[Hinton and Salakhutdinov, 2006] Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

[Hodgkin and Huxley, 1952] Hodgkin, A. L. and Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500.

[Izhikevich, 2006] Izhikevich, E. M. (2006). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 17(10):2443–2452.

[Jonke et al., 2017] Jonke, Z., Legenstein, R., Habenschuss, S., and Maass, W. (2017). Feedback inhibition shapes emergent computational properties of cortical microcircuit motifs. *arXiv:1705.07614*.

[Jouppi et al., 2017] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., and others (2017). In-datacenter performance analysis of a tensor processing unit. *arXiv preprint arXiv:1704.04760*.

[Kim et al., 2015] Kim, Y., Zhang, Y., and Li, P. (2015). A reconfigurable digital neuromorphic processor with memristive synaptic crossbar for cognitive computing. *ACM Journal on Emerging Technologies in Computing Systems*, 11(4):1–25.

[Koelmans et al., 2015] Koelmans, W. W., Sebastian, A., Jonnalagadda, V. P., Krebs, D., Dellmann, L., and Eleftheriou, E. (2015). Projected phase-change memory devices. *Nature Communications*, 6:8181.

[Kuzum et al., 2012] Kuzum, D., Jeyasingh, R. G. D., Lee, B., and Wong, H.-S. P. (2012). Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing. *Nano Letters*, 12(5):2179–2186.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee and Seung, 1999] Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

[Lee and Seung, 2001] Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562.

[Lukoševičius and Jaeger, 2009] Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

[Maass, 2000] Maass, W. (2000). On the computational power of Winner-Take-All. *Neural Computation*, 12(11):2519–2535.

[Marder and Goaillard, 2006] Marder, E. and Goaillard, J.-M. (2006). Variability, compensation and homeostasis in neuron and network function. *Nature Reviews Neuroscience*, 7(7):563–574.

[Markram, 2012] Markram, H. (2012). The human brain project. *Scientific American*, 306(6):50–55.

[Markram et al., 1997] Markram, H., Lübke, J., Frotscher, M., and Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, 275(5297):213–215.

[Masquelier and Thorpe, 2010] Masquelier, T. and Thorpe, S. J. (2010). Learning to recognize objects using waves of spikes and spike timing-dependent plasticity. In *2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

[Mead, 1990] Mead, C. (1990). Neuromorphic electronic systems. *Proceedings of the IEEE*, 78(10):1629–1636.

[Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Modha et al., 2011] Modha, D. S., Ananthanarayanan, R., Esser, S. K., Ndirango, A., Sherbondy, A. J., and Singh, R. (2011). Cognitive computing. *Communications of the ACM*, 54(8):62.

[Moraitis et al., 2017] Moraitis, T., Sebastian, A., Boybat, I., Le Gallo, M., Tuma, T., and Eleftheriou, E. (2017). Fatiguing STDP: Learning from spike-timing codes in the presence of rate codes. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

[Neftci et al., 2013] Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., and Cauwenberghs, G. (2013). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in neuroscience*, 7.

[Nessler et al., 2009] Nessler, B., Pfeiffer, M., and Maass, W. (2009). STDP enables spiking neurons to detect hidden causes of their inputs. In *Advances in Neural Information Processing Systems*, pages 1357–1365.

[O'Connor et al., 2013] O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., and Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, 7.

[Pantazi et al., 2016] Pantazi, A., Woźniak, S., Tuma, T., and Eleftheriou, E. (2016). All-memristive neuromorphic computing with level-tuned neurons. *Nanotechnology*, 27(35):355205.

[Papandreou et al., 2011] Papandreou, N., Pozidis, H., Pantazi, A., Sebastian, A., Breitwisch, M., Lam, C., and Eleftheriou, E. (2011). Programming algorithms for multilevel phase-change memory. In *2011 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 329–332. IEEE.

[Parisien et al., 2008] Parisien, C., Anderson, C. H., and Eliasmith, C. (2008). Solving the problem of negative synaptic weights in cortical models. *Neural computation*, 20(6):1473–1494.

[Qiao et al., 2015] Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., and Indiveri, G. (2015). A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, 9.

[Querlioz et al., 2011] Querlioz, D., Bichler, O., and Gamrat, C. (2011). Simulation of a memristor-based spiking neural network immune to device variations. In *2011 International Joint Conference on Neural Networks (IJCNN)*, pages 1775–1781. IEEE.

[Rumelhart et al., 1985] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

[Sajikumar et al., 2014] Sajikumar, S., Morris, R. G. M., and Korte, M. (2014). Competition between recently potentiated synaptic inputs reveals a winner-take-all phase of synaptic tagging and capture. *Proceedings of the National Academy of Sciences*, 111(33):12217–12221.

[Schemmel et al., 2010] Schemmel, J., Bruderle, D., Grubl, A., Hock, M., Meier, K., and Millner, S. (2010). A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *2010 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1947–1950. IEEE.

[Sebastian et al., 2014] Sebastian, A., Le Gallo, M., and Krebs, D. (2014). Crystal growth within a phase change memory cell. *Nature Communications*, 5.

[Serrano-Gotarredona et al., 2013] Serrano-Gotarredona, T., Masquelier, T., Prodromakis, T., Indiveri, G., and Linares-Barranco, B. (2013). STDP and STDP variations with memristors for spiking neuromorphic learning systems. *Frontiers in Neuroscience*, 7.

[Shimada and Torikai, 2015] Shimada, N. and Torikai, H. (2015). A novel asynchronous cellular automaton multicompartment neuron model. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(8):776–780.

[Sidler et al., 2017] Sidler, S., Pantazi, A., Woźniak, S., Leblebici, Y., and Eleftheriou, E. (2017). Unsupervised learning using phase-change synapses and complementary patterns. In *2017 ENNS International Conference on Artificial Neural Networks (ICANN)*. (Accepted).

[Sjöström and Häusser, 2006] Sjöström, P. J. and Häusser, M. (2006). A cooperative switch determines the sign of synaptic plasticity in distal dendrites of neocortical pyramidal neurons. *Neuron*, 51(2):227–238.

[Sjöström et al., 2008] Sjöström, P. J., Rancz, E. A., Roth, A., and Häusser, M. (2008). Dendritic excitability and synaptic plasticity. *Physiological Reviews*, 88(2):769–840.

[Song et al., 2000] Song, S., Miller, K. D., and Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nature Neuroscience*, 3(9):919–926.

[Spratling, 2006] Spratling, M. W. (2006). Learning image components for object recognition. *Journal of Machine Learning Research*, 7(May):793–815.

[Stuart et al., 2016] Stuart, G., Spruston, N., and Häusser, M. (2016). *Dendrites*. Oxford University Press.

[Suri et al., 2011] Suri, M., Bichler, O., Querlioz, D., Cueto, O., Perniola, L., Sousa, V., Vuillaume, D., Gamrat, C., and DeSalvo, B. (2011). Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction. In *2011 IEEE International Electron Devices Meeting (IEDM)*, pages 4–4. IEEE.

[Tuma et al., 2016a] Tuma, T., Le Gallo, M., Sebastian, A., and Eleftheriou, E. (2016a). Detecting correlations using phase-change neurons and synapses. *IEEE Electron Device Letters*, pages 1–1.

[Tuma et al., 2016b] Tuma, T., Pantazi, A., Le Gallo, M., Sebastian, A., and Eleftheriou, E. (2016b). Stochastic phase-change neurons. *Nature Nanotechnology*, 11(8):693–699.

[Van Rullen et al., 1998] Van Rullen, R., Gautrais, J., Delorme, A., and Thorpe, S. (1998). Face processing using one spike per neurone. *Biosystems*, 48(1):229–239.

[Vincent et al., 2010] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

[Vinyals et al., 2015] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3156–3164.

[Von Neumann, 1958] Von Neumann, J. (1958). *The Computer and the Brain*. Mistress Hepsa Ely Silliman Memorial Lectures: Mistress. Yale University Press.

[Wang et al., 2016] Wang, Q., Kim, Y., and Li, P. (2016). Neuromorphic processors with memristive synapses: Synaptic interface and architectural exploration. *ACM Journal on Emerging Technologies in Computing Systems*, 12(4):1–22.

[Weston et al., 2014] Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *ArXiv e-prints*.

[Woźniak et al., 2015] Woźniak, S., Almási, A.-D., Cristea, V., Leblebici, Y., and Engbersen, T. (2015). Review of advances in neural networks: Neural design technology stack. In *Proceedings of ELM-2014 Volume 1*, pages 367–376. Springer.

[Woźniak et al., 2017a] Woźniak, S., Pantazi, A., Leblebici, Y., and Eleftheriou, E. (2017a). Feature learning using synaptic competition in a dynamically-sized neuromorphic architecture. In *International Conference on Rebooting Computing (ICRC)*. (Accepted), IEEE.

[Woźniak et al., 2017b] Woźniak, S., Pantazi, A., Leblebici, Y., and Eleftheriou, E. (2017b). Neuromorphic system with phase-change synapses for pattern learning and feature extraction. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

[Woźniak et al., 2017c] Woźniak, S., Pantazi, A., Sidler, S., Papandreou, N., Leblebici, Y., and Eleftheriou, E. (2017c). Neuromorphic architecture with 1M memristive synapses for detection of weakly correlated inputs. *IEEE Transactions on Circuits and Systems II: Express Briefs*, pages 1–1.

[Woźniak et al., 2016] Woźniak, S., Tuma, T., Pantazi, A., and Eleftheriou, E. (2016). Learning spatio-temporal patterns in the presence of input noise using phase-change memristors. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 365–368. IEEE.

[Wu et al., 2015] Wu, X., Saxena, V., Zhu, K., and Balagopal, S. (2015). A CMOS spiking neuron for brain-inspired neural networks with resistive synapses and in situ learning. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(11):1088–1092.

✉ *Stanislaw Wozniak*
c/o K.Kryszczuk
Badstrasse 5
CH-8134 Adliswil
Switzerland

☎ *+48 600 955 339*
www.swoz.org/cv
swoz@onet.eu

Stanisław Woźniak

Curriculum Vitae

General Information

Born: 22 May 1986 Marital status: Married Nationality: Polish Swiss permit type: B

Education

Nov 2012 – Dec 2017	**PhD**, *EPFL (École Polytechnique Fédérale de Lausanne)* Dissertation title: *Unsupervised Learning of Phase-Change-Based Neuromorphic Systems*	Lausanne, CH
Feb–Aug 2015	**Visiting student**, *ETHZ (Eidgenössische Technische Hochschule Zürich)*	Zürich, CH
2009 – 2010	**MSc in Computer Science**, *PUT (Poznan University of Technology)*	Poznań, PL
2008 – 2009	**Exchange student**, *Universität Stuttgart*	Stuttgart, DE
2005 – 2008	**BSc in Computer Science**, *PUT (Poznan University of Technology)*	Poznań, PL

Professional Experience

Nov 2012 – Oct 2017	**Predoctoral Researcher**, *IBM Research – Zurich* Neuromorphic Computing & Services Research	Rüschlikon, CH
Jul–Oct 2012	**Windows Phone SDET Intern**, *Microsoft*	Redmond, WA, US
Mar–Jun 2012	**IBM "Great Minds" Intern**, *IBM Research – Zurich*	Rüschlikon, CH
Jan–Jun 2011	**Virtualization Consultant**, *PUT (Poznan University of Technology)*	Poznań, PL
Sep 2010 – Feb 2011	**Researcher**, *WUT (Warsaw University of Technology)* SYNAT project: Social network platform for Polish scientists, Graph OLAP	Warszawa, PL
Nov 2009 – Aug 2010	**Project Manager and Java Developer**, *Volkswagen* MSc project: Logistics budget simulation tool for VW Transporter & VW Caddy	Poznań, PL
Feb–Apr 2010	**C# ASP.NET Developer**, *Alliance Technology*	Poznań, PL
Jun–Aug 2008	**co-Architect, MSSQL ETL Developer**, *Eskulap* BSc project: Business Intelligence module for Hospital Information System	Poznań, PL
2007 – 2008	**Start-up co-Founder (seed stage MVP)**, *Amnis Media* Steganography-based DRM for e-publishing	Poznań, PL

Notable Awards & Accomplishments

2016	IBM A-Level Accomplishement: Service Delivery Analytics with GTS DataHub
2015	IBM Outstanding Accomplishement: Dynamic Automation for GTS
2009	Volkswagen Best Student Scholarship
2008	Polish Ministry of Science & Higher Education Academic Achievement Scholarship
2005	Polish Prime Minister Prof. Marek Belka Achievement Scholarship

Skills

Languages	Polish (native), English (fluent), German (advanced), French (basic)
Coding	C#, C++, C, CUDA, Java, PHP, XHTML, XPath, SQL, Python, Matlab

Publications

[1] S. Woźniak, A. Pantazi, Y. Leblebici, and E. Eleftheriou, "Feature learning using synaptic competition in a dynamically-sized neuromorphic architecture," in *International Conference on Rebooting Computing (ICRC)*. (Accepted), IEEE, 2017.

[2] S. Sidler, A. Pantazi, S. Woźniak, Y. Leblebici, and E. Eleftheriou, "Unsupervised learning using phase-change synapses and complementary patterns," in *2017 ENNS International Conference on Artificial Neural Networks (ICANN)*, 2017.

[3] S. Woźniak, A. Pantazi, Y. Leblebici, and E. Eleftheriou, "Neuromorphic system with phase-change synapses for pattern learning and feature extraction," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017.

[4] S. Woźniak, A. Pantazi, S. Sidler, N. Papandreou, Y. Leblebici, and E. Eleftheriou, "Neuromorphic architecture with 1M memristive synapses for detection of weakly correlated inputs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2017.

[5] A. Pantazi, S. Woźniak, T. Tuma, and E. Eleftheriou, "All-memristive neuromorphic computing with level-tuned neurons," *Nanotechnology*, vol. 27, no. 35, p. 355205, 2016.

[6] S. Woźniak, T. Tuma, A. Pantazi, and E. Eleftheriou, "Learning spatio-temporal patterns in the presence of input noise using phase-change memristors," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 365–368.

[7] A.-D. Almási, S. Woźniak, V. Cristea, Y. Leblebici, and T. Engbersen, "Review of advances in neural networks: Neural design technology stack," *Neurocomputing*, vol. 174 A, pp. 31–41, 2016.

[8] S. Woźniak, A.-D. Almási, V. Cristea, Y. Leblebici, and T. Engbersen, "Review of advances in neural networks: Neural design technology stack," in *Proceedings of ELM-2014 Volume 1*. Springer, 2015, pp. 367–376.

[9] T. Morzy and S. Woźniak, "Prototype OLAP processing system for multidimensional analysis of graph of the network of interconnections between scientists," SYNAT Project, Tech. Rep., 2012.

[10] ——, "Methods of networked structures analysis and application in bibliographic and scientific social networks," in *Methods of Artificial Intelligence in the Processes of Data Acquisition and Preprocessing, and for Analyzing Network Structures and their Dynamics*. SYNAT Project, 2011.

[11] ——, "The analysis of the possibility of using data mining and exploration of the Internet with the use of social technologies to create multicriteria evaluation environment of Polish scientists and their dissertations," SYNAT Project, Tech. Rep., 2010.

Patent Applications

[1] S. Wozniak and A. Pantazi, "Neuromorphic architecture for feature learning using a spiking neural network," U.S. Patent App. 15/725 320, Oct. 5, 2017.

[2] S. Sidler, S. Wozniak, and A. Pantazi, "Neuromorphic architecture for unsupervised classification tasks using information from complementary patterns," U.S. Patent App. 15/680 140, Aug. 17, 2017.

[3] S. Wozniak and A. Pantazi, "Neuromorphic architecture for unsupervised pattern detection and feature learning," U.S. Patent App. 15/264 081, Sep. 13, 2016.

[4] A. Pantazi, S. Wozniak, and T. Tuma, "Neuromorphic architecture with multiple coupled neurons using internal state neuron information," U.S. Patent App. 15/189 449, Jun. 22, 2016.