Precise timing and computationally efficient learning in neuromorphic systems

Omar Oubari

THÈSE DE DOCTORAT
DE SORBONNE UNIVERSITÉ

Spécialité : Ingénierie Neuromorphique

École doctorale nº391: Sciences mécaniques, acoustique, électronique et robotique

réalisée

à l'Institut de la Vision - Équipe vision et calcul naturel

sous la direction de Ryad B. Benosman et Sio-Hoi Ieng

présentée par

Omar Oubari

pour obtenir le grade de :

DOCTEUR DE SORBONNE UNIVERSITÉ

Sujet de la thèse :

Precise timing and computationally efficient learning in neuromorphic systems

soutenue le 14 décembre 2020

devant le jury composé de :

Pr.	Alejandro Linares Barranco	Rapporteur
Pr.	Emre Neftci	Rapporteur
Dr.	Laurent Perrinet	Examinateur
Dr.	Sylvain Argentieri	Examinateur
Pr.	Ryad B. Benosman	Directeur de thèse
Dr.	Sio-Hoi Ieng	Co-Directeur de thèse

Precise timing and computationally efficient learning in neuromorphic systems

Abstract: From image recognition to automated driving, machine learning nowadays is all around us and impacts various aspects of our daily lives. This disruptive technology is rapidly evolving at a huge cost in terms of energy consumption. Machine learning models are usually trained on powerful GPUs, limiting the potential for edge computing. Processing data locally instead of relying on cloud computing brings about improvements in speed and latency, which are essential for real-time applications. The field of *neuromorphic engineering* tries to solve the energy bottleneck problem through bio-inspired hardware and computation techniques. In particular, neuromorphic vision sensors feature independent pixels that asynchronously generate millions of events per second with high temporal precision, depending on the dynamics of a visual scene. The goal of this thesis is to take advantage of precise timing on neuromorphic architectures in order to develop computationally-efficient learning algorithms. We approach the issue through two different perspectives: spiking neural networks and probabilistic models. We introduce a delay-learning rule for spiking neural networks that relies on highly redundant sparse connectivity. We also develop bio-inspired learning techniques on a dedicated hardware with ultra-low power requirements and latency. The system implements synaptic plasticity using a memristive crossbar array to learn from the output of event-based vision sensors in the context of autonomous driving. When working with very large streams of events, we introduce two clustering techniques based on Gaussian mixture models that set a new state of the art in terms of computational efficiency.

Keywords: Neuromorphic engineering, event-based processing, energy efficiency, spiking neural networks, probabilistic models

Apprentissage dans des systèmes neuromorphiques: précision temporelle et efficacité calculatoire

Résumé : De la reconnaissance d'image à la conduite autonome, l'apprentissage machine est omniprésent dans notre vie quotidienne. Cette technologie de rupture évolue rapidement mais consomme énormément d'énergie. Les modèles d'apprentissage actuels utilisent généralement des GPU puissants qui ne permettent pas de traiter des données localement, alors que ceci améliore la vitesse et la latence indispensables aux applications en temps réel. De ce fait, le domaine de *l'ingénierie neuromorphique* tente de résoudre ce problème et de baisser le budget énergétique grâce à l'introduction des systèmes et des techniques d'apprentissage bio-inspirés comme notamment les capteurs de vision événementiels. Ces derniers comportent des pixels indépendants qui génèrent de manière asynchrone des millions d'événements par seconde avec une grande précision temporelle, en fonction de la dynamique d'une scène visuelle. Le but de cette thèse est de tirer profit de la précision temporelle des architectures neuromorphiques afin de développer des algorithmes d'apprentissage efficaces. Nous abordons la problématique selon deux approches différentes : celle des réseaux de neurones impulsionnels et celle des modèles probabilistes. D'abord, nous présentons un système d'apprentissage à délai pour les réseaux de neurones impulsionnels qui repose sur une connectivité éparse et hautement redondante. Ensuite, nous proposons des techniques d'apprentissage bio-inspirées sur du matériel dédié à basse latence et consommation énergétique. Le système implémente la plasticité synaptique à l'aide d'un réseau de memristors, pour l'apprentissage à partir de capteurs de vision événementiels dans le contexte de la conduite autonome. Enfin, nous introduisons deux techniques de partitionnement basées sur des modèles de mélange de gaussiennes qui établissent un nouvel état de l'art en termes d'efficacité de calcul.

Mots clés : Ingénierie neuromorphique, traitement des signaux événementiels, efficacité énergétique, réseau de neurones impulsionnels, modèles probabilistes

Acknowledgements

I would first like to thank my supervisor Dr. Ryad Benosman for giving me the time and opportunity to enter a new field of research, for giving me the freedom to explore my own ideas, and also for providing me with a nice environment to work in.

I would also like to thank my co-supervisor Dr. Sio-Hoi Ieng for his valuable guidance and advice throughout my thesis, and for proving me with the exciting opportunity to meet and work with various collaborators on a trans-disciplinary project.

I am grateful to Dr. Georgios Exarchakis for sharing some of his ideas with me, and for his patience and support in some of the more mathematically advanced concepts of this thesis.

I am also grateful to Sylvain Saighi and the rest of the team at IMS Bordeaux for all the fruitful discussions about integrated circuit design.

I also wish to thank Gregor Lenz for being a true brother and keeping me sane during these past few years; Alexandre Marcireau for making me into a better programmer; and the rest of my teammates for all the adventures.

I thank *El bira 7aram*, "bas ntebho mish 3am na3mol di3ayat". I also thank *The Usual* for all the unforgettable memories we've shared these past few years.

Finally, I would like to thank my sister Rewa for being a role model in perseverance and determination and for always believing in me; my parents, Bassem and Rana, for their unconditional love and support; and the rest of my family for simply being the best.

Publications

[1] G. Haessig, M. B. Milde, P. V. Aceituno, O. Oubari, J. C. Knight, A. van Schaik, R. B. Benosman, and G. Indiveri, "Event-Based Computation for Touch Localization Based on Precise Spike Timing," *Frontiers in Neuroscience*, vol. 14, p. 420, May 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00420/full

[2] P.-H. Prévot, K. Gehere, F. Arcizet, H. Akolkar, M. A. Khoei, K. Blaize, O. Oubari, P. Daye, M. Lanoë, M. Valet, S. Dalouz, P. Langlois, E. Esposito, V. Forster, E. Dubus, N. Wattiez, E. Brazhnikova, C. Nouvel-Jaillard, Y. LeMer, J. Demilly, C.-M. Fovet, P. Hantraye, M. Weissenburger, H. Lorach, E. Bouillet, M. Deterre, R. Hornig, G. Buc, J.-A. Sahel, G. Chenegros, P. Pouget, R. Benosman, and S. Picaud, "Behavioural responses to a photovoltaic subretinal prosthesis implanted in non-human primates," *Nature Biomedical Engineering*, vol. 4, no. 2, pp. 172–180, Feb. 2020. [Online]. Available: http://www.nature.com/articles/s41551-019-0484-2 xii, 9, 16, 18

[3] O. Oubari, G. Exarchakis, G. Lenz, R. Benosman, and S.-H. Ieng, "Computationally efficient learning on very large event-based datasets," *In preparation.*

[4] G. Exarchakis, O. Oubari, G. Lenz, R. Benosman, and S.-H. Ieng, "Efficient clustering by sampling-based training of a gaussian mixture model," *In preparation.*

[5] G. Lenz, O. Oubari, G. Orchard, S.-H. Ieng, and R. Benosman, "Neural computation using precise timing on loihi," *In preparation.*

Open-source software

- **Hummus**: Event-based spiking neural network simulator (chapters 2 and 3)
- **Peregrine** : Framework for efficient Gaussian mixture models (chapter 4)
- **e-PSC**: Probabilistic sparse coding algorithms for event-based data
- **Tonic (minor contribution)** : Library for event datasets and data augmentation
  - Added support for averaged time surfaces [1]

Contents

Introduction

Computer vision started taking shape in the early 1960s as a way to automate image analysis and achieve a full understanding of visual scenes by extracting 3D geometrical information [2, 3]. Building on Hubel and Wiesel's pioneering work on edge detection and orientation selectivity in neurons of the visual cortex, and on hierarchical models of the visual system [4, 5], researchers gradually moved away from the 3D blocks representation and started focusing on low-level features such as edges, corners and curves that get increasingly complex as we move along the hierarchy. This approach to computer vision led to major breakthroughs in artificial intelligence (AI) and inspired state of the art deep learning models such as convolutional neural networks (CNN) [6, 7, 8] and recurrent neural networks (RNN) with applications in facial recognition, self-driving cars and natural language processing among others.

Approaches based on neural networks have yielded impressive results as of late. Machines are now more than capable of outperforming humans in difficult tasks such as mastering the game of Go [9] or even creating artistic images based on a painter's style [10]. In the words of Alan Turing (1951), "it seems probable that once the machine thinking method had started, it would not take long to outstrip our feeble powers". The reality however is more complicated. Tasks such as context awareness and common-sense understanding are easily solved by humans, yet still pose a significant challenge for machines [11]. Furthermore, training deep learning models for large tasks requires massive datasets and powerful systems with high performance graphics processing units (GPU) that consume a lot of energy and are only available to a handful of research laboratories and server farms belonging to large companies [12]. These requirements hinder the deployment of machine learning models and limit Internet of Things (IoT) devices to cloud computing, which is not ideal for embedded systems without a reliable internet connection (e.g. satellites equipped with cameras) or real-time applications requiring fast processing. Nevertheless, combining algorithmic breakthroughs like transfer learning [13], the lottery ticket hypothesis [14] and TinyML [15], with AI hardware accelerators such as Google's tensor processing unit (TPU) [16] or Intel's vision processing unit (VPU) [17], paves the way for energy-efficient inference at the edge. On-device training however, remains an open issue.

In contrast, the brain is capable of sophisticated cognitive functions such as learning with a mere 20 watts of power [18]. It remains unclear how the brain actually accomplishes this feat; even so, the sparsity of neural spike-based computation is thought to be at the centre of the learning process [19, 20]. In the early 1990s, researchers designed a silicon retina inspired by the human visual system, effectively establishing the field of

neuromorphic engineering which aims to solve the energy bottleneck through processors and computation techniques inspired by biological systems [21, 22, 23]. Instead of working at a fixed frequency as is the case in classical Von Neumann architectures, neuromorphic hardware only responds to changes in the data, resulting in a dramatic reduction in power consumption, lower bandwidth, and a high temporal resolution [24]. Latest advances include processors such as the Intel Loihi chip with spiking neural networks-based learning and inference [25], high definition vision sensors [26, 27], and artificial cochleas [28].

Neuromorphic cameras such as the *Asynchronous Time-based Image Sensor* (ATIS) [29] have independent pixels that emit spatially and temporally sparse events with a very low latency, in response to brightness changes in a visual scene. Properly taking advantage of all the benefits of neuromorphic vision sensors requires a novel approach to computer vision and machine learning techniques. State of the art algorithms designed for handling streams of events either (i) convert them into dense frames that can be passed to conventional CNNs, at the expense of the benefits gained from using event-based sensors, or (ii) rely on data-driven approaches that individually handle each event to extract spatio-temporal features. The latter approach while very promising, quickly becomes very expensive as streams of events get larger.

The goal of this thesis is to exploit precise timing on neuromorphic architectures towards a more computationally-efficient learning. With Moore's law becoming less valid due to physical limitations to transistor scaling, it is increasingly complicated to get better processor performance. Meanwhile, artificial intelligence models require more power than ever as the demand increases across multiple fields. Neuromorphic architectures and low power machine learning strategies have the potential to make edge computing, or in other words, on-chip learning, a reality. The gains in processing speed and computational efficiency [30] can be particularly useful for low-power embedded systems with event-based cameras directed towards fast motion tasks and pattern recognition applications. Examples include gesture recognition in mobile phones [31], or even smarter and more natural implant stimulation in neuromorphic vision restoration systems [32, 33, 34] that extract relevant features from the environment without compromising the high temporal resolution of event-based cameras. Coming up with new ways to take advantage of precise timing in learning tasks can also help elucidate some of the obscure mechanisms behind the brain's efficient cognitive functions. We approach this issue through two different angles: spiking neural networks and probabilistic models.

- Spiking neural networks are the natural choice for dealing with the output of event-based vision sensors since they compute with voltages intrinsic to the system, instead of expensive floating points. We introduce a delay learning rule aimed at exploring the benefits of *temporal coding*, where the precise timing of spikes holds meaningful information [35]. In parallel, we also explore a hardware spiking neural network with memristive synapses as part of the EU-funded ultra-low power event-based camera (ULPEC) project. The device is embedded on a system on chip (SoC) with the goal of designing a low-power memristor-based visual data processing system targeted at autonomous driving and the recognition of traffic events.

- Probabilistic models work with probability distributions that can model uncertainty inherent in natural data. We introduce a framework for novel sublinear complexity

clustering algorithms that are perfectly suited for dealing with large streams of events such as those generated by the latest HD event-based cameras [26, 27].

Chapter 1

State of the art

1.1 The visual system

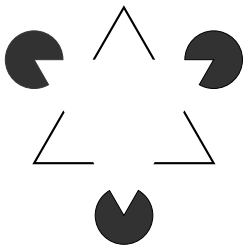


Fig. 1.1 *Edge completion in Kanizsa's triangle. When we look at an image with missing parts, the brain tends to fill in the blanks and form a recognisable pattern. This phenomenon is thought to have been advantageous for survival, particularly for identifying predators when limited information is available.*

Pattern recognition is a fundamental cognitive process that forms the basis of learning in human and non-human animals alike. For instance, the ability to identify sound and environmental cues can be advantageous for growth and survival [36, 37]. Nowadays, this process appears in every aspect of our daily lives through facial recognition, language comprehension or simply the identification of various objects around us. In the visual system, pattern recognition begins in the retina. Bipolar and retinal ganglion cells can extract important features from the available information before further visual processing in downstream brain areas [38, 39]. The brain is capable of carrying out these powerful computations with only 20 watts [18], and therefore merits a closer look in order to improve modern pattern recognition and feature extraction algorithms. This section provides an overview of the visual system and some of the principles governing pattern recognition in the retina.

1.1.1 Visual processing in the retina

Human beings rely heavily on visual perception to make sense of the world around them, to such an extent that the visual cortical areas occupy a large part of the cerebral cortex.

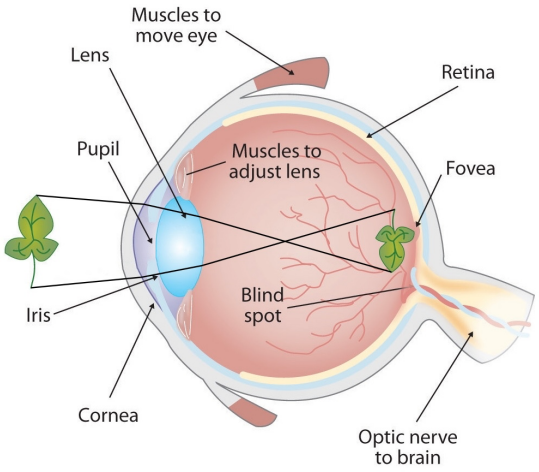


Fig. 1.2 *The human eye. Incoming light is focused on the retina after passing through the cornea, pupil and lens. The retina then extracts visual information through specialised neurons for further processing by the visual cortex. Adapted from [40].*

Vision begins when light passes through the eye, into the retina, where it is converted to electrical signals that can be processed by downstream areas. This phenomenon is called phototransduction. In the visual cortex, the visual information is interpreted into meaningful representations of the world through the interaction between various neurons involved in colour perception, motion, object recognition, and orientation and direction selectivity [40].

The bulk of the eye is designed to focus light reflected from the visual field onto the retina through a process called *refraction*. More specifically, incoming light is first refracted by the cornea before passing through the pupil. The iris is in charge of controlling the diameter of the pupil in order to regulate the amount of light entering the eyeball. Light rays are refracted once more by the lens, to a more finely adjusted focus. This process can be visualised in Fig. 1.2.

The retina is a highly organised structure consisting of layers of neurons specialised in capturing and processing visual information (Fig. 1.3). Photoreceptors are responsible for light transduction into electrical signals that are propagated to subsequent layers of retinal neurons. The retina typically contains two types of photoreceptors, rods and cones. Rods mediate vision in low-light conditions. They are mostly concentrated at the outer edges of the retina, and have a low spatial acuity. Cones on the other hand, are involved in colour vision, and are mostly the only cells present in the centre of the retina.

Further downstream, bipolar cells transmit signals from photoreceptors to ganglion cells, which digitalise the visual information into time-coded spike trains that are relayed to the brain through the optic nerve [41, 42]. Bipolar and ganglion cells form an image processing network capable of simplifying and extracting important features such as motion. These retinal neurons are of two types: ON bipolar and ganglion cells that are active in the presence of light, and OFF bipolar and ganglion cells that emit spikes in darkness. These

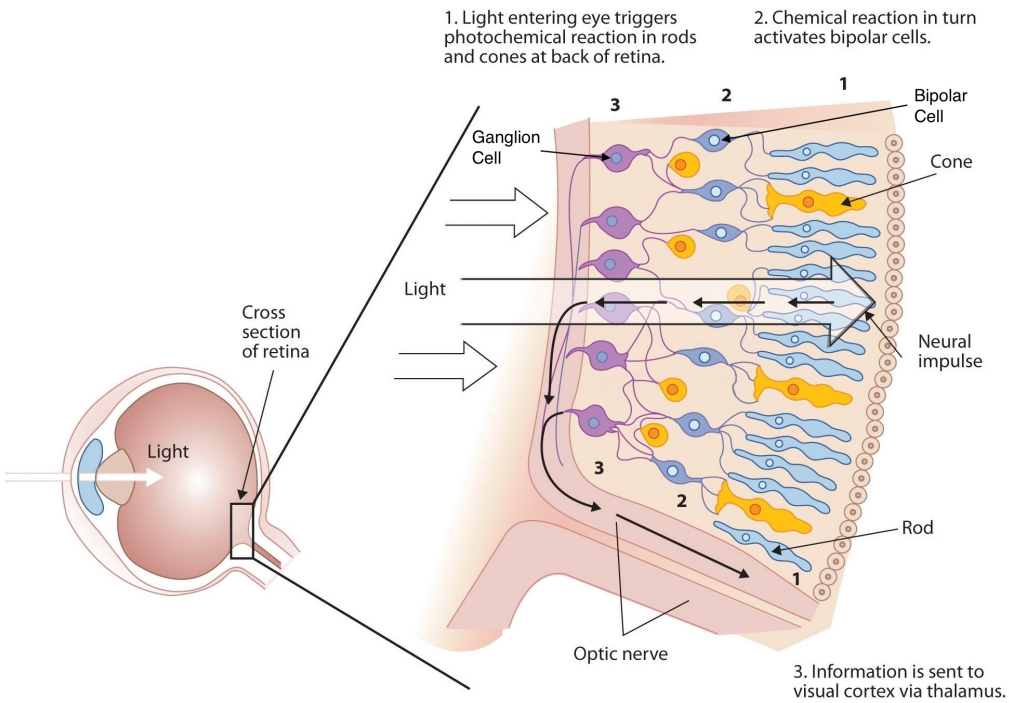


Fig. 1.3 *Organisation of the retina. Light passing through the retina is captured by rods and cones photoreceptor cells and converted into an electrical signal that propagates to downstream retinal neurons, mainly bipolar and ganglion cells, in charge of processing the visual information before sending it to the brain via the optic nerve. Adapted from [40].*

cells have a concentric organisation characterised by a centre-surround antagonism [43]. For instance, ON-centre ganglion cells have an OFF-surround field and vice versa. This antagonism serves to enhance contrast, promotes edge detection and even plays a role in colour vision. Some specialised ganglion cells are also capable of temporal processing and respond selectively to moving stimuli. The retina is even capable of using predictive coding to compensate for delays in signal transmission [38, 44].

The visual system is able to analyse and understand a visual scene within 100ms [45]. In that regard, the retina adopts a first-spike spatiotemporal coding scheme where the relative timing of spikes is considered to carry meaningful information. In contrast, poissonian rate coding schemes integrate spikes over discrete time intervals in order to correlate the firing rate of a retinal neuron to the strength of a stimulus. Taking advantage of precise spike timing allows for the visual cortical pathways to efficiently decipher retinal information [46, 47].

1.1.2 Visual processing in cortical pathways

The visual system, unlike other sensory nervous systems, performs most of the visual processing in the retina. The visual centres of the brain use the information received

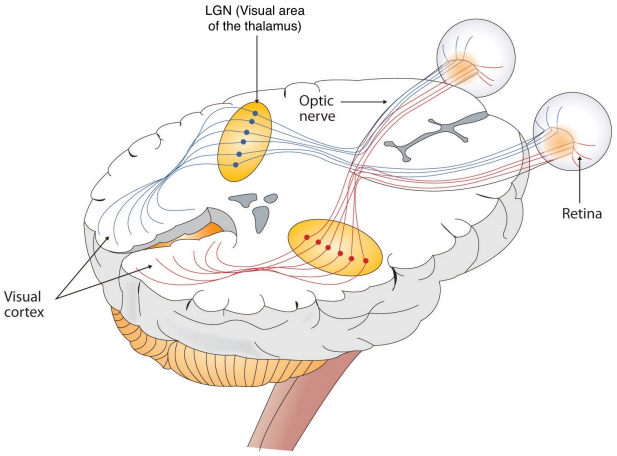


Fig. 1.4 *Anatomy of the visual system. Retinal signals are sent to the visual cortex via the thalamus. Each visual area of the thalamus, also known as the lateral geniculate nucleus (LGN), receives input from both eyes. Adapted from [40].*

from retinal ganglion cells for multiple purposes: mediating subconscious optical reflexes that control the pupil and lens, regulating the circadian rhythm to the 24-hour day-night cycle, centring a visual target on the fovea, and visual processing to extract meaningful information [48].

The optic nerve of each eye sends retinal spike trains to the visual cortex via the thalamus. More precisely, the visual pathway passes through the lateral geniculate nuclei (LGN) which organise and integrate the input from both eyes. The LGN maintains a retinotopic mapping which is useful for enhancing the contrast discrimination brought about by centre-surround antagonism. Neurons from the LGN project directly to the primary visual cortex (V1). At this point, the retinal information splits into information processing streams that traverse other areas of the brain. The dorsal stream, or the "where/how pathway", is concerned with spatial location, motion and depth, and essentially facilitates visually-guided actions such as grabbing an object [49]. The ventral stream otherwise known as the "what" pathway is involved in object recognition, form analysis and colour [50, 51, 52]. However, recent studies suggest that the two visual processing pathways might not be as functionally distinct as once thought [53].

The ventral stream of the visual pathway dealing with pattern recognition is thought to be organised in a hierarchical manner [5, 4]. The outline of all objects can be broken down into a series of linear segments. As one moves along the hierarchy of the ventral stream, neurons gradually become responsive to increasingly complex stimuli, starting from oriented bars, gratings and edges in the primary visual cortex and ending with objects and faces in higher areas of the visual pathway. Instead of the centre-surround concentric receptive fields found in the retina and the LGN, neurons in the primary visual cortex have an elongated receptive field that responds to bars or gratings at a preferred

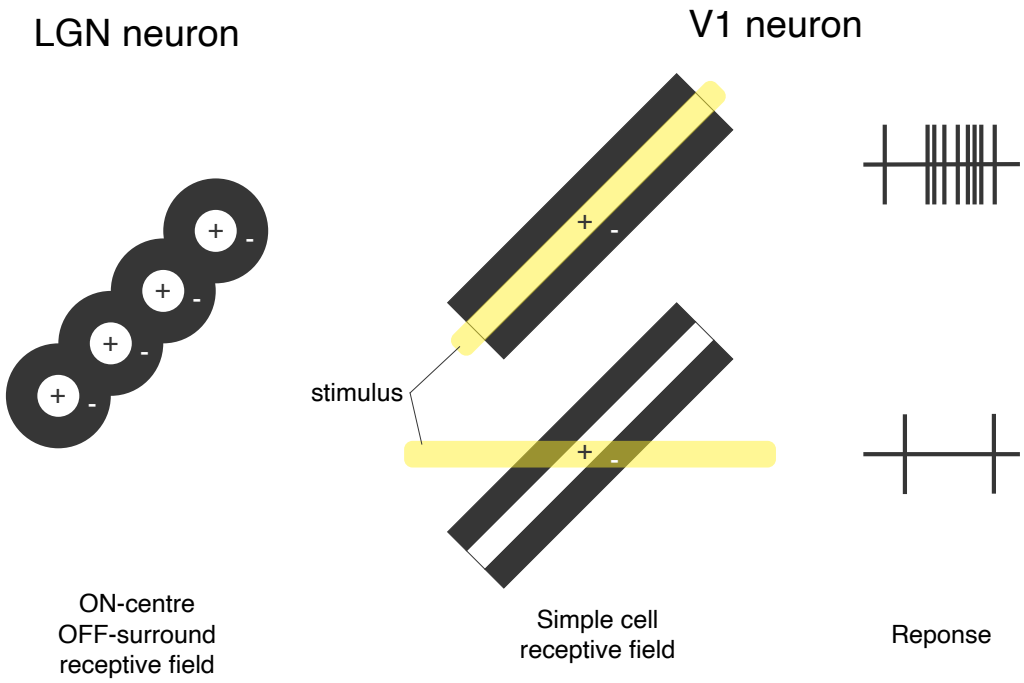


Fig. 1.5 *Receptive fields of LGN and V1 neurons. LGN neurons have a centre-surround antagonism that is used to enhance luminance contrast discrimination. V1 neurons have elongated receptive fields that respond to bars of light at a preferred orientation.*

orientation (Fig. 1.5). Orientation selectivity and contrast discrimination are achieved by flanking the stimulus with inhibitory regions. These neurons come in a variety of types that provide information about the orientation, direction, length, corners and edges of a stimulus [54]. This information, in combination with input from other brain areas, is also used for depth and motion analysis. We can distinguish two types of orientation-selective cells in the primary visual cortex: simple cells respond to oriented bars and gratings at a precise location within their receptive fields; complex cells on the other hand respond anywhere within the receptive field. The hierarchy of the ventral visual pathway integrates the input from simple and complex cells to construct increasingly complex features and develop translation invariance to them [55].

This hierarchical model of vision is considered the source of inspiration for popular deep learning architectures such as convolutional neural networks (CNN) which aim at solving computer vision tasks [6, 7]. These architectures will be further explored in section 1.3.

1.2 Event-based vision sensors

Computer vision is a branch of artificial intelligence that allows machines to analyse and interpret visual scenes. The first applications of computer vision dealt with optical character recognition [56]. Nowadays, increasingly sophisticated algorithms have expanded

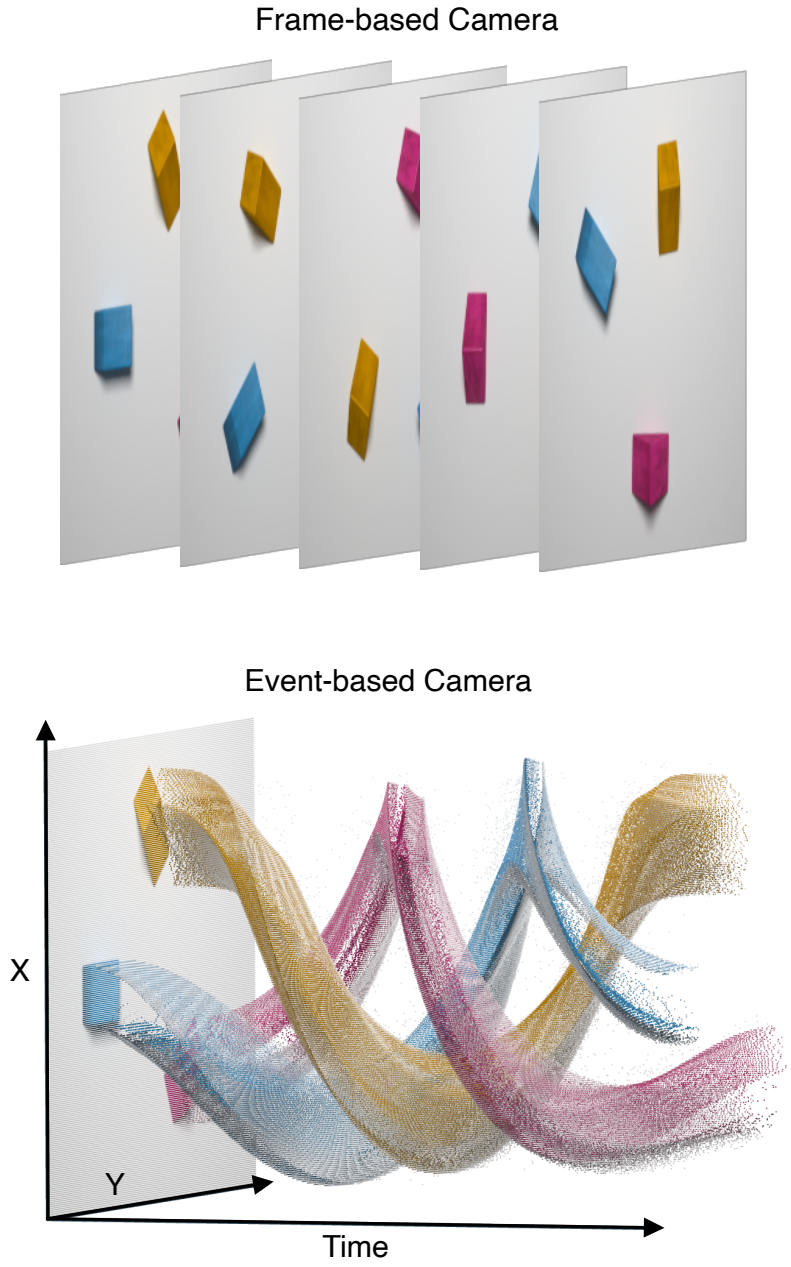


Fig. 1.6 *Output of conventional cameras and event-based cameras.* ***(Top)*** *Conventional frame-based cameras capture dense images at fixed time intervals.* ***(Bottom)*** *Event-based cameras independently update each pixel in an asynchronous manner depending on luminance changes in a visual scene. The output is a spatially sparse representation with microsecond time resolution. Colours are artificially injected for better visualisation. Data courtesy of Alexandre Marcireau.*

the capabilities of computer vision into numerous fields such as facial recognition [57], recognition of traffic events for self-driving cars [58], and medical diagnosis [59]. These algorithms have traditionally relied on conventional cameras that capture dense frames at fixed time intervals [60]. A global shutter simultaneously captures light from all pixels, over a specific exposure time, resulting in redundant visual data representing an average luminance.

Unlike frame-based vision sensors, the human retina can adapt to a broad range of illumination levels. As seen in section 1.1, photoreceptors propagate analog signals for further processing in downstream layers of the retina. In the early 1990s, Misha Mahowald and Carver Mead devised a silicon retina that mimics the behaviour of biological retinas to a certain extent, effectively starting the field of neuromorphic engineering [21, 22, 23]. Early iterations of the silicon retina included a logarithmic pixel design, a spatial contrast discrimination circuit based on the centre-surround antagonism observed in bipolar cells [43], and circuits dedicated to pre-processing the visual scene [61, 62]. The vast majority of computer vision research was at the time, and still is to this day, dominated by digital computers with Von Neumann architectures. Developing novel algorithms compatible with the fully analog design of silicon retinas was therefore quite a challenging task.

The Dynamic Vision Sensor (DVS) developed in 2008 [63], streamlined the architecture of silicon retinas by dropping the spatial contrast enhancement circuit and adopting the Address Event Representation (AER) protocol [64]. This communication protocol converts an analog signal into a spatially sparse stream of events, or spikes, with sub-millisecond temporal resolution. We can visualise the output of these event-based vision sensors in contrast to conventional frame-based cameras in Fig. 1.6. Combining the silicon retina's analog circuits with a digital output solves the compatibility problem with conventional computer architectures and makes it easier to transfer the data between chips or develop machine learning algorithms that analyse and interpret a visual scene. This latest generation of vision sensors reaches a new level of performance by independently updating each pixel in an asynchronous manner. An event is triggered when the logarithmic luminance of a pixel crosses a threshold (Fig. 1.7), and is characterised by a set of spatial coordinates x and y, a timestamp t and a polarity p that encodes either an increase (ON events) or a decrease (OFF events) in luminance. The total amount of events is directly driven by the activity of a scene. Consequently, event-based cameras need to be either constantly in motion to update the visual scene, or capture moving targets when the camera is static. This event-based architecture holds several advantages over conventional cameras:

- **high dynamic range:** Event-based cameras can operate under a broad range of illumination levels due to a significantly high dynamic range ($> 120dB$) compared to conventional cameras (60dB).
- **low motion blur:** Conventional cameras can lead to streaking of rapidly moving objects when an image being captured changes during a single exposure time. This motion blur phenomenon is less of an issue in event-based cameras as the pixels are independent and do not rely on a global exposure time. Nevertheless, motion blur in the order of 1ms still occurs in event-based cameras, and is typically the

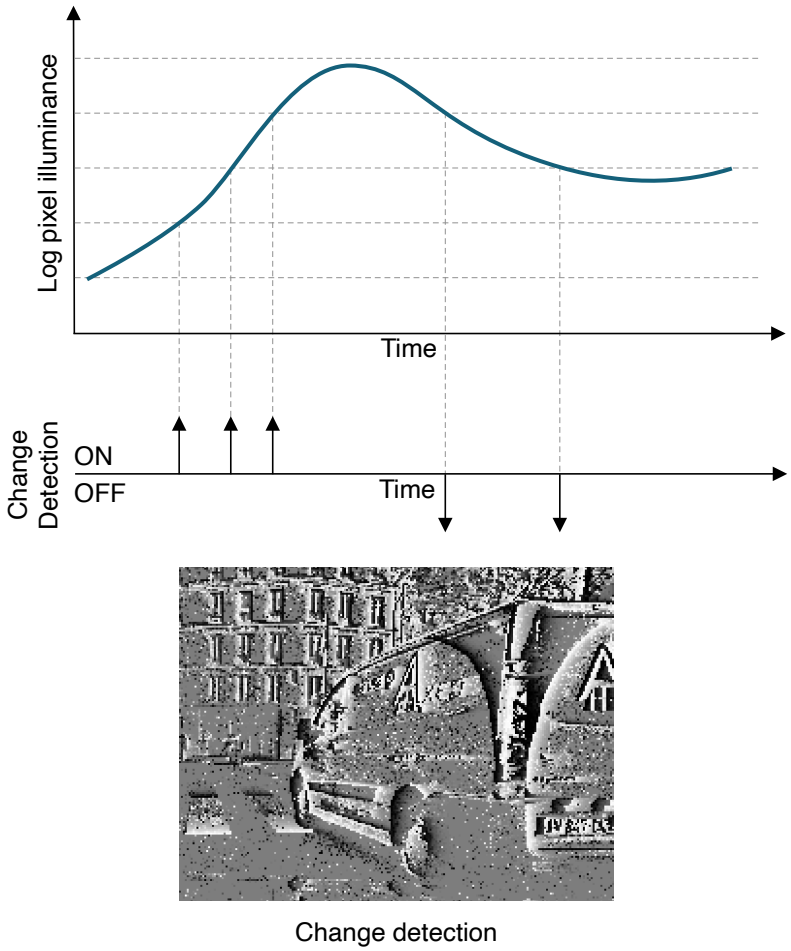


Fig. 1.7 *Behaviour of a single pixel in an event-based vision sensor. A new event is triggered when the logarithmic luminance of a pixel crosses a user-defined threshold. ON events (bright pixels) indicate an increase in luminance while OFF events (dark pixels) represent a decrease in luminance.*

result of multiple events being generated by the same pixel in response to a change in luminance [24].

- **high temporal resolution and low latency:** Event-based cameras can detect luminance changes with sub-millisecond temporal resolution and output events at a very low latency, making them ideal for high speed motion perception. The DVS for example, has a minimum latency of 15µs [63], but real-world usage brings the latency in the order of a few milliseconds [24].
- **low power:** The power consumption of an event-based vision sensor is reduced to less than 10mW by eliminating the redundancy in the data and only updating relevant pixels [65]. Efficient algorithms would need to be developed in parallel to fully take advantage of the reduction in energy demands.

Several cameras that behave similarly have been developed since then, with reduced sensor

noise [66] and improvements to the dynamic range [67], sensitivity [68, 69], latency [70] and spatial resolution [71]. Noteworthy event-based cameras include the ATIS [29], the Dynamic and Active-pixel Vision Sensor (DAVIS) [70], and a DVS camera developed by Samsung [71]. Lately, the industry's increased interest in neuromorphic technology led to the development of the first HD event-based cameras by Celepixel [26, 72] (Celex) and Prophesee [27]. This technology opens the door to a wide variety of applications where fast motion perception, precise timing and energy efficiency are important. Some examples include autonomous cars that use visual processing for navigation [1, 73], positioning and collision avoidance in robotics [74], and even medical applications such as visual prosthesis [34].

1.3 Pattern recognition on event streams

Extracting features from a stream of events is not particularly straightforward. The asynchronous and sparse nature of the data is incompatible with gradient-based optimisation algorithms such as backpropagation which form the backbone of modern deep learning [75, 6, 7]. The precise timing of events adds further complexity to the task, since standard computer vision algorithms do not consider the temporal dimension. The neuromorphic community is constantly experimenting with new ways to represent streams of events in order to facilitate the development of algorithms that can handle the spatial and temporal sparsity of the data. In the context of pattern recognition for event-based data, algorithms fall into two main categories: *event-to-frame* methods that generate frames from batches of events to take advantage of decades of computer vision research, and *event-by-event* methods that process each event individually and require rethinking from the ground up. Lesser known approaches consist in voxelising the stream of events and passing it through a CNN [76, 77] or a broad learning system (BLS) [78] which is more suitable for learning in a big data environment since it does not require any layer stacking. Other approaches involve converting the stream of events into geometric data structures such as 3D point clouds that can be processed by neural network architectures [79, 80] based on PointNet [81], or graphs [82] that can be used as input to a CNN.

1.3.1 Event-to-frame methods

The first approach to feature extraction in the context of computer vision involves converting event streams into image sequences through simple approaches like binning events [73, 83], counting events [84] or even summing polarities [85] in a pixel-wise manner over a user-defined integration time. More recently, [86] was able to reconstruct high-quality frames by passing temporal bins of events through a recurrent CNN. The generated frames provide a familiar two-dimensional grid representation that is compatible with standard computer vision algorithms such as CNNs, at the detriment of the sparsity and temporality of event data [85, 86, 87]. The reliance on dense data and optimisation-based methods negates most of the advantages of neuromorphic vision sensors, particularly the energy efficiency aspect which is central to the neuromorphic computing paradigm. Nevertheless, frame integration procedures that include a decay based on past events such as leaky surfaces [88] or memory surfaces [89] are better able to preserve the time resolution of event-based vision sensors, and are generally more robust to noise. [88] takes

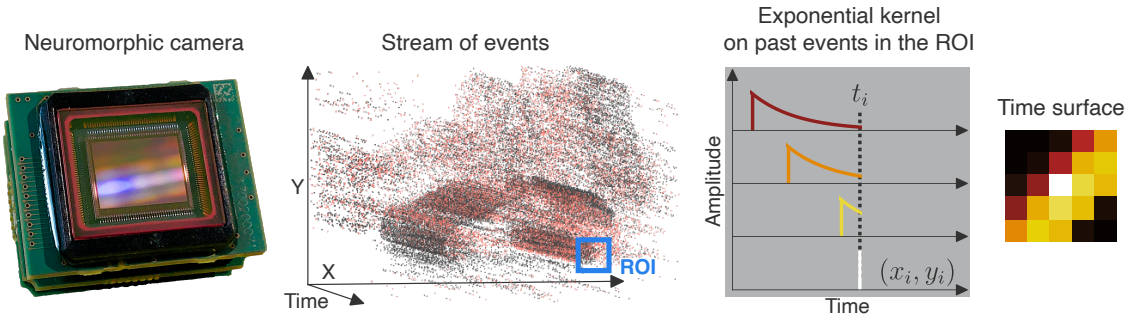


Fig. 1.8 *Time surface generation pipeline. For a given stream of events, we apply an exponential decay on past events in a region of interest (ROI) centred around each event. The resulting grid structure describes the spatio-temporal context around an event. The dashed line represents the timestamp of the centre event.*

it a step further with fcYOLE and tries to re-inject sparsity into CNNs by only looking at regions in frames where events are received, bridging the gap between fully frame-based and fully event-based algorithms.

1.3.2 Event-by-event methods

The algorithms introduced throughout this doctoral thesis fall under this broad category. These methods trigger a series of computations at each event, taking advantage of all the benefits of neuromorphic vision sensors (see section 1.2). Feature extraction algorithms that work on an event-by-event basis generally aim for a low power consumption that enables on-device processing towards IoT applications [31], and a low latency which is very useful in tasks requiring a fast reaction time [1, 90]. Making algorithms more energy efficient can be achieved through biologically plausible architectures and by exploiting the sparsity of event-based data [91]. Indeed, neuromorphic vision sensors naturally extract spatio-temporal features and enhance edges which greatly simplifies the end-to-end classification pipeline. Utilising the high temporal resolution of neuromorphic vision sensors not only contributes greatly towards lower latency algorithms but also offers a level of performance that is superior or matches state-of-the-art frame-based algorithms in some recognition tasks [92, 1]. In any case, simpler computations are usually preferred in event-based programming due to the large number of events generated by neuromorphic vision sensors. To put this into perspective, an ATIS vision sensor generates on average one million events per second. Spiking Neural networks (SNN) are a natural choice for dealing with event streams on an event-by-event basis [93, 92, 94, 95, 96, 97]. We can also make use of handcrafted grid-like structures called time surfaces that take into consideration the spatio-temporal context around each event [90, 98, 99, 100, 1].

a) Time surface based algorithms

For a given stream of events, we look into the spatio-temporal neighbourhood of each event e_i to create a time surface as formally introduced in [100], which resembles a local image patch that represents the temporal relation between events. The surface is locally updated

using a decay function within a $(2r+1)^2$ spatial neighbourhood σ_i, centred on (x_i, y_i) at time t_i (Fig. 1.8). We use an exponential kernel to decay the most recent activations such that:

$$\begin{array}{rcl} S_i : \sigma_i \in \mathbb{R}^2 & \rightarrow & \mathbb{R} \\ (x_k, y_k)^T & \mapsto & S_i(x_k, y_k) = \exp\left(\frac{t_k - t_i}{\tau}\right) \end{array} \tag{1.1}$$

where τ is the decaying factor reflecting the scene dynamic. Since t_i is the time of the current and most recent event, we always have $t_k \leq t_i$.

The exponential decay is a costly function to compute. As an alternative, we can use a piece-wise linear function to approximate the exponential decay and speed up computation without dramatically affecting the performance in classification tasks [90], such that:

$$\begin{array}{rcl} S_i : \sigma_i \in \mathbb{R}^2 & \rightarrow & \mathbb{R} \\ (x_k, y_k)^T & \mapsto & S_i(x_k, y_k) = max(0, \frac{t_k - t_i}{\tau'} + 1), \end{array} \tag{1.2}$$

where τ' is a decay constant similar to τ.

The HOTS algorithm proposed in [100] and improved upon by [31] makes use of pattern matching to learn from time surfaces a set of representative features referred to as prototypes, using an online clustering technique. It is important to note that any clustering technique, online or offline can be used in this step. The algorithm can be employed in a hierarchical manner by stacking layers that have a different spatial neighbourhood size or decaying factor τ. Prototypes from the final layer of this pattern matching network are gathered and used in a distance-based linear classifier such as the k-nearest neighbours algorithm (k-NN) [31], or more complicated classifiers, namely extreme learning machines (ELM) which are feedforward neural networks that converge faster than networks trained using backpropagation [90]. Although, these classifiers cannot reach the performance of backpropagation-based training methods.

HOTS prompted the development of various algorithms centred around representing an event as a spatio-temporal context. FEAST [99] extends HOTS with adaptive thresholds, or in other words, a homeostasis mechanism that promotes competition and regulates the overall activity of prototypes [101, 102]. HATS [1] reduces temporal noise in time surfaces by taking into consideration all past events in a ROI, instead of limiting it to the most recent ones. HATS completely bypasses the pattern matching step by averaging the new time surfaces into a histogram that can be used to train a support vector machine (SVM) classifier. DART [98] builds upon the fact that rotations and scale changes in a Cartesian system appear as translations in the log-polar domains [103], and adds rotation and scale invariance by encoding the spatio-temporal context as a log-polar grid instead of a Moore neighbourhood. DART, along with the descriptor proposed in [90], tries to solve the time surface's inability to handle variable motion and different data rates by using a decay kernel dependent on incoming events instead of a time constant τ.

Time surface-based methods are promising in regards to pattern recognition. However, event-by-event methods are data-driven, and therefore largely bound by the number of events in a visual scene. Feature extraction algorithms have so far been tested in small event-based datasets with very low spatial resolution vision sensors. With the advent of high definition cameras that can output up to 50 million events per second [26, 27], methods

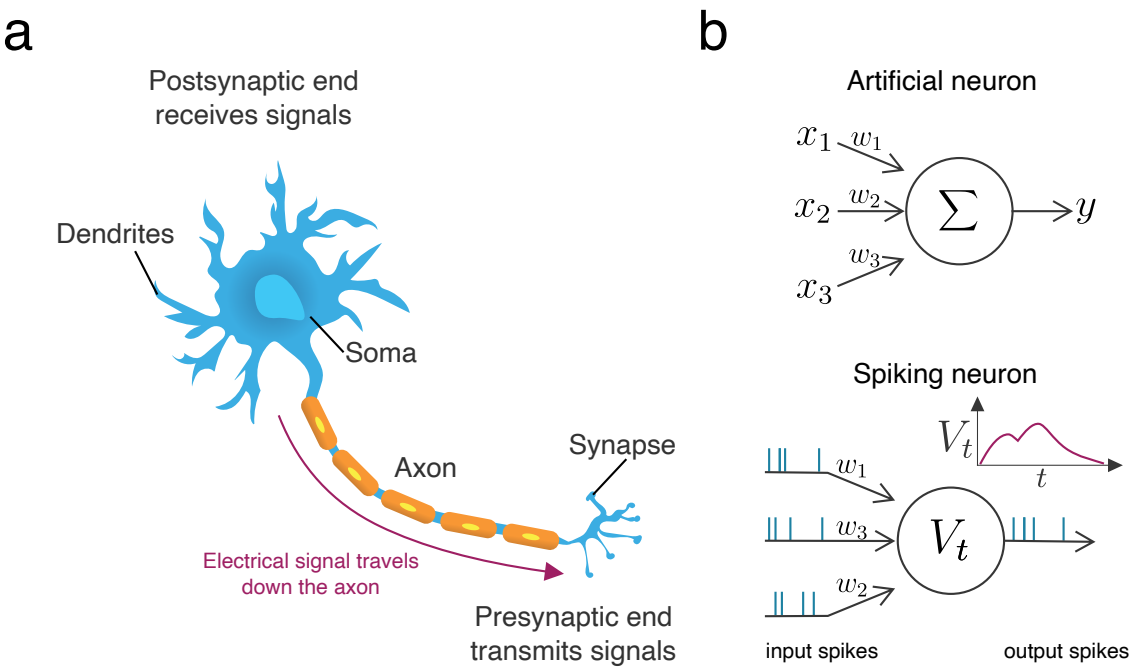


Fig. 1.9 *Neural computation.* ***(a)*** *In a biological neuron, electrical signals travel through the axon and are propagated to a postsynaptic neuron via synapses.* ***(b)*** *Artificial neurons take continuous values as inputs and instantly compute the output. Spiking neurons in comparison communicate with spikes and have an internal state V_t that integrates spikes over time.*

relying on clustering such as HOTS and DART would be unable to handle the stream of events in a reasonable time frame. For this particular reason, apart from research labs, key players in the industry are as of now relying on event-to-frame strategies [104].

b) Spiking neural networks

SNNs are widely considered to be the third generation of artificial neural networks (ANN) [35]. Much like their biological counterpart, spiking neurons communicate via sparse and asynchronous binary signals that are transmitted and processed in a massively parallel manner, and are therefore able to directly process the output of neuromorphic vision sensors. The membrane potential V_t of a neuron integrates incoming spikes over time (Fig. 1.9). The neuron emits a spike of its own once V_t reaches a certain threshold, and propagates it throughout the network via synapses characterised by a weight and optionally, a delay [105]. Spike-based communication is however costly to simulate on the classical Von Neumann architecture, and dedicated hardware is usually required in order to unlock the full potential of SNNs [106, 107, 25], namely, energy-efficiency and real-time inference [108].

SNNs have had great success in robotics applications [74, 109, 110]. Their performance was until recently lacking in pattern recognition tasks compared to the state of the art in deep learning networks [111, 112, 113]. Recent advances have shown great promise in

bridging the performance gap in object recognition tasks. Learning methods for SNNs can be split into three broad categories:

- **Hebbian-based learning:** SNNs can be trained with biologically plausible unsupervised local learning rules, the most popular one being Spike-timing-dependent plasticity (STDP) [114, 19]. The majority of Hebbian learning rules are based on synaptic weight modulation [113]. Long-term potentiation (LTP), or an increase in synaptic weight, occurs when a presynaptic neuron fires before a postsynaptic neuron. Whenever the opposite scenario occurs, synaptic weights are decreased in a process called long-term depression (LTD). This process is considered local since it only depends on information available to a given pre- and postsynaptic neuron pair. STDP-based feedforward architectures dedicated to pattern recognition have been well-explored [19, 113, 96, 115], and despite not being competitive compared to the state of the art, their simplicity is coveted in low power end-to-end object recognition systems [116, 117]. The state of the art in Hebbian-based learning includes supervised variants of STDP [118] and deep architectures that utilise multiple STDP-based convolution and pooling layers in an effort to gain more abstract and representative features [119, 97, 120].

- **ANN-to-SNN conversion:** SNNs can also be converted from state of the art ANNs trained with conventional optimisation-based deep learning algorithms [121, 122, 123, 124, 125]. The weights of the trained ANN are used in spiking neurons for inference. Until recently, these conversion strategies were incompatible with the temporal structure of SNNs. With the advent of streaming rollouts that essentially enable propagation delays in ANNs [126, 127], these networks can now achieve state of the art results in object recognition with event-based datasets. ANN-to-SNN however remains a costly method bound by the current limitations of deep learning when it comes to energy-efficiency.

- **Gradient-based learning:** Gradient-based approaches cannot be applied to SNNs in a straightforward manner due to the non-differentiability of spikes activity. In fact, a spike is modelled by the Heaviside step function $\mathcal{H}(x)$. Backpropagating the error through a spiking neuron leads to zero gradients and no weight updates, since the derivative of $\mathcal{H}(x)$, the Dirac function, is zero everywhere and undefined at $x = 0$. Gradient-based learning can be achieved in various ways. For instance, recent work suggests approximating the behaviour of spiking neurons with a differentiable surrogate function, paving the way for SNN training algorithms that can make use of the error backpropagation optimisation method [94, 128, 92, 93]. Other methods of computing a gradient include latency learning [129, 130] and using continuously valued neuron activity [95]. This category of direct training algorithms for SNNs can offer the best compromise between performance and energy-efficiency, for instance through layer-wise local losses [131, 93] or low precision weights [130, 132].

Throughout these learning rules, spiking neurons can employ different strategies for computation. *Rate coding* takes into consideration the mean firing rate of a single neuron or a population of neurons as an informational parameter [133]. SNNs being largely data-driven, this encoding strategy can quickly become more costly than frame-based methods especially when a great deal of spikes are required. Furthermore, visual perception

is incredibly fast. *Temporal coding* positions itself as an alternative strategy that assigns meaning to the precise timing of spikes [35], reducing the number of spikes required to encode information. Whether by relying on the first spike latency [38] or by looking at the relative firing order of neurons [134], strategies based on precise timing provide a reasonable explanation in regards to the brain's energy-efficient learning. In fact, a number of studies show the importance of temporal coding in sensory systems [135, 136, 137, 138, 139]. Beyond STDP [19] which has recently hit a wall in terms of performance, three-factor learning rules that combine Hebbian learning and neuromodulation can take advantage of temporal coding for more complex algorithms based on supervised or reinforcement learning [140, 119, 141].

1.4 Efficient hardware for machine learning

In environments with power, latency or bandwidth constraints such as autonomous vehicles [1, 73] or mobile phones [31], deploying machine learning algorithms at the edge and reducing the dependency on cloud computing is increasingly sought after. The previously explored event-based algorithms have the potential to make on-device computing a reality when co-designed with dedicated hardware. Indeed, hardware and software integration is essential for pushing the limits in terms of delivering high performance with a small power footprint. In fact, most modern computer systems such as our phones and laptops, include co-processors designed to accelerate specific tasks which cannot be efficiently handled by the central processing unit (CPU). In vision and machine learning tasks, graphical processing units (GPU) are by far the most widely used co-processors due to their speed and flexibility. The high memory bandwidth and large number of cores makes it particularly well suited for deep learning models as they enable parallel computation on big datasets [7]. Reconfigurable devices such as field-programmable gate arrays (FPGA) offer lower latency and better power-efficiency than GPUs by making it easier to adapt the hardware according to the needs of the algorithm [142]. For maximum speed and efficiency however, application-specific integrated circuits (ASIC) can be designed to perform a narrow set of operations such as processing neural networks at the cost of programmability. Examples of ASIC chips include Google's tensor processing unit (TPU) [143, 16] and Intel's vision processing unit (VPU) [144]. Through low precision floating-points and hardware-level optimisations these chips can significantly speed up inference. The Edge TPU for instance, is capable of performing four trillion operations per second using only two watts [143]. CPUs, GPUs, and AI accelerators are held back by the transfer rates between the processing and memory units. Neuromorphic chips emulating neural networks in hardware, tackle this problem, known as the Von Neumann bottleneck, with architectures that have collocated processing and memory units, significantly improving parallelism and energy efficiency [145]. These inherently low power chips rely on the event-based computing paradigm to dramatically improve training and inference speeds [107, 146, 147], bringing us closer to full edge computing without compromising the energy budget. Each core in these processors can be implemented using digital or mixed-signal circuits. Digital neuromorphic manycore processors include the IBM TrueNorth chip [147], SpiNNaker [107], the Intel Loihi chip [25], ODIN [148], and more recently, the Akida neural processor [149]. Mixed-signal neuromorphic processors such as the DYNAP-SE [146], BrainScaleS [150]

and Neurogrid [151] can take advantage of the device's physical properties for an even lower power footprint, but they are inherently noisy and require a physical circuit per neuron or synapse which can take up more space on silicon.

1.5 Visual restoration in non-human primates

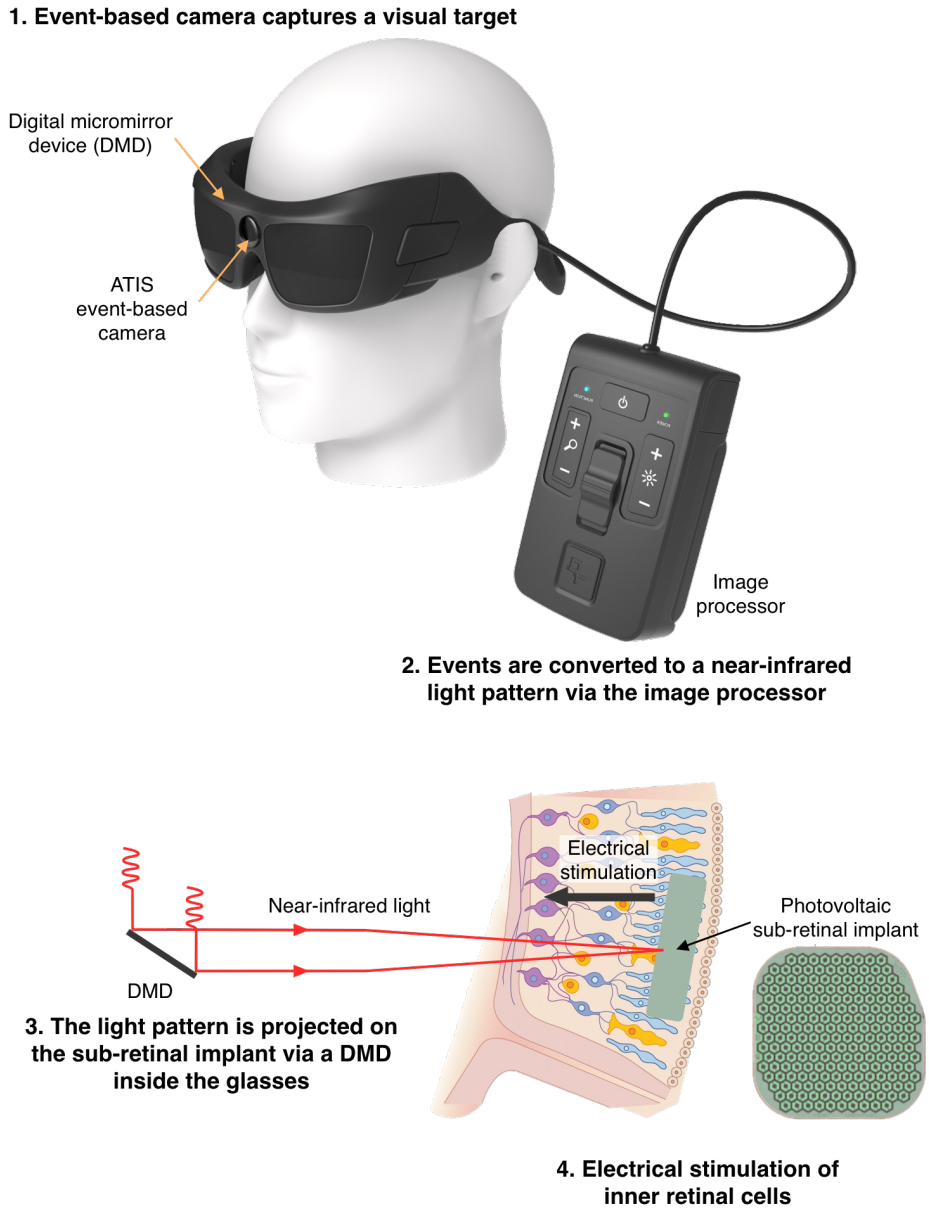


Fig. 1.10 *Prototype for the PRIMA vision restoration system with a photovoltaic subretinal implant that stimulates remaining retinal cells. This prototype was based on the IRIS II vision restoration system which included an ATIS event-based vision sensor that offers better compatibility with the biological retina than conventional cameras [152, 32]. Implant picture courtesy of Paul-Henri Prevot.*

Retinal degenerative diseases such as dry age-related macular degeneration (AMD) and retinitis pigmentosa (RP) are highly prevalent diseases that can lead to the progressive degeneration of photoreceptors and partial blindness [153]. Dry AMD for instance progresses into a loss of central vision. Development of vision restoration solutions has been progressing steadily. In fact, vision restoration systems based on retinal implants have already reached clinical trials [32]. Early prototypes of the PRIMA vision restoration system developed by Pixium Vision exploited the high temporal resolution of event-based cameras for a more natural simulation of the retina. The stream of events is converted to a near-infrared light patterns that are used to stimulate remaining bipolar cells and retinal ganglion cells through a photovoltaic subretinal implant that converts non-visible near-infrared light into an electrical current. The full protocol can be seen in Fig. 1.10.

While frame-based video cameras are more commonly used in vision restoration systems [154], psychophysical experiments showed a noticeable improvement when performing motion perception tasks with high temporal frequency simulations [155, 33, 156]. The sub-millisecond temporal resolution of event-based cameras could therefore make it easier for patients to solve everyday tasks involving moving stimuli. In fact, the biological retina also functions at a high temporal resolution [157, 38]. Working with an event-driven vision sensor increases the compatibility of the stimulus between the implant and downstream layers of the retina, and holds additional benefits, including an enhanced contrast discrimination, or in other words, built-in edge detection, and a high dynamic range that would allow the vision sensor to easily function under a wide range of illuminations.

As part of the pre-clinical validation phase, we wanted to demonstrate the efficacy of the PRIMA vision restoration strategy through a series of behavioural tests on non-human primates implanted with the photovoltaic subretinal implant [34]. As a model for the PRIMA system, we developed a split-lamp setup that allows ocular movement tracking for analysis purposes, and achieves very precise *in vivo* near-infrared activations of the subretinal implants at a high refresh rate. The behavioural experiments, my personal contribution to this work, consisted in studying the saccadic response of a non-human primate from a central fixation point, towards visual stimuli displayed for $500ms$ in the peripheral visual field under near-infrared and visible light. The non-human primates had healthy retinas with the exception of a blind zone over the implant. Fig. 1.11 shows the response of two non-human primates to implant activation by near-infrared visual stimuli. Visible stimuli elicit saccades everywhere except over the implant, while non-visible near-infrared stimuli only elicit saccades over the implant, demonstrating the primates' ability to perceive stimuli in their blind zone.

This project, published in Nature Biomedical Engineering [34], is one of the primary motivations for the work presented in this doctoral thesis. Indeed, despite impressive clinical trial results on other vision restoration systems such as the IRIS II or the Alpha IMS [32, 158], the restored vision remains quite limited and highly variable across patients, mainly due to the brain's ability to reorganise itself and adapt to changes. Applying machine learning algorithms to extract important features from visual scenes can help ease the burden on remaining retinal cells by only sending the relevant visual information for a particular task, potentially leading to improvements in the restored vision and facilitating neural plasticity. Designing algorithms that can handle streams of events is a challenging

task currently under active research by the neuromorphic community. For that reason, the latest iteration of the PRIMA vision restoration system reverted back to a frame-based solution, losing in the process, the bio-compatibility and the improvements in motion perception which are important to a patient's quality of life [155, 33, 156].

Augmenting devices such the PRIMA vision restoration system with machine learning requires algorithms and dedicated hardware capable of learning on the fly and adapting to a broad range of tasks in a power and thermal-constrained environment. Frame-based strategies are not particularly well-suited for the task. Building on the event-by-event pattern recognition techniques seen in section 1.3, we propose, in the following chapters, computationally-efficient unsupervised machine learning methods, as well as a bio-inspired end-to-end hardware implementation of SNNs that can take advantage of all the benefits offered by event-based cameras such as the high temporal resolution, towards advanced vision applications with very low power and latency requirements.

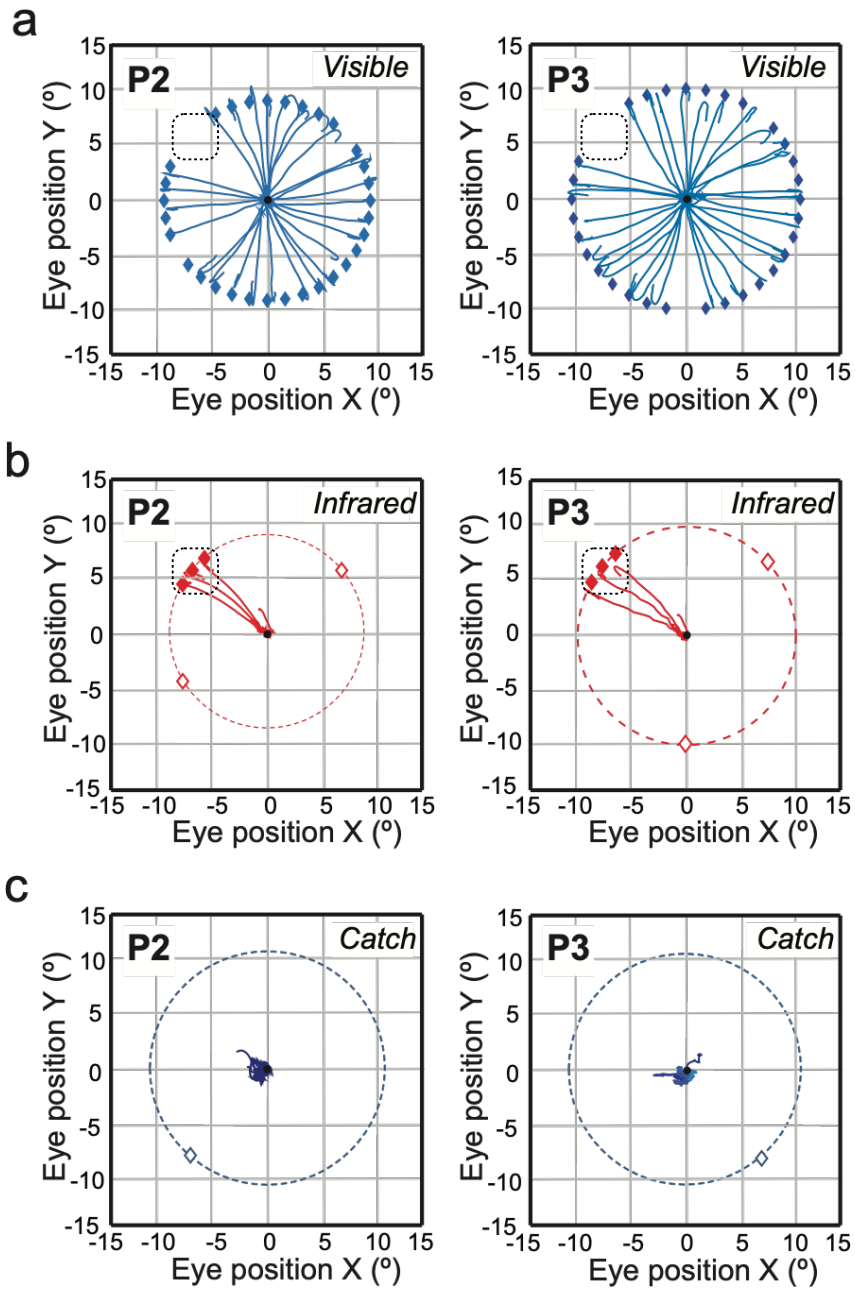


Fig. 1.11 *Subretinal implant activation in two non-human primates [34]. After fixating on a central spot (black dot), visible and near-infrared visual stimuli are displayed every* 10° *in the peripheral visual field.* ***(a)*** *Visible stimuli correctly elicit a saccadic response by the non-human primates everywhere except over the implant (delimited by the dotted black square).* ***(b)*** *Near-infrared stimuli only elicit saccades over the implant.* ***(c)*** *Catch trials allow us to control for external light sources as potential phosphene generators over the implant. Visual stimuli were displayed without a visible or near-infrared light source.*

Chapter 2

A Myelin plasticity model for spiking neural networks

2.1 Introduction

Spiking neural networks (SNN) have traditionally relied on weight-based synaptic plasticity for supervised and unsupervised learning. Besides the Hebbian-based STDP learning rule and the backpropagation-based SNN algorithms discussed in section 1.3, [159] proposed the tempotron model which modulates synaptic weights in a supervised manner. State of the art algorithms cannot yet properly explain the learning mechanisms governing biological neurons. Studies suggest precise spike timing and temporal coding play an important role in learning and memory formation [160]. Spike transmission delays however, have not been thoroughly explored in regards to understanding spatio-temporal sequences, and they are generally regarded as a non-plastic process. [114] for instance, adds beneficial noise to an STDP-based network by randomly initialising delays, and keeping them fixed.

Spikes are delivered to a neuron's post-synaptic partners through its axon with a transmission delay dictated by the axon's conduction velocity. The conduction velocity is dependent on both the diameter of the axon and the thickness of the *Myelin* sheath around it [161]. Myelin is a phospholipid substance formed by glial cells and its presence increases the conduction velocity of axons by wrapping around them and acting as an electrical insulator. Auditory neurons for example, act as coincidence detectors by using interaural time differences for sound localization [162, 163]. This particular mechanism involves precisely tuned delay lines. Furthermore, it has recently been shown that the myelination of axons can be influenced by neural activity [164, 165, 166, 167] suggesting that a form of myelin plasticity is also at work – something that should be taken into consideration when developing learning algorithms for SNNs [168, 169].

By optimizing conduction delays, a myelin plasticity-based model paves the way for directly learning the time dynamics of incoming spikes and extracting meaningful spatio-temporal patterns. However, the fundamental mechanisms dictating this type of plasticity is still poorly understood, and there is a need to understand exactly how changes in conduction velocity actually promote learning. Previous conduction delay plasticity

algorithms have often not been tested with practical tasks such as pattern recognition and clustering [170, 171]. The deltron [172] takes inspiration from the tempotron model [159] to adjust conduction delays through gradient descent dynamics. [173] extended the polychronisation model developed by [114] to include learnable conduction delays for classification and [174] applied this approach to pattern storage. [175] developed a probabilistic delay learning model which adjusts conduction delays and synaptic weights. However, [175] used this to classify time-invariant databases such as MNIST, which have no temporal structure, making them a poor choice for evaluating computation based on spike timing.

This chapter introduces a model for myelin plasticity that specialises postsynaptic neurons to specific patterns through axonal delay modulation. The algorithm in question can be unsupervised or supervised when a teacher signal with the desired spike timing is available. We will start by introducing the learning rule and testing it on a one-dimensional artificial dataset as a proof of concept. We will then present a real-life application to this delay learning rule by solving a touch localisation task on a real dataset.

2.2 Methodology

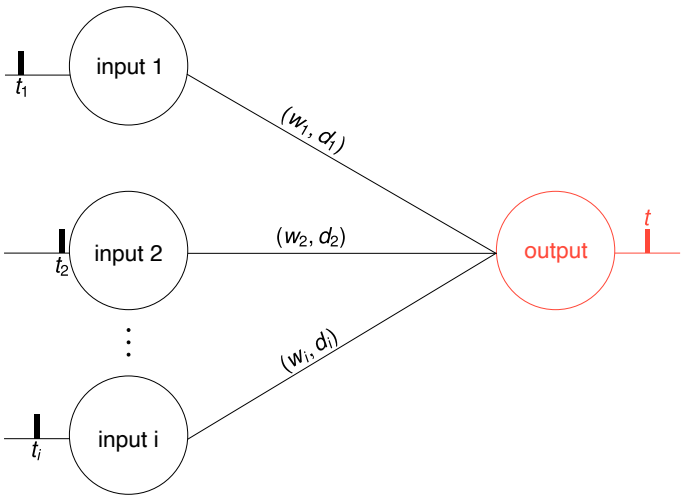


Fig. 2.1 *Leaky integrate and fire model. The presynaptic input neuron sends a spike to the postsynaptic output neuron at time t_i, via a synapse i characterized by a weight w_i and a conduction delay d_i. The output neuron receives the spike at time s_i, which is then integrated into the membrane potential $V(t)$. Once $V(t)$ passes the membrane threshold V_{th}, the output neuron fires, its membrane potential is reset and the neuron goes through a refractory period.*

Drawing inspiration from the myelin plasticity discussed in section 2.1 and from previous work on delay shifts [172, 176, 94], we will develop in this section an algorithm tailored for extracting temporal features by modulating conduction delays. The proposed model uses gradient descent dynamics to synchronize spikes emitted by pre-synaptic neurons, by adjusting delays on the most recently active synapses within an experimentally set temporal window (Fig. 2.2). Whenever a neuron fires, mutual inhibition is used to ensure

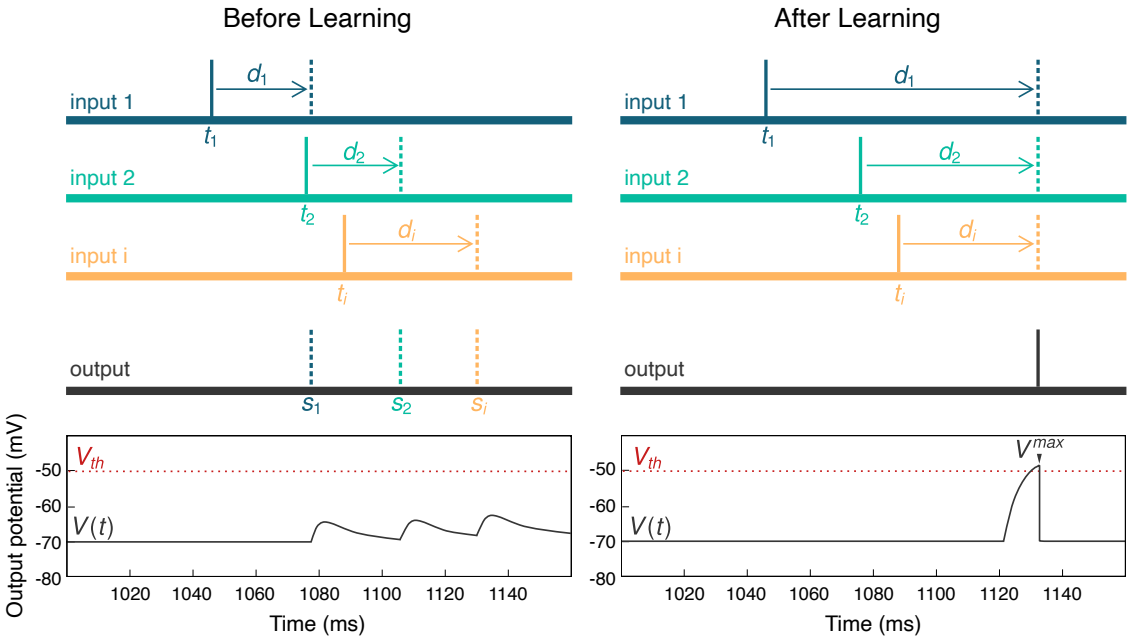


Fig. 2.2 *Delay learning rule. (**Left**) State of the SNN before learning delays. An input neuron emits a spike (vertical bar) from a synapse i at time t_i. The spike is received by the output neuron at time s_i after a delay d_i. The output neuron in this case does not fire because $V(t) < V_{th}$ represented here by the red dashed line. (**Right**) State of the SNN after learning delays. Delays are gradually modulated to make the input neurons fire concurrently. The output neuron's membrane potential is maximised, and reaches a value $V^{max} \geq V_{th}$.*

that neurons specialise to a particular temporal pattern. The delay plasticity model works in conjunction with leaky integrate and fire (LIF) neurons described by the following differential equation:

$$\tau_m \cdot \frac{dV(t)}{dt} = E_L - V(t) + R_m \cdot I(t) \tag{2.1}$$

where R_m is the membrane resistance, and τ_m is the membrane time constant. The LIF model, one of the simplest ways to describe neuronal activity, was chosen based on its ability to asynchronously update the state of a neuron using the timestamp of input events. Each spike triggers a current injection that gradually increases the membrane potential of a neuron. Once the membrane potential reaches a certain threshold, the neuron fires and propagates a spike throughout the network via synapses characterized by a weight w and a delay d. The neuron's potential is then reset back to its resting state.

We chose an exponential excitatory post-synaptic current (ESPC) shape such that the input current $I(t)$ at time t is:

$$I\left(t\right) = I_{inj} \cdot \sum_i w_i \cdot e^{-\frac{t-s_i}{\tau_{syn}}} \cdot \mathcal{H}\left(t - s_i\right) \tag{2.2}$$

where I_{inj} is the injected current every time a neuron fires, w_i is the synaptic weight of synapse i, τ_{syn} is the synaptic time constant, $\mathcal{H}(t)$ is the Heaviside step function, and s_i is the time of arrival of a spike, from a presynaptic neuron at time t_i, to a post-synaptic neuron with a delay d_i, such that $s_i = t_i + d_i$.

When we study the dynamics of a single synapse i, we remove the discontinuities caused by the input signal by focusing on the range $[s_i, t]$ where $\mathcal{H}(t) = 1$. Assuming initial conditions such that $V_i(s_i) = E_L$, we are restricting the network to only one spike per synapse. The membrane potential is reset between each training example through a winner-takes-all (WTA) algorithm and the membrane equation now has a solution:

$$V_i(t) = E_L + \frac{R_m \cdot I_{inj} \cdot w_i \cdot \tau_{syn}}{\tau_m - \tau_{syn}} \cdot \left(e^{-\frac{t-s_i}{\tau_m}} - e^{-\frac{t-s_i}{\tau_{syn}}} \right) \tag{2.3}$$

The time course of the potential follows a bi-exponential model with a finite rising time. In order to maximise the membrane potential of a post-synaptic neuron – and ultimately associate it with a particular temporal pattern – we compute the gradient of the neuron's potential $\frac{\partial V(t, s_i)}{\partial s_i}$ and modulate d_i until all spikes are aligned. The model assumes only one spike per synapse because the precise timing of each spike is considered to hold relevant information. The partial derivative of $V(t, s_i)$ with respect to s_i can then be written as:

$$\frac{\partial V(t, s_i)}{\partial s_i} = \frac{R_m \cdot I_{inj} \cdot w_i}{\tau_m - \tau_{syn}} \cdot (e^{-\frac{t-s_i}{\tau_m}} - e^{-\frac{t-s_i}{\tau_{syn}}}) \tag{2.4}$$

The delay update rule can be represented by the following equation:

$$d_i^{t+1} = d_i^t + \eta \cdot \frac{\partial V(t, s_i)}{\partial s_i} \tag{2.5}$$

where η represents the learning rate of a neuron, with $\eta > 0$. We decay the learning rate across iterations to avoid large gradient steps. The myelin plasticity model can easily be adapted into a supervised learning algorithm by aligning input spikes at a desired time via a teacher signal, instead of aligning the spikes to the last arriving input s_i.

2.3 Validation experiments

We decided to start by classifying randomly generated one-dimensional spike patterns. We wanted to confirm the algorithm's ability to learn time dynamics by specialising a temporal pattern to an output neuron. In that regard, we generated four different patterns consisting of ten presynaptic neurons firing once, at different times within a 20ms time window. Each pattern was presented 100 times to a single-layer SNN with ten input (presynaptic) neurons and four output (postsynaptic) neurons, one for each of our classes. To make the problem more complicated, up to 2ms of time jitter was added to the pattern.

We used biologically plausible values for all network parameters, with a membrane threshold set to -50mV, a refractory period of 3ms, and a resting membrane potential equal to

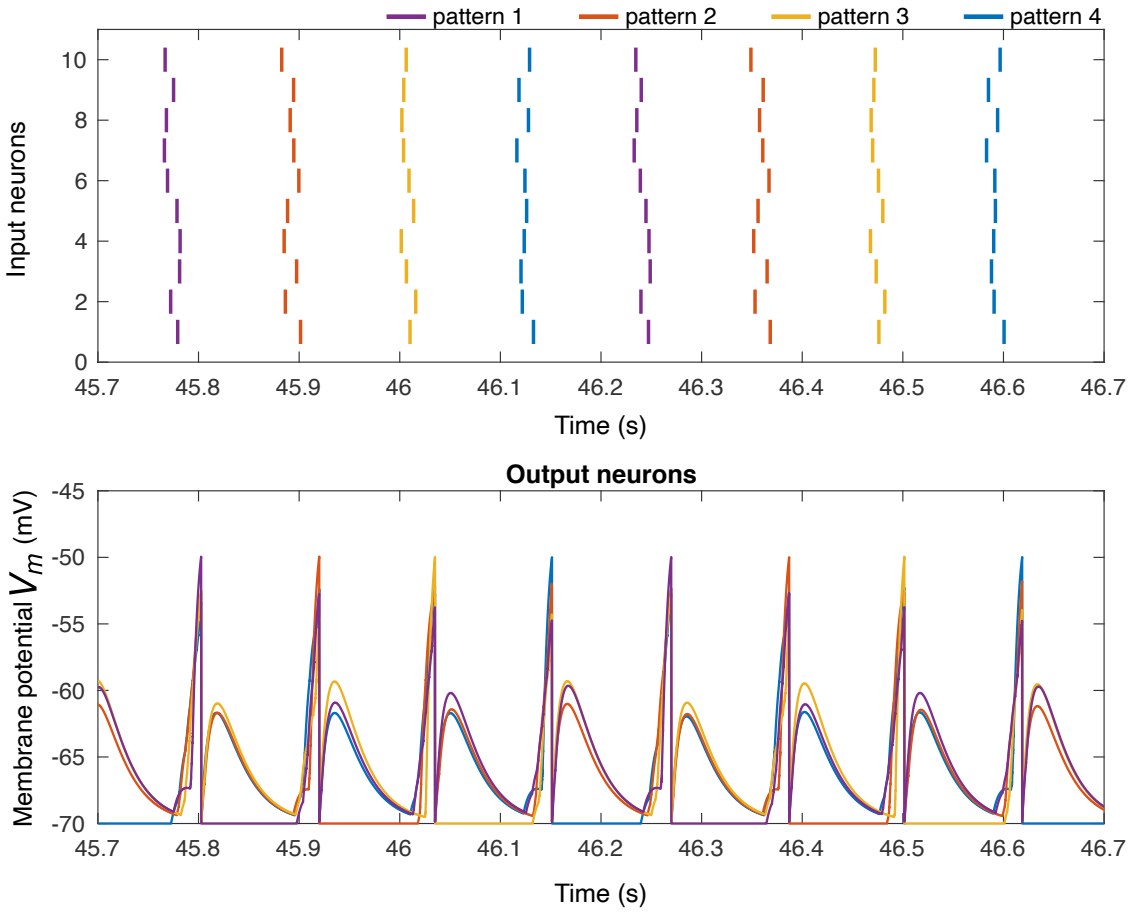


Fig. 2.3 *Myelin plasticity learning rule on a one-dimensional pattern recognition tasks consisting of four different patterns, each with ten input neurons that fire at different times within a 20ms time window.* ***(Top)*** *Raster plot representing the four patterns in different colours.* ***(Bottom)*** *Membrane potential V_m for the four output neurons after training with the myelin plasticity rule. Each output neuron has maximised its membrane potential to a particular temporal pattern.*

-70mV. We set the membrane resistance $R_m = 1\text{M}\Omega$ for simplicity. The membrane time constant is set to $\tau_m = 25$ms, the synaptic time constant to $\tau_{syn} = 10$ms, and the injected current is set to $I_{inj} = 80$nA. Finally, we keep the learning rate at $\eta = 1$.

Fig. 2.3 shows a raster plot of each of the four patterns, represented in different colours for better visibility. After learning, each postsynaptic neuron responds optimally to a specific pattern as seen by the membrane potential plots. This is largely due to the gradient ascent dynamics of the myelin plasticity algorithm. Fig. 2.4 shows the peristimulus histograms (PSTH) for the postsynaptic neurons that specialised to the first and second pattern before and after learning. Peristimulus time histograms allow us to visualise the timing and firing rate at which neurons fire in relation to a reference event, which in our case is the firing time of the postsynaptic neuron. More specifically, the input neurons are binned according to the time difference from the reference timestamp. Before learning, the input neurons

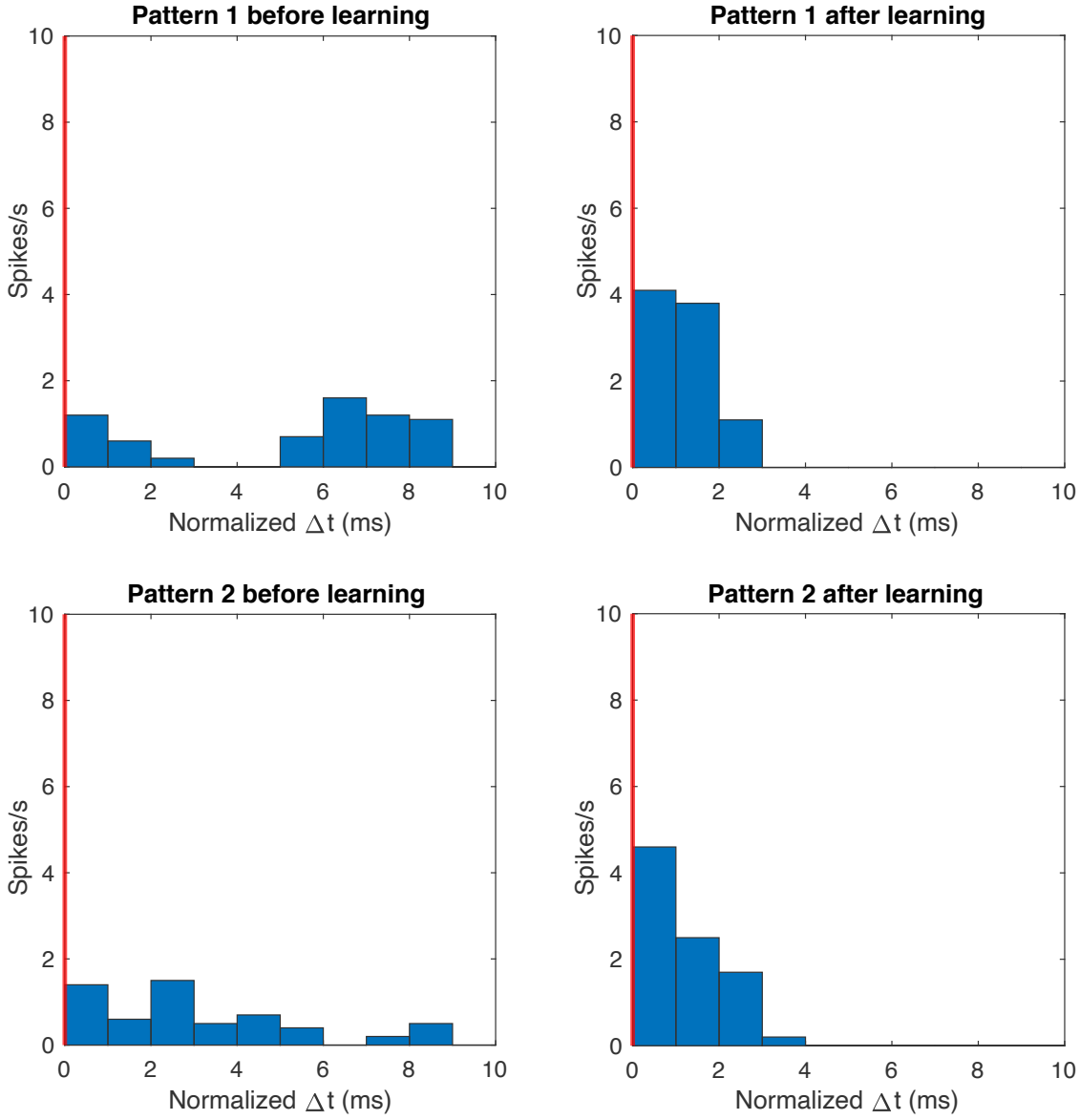


Fig. 2.4 *Behaviour of output neurons with myelin plasticity learning. For simplicity, we only show two out of four output neurons that specialised for the first and second patterns. The plots represent a peristimulus time histogram (PSTH) averaged across the first and last* 10 *output neuron spikes before and after learning, and tell us about the time at which the input neurons are firing. Neurons are binned according to the time difference from a reference timestamp (vertical red line) which is chosen to be the postsynaptic neuron's firing time. Before learning, neurons are firing at a similar firing rate throughout the range of time differences (up to* 10*ms). After learning, most input neurons are firing within 4ms of each other.*

are firing at a similar rate across all time difference bins. After multiple repetitions of the patterns, learning is turned off for cross-validation purposes. The PSTH after learning has a higher firing rate for input neurons that spike within 4ms of the reference time, indicating a lower variability in the spike times. The time differences therefore converge towards zero due to the myelin plasticity learning rule, and each output neuron only responds to a particular pattern.

2.4 Touch localisation with precise timing

In terms of information transmission strategies, the advantages of rate coding have been thoroughly explored in the past decade [177, 178]. Rate coding makes use of the mean firing rate of spiking neurons to encode continuous values, with individual spikes being emitted according to a random Poisson distribution [179]. This differentiable output is compatible with conventional artificial neural networks which are widely considered to be the gold standard in terms of pattern recognition tasks. Biological systems also use the precise timing of spikes to convey information [137, 138, 180, 139]. This strategy called temporal coding is associated with fast visual processing [46], sound localisation [162, 181], fine motor control [182], arbitrary nonlinear function approximation [183, 184], and even for temporal pattern classification [185, 186, 187]. Considering performance in pattern recognition tasks is usually based on classification accuracy and ignores latency and computational efficiency, the context in which temporal coding tasks can outperform their rate-based equivalent are a topic of current research.

Sand scorpions have the ability to accurately localise their prey using temporal information based on vibrations from the environment that are detected using sensory organs situated on their legs [188]. In this section we want to show that precise timing can be useful by presenting a practical example based on the sand scorpion. The task consists in localising the source of a vibration caused by tapping on a surface, via the spatio-temporal pattern detected by an array of sensors [189]. More specifically, we want to assess the myelin plasticity model's ability to pick up both the direction and distance of the source from the sensor device.

2.4.1 Methodology

As part of the data collection process, we rely on a neuromorphic tactile sensor consisting of 8 piezoelectric accelerometers arranged in a regular octagon pattern (Fig. 2.5B). Each accelerometer is connected to a microcontroller in charge of converting the analog signal into events via level crossing. The dataset collected with this electronic device consisted of 10 repetitions of 32 different stimuli elicited to tapping on a wooden surface from 8 different directions, each separated by a 45° angle, and 4 different distances from the sensor (200, 400, 600 and 800mm). Since wood does not attenuate the signal as much as sand, a high temporal resolution becomes a requirement in order for precise timing-based algorithms to accurately mimic the localisation abilities of a sand scorpion.

The tactile sensor itself solves the localisation problem using an analytical solution that is explained in detail in [189]. Indeed, while other solutions exist, we address the localisation

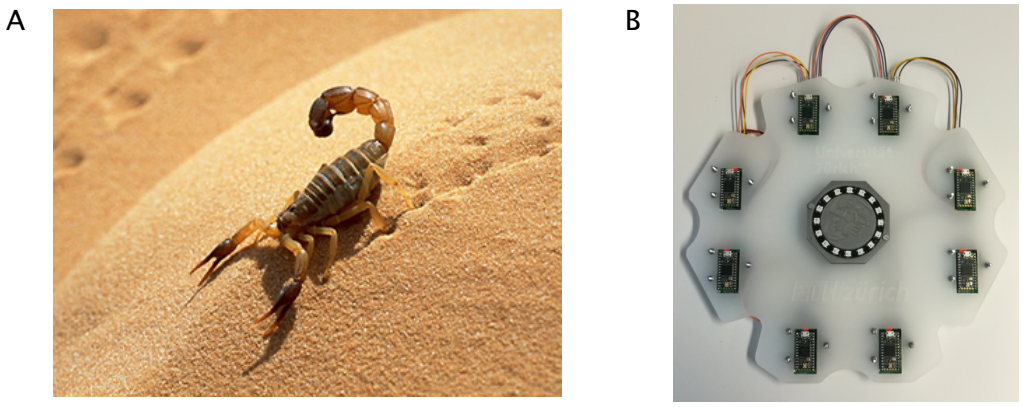


Fig. 2.5 ***(A)*** *Image of a sand scorpion.* ***(B)*** *Touch localisation prototype [189].*

task by applying three constraints to our model, all of which are satisfied by the myelin plasticity algorithm:

1. Only information which is local to a given pre-post synaptic neuron pair is used to update synaptic parameters.
2. No a priori knowledge of the sensor system is required.
3. The model parameters must be updated in an unsupervised manner.

2.4.2 Results

The myelin plasticity network consists of eight pre-synaptic neurons, sparsely connected in a random fashion to 50 LIF neurons. Sparsity is achieved by limiting the number of connections towards a LIF neuron to $N = 4$. Synaptic delays are randomly initialized according to a normal distribution with a mean of $\mu = 0.5\,\text{ms}$, and a standard deviation $\sigma = 0.3\,\text{ms}$ and a fixed weight equal to $\frac{w_0}{N}$ with $w_0 = 1$. The resting potential is set to $E_L = -70\,\text{mV}$. The LIF neuron's decay constant is set to $\tau_m = 2\,\text{ms}$, the synaptic time constant is set to $\tau_{syn} = 1\,\text{ms}$ and the injected current I_{inj} is set to $80\,\text{nA}$ to make sure that each presented spike train is capable of causing a LIF neuron to fire. The learning rate starts at $\eta = 1$ and decays by 10% after every 100 input spike trains to help the network converge towards a local minimum.

Each postsynaptic neuron that responds starts specialising to a particular pattern by synchronising its input spikes through a change in synaptic delays following Eq. 2.5. The winner-take-all mechanism ensures that no other postsynaptic neurons synchronise their input spikes. With each subsequent presentation of the pattern, the time differences between input spikes gradually converge towards zero (Fig. 2.6A).

With a temporal resolution of 1ms, the myelin plasticity model successfully recovers all eights angles with a 100% accuracy, as each angle is represented by at least one LIF neuron (Fig. 2.6B). While previous implementations of learned delays relied on an all-to-all connectivity scheme [172], the myelin plasticity model can obtain similar performances with fewer connections through a randomly connected sparse network (Fig. 2.7). Additionally,

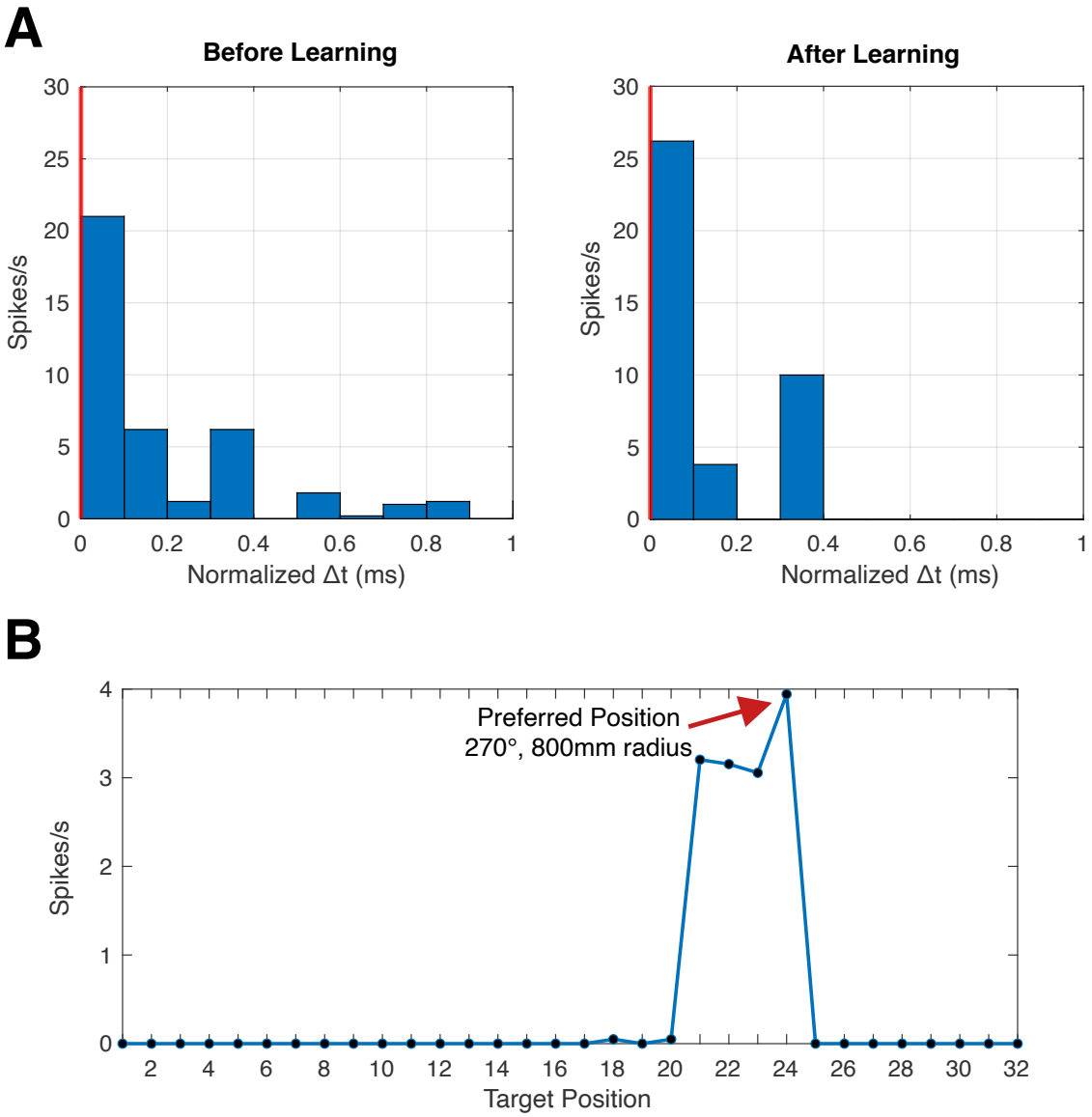


Fig. 2.6 *Behaviour of a postsynaptic neuron with myelin plasticity.* ***(A)*** *Peristimulus time histogram averaged across 50 data points at the beginning and towards the end of the simulation. The vertical red line represents the postsynaptic neuron's firing time, chosen as a reference from which the time difference is calculated. After learning, more presynaptic neurons fire with a lower time difference compared to the postsynaptic firing time, due to the synchronisation of input spikes.* ***(B)*** *Tuning curve of the postsynaptic neuron averaged across all the data points where the neuron fired, linking the firing rate to each of the 32 different stimuli positions. Positions 21 to 24 correspond to the distances 200, 400, 600 and 800mm respectively, at a 270° angle [189].*

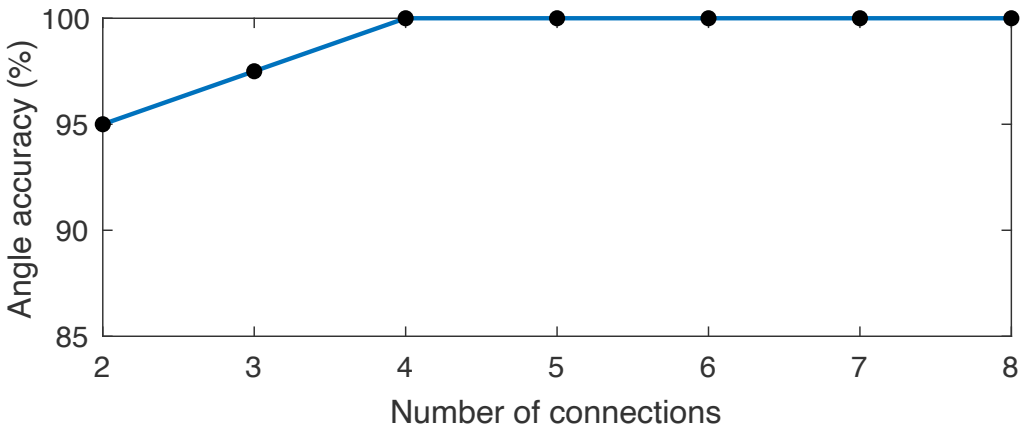


Fig. 2.7 *Average angle accuracy for an increasingly sparse synaptic delay network. We connected each LIF neuron to a random subset of 8 pre-synaptic neurons representing the hardware sensors. We varied the size of the subset across 35 different trials in order to assess the smallest number of connections between pre-synaptic and LIF neurons capable of preserving an accurate temporal representation of all 8 angles. A sparse synaptic delay network with only 2 connections per LIF can already represent all angles with an average accuracy of 95% and we can achieve 100% accuracy with only 4 connections [189].*

the network successfully represents all eight angles with a 95% accuracy when the number of connections per postsynaptic neuron are reduced even further to $N = 2$, but at least four connections are required for best performances. In contrast to delay learning models with fully-connected networks, a sparsely connected neural network with random connectivity, in addition to being more efficient to simulate, increases robustness against systematic noise originating from faulty sensors.

For each angle, the next step in touch localisation is to determine whether the myelin plasticity model can differentiate between stimuli at different distances. By looking into the average firing rate of postsynaptic neurons for each of the 32 positions, we can identify individual neurons that respond more frequently to specific positions after learning with the myelin plasticity model (Fig. 2.6B). However, a more in-depth analysis shows that neurons specialising to a particular angle have similar membrane potentials across distances, and can therefore respond to more than one position. While a preferred distance can be inferred by analysing the average firing rates across multiple repetitions, it cannot be trivially obtained in an online manner unless the WTA scheme is disabled after learning is done, and a supervised "voting" process is implemented. Indeed, the temporal signatures across distances are not significantly different due to the slow attenuation of the waves being measured. A simple solution involves recording stimuli on a surface with higher attenuation such as sand. Adding sensors dedicated to measuring the amplitude of a signal would also be beneficial for estimating the distance to the stimuli, assuming the same pressure is used to elicit the taps.

In [189] we compare the myelin plasticity model to an analytical solution as well as four different models based on spiking neural networks that classify spatio-temporal patterns

using both supervised and unsupervised learning rules. All of the approaches are able to distinguish between the angles, but the distances remain an open issue. Most strategies require supervision or knowledge of the geometry of the sensor system with the exception of the unsupervised structural plasticity model which pairs an STDP for synaptic weights with an STDP for synaptic time constants. However, the reliance on two STDP learning rules makes it quite expensive to simulate compared to the myelin plasticity model. My personal contribution to this published work involves solving the touch localisation task using myelin plasticity.

Table 2.1: Touch localisation with various models based on precise timing [189].

Method	Angle accuracy (%)	Distance accuracy (%)
Analytic solution	99.7	73.4
Temporal coincidence	100	37.5
Complex weights and delays	100	100*
Temporal difference encoders	100	N.A.
Myelin plasticity	100	N.A.
Structural plasticity	100	N.A.

(*) The complex weights and delays method requires an external linear classifier to successfully differentiate between distances.

2.5 Discussion

We presented a myelin plasticity model for learning spatiotemporal patterns in an unsupervised manner. This method relies on a gradient ascent approach which maximises the membrane potential of postsynaptic neurons by aligning spikes originating from the previous layer. After a few iterations, and with the help of a winner-takes-all mechanism, the postsynaptic neurons end up specialising to a particular pattern.

Similar delay learning rules that are based on the principle of coincidence detection such as the Deltron [172] and the Bayesian approach explored by [175] relied on fully connected networks. Instead, we use a partially connected network with random connectivity that exhibits an equivalent or better performance but with fewer connections, making it more suitable for hardware implementations. The increased sparsity can **(i)** reduce the complexity of the network by having a lower number of synapses to update, **(ii)** improve efficiency if implemented on analog or mixed signal hardware [174, 190], **(iii)** decrease the overall training time, and finally **(iv)** partial connectivity improves generalisation by learning local features instead of synchronising spikes from all input connections as is the case in the fully connected network topology.

The current-based LIF model with double exponential synapses does not have an analytical solution and therefore cannot behave in a fully event-based manner. As an improvement, we can use a box-shaped excitatory post-synaptic current instead of the double exponential model to eliminate the time dependency of the input current. This essentially removes the decay term which greatly simplifies the membrane equation solution, giving rise to an event-based LIF neuron with complex current dynamics. In order to solve more

complicated classification tasks with the myelin plasticity model, a weight-based synaptic plasticity rule needs to be combined with the delay learning rule in order to be able to extract spatiotemporal patterns from extremely noisy signals instead of just synchronising spikes. The benefits of using both delays and weights for pattern recognition was first outlined in [114] with the theory of polychonisation, and then further evidenced by [175]'s work on probabilistic delay learning. In our case, implementing a resource-dependent synaptic plasticity [191] to work alongside the myelin plasticity rule seems to be the logical step forward for Hebbian learning in SNNs.

Chapter 3

Bio-inspired learning in a low-power visual data processing system

3.1 Introduction

Developing systems that are able to process the high temporal resolution information output by event-based sensors without consuming much energy is a challenging yet important task. In addition to lasting longer, energy-efficient systems have the potential to enable advanced visual data processing in power-limited environments such as a car.

The latest high definition event-based cameras [26, 72] can generate millions of events per second. When dealing with such a high bandwidth, the sub-millisecond temporal resolution and power efficiency of event-based systems quickly become limited by the Universal Serial Bus (USB) protocol used to transfer data to a computer for processing with computer vision and machine learning algorithms [192, 193]. An obvious solution consists in converting the events into dense frames through one of the techniques discussed in section 1.3.1. Artificial intelligence hardware accelerators can then be leveraged for efficient inference with pre-trained deep learning models, losing in the process most of the benefits of neuromorphic vision sensors [143, 144]. We can circumvent this problem and work in an event-based manner by directly interfacing the neuromorphic vision sensor with a data processing CMOS architecture.

Spiking neural networks (SNN) are the natural choice for dealing with streams of events, especially as they are built to be resilient to noise which can be quite prevalent in neuromorphic vision sensors. Their massively parallel structure and recurrent connectivity is however, very costly to simulate on CPU processors. The main bottleneck of the Von Neumann architecture, contributing towards a higher power consumption and a reduction in processing speed, is a result of the separation between the processor and the memory [194, 147]. While GPU-based solutions are much better at simulating SNNs [195], they ultimately suffer from the same memory access bottleneck [196, 197]. Looking into biological neural networks for inspiration, dedicated neuromorphic architectures with an event-based communication protocol and co-located processing and memory units can overcome the limitations of the Von Neumann architectures and have the potential to

outperform classical computing in terms of edge AI [147, 198]. These architectures are the ideal candidates for extremely low power and highly scalable hardware implementations of SNNs [117].

The goal of the *Ultra-Low Power Event-Based Camera* (ULPEC) project, presented in this chapter, is to develop a visual data processing system with ultra-low power requirements and ultra-low latency, aimed at the recognition and classification of traffic events towards autonomous driving. Indeed, when it comes to self-driving cars, designing an energy-efficient system-on-chip (SoC) becomes all the more necessary due to the limited power available in the car. This can be achieved by reducing the latency and processing the information closer to the sensor. The system consists of an event-based vision sensor connected to a mixed signal hardware implementation of an SNN with solid-state synapses called memristors [199, 200]. The memristor is a passive two-terminal component with a variable conductance that can be adjusted by injecting voltage pulses. Simply put, memristors can be thought of as plastic synapses where the conductance is essentially the synaptic weight, or in other words, the strength of a connection between two nodes. We can take advantage of this property and evolve the memristor conductance according to the biologically plausible Hebbian spike-timing-dependent plasticity (STDP) learning rule [201], bringing about a self-adapting electronic architecture capable of unsupervised learning without any external control on the synaptic weights. In fact, the synaptic strength is purely determined by the timing and coincidence of events originating from the neuromorphic vision sensor.

The ULPEC project is the outcome of a consortium of three universities, Sorbonne University, the University of Bordeaux, and the University of Twente, the CNRS research institute, and technology companies including Prophesee, Bosch and IBM Research. This multidisciplinary work encompasses various fields such as material science, integrated circuit design, and machine learning, and is meant to strengthen Europe's leadership in miniaturised bio-inspired smart integrated systems. Designing the visual data processing system can be summarised into three main tasks that were handled by different collaborators according to the required expertise. For clarity, my contribution within the ULPEC project is highlighted in bold.

1. delivering the memristor technology capable of multiple resistance states
2. integrating memristors with CMOS technology on an industrial scale
3. **finding a suitable architecture and processing algorithm to implement the hardware neural network**

In order to test the ULPEC visual data processing system in a real-world situation, Prophesee created N-CARS, a two-class dataset consisting of 12336 car samples (Fig. 3.1) and 11693 background samples [1], each example having a resolution of 64×56 pixels and a duration of 100ms. The dataset was recorded from various driving sessions using an ATIS event-based camera [29] mounted behind the windshield of a car. A newer, much larger version of the dataset was recently made available [202], adding further complexity to the task at hand.

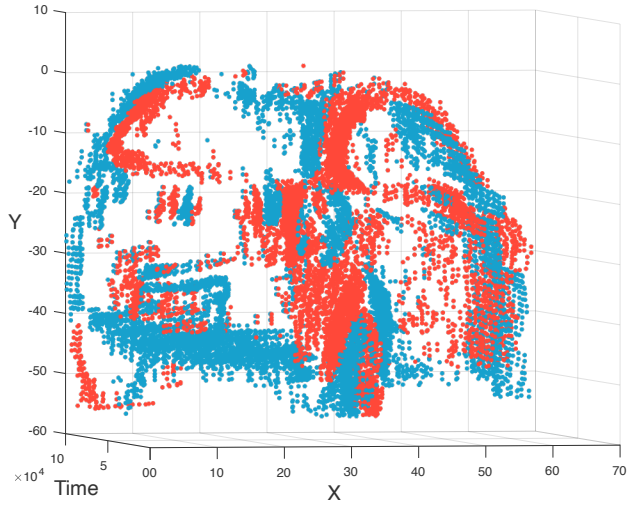


Fig. 3.1 *Example of a car from the N-CARS dataset [1]*

3.2 The memristor: fourth fundamental circuit element

The memristor, short for memory resistor, is a passive two-terminal electrical component that is considered to be the fourth fundamental circuit element alongside the resistor, capacitor and inductor [199]. The memristor was originally hypothesised to link the magnetic flux Φ to the electrical charge q. More intuitively, its behaviour can be expressed in terms of voltage and current according to the following equation:

$$V = M(q) \cdot I \tag{3.1}$$

where the memristance M represents the resistance that varies depending on the history of the electrical charge passing through the device. In this device, the relationship between voltage and current is highly non-linear and is often described by a pinched hysteresis loop [203, 204]. The definition of a memristor was later expanded to include all two-terminal non-volatile memory devices that exhibit resistance switching properties [205]. In a nutshell, the resistance of a memristor can be programmed by applying voltage pulses, giving rise to interesting use cases such as a nano-scale low power alternative to flash memory [206] and an artificial synapse with adjustable weights that can be used for neuromorphic computing applications [201]. Memristors can be categorised into two groups: (i) molecular and ionic thin film memristors, and (ii) purely electronic memristors [207, 208].

Molecular and ionic thin film memristors: This category includes resistive random-access memory devices (ReRAM) and other kinds of memristors which are built with a metal-insulator-metal (MIM) configuration [200, 209]. By applying positive or negative voltage pulses, these devices can switch between a high resistance (OFF) and low resistance (ON) state through the formation or breakdown of conductive filaments in the insulator layer between the two terminals. Our collaborators at the CNRS and the university of Twente did not consider this particular type of memristors for the ULPEC visual data

processing system due to the devices' reliance on significant material structural changes and defects which can cause reliability and endurance issues [210, 211]. Stable resistance states need to be guaranteed in order for memristors to properly emulate synaptic plasticity mechanisms, towards unsupervised learning in hardware SNNs.

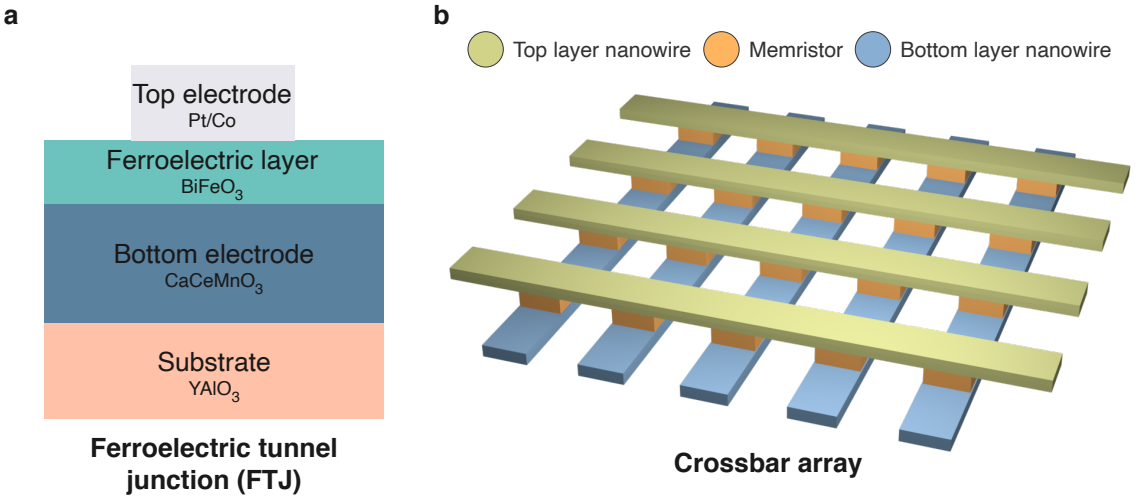


Fig. 3.2 *FTJ based memristor design.* ***(a)*** *Cross-section of a FTJ consisting of a nanoscale ferroelectric thin film surrounded by two metal electrodes. The polarisation of the ferroelectric material can be flipped with the application of a voltage pulse, leading to resistance switching behaviour [212]. Here, a $YAlO_3$//$CaCeMnO_3$//$BiFeO_3$ FTJ stack is shown, however, other substrates and electrode materials have also been used in the literature [213].* ***(b)*** *Diagram of a memristive crossbar array which can be integrated into CMOS to realise a SoC device such as the ULPEC visual data processing system.*

Purely electronic memristors: The two main types of memristors in this category are spintronic memristors [214] and ferroelectric memristors based on ferroelectric tunnel junctions (FTJ) [204, 212]. Resistance switching is mediated entirely through electronic effects and does not rely on structural changes, potentially resolving the reliability, endurance and speed of switching issues seen in molecular and ionic thin film memristors. Spintronic memristors use magnetisation to change the spin direction of electrons. While interesting, the technology needs more time to mature before being used in realistic scenarios [214]. FTJs on the other hand, consist of a bottom and a top metal electrode separated by a nanoscale ferroelectric crystalline thin film grown on a substrate (Fig. 3.2a). The thin film is made of nanoscale ferroelectric domains, each with its own polarisation state. An up domain corresponds to a low resistance ON state, and a down domain indicates a high resistance OFF state. In a ferroelectric memristor, in addition to the typical ON or OFF configuration, intermediate resistance states can be stabilised by reversing the polarisation of an increasing number of ferroelectric domains depending on the amplitude of the voltage pulses [204]. This process can be seen in Fig. 3.3 through piezoresponse force microscopy (PFM) images which show the percentage of down domains with respect to the resistance state.

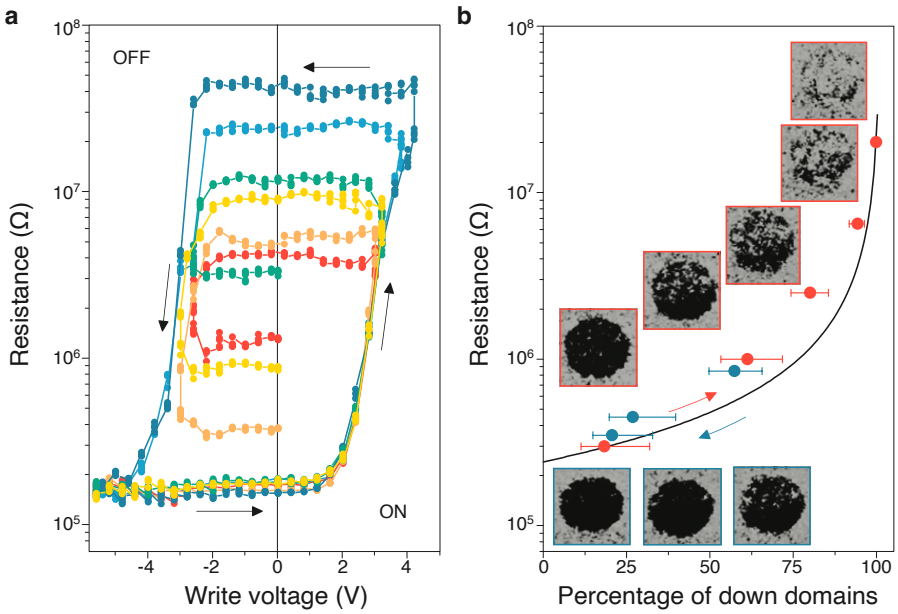


Fig. 3.3 *Resistance switching in ferroelectric memristors.* ***(a)*** *Multiple intermediate resistance states can be reached between the ON and OFF configurations by applying voltage pulses of different amplitudes.* ***(b)*** *PFM phase images showing the polarisation state of the nanoscale ferroelectric domains under the different resistance states. Dark ferroelectric domains are in a low resistance state. Adapted from [204].*

Ferroelectric memristors have several advantages, including large ON/OFF ratios of up to 10^4, switching between resistance states in under 10ns, low operation energy as well as high endurance and reliability [212, 201, 204, 215]. Furthermore, ferroelectric memristors have proven capable of evolving according the Hebbian STDP unsupervised learning mechanism [201], making them suitable for developing self-evolving neuromorphic architectures such as the ULPEC visual data processing system. Moving beyond a single device, ferroelectric memristors can be integrated into a large scale crossbar (Fig. 3.2b), allowing them to regulate the electrical conductivity between two layers of nanowire which can be connected to CMOS technology through various wafer bonding strategies (e.g. flip-chip bonding). From a scalability perspective, epitaxial growth on silicon instead of other single crystal substrates combined with a direct wafer bonding approach, paves the way for industrial scale back-of-the-line integration with microelectronics owing to compatibility benefits. This challenging task is under active development by the scientific community [216, 217] and our collaborators within the ULPEC consortium.

3.3 System and network architecture

3.3.1 General system on chip design

To come up with a suitable architecture for the visual data processing system and to easily assess learning performance, system-level simulations are done using Hummus, an

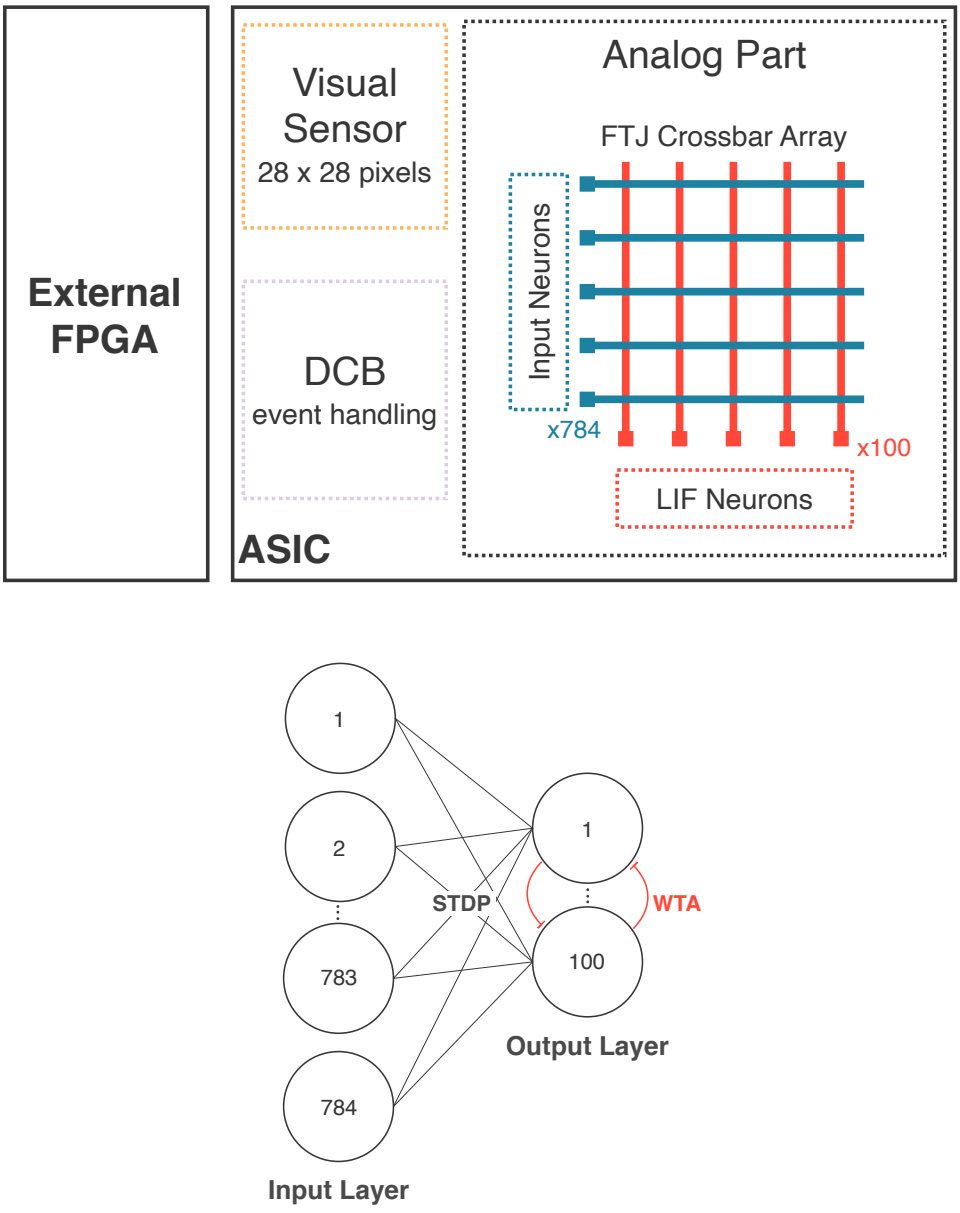


Fig. 3.4 *General architecture of the ULPEC visual data processing system.* ***(Top)*** *The SoC includes an event-based vision sensor, the FTJ crossbar array which connects two layers of analog CMOS neurons, namely input and leaky integrate and fire (LIF) neurons, the digital control block (DCB) in charge of handling event and spike to and from the LIF neurons, and finally, an external FPGA that supplements the analog part with digital functions to further complexify the behaviour of the CMOS neurons.* ***(Bottom)*** *Single-layer all-to-all neural network architecture of the FTJ crossbar array.*

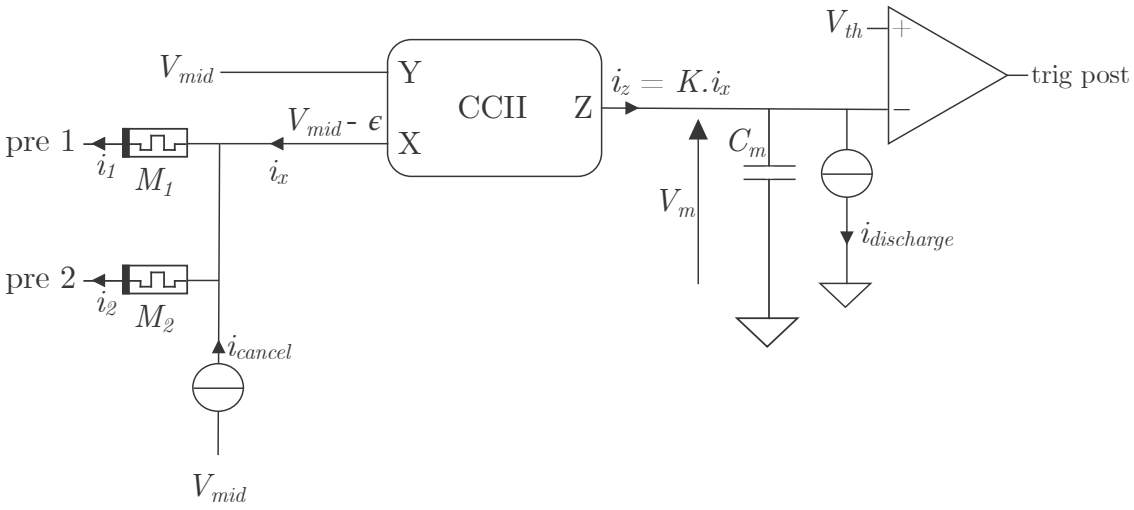


Fig. 3.5 *Circuit-level design of a postsynaptic LIF neuron. Active pixels on the event-based vision sensor drive their corresponding memristor, denoted as M, by applying a voltage pulse. The total contribution of active memristors, i_x is passed through the CCII [218], a current conveyor which scales down and forwards positive currents ($i_x > 0$) in order to charge the LIF neuron's capacitor C_{mem}. i_{cancel} represents the current offset which takes into account the non-negligible impact of inactive memristors. Inside the neuron, a constant leakage current $i_{discharge}$ is used to get the expected LIF behaviour. Taken from [117]*

.

event-based spiking neural network simulator developed in C++ throughout this thesis (see appendix A). The explored architectures follow a general design for the system which is outlined in Fig. 3.4.

The ULPEC system is an application-specific integrated circuit (ASIC) that includes several fixed-function blocks, each of which is designed to efficiently perform specific tasks. The chip includes a 28 × 28-pixel resolution event-based vision sensor [29] connected to a digital control block (DCB) in charge of implementing a winner-takes-all (WTA) mechanism and handling events and spikes to and from an analog neural network circuit. The network consists of an input layer of 784 pulse generators, one for each pixel, fully-connected to an output layer of 100 leaky integrate and fire (LIF) neurons via a crossbar array of FTJ memristors that adjust their conductance (synaptic weight) according to the Hebbian-based STDP learning rule (Fig. 3.4). The ASIC is coupled with an external FPGA for a more versatile solution, e.g. initialising the synaptic weights (memristor resistance states), using different input data types, using external classifiers and detecting misbehaving output neurons. The workflow of the ULPEC visual data processing system is the following:

1. The event-based vision sensor captures visual information in an asynchronous manner [29]. Each individual pixel is mapped to an input neuron on the FTJ crossbar array.

2. Events emitted by the sensors are sent to the DCB, generating a signal that causes the relevant input neurons to emit a voltage pulse used to drive the memristors. The sum of the currents in active memristors i_x is forwarded to the output LIF neurons via a second-generation current conveyor (CCII) in the case where i_x is positive. Forwarding a unipolar current avoids unnecessary discharge at the output i_z of the current conveyor. The capacitors C_m of the output neurons get charged in the process, leading to an increase in membrane potential on the output neurons (Fig. 3.5).

3. Whenever an output neuron fires, a feedback voltage pulse triggered by the DCB is applied to the relevant memristors which enter a writing phase that adjusts their conductance. This behaviour reproduces a simplified multiplicative STDP learning mechanism that depends on the previous state of the memristor.

4. A winner-take-all (WTA) mechanism implemented on the DCB is used to inhibit the other output neurons by resetting their potential, and its own, to zero. This lateral inhibition process promotes competition and prevents the postsynaptic neurons from learning similar patterns. This is further accentuated by the use of an event-based refractory period that deactivates a recently active output neuron until a set number of other output neurons have fired. The counters necessary to keep track of each output neuron's refractory period are implemented on the FPGA.

3.3.2 FTJ memristor model

a) The STDP learning rule

Pending availability of the exact memristor prototype that will be integrated into the chip, it was necessary to work with a qualitative model for FTJ memristors. To simplify this theoretical study, voltage fluctuations due to noise and cycle-to-cycle and device-to-device variations are ignored. We based the model on a $YAlO_3//CaCeMnO_3//BiFeO_3$ FTJ stack (Fig. 3.2). For this particular type of ferroelectric memristor, the threshold voltage beyond which resistance switching occurs, is roughly around 1V [201]. The conductance change of the memristors when submitted to a rectangular voltage pulse is reflected by the equation:

$$\Delta G = \begin{cases} A^+ \cdot (G_{max} - G_0) & \text{for } V_{syn} \leq -1.2V \text{ (LTP)} \\ A^- \cdot (G_0 - G_{min}) & \text{for } V_{syn} \geq +1.2V \text{ (LTD)} \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where G_0 is the initial conductance before the writing phase, G_{max} and G_{min} are the upper and lower values bounding the conductance, V_{syn} is the voltage applied on the memristors, and A^+ and A^- are experimentally set learning rates. This time-independent variant of the STDP learning rule introduced in the literature [219, 96, 117], is simple to implement on hardware. Whenever $V_{syn} \leq -1.2V$, the memristor enters a long-term potentiation (LTP) phase where the conductance is increased. Conversely, when $V_{syn} \geq +1.2V$, the memristor enters a long-term depression (LTD) phase where the conductance is decreased.

STDP is implemented using waveform superposition as shown in Fig. 3.6. After passing the membrane voltage threshold V_{th}, when an output neuron fires, it emits a feedback

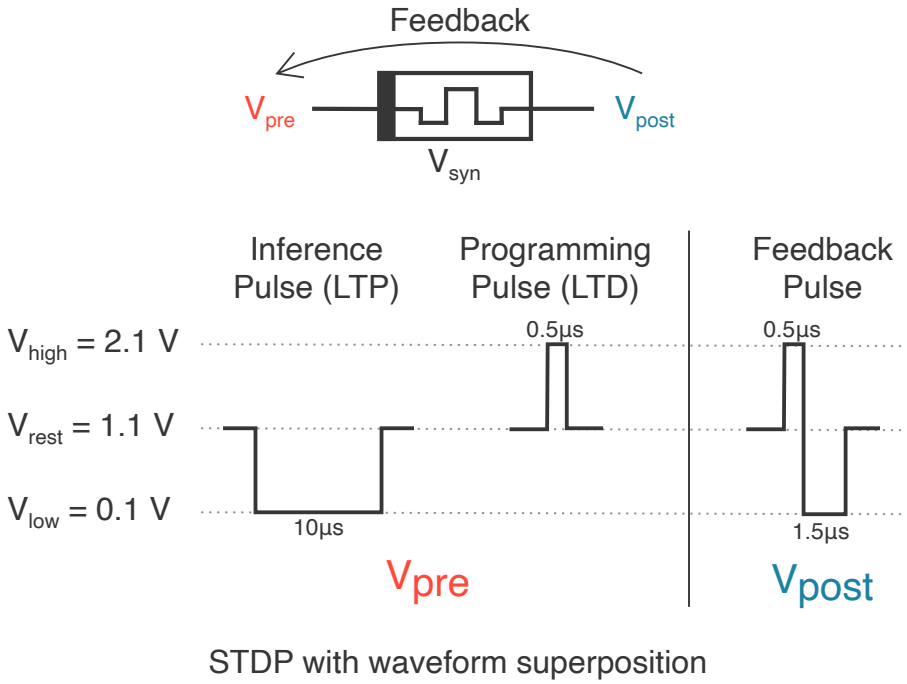


Fig. 3.6 *Overview of the demonstrator mechanism used to implement STDP. When a postsynaptic neuron fires it emits a feedback pulse towards the presynaptic neurons. An inference pulse is emitted by the presynaptic neuron when the corresponding pixel was recently active. A programming pulse is emitted otherwise. By overlapping the pulses originating from the presynaptic and postsynaptic neurons, the voltage applied on the memristors becomes enough to pass the potentiation and the depression threshold in the case of the active and inactive memristors respectively.*

pulse towards the presynaptic neurons. The input neurons themselves can generate two different pulses depending on the conditions:

- a 10µs inference pulse is emitted towards a memristor in response to an active pixel.
- a 0.5µs programming pulse is emitted towards inactive memristors once an output neuron fires

The size of the capacitor that can be realistically included in the system only allows for about 10µs of integration time. Taking that into consideration, the superposition of the inference pulse with the feedback pulse allows the voltage V_{syn} applied on the memristor to cross the LTP threshold, increasing the conductance of the active memristors. On the other hand, the LTD threshold can be crossed by generating a programming pulse and superposing it with the feedback pulse, leading to a decrease in conductance. This can be described by the following equation:

$$V_{syn} = V_{pre} - V_x \tag{3.3}$$

where V_{pre} represents the inference/programming pulse originating from the input neuron,

and V_x represents the feedback pulse emitted by the output neuron.

b) Validation

When looking at the evolution of the resistance with respect to the input voltage, memristive properties are usually verified by the presence of a hysteresis loop [199]. Looking at the conductance-voltage characteristic in Fig. 3.7 allows us to confirm the voltage-dependent resistance switching properties of our model as well as the parallel with STDP-based synaptic weight modulation. As a reminder, the conductance is inversely proportional to the resistance. By varying the amplitude of the voltage pulses, the memristor can reach different intermediate states. The number of resistance states a memristor can reach between the ON and OFF configurations determines the synaptic weight precision.

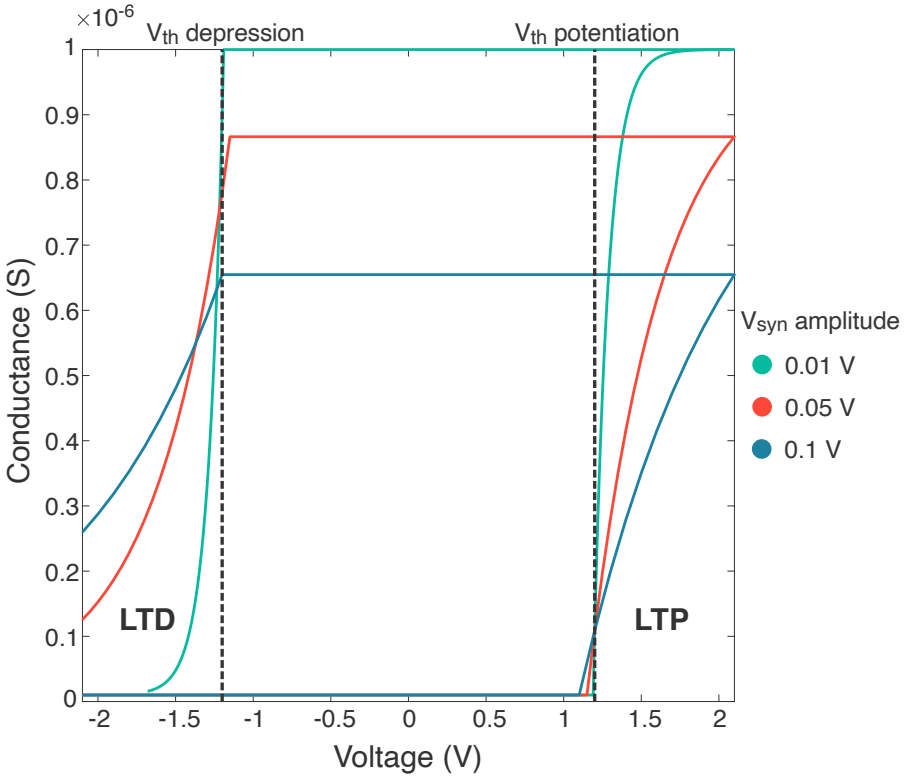


Fig. 3.7 *Voltage versus conductance curve of an FTJ memristor modeled with an in-house SNN simulator. The model shows a clear hysteresis loop, indicating memristive behaviour and a capacity for resistance switching. Driving memristors with* 10µs *long write voltage pulses of increasingly large amplitudes* V_{syn} *leads to multiple intermediate resistance states that can reproduce the STDP learning mechanism.*

3.3.3 CMOS leaky integrate and fire neuron model

The learning performance of the ULPEC system is greatly affected by the dynamics of the LIF neuron. It is then essential to accurately reproduce the CMOS neuron behaviour.

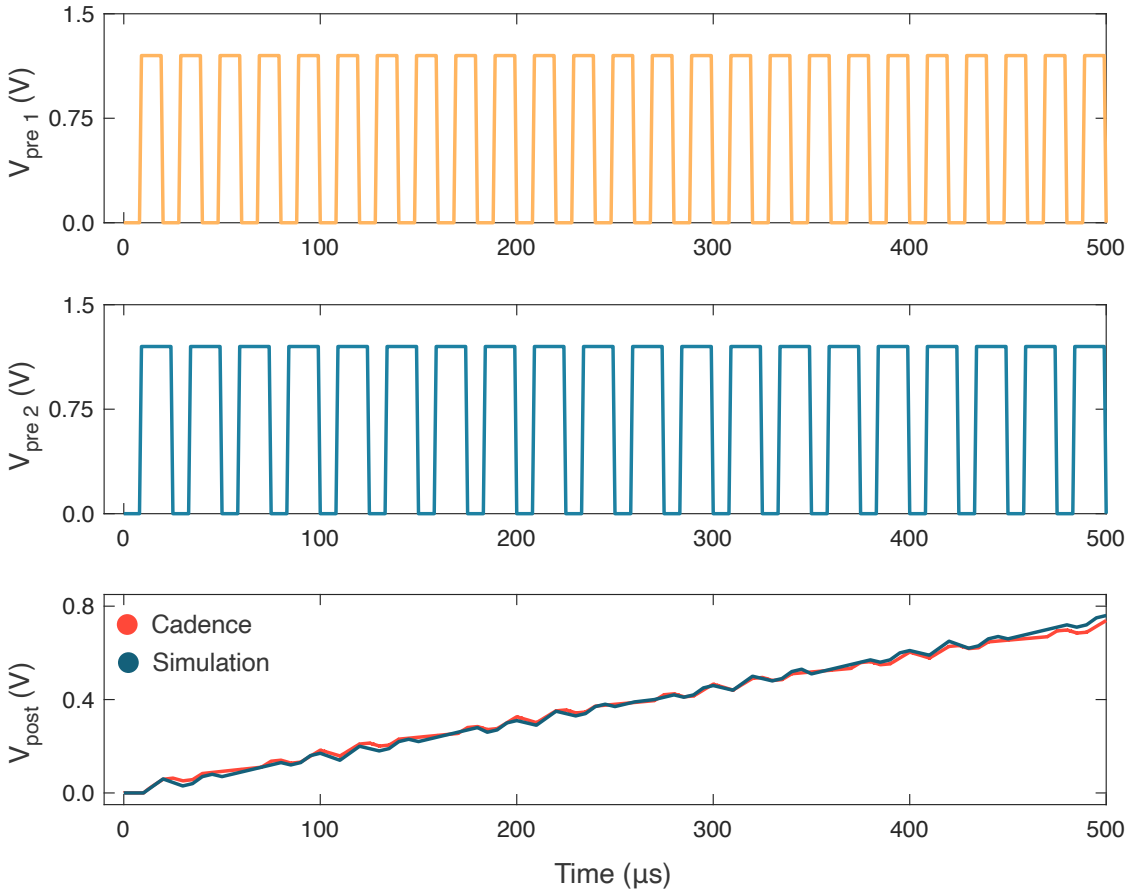


Fig. 3.8 *CMOS LIF neuron electrical simulations in Cadence, compared to a model developed with an in-house SNN simulator. Our model manages to accurately reproduce the hardware neuron. The voltage threshold was set to* $V_{th} = 1.2V$*, The memristors had a resistance* $R = 100k\Omega$*, the leakage current* $i_{discharge} = 12nA$ *and current scaling factor* $K = \frac{1}{725}$*. Cadence simulation data provided by Charly Meyer from IMS Bordeaux.*

Accordingly, we compare our model to electrical simulations done in Cadence, a software suite aimed at prototyping ASIC designs and simulating circuits (Fig. 3.8).

The current flowing from active memristors is sent through the current conveyor (CCII) to the neuron when $i_x > 0$ and the conveyor charges the post neuron membrane capacitor C_m by copying i_x, increasing the membrane potential V_m in the process. A constant leakage current $i_{discharge}$ is used to get the potential decay associated with LIF neurons. Due to the analog nature of the circuit, parasitic currents originating from inactive memristors are taken into account using a counterbalancing current i_{cancel}. With respect to the notations shown in Fig 3.5, the membrane potential can be modeled by the following differential equation:

$$\Delta V_m = \frac{K \cdot i_x \cdot \Delta t}{C_m} - \frac{i_{discharge} \cdot \Delta t}{C_m} \tag{3.4}$$

$$i_x = \sum_{j:active} G_j * V_j - i_{cancel} \tag{3.5}$$

$$i_{cancel} = \frac{\epsilon}{R_{network}} \tag{3.6}$$

where $R_{network}$ is the resistance of all inactive synapses and ϵ represents the voltage offset due to inactive synapses. For simplicity, we can assume $\epsilon = 0$ in our simulations without significantly altering the results of our model.

3.3.4 Choice of classifier

After training the neural network with STDP we explore two classifiers for inference. The first is a really simple heuristics-based classifier that is fully compatible with the design and low power budget of the ULPEC system [117]. The second is a stronger logistic regression classifier that requires training on FPGA, without compromising the low latency and energy-efficiency of the ULPEC system [220].

a) Heuristics-based classifier

After passing through one epoch of the training set, we label the output neurons using experimentally set parameters. If 50% of a neuron's last 10 spikes respond to the same class, then we label that neuron accordingly. Any neurons that do not specialise to a particular class are discarded.

b) Logistic regression classifier

With a logistic regression classifier, the workflow of the ULPEC visual data processing system is the following:

1. Train the network with one epoch of the training dataset via STDP
2. Start training the logistic regression as soon as the STDP starts to converge by counting the number spikes output by each LIF neuron for a particular data point (visual data presentation)
3. Use the fitted logistic regression model for real-time inference

Implementation To preserve the low latency in our simulations, we implement an online logistic regression capable of directly classifying spikes from the output layer while the STDP-based SNN is still running. In that regard, the C++ front-end of PyTorch, a popular machine learning framework, is tightly integrated into our own SNN simulator. The logistic regression classifier used in our simulations is basically a two-layer spiking network: (i) neurons in the first layers count the spikes output by each LIF neuron, resulting in a feature vector that is used to fit the classifier and for inference. (ii) the

second layer has one neuron per class and is simply used to represent the prediction outcome as a spike.

Mathematical description We train the logistic regression model using stochastic gradient descent on the data points [221, 222]. By adding the activity of neurons over time, we create features S_m that we can use to train a predictive model on labels for the input. The labels Y take values in $\{1, \ldots, K\}$, where K is the number of classes, denoting the class to which an input belongs. We model the probability of an input belonging to class k using the distribution $\hat{P}(Y)$ defined as:

$$\hat{P}(Y = k|W, b, S) = \frac{\exp\left(\sum_m W_{k,m} S_m\right) + b_k}{\sum_k \exp\left(\sum_m W_{k,m} S_m\right) + b_k} \tag{3.7}$$

to find the optimal values for the weight matrix W and bias b we minimise the negative log-likelihood:

$$\mathcal{L}(W, b) = -\sum \log\left(\hat{P}(Y = k|W, b, S)\right) \tag{3.8}$$

we can also use a regularisation term for W:

$$\mathcal{L}(W, b) = -\sum \log\left(\hat{P}(Y = k|W, b, S)\right) + \frac{\lambda}{2} \sum_{k,m} W_{k,m}^2 \tag{3.9}$$

where λ is a regularisation parameter. In order to minimize Eq. 3.9, we use stochastic gradient descent. Stochastic gradient descent iteratively optimises the data using the following update rules:

$$W_{k,m}^{new} = W_{k,m}^{old} - \eta \nabla_W \mathcal{L}(W, b) \tag{3.10}$$

$$b_k^{new} = b_k^{old} - \eta \nabla_b \mathcal{L}(W, b) \tag{3.11}$$

where η is the learning rate and the gradients $\nabla_W \mathcal{L}(W, b)$, and $\nabla_b \mathcal{L}(W, b)$ are given by:

$$g_W = \nabla_W \mathcal{L}(W, b) = \left[S_m \left(\sum_m W_{k,m}^{old} S_m - \delta(Y = k) \right) + \lambda W_{k,m}^{old} \right] \tag{3.12}$$

$$g_b = \nabla_b \mathcal{L}(W, b) = \left[\sum_m W_{k,m}^{old} S_m - \delta(Y = k) \right] \tag{3.13}$$

such that $\delta(Y = k) = 1$ when the labels $Y = k$ and is 0 otherwise. Stochastic gradient descent often leads to instabilities that can be ameliorated by averaging the updates over multiple data points S and Y. The update rules with averaging over mini-batches of size N then become:

$$\bar{g}_W = \frac{1}{N} \sum_n \left[S_m^{(n)} \left(\sum_m W_{k,m}^{old} S_m^{(n)} - \delta\left(Y^{(n)} = k\right) \right) + \lambda W_{k,m}^{old} \right] \tag{3.14}$$

$$\bar{g}_b = \frac{1}{N} \sum_n \left[\sum_m W_{k,m}^{old} S_m^{(n)} - \delta\left(Y^{(n)} = k\right) \right] \tag{3.15}$$

Using the stochastic gradient descent update rules (Eq. 3.14, and 3.15) guarantees convergence to the global optimum, assuming a reasonable learning rate η is selected.

3.4 Numerical experiments

The following parameters were used across all simulations unless otherwise stated.

- the leakage current $i_{discharge}$ is 100pA
- the membrane capacitor C_m is 1pF
- the voltage offset was chosen to be 0 for simplicity
- the initial conductance values were uniformly drawn between $G_{min} = 10^{-9}$S and $G_{max} = 10^{-7}$S
- the scaling factor K is 1/12.5
- the membrane voltage threshold $V_{th} = 1$V
- the refractory period on presynaptic neurons is equivalent to 25µs
- the refractory period on postsynaptic neurons is equivalent to 10 successive spikes from other postsynaptic neurons
- the recordings are spatially cropped to fit the 28×28 vision sensor
- the logistic regression was trained over 70 epochs with a learning rate $\eta = 0.1$

3.4.1 Handwritten digit classification: the N-MNIST dataset

a) 3-class N-MNIST

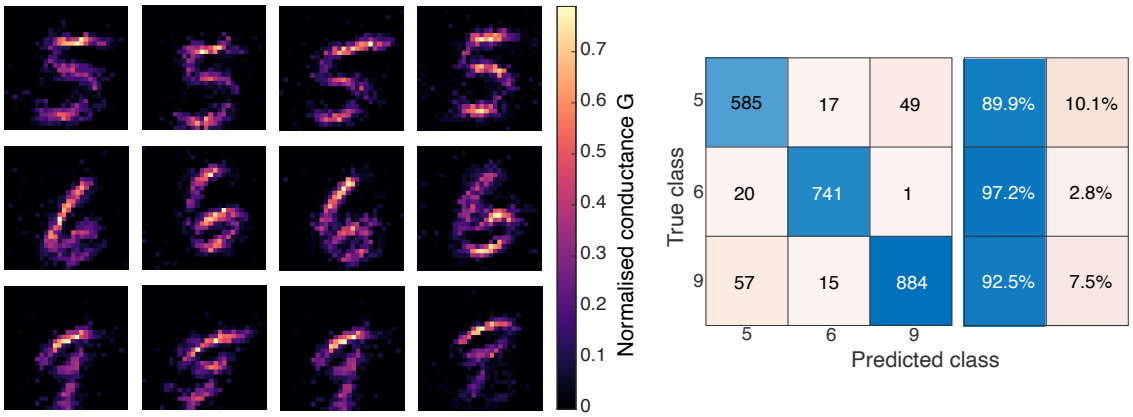


Fig. 3.9 *Simulation results for a 3-class N-MNIST learning task with the heuristics-based classifier.* ***(Left)*** *Normalised conductance maps towards the end of the training phase.* ***(Right)*** *Confusion matrix with an overall accuracy of* 93.08%.

The learning performance of the single-layer SNN is evaluated on N-MNIST, a handwritten digit recognition dataset, when using both the internal heuristics-based classifier and the external logistic regression. N-MNIST is an event-based version of the popular MNIST dataset, consisting of 60000 training and 10000 testing samples of 28×28-pixel handwritten

digits with values ranging from 0 to 9. To generate events from images, each digit is moved in front of an event-based camera following a sequence of three 100ms saccades [223]. The all-to-all connected network being quite expensive to simulate, only the ON polarity events occurring during the first 100ms saccade are used. For all simulations involving the logistic regression classifier the STDP learning rate is decreased to $A^{+/-} = 0.01$ compared to a learning rate $A^{+/-} = 0.1$ for the heuristics-based classifier, as the logistic regression typically needs to be trained on a stable dataset.

In order to be comparable with the relevant literature, only three classes, the 5, 6 and 9 digits, are used to begin with [96, 117]. With 100 neurons, the network can achieve an accuracy of 93.08% (Fig. 3.9), a result in accordance with the current state of the art which is at 92.1% with the same number of neurons [117], and 93.68% with 400 neurons [96].

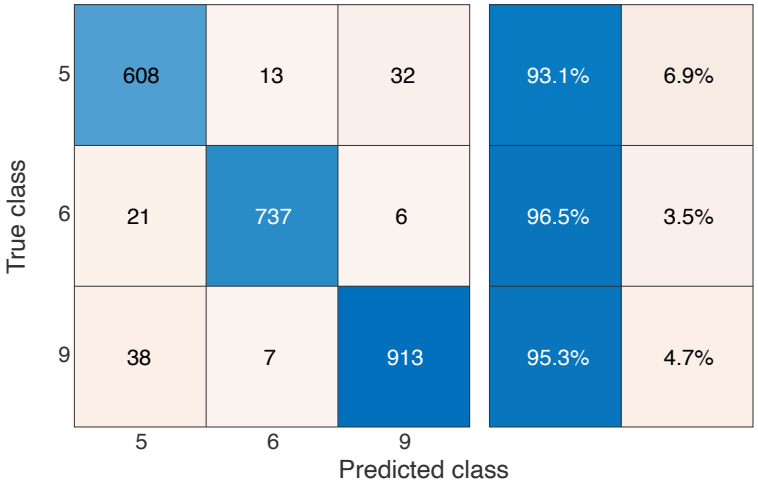


Fig. 3.10 *Confusion matrix for the 3-class N-MNIST task with the logistic regression classifier. An accuracy of* 95.07% *is achieved.*

As expected, the logistic regression classifier performs better on the same subset of N-MNIST handwritten digits. An accuracy of 95.07% is reached when the classifier is trained with the number of spikes output by each LIF neuron over the last 1000 data points, a term referring to a presentation of an event-based recording, or a digit in the case of N-MNIST (Fig. 3.10). To confirm the increase in accuracy is due to the logistic regression itself, simulations with the internal heuristics-based classifier are repeated. This time around, neuron labelling is done based on the last 1000 spikes, up from 10. Using a higher number of spikes for labelling does not improve the classification accuracy. The heuristics-based classifier is therefore quite limited and would not scale well on more complicated datasets.

b) 10-class N-MNIST

The 3-class N-MNIST learning task is too simple to properly explore the benefits of using a logistic regression classifier instead of a heuristics-based classifier. This particular subset of the N-MNIST dataset is used in the literature when simulation speed is unreasonably

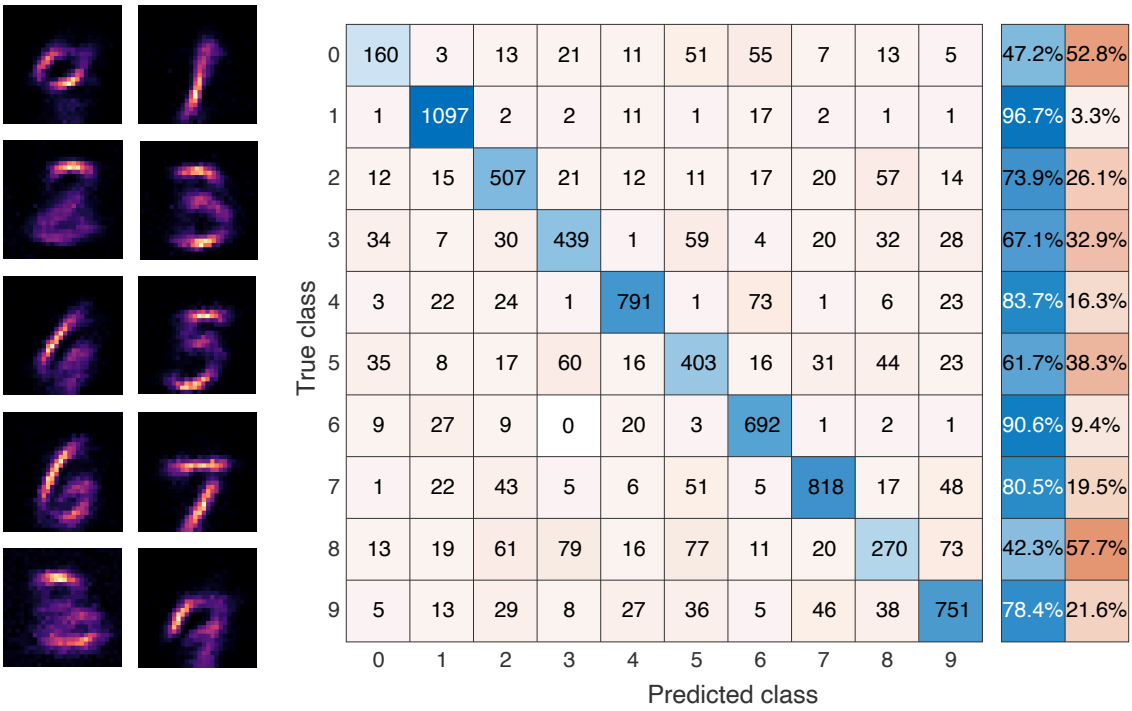


Fig. 3.11 *Simulation results for a 10-class N-MNIST learning task with the logistic regression classifier.* ***(Left)*** *Examples of conductance maps for each class in the full N-MNIST dataset.* ***(Right)*** *Confusion map for the full N-MNIST dataset with a 76.2% accuracy.*

long due to an unoptimised simulator [96, 117]. As this is not an issue with our in-house simulator, the full 10-class N-MNIST dataset can be tested. An accuracy of 61% is reached with the heuristics-based classifier. The same experiment is repeated with the logistic regression classifier. With all ten classes of the N-MNIST dataset, the STDP-based network yields a classification accuracy of 76.2% (Fig. 3.11), a 15% increase over the heuristics-based classifier with an equivalent network.

3.4.2 Critical parameters

a) Logistic regression training set size

The single-layer network with a logistic regression classifier is now trained over two epochs on the 10-class N-MNIST task in order to find the best possible accuracy when the STDP learning rule and the classifier are not clashing with each other. Indeed, the first epoch is used to extract features from the stream of events using the STDP. The logistic regression is then trained in a second epoch, after the synaptic weights have completely stabilised. A classification accuracy of 79.2% is reached, which is a 3% increase over the one-epoch method where the synaptic weights and the classifier are trained at the same time (Fig. 3.12). From an energy efficiency perspective, having two full passes over the data is simply too costly, especially considering the small payoff in classification accuracy. The one-epoch strategy which starts training the classifier once synaptic weights start

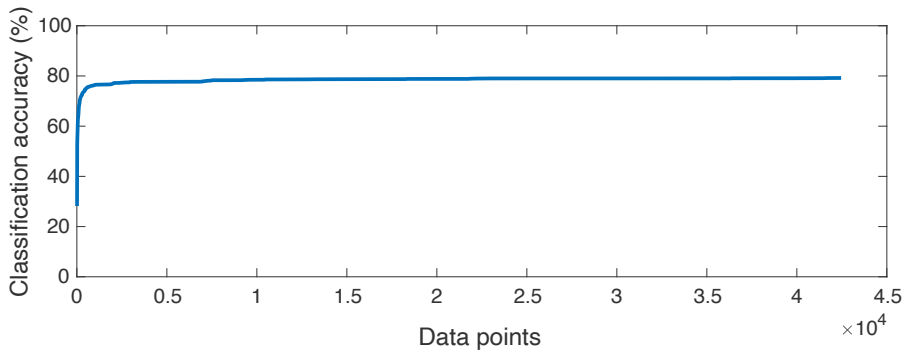


Fig. 3.12 *Plot of the accuracy according to the number of data points with the STDP and the logistic regression trained on separate epochs of the full N-MNIST training dataset.*

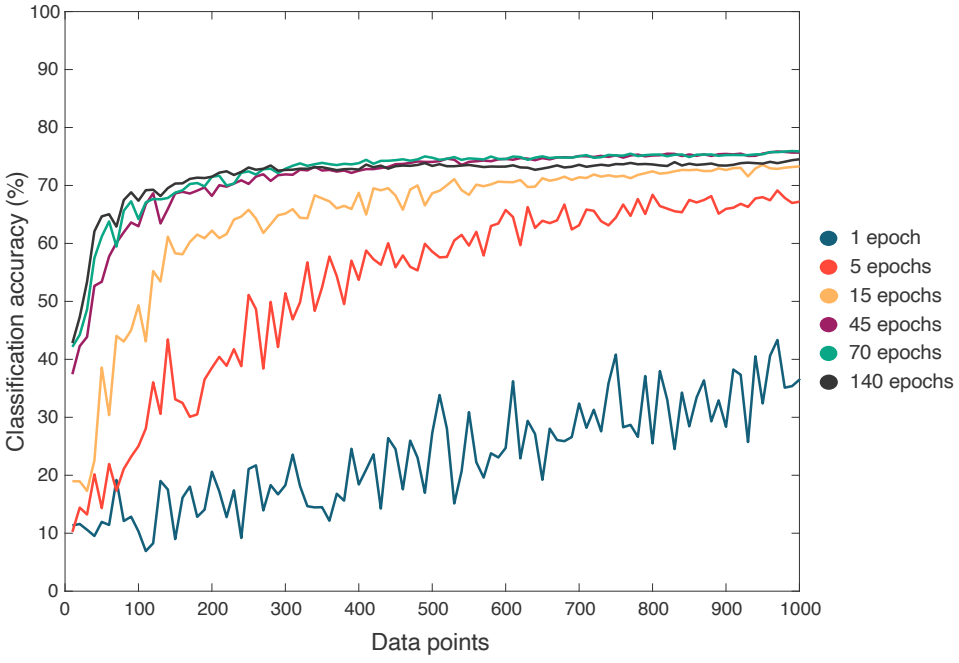


Fig. 3.13 *Classification accuracy with a logistic regression trained over a different number of epochs.*

stabilising, seems to be the best choice for a single-layer SNN with 100 fully connected neurons.

b) Optimal number of epochs for training the logistic regression

A logistic regression fitted with the stochastic gradient descend (SGD) algorithm usually goes through multiple epochs of the dataset, which in our case, is the number of spikes output by every LIF neuron for each of the last 1000 N-MNIST digit recordings. The optimal number of epochs is explored, starting with one epoch, the closest scenario to a fully online architecture, and gradually increasing the number of epochs until a maximum of 140 epochs (Fig. 3.13). Using 45 to 70 epochs yields the best classification accuracy. Considering the small size of the dataset used to train the classifier, training with more than 70 epochs cause overfitting, and anything less than 5 epochs causes a significant drop in classification accuracy.

3.4.3 Results with alternative event-based datasets

Table 3.1: Classification accuracy compared to other event-based algorithms

	POKER-DVS	**N-CARS**	**N-MNIST**
ULPEC SNN + logistic regression	100%	53.2%	79.2%
H-FIRST [224]	91.6%	56.1%	71.2%
HOTS [100]	$95-100\%$	62.4%	80.8%
Gabor-SNN		78.9%	83.7%
HATS [1]		90.2%	99.1%

POKER-DVS A deck of 99 cards belonging to one of four suits, diamonds, clubs, spades and hearts, is manually flipped in front of an event-based camera [225]. The POKER-DVS dataset is particularly interesting, as the time domain is more relevant than the N-MNIST digits which move at a constant speed within each saccade. The POKER-DVS dataset is however, heavily unbalanced. To solve this issue, the single-layer network is trained with only 15 data points from each of the four classes, and the rest is used in the testing set. Moreover, the logistic regression is only trained over 20 epochs, mainly due to the fact that this event-based dataset is simpler than the 10-class N-MNIST. The network reaches an accuracy of 100%, an improvement over other event-based feature extraction algorithms such as H-FIRST [224] or HOTS [100].

N-CARS The visual data processing system is mainly targeted towards autonomous driving applications. To this end, the single-layer network with a logistic regression classifier is tested on N-CARS, a binary classification problem that involves differentiating between cars and background samples recorded using a 304×240-pixel resolution event-based camera and downsampled to a 64×56-pixel resolution [1]. Compared to POKER-DVS and N-MNIST, this dataset is significantly more complex due to variability in the scale,

speed and direction of incoming cars, especially considering the restrictions imposed by the general hardware design. The presence of a random background class is also potentially problematic in a fully connected single-layer network, as each LIF neuron learns global features instead of picking up local features in the visual scene. In order to avoid further downsampling, each recording is cropped to a 28×28-pixel window, discarding in the process any events falling outside of the region of interest. Furthermore, similarly to the N-MNIST dataset, only the ON polarity events are taken into consideration, to avoid driving the memristors with too many events. Simulations on the N-CARS dataset yield an accuracy of $53.6 \pm 0.39\%$. Scaling it up to the full 64×56 dataset yields an accuracy up to 60.25%. As expected, a fully connected network that learns global features does not fare well under realistic conditions. A different network architecture that is able to extract local features is likely more suitable for automotive applications.

3.4.4 Hardware design trade-offs

Considering the low number of neurons in the output layer, taking a step towards a hybrid architecture implementing a logistic regression classifier on the external FPGA can significantly improve the performance of the single-layer spiking network without significantly impacting the energy-efficiency [220]. In fact, high performance implementations are readily available.

As for the optimal training strategy, learning the synaptic weights using STDP and fitting the regression model during the same data pass offers the best power-performance trade-off. Fig. 3.14 suggests using at least 200 data points as part of the logistic regression training set, the performance is otherwise significantly impacted.

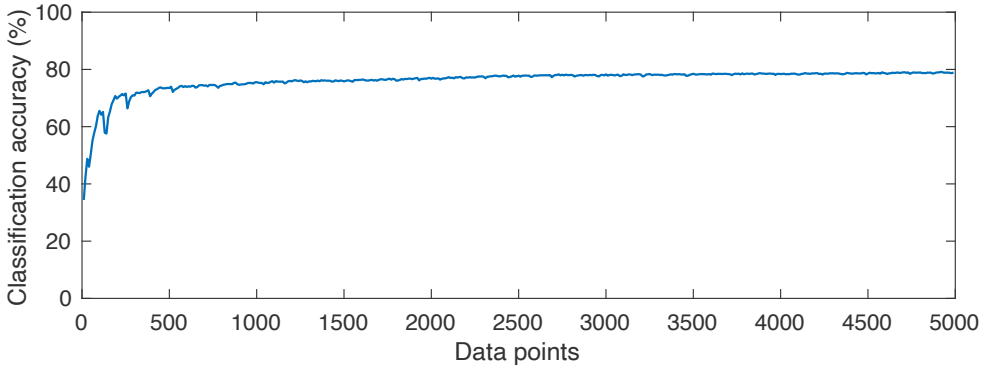


Fig. 3.14 *Plot of the classification accuracy on N-MNIST according to the number of data points used in the logistic regression training set. In this experiment, the synaptic weights and the classifier are trained during the same epochs. Training with* 200 *data points is the bare minimum to start getting a decent classification accuracy considering the low number of neurons used.*

3.5 Scalability analysis: towards a full-scale neuromorphic vision system

The architecture of the ULPEC system as seen in the previous section was chosen with the early motivation of providing a versatile learning capacity, so the network can handle a broad spectrum of learning tasks. The achieved performance, obtained via simulations based on qualitative models of the memristor and LIF neuron, is on par with equivalent Hebbian-based architectures [96]. However, the visual data processing system has so far not been extensively tested on automotive tasks such as car detection on the N-CARS dataset [1]. Specialising and adapting the current architecture for these complex scenarios is necessary in order to push the performance limits of the network without significantly compromising the target power budget.

This scalability study is driven by two objectives: **(i)** improving the performance of the single-layer SNN architecture by measuring the impact of varying a number of parameters such as the number of neurons, the memristor conductivity range, and the number of data points used to train the logistic regression classifier. A data point refers to the number of spikes output by every LIF neuron for a particular pattern presentation. **(ii)** Refining the network by re-routing neuron connections and pruning synapses in an optimal way, towards a more task-specific learning. In that regard, we propose a new architecture based on local connectivity that improves both the performance and energy efficiency of the ULPEC system, making it easier to scale up the network with the same resources available to the fully-connected architecture.

3.5.1 Improved architecture with local sparse connectivity

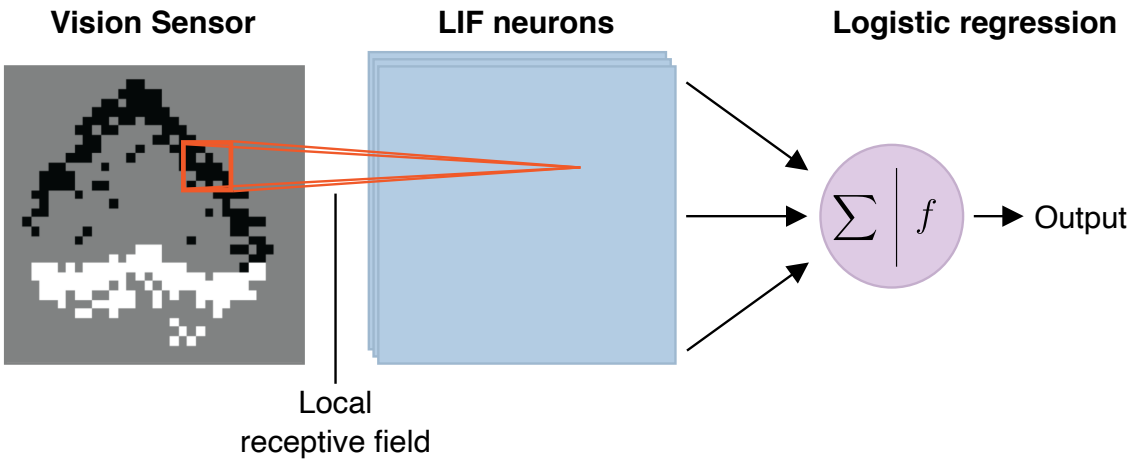


Fig. 3.15 *The ULPEC visual data processing system with sparse connectivity instead of a fully-connected layer. Each LIF neuron is connected to a small region of the input space called a local receptive field.*

The scalability problem is directly related to the dimensions of the current network which is defined by the number of neurons and the number of connections. The physical size of the chip is ultimately bounding the two parameters and limiting the complexity of

the network. As previous stated, in its current form, the demonstrator is built with a maximum of $28 \times 28 = 784$ input neurons, mapped one-to-one to the pixels of the vision sensor, defining the input layer. Each of these neurons are connected to all of the 100 output neurons, setting the number of memristive synapses to $784 \times 100 = 78400$ (Fig. 3.4). From this starting point, we are measuring the impact of these two and other parameters on the learning performances.

Pioneering works found in literature [96, 113] are all trained to classify the N-MNIST dataset as it is used in the field to set the baseline of the learning performance of an SNN. As previously mentioned, to convert the original MNIST dataset into events, the digits are moved at a constant speed in front of an event-based camera following a sequence of three 100ms saccades [223]. This controlled environment greatly simplifies the dataset, allowing the fully-connected network to achieve a good performance considering the limitations imposed by the hardware. In fact, the constant speed in each saccade renders the time domain mostly irrelevant as only the spatial features differ between classes. Furthermore, N-MNIST lacks any sort of translation, mirroring or scaling problems which is regularly encountered on more complex datasets.

The goal of the ULPEC visual data processing system is to explore more realistic automotive tasks through the use of N-CARS, a car classification dataset [1]. A fully connected architecture poses two major problems in that regard:

- it is designed for global features learning by feeding all input pixel to each LIF neuron. This is not ideal considering cars can come into the camera's field of view from different directions, and the scale is not necessarily the same.
- the high number of synapses can also significantly slow down learning and increase the energy budget.

These problems are mitigated in convolutional neural networks (CNN) [7] and deep SNNs [97, 120, 129] by using low-level local features that are passed through a hierarchical model to increase abstraction and learn higher-level features. The use of local features in CNNs has largely been supported by observations in the mammalian visual cortex [4, 5, 226] but also from natural image statistics [227, 228], which imply that the behaviour of a vision sensor (pixel) is significantly more correlated with the vision sensors that are near than those further away. The increased significance of local structure in an image allows for a neural network design where neurons in one layer are only sparsely connected with neurons in the previous layer.

The new architecture proposed here, introduces a similar strategy to create a sparser, and thus more efficient, architecture. In this new architecture, the LIF neurons are not connected to all pixels of the event-based vision sensor. Instead, a square window of size $k \times k$ is scanned across the input pixel space with a stride s, linking each spatial region to a different LIF neuron (Fig. 3.15). Depending on the stride, these spatial regions, or local receptive fields, can have a certain overlap which reduces the downsampling inherent to this method. Furthermore, multiple sublayers of LIF neurons can be defined by connecting each receptive field to more than one neuron. Unlike their CNN counterparts, sublayers do not introduce a parameter sharing scheme as it would significantly burden the hardware and increase the number of memristors that simultaneously need to be in a programmable

state. They are simply meant to enable learning more than one feature per receptive field. Setting different weights to each neuron means that we no longer treat images as stationary.

Defining $i_w \times i_h$ as the dimensions of the vision sensor, the dimensions of the output layer of LIF neurons are given by:

$$o_w \times o_h = \left(\frac{i_w - k}{s} + 1\right) \times \left(\frac{i_h - k}{s} + 1\right). \tag{3.16}$$

This particular equation corresponds to an architecture with no zero padding and non-unit stride; some pixels on the edge of the vision sensor can therefore be ignored depending on the selected parameters. The proposed architecture brings about significant improvements in cutting down the simulation time of our qualitative models by a large margin, contributing towards a more comprehensive scalability analysis. Considering common receptive field sizes such as 3×3, 5×5, or 7×7, each neuron is connected to a maximum of 49 memristive synapses, instead of 784 with the fully-connected architecture.

3.5.2 Scalability and performance analysis on N-MNIST

Simulation parameters are the same as in section 3.4 across both the fully-connected and the sparse architecture. The STDP is symmetrical with learning rates $A^+ = 0.01$ and $A^- = 0.01$. As for the logistic regression classifier, we use a stochastic gradient descent optimiser that fits the model over 70 epochs, with a batch size of 128 data points, a learning rate of 0.01 and a weight decay of 0.01. Experiments on the local sparse architecture are averaged over 5 trials. The fully-connected network is significantly slower to simulate and is therefore only tested over 2 trials.

The topology of the sparse network is arbitrarily chosen to get exactly 100 neurons according to Eq. 3.16. The output layer consists of four sublayers with a $k \times k = 7 \times 7$ receptive field window size and a stride $s = 5$, in other words, 25 LIF neurons per sublayer, each with only 49 synapses. The impact of the receptive field window size is discussed in a second step. With these parameters, the network only considers a 27×27 region of the input pixel space, losing in the process one row and one column of pixels which does not have an impact on the classification performance. As in previous reports, we only consider ON polarity events occurring during the first 100ms of each digit recording.

a) Differences in learning across architectures

Before analysing the impact of varying critical parameters, it is important to understand the differences in learning between both network architectures. The local features learned in the sparse network represent oriented edges and other low-level characteristics of the digits, whereas the fully-connected network learns global features from the full input space. This difference can be visualised through the conductance maps of LIF neurons (Fig. 3.16).

Learning with local features helps more easily discriminate between the 10 N-MNIST classes more easily, and provides a more balanced solution. This is apparent in Fig. 3.17

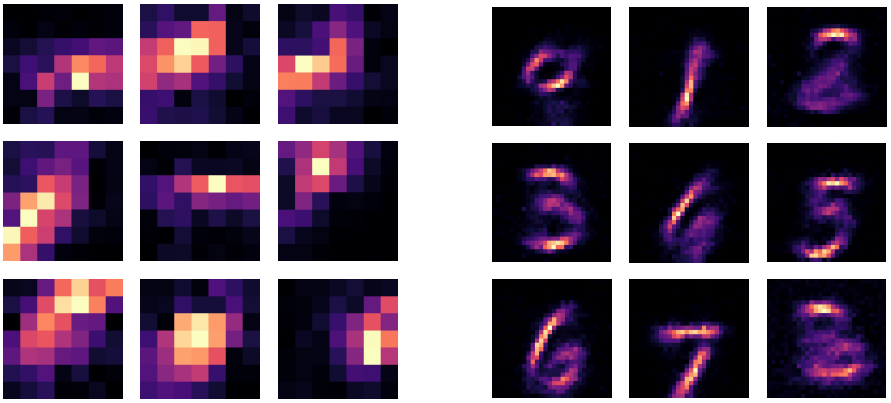


Fig. 3.16 *Examples of conductance maps for LIF neurons with N-MNIST.* ***(Left)*** *Local features with the sparse network.* ***(Right)*** *Global features with the fully-connected network.*

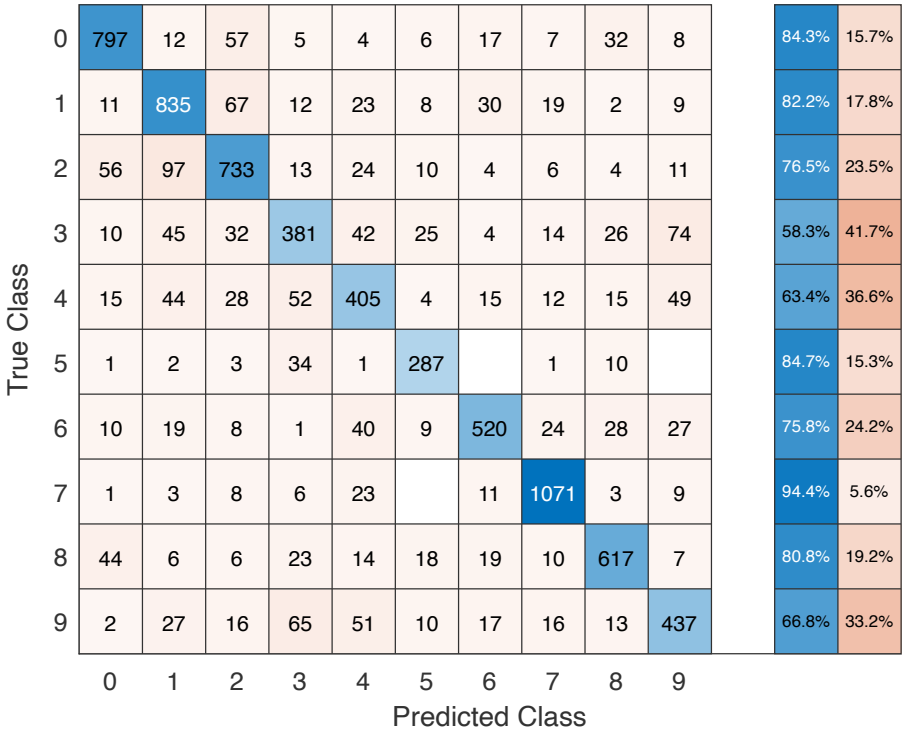


Fig. 3.17 *Confusion map for a 10-class N-MNIST learning task with the sparse architecture. An average accuracy of* $76.59 \pm 2.17\%$ *is reached with 1000 data points.*

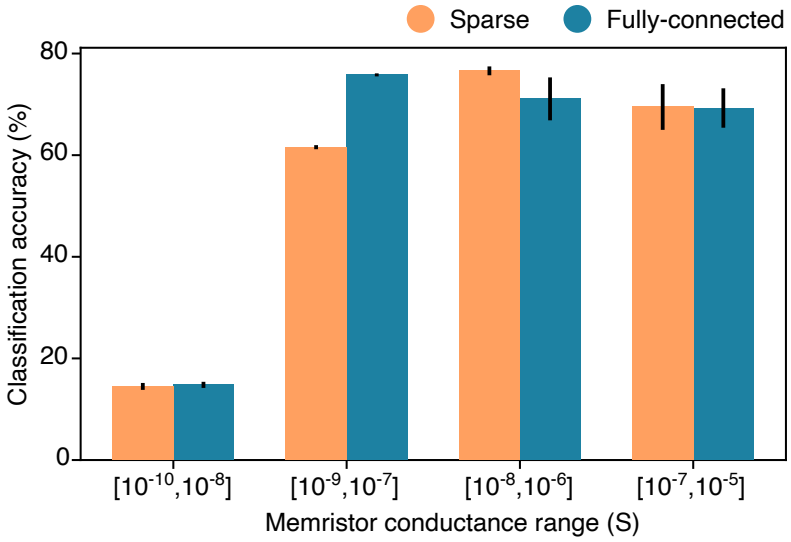


Fig. 3.18 *Classification accuracy depending on the memristor conductance range for the local sparse and fully-connected architectures. The logistic regression is trained with* 1000 *data points.*

where every class has a non-random classification accuracy, in contrast to the fully-connected architecture that frequently misclassifies closely related classes such as the 0 and 8 digits (Fig. 3.11). When training the logistic regression with 1000 data points, in other words the final 1000 digit presentations, the local sparse network yields an overall classification accuracy of $76.59 \pm 2.17\%$ which is in line with previously obtained results on the fully-connected architecture.

The scalability analysis involves changing parameters that directly impact the hardware design, including the number of neurons, memristor conductance range, and the dataset size used to train the classifier, towards achieving better performances on automotive datasets. While the fully-connected architecture is explored in that context, the local sparse architecture significantly speeds up simulation time, allowing us to more easily scale up the network and assess its stability over multiple trials. We ensure consistency of the results across network architectures by comparing the two architectures for some parameter configurations, while we further explore the parameter space using the more efficient sparse network.

b) Memristor conductance

The memristor conductance has a significant impact on the performances of the network. As we introduce the local sparse architecture to substitute the original fully-connected network, it is important to examine the sparse network behaviour under various conductance ranges that are taken from the literature [204, 212, 201]. To measure the impact of the conductance we fix the scaling factor K which regulates the current going through the current conveyor into the analog neuron [218], to $K = 1/12.5$. This particular value is selected for comparability with the literature [117].

Considering an ON/OFF ratio of 100, we run simulations on the full N-MNIST task under four different ranges of conductance G (Fig. 3.18). When $G \in [10^{-10}S, 10^{-8}S]$ the LIF neurons are underactive in both architectures, resulting in a network that is unable to learn any features. The drop in performance can be compensated by a higher scaling factor K which increases the current injected into the LIF neurons, resulting in more spikes that can be used to train the logistic regression classifier.

When we the increase the scaling factor to $K = 1$ the sparse network yields an accuracy of $63.36 \pm 0.95\%$. On hardware, the possible values of K are bounded by the size of the capacitor C_m which can be estimated according to the following equation derived from the LIF model (Eq. 3.4):

$$C_m = \frac{\Delta t(K \cdot i_x - i_{discharge})}{\Delta V_m} \tag{3.17}$$

Considering the sparse architecture has around two simultaneously active memristors, each with a 10kΩ resistance and driven by a 1V pulse over $\Delta t = 10\mu s$, the 1pF capacitor present in the first ULPEC demonstrator is sufficient to handle a value of $K = 1$ for a target $V_m = 1V$.

Moving on to higher conductance ranges, the fully connected network performs better than the sparse network when $G \in [10^{-9}S, 10^{-7}S]$, reaching an average accuracy of $75.44 \pm 0.29\%$. The local sparse architecture on the other hand, is optimal when $G \in [10^{-8}S, 10^{-6}S]$ with an accuracy of $76.2357 \pm 0.87\%$. In general, the sparsely-connected architecture receives less input than the fully-connected one, degrading its learning performance on lower conductance ranges. On the other hand, when the neuronal activity is too high, the logistic regression can more easily start overfitting. As before, the scaling factor can be adjusted so that each network architecture behaves similarly across various memristor conductance ranges, but the 1pF capacitor chosen in the ULPEC demonstrator might be a limiting factor in regards to the possible values of K.

c) Number of neurons

We proceed to explore the evolution of the classification performance with an increasing number of neurons (Fig. 3.19). We vary the number of neurons by keeping the stride fixed and adjusting the number of LIF neuron sublayers. With a logistic regression trained on 1000 data points, the fully-connected architecture peaks at 500 neurons with average accuracy of $79.34 \pm 0.36\%$ and subsequently starts decreasing with larger networks. This is likely due to the fact that the fully-connected network is more prone to overfitting the training data with a high number of neurons. More data points could therefore be needed for the model to converge. The accuracy of the sparse network on the other hand continues to increase until 1000 neurons and reaches an accuracy of $82.01 \pm 1.07\%$, presumably because of the inherent constraint for local statistics that is verified as optimal by the wider literature in computer vision.

In order to find whether a higher number of neurons requires a larger logistic regression training dataset, we evaluate the accuracy of the sparsely-connected network with 100 and 1000, according to the number of data points used to train the classifier (Fig. 3.20).

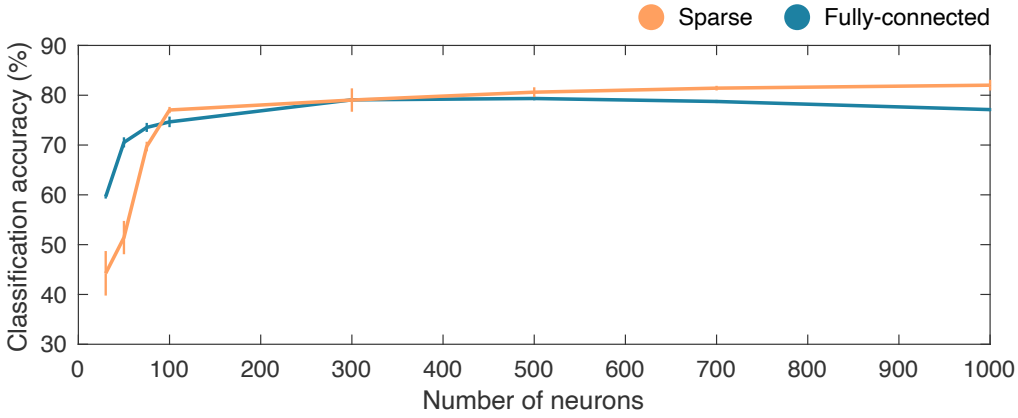


Fig. 3.19 *Classification accuracy with an increasing number of neurons for the sparse and the fully-connected architecture. The logistic regression is trained with* 1000 *data points.*

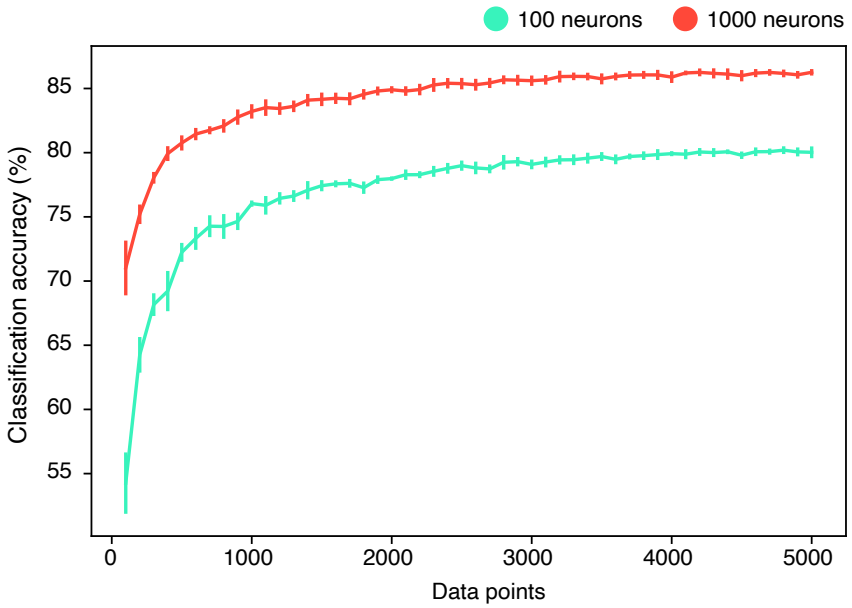


Fig. 3.20 *Accuracy of the sparse architecture with* 100 *and* 1000 *neurons according to the number of data points used to train the logistic regression.*

In both cases, training the logistic regression with more than 1000 data points leads to diminishing returns, and the cost of increasing the size of the training dataset might therefore be too steep to pay.

With 5000 data points, a local sparse network with 100 neurons yields an average accuracy of $78.88 \pm 0.65\%$, a 2% increase over using 1000 data points. Increasing the number of neurons to 1000 improves the accuracy to $86.54 \pm 0.27\%$, matching the performance of event-based algorithms such as HOTS [100].

3.5.3 Scalability and performance analysis on N-CARS

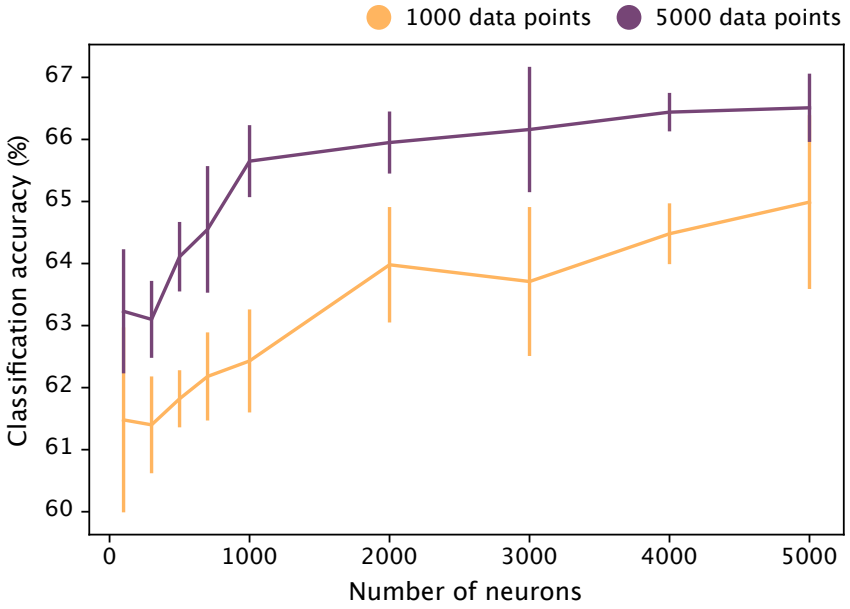


Fig. 3.21 *N-CARS classification accuracy with an increasing number of neurons for the sparsely-connected architecture with* 5×5 *windows and no overlaps between receptive fields. The N-CARS dataset is cropped to fit the* 28×28 *vision sensor on the demonstrator. The classification accuracy is evaluated with a logistic regression trained on* 1000 *and* 5000 *data points.*

On a 28×28 cropped N-CARS dataset, a sparsely-connected network with 7×7 windows, a stride $s = 5$ and 4 sublayers (100 neurons) yields an accuracy of $58.99 \pm 0.75\%$, a 5% improvement over the fully-connected architecture. In order to work with the full 64×56 input space while preserving the same number of neurons, we increase receptive fields to 13×13 windows. This does not seem to offer any advantages, and it increases the number of synapses connected to each neuron. Moving forward, we decide to work exclusively with the cropped dataset.

We decrease the receptive field size to 5×5 windows and remove any overlap by keeping the stride at $s = 5$. In Fig. 3.21, we explore the impact of increasing the number of neurons on the classification accuracy, starting from 100 neurons up to 5000 neurons. With 100 neurons, the network reaches an accuracy of $61.48 \pm 1.4\%$ for a logistic regression trained on 1000 data points, and $63.23 \pm 1.0\%$ for 5000 data points. The accuracy consistently

stays below 70% even when we significantly increase the number of neurons. Extracting features with an unsupervised STDP does not allow us to reach satisfactory performances on complex datasets. With a single layer network, a supervised variant of STDP can be expected to slightly improve the performance, but exploring a multi-layer network by stacking FTJ crossbar arrays seems like a more sensible solution for automotive tasks.

3.5.4 Energy consumption analysis

The sparse architecture is based on the same FTJ crossbar design shown in Fig. 3.4, with each of the output neurons connected only to a local subset of the input neurons defined by the size of the receptive field. The sparse connectivity can be cast into the hardware by removing specific junctions on the crossbar.

We give here, a comparative estimation of the write energy per training pattern consumed by the memristors for the fully-connected network and the sparse network, within a conductance range of $[10^{-9}S, 10^{-7}S]$. The write energy is a measure for the energy consumption per weight update. It can be directly affected by varying:

- the number of synapses per neuron (by setting the size of the receptive field window)
- the number of spikes produced per training pattern.

In that range of conductances, neurons are underactive in the sparse architecture, which leads to an accuracy of 62.04±1.52% on the N-MNIST learning task, corresponding to a 15% drop compared to the fully-connected network (Fig. 3.18). To reach the same comparable accuracy within this conductance range, we have to act on three parameters:

- we increase the factor K which raises the output neurons activity
- we decrease the output neurons refractory period from 10 to 5. This does not pose a problem as not all neurons are in competition with each other
- we initialize the synaptic conductances according to a uniform distribution within the interval $[G_{max}/2, G_{max}]$, with $G_{max} = 10^{-7}S$

With a window size of 7×7 for the receptive field, a stride $s = 5$, and increasing the scaling factor to $K = 1$ the model yields approximately 145 spikes per training pattern, similar to the spike count at the higher conductance range (Fig. 3.18). Using a logistic regression classifier trained with only 1000 data points, we obtain an accuracy of $75.79 \pm 1.52\%$. This particular value of K could require a slightly larger capacitor than 1pF. For integration concerns, a smaller value of $K = 1/2$, leading to a trade-off of 2% drop in performance, might be considered.

With all the parameters kept unchanged but the size of the receptive field reduced to a 5×5 window, we achieve an accuracy of $74.20 \pm 1.04\%$ for an average of 123 spikes per training pattern. Compared to the 7×7 window, the number of connections per output neuron is divided by 2 with a loss on accuracy of 1%. Networks with larger receptive fields do not show significant improvement in performance, and the 5×5 window appears to provide the best trade-off between number of connections and classification performance.

Given an equivalent classification performance, the write energy consumed by memristors

per training pattern can be roughly estimated by accounting for the average number of post-synaptic spikes since:

$$\text{Write energy} = \#\text{spikes} \times \#\text{synapses} \times \text{pulse voltage} \times \text{pulse current} \times \text{pulse duration}$$

With the assumption that a spike from an output neuron applies a pulse voltage of 1V over 10nS to a memristor with the conductance set to $G_{max} = 10^{-7}$S, we can now estimate the write energy consumed for both network architectures:

- the fully-connected network produces on average 13 spikes per training pattern with the N-MNIST dataset, for 784 synapses. This leads to an overall write energy of $\sim$ **10pJ**.
- the sparsely-connected network produces on average 123 spikes per training pattern, for a total of 25 synapses, hence the write energy consumed by the memristors amounts to $\sim$ **3pJ**.

As shown by these first estimates, the sparse connectivity reduces the energy consumed by the synapses for weight updates, by a factor of 3 during the training phase. This perspective is important as it suggests that we improve the energy efficiency of the physical network with minimal modification and little loss in accuracy. The saved resources can then be redirected to other use such as increasing the size of the vision sensor and/or the number of neurons for a fixed energy budget or allocating more energy for computation purposes. Furthermore, the analog circuit design can be simplified by not having to control for the case where all synapses are simultaneously active.

3.6 Discussion

Through a tight collaboration with academic and industrial partners working in the fields of microelectronics and material sciences, we develop a neuromorphic vision system capable of learning with ultra-low power requirements and ultra-low latency. The system in question implements a fully-connected single-layer STDP-based SNN with memristive synapses for real-time unsupervised feature extraction. After identifying a suitable architecture, we provide a study on the scalability of the visual data processing system aimed at improving the learning performance by re-sizing the network in a broad sense. In that regard, we measure the impact of the number of output neurons, and we explore an alternative sparsely-connected network that improves the accuracy and makes the system even more energy-efficient. Providing a clear insight to these two parameters allows us to propose a clear recommendation in scaling up the system: is the increase in computing resource necessary, is it feasible for an actual physical implementation and what can we expect for an automotive learning task?

With the current unsupervised STDP algorithm, scaling up the system by increasing the number of neurons only leads to moderate improvements in performance. This is particularly true for more complicated learning tasks. Nevertheless, the sparsely-connected network appears to have a decreased sample complexity, i.e. it requires fewer data points for training, and typically outperforms the former model in classification accuracy while being three times more efficient in terms of the energy required to update the synaptic

weights. Each neuron accounts only for local structure, and it is therefore necessary to obtain more spikes in order to identify an image. However, the energy required for a write operation decreases overall, despite the increased number of spikes.

Our study suggests that a sensible compromise between energy and performance would be a sparsely-connected network with around 300 neurons. Increasing the number of neurons however, cannot be easily achieved in the demonstrator as the size of the chip will be significantly affected. A 3D crossbar architecture providing multi-layer support, along with a supervised STDP [141, 93, 132], instead of continuing to rely on purely-analog STDP-mediated feature extraction techniques.

From a manufacturing perspective, growing ferroelectric memristors on silicon, the end goal of the ULPEC project, makes the technology directly compatible with CMOS and enables industrial-scale integration with microelectronics. Memristor technology is commonly associated with in-memory computing [200], spiking neural networks [201] and deep learning accelerators [229]. Memristors are also being explored in the context of probabilistic computing [230, 231]. This alternative computing paradigm relies on probability bits (p-bit) that can fluctuate between 0 and 1, paving the way for energy-efficient implementations of probabilistic models and inference methods.

Chapter 4

Probabilistic models for event-based data

4.1 Introduction

Neuromorphic vision sensors generate millions of events per second depending on the dynamics of a visual scene, by separately tracking changes in each pixel in an asynchronous manner and with microsecond temporal precision. With the advent of high definition neuromorphic cameras [72, 26, 27], event-by-event algorithms struggle to stay competitive, and more often than not, binning events into frames is a more feasible approach.

The focus of this doctoral thesis has so far been on leveraging temporal learning algorithms for spiking neural networks, towards a more energy-efficient feature extraction on event-based data. These neural networks can be quite expensive to simulate, and usually require dedicated neuromorphic architectures to be competitive (see chapter 3).

Algorithms such as HOTS [100] have previously explored various clustering techniques for feature extraction, none of which are particularly well suited for dealing with an increasingly large quantity of data due to memory and time constraints. Probabilistic event generation models that take into account sensor noise can be represented by a Gaussian distribution [232, 65]. With that in mind, we introduce two efficient learning algorithms for Gaussian Mixture Models (GMM) with sublinear computational complexity:

- The first approach is based on the Expectation Maximisation (EM) algorithm and relies on a stochastic approximation of the E-step over a truncated space in order to maximise the variational lower bound, reducing the computational burden and speeding up the learning process.
- For even better efficiency, the second approach identifies non-zero areas of the posterior before computation for efficient inference. In each EM iteration, the posterior is estimated over a truncated space that is iteratively improved based on a similarity matrix.

The resulting approaches have state-of-the-art efficiency without compromising stability,

making them attractive candidates for extracting features from massive streams of events. We start by comparing our algorithms to popular clustering techniques on a variety of datasets, to highlight their applicability as a general-purpose learning strategy. Furthermore, we evaluate model performance in feature extraction for a classification task on very large vision datasets from event-based cameras.

4.2 Sub-linear stochastic learning in a Gaussian mixture model for event-based data

A Gaussian mixture model (GMM) is a probabilistic machine learning model with a wide variety of applications in speech recognition [233], computer vision [234, 235], astrophysics [236] and precision medicine for instance [237]. For computer vision in particular, using GMMs to identify patterns of local image features is arguably one of the simplest popular approaches [238]. Parameters of the model are typically optimised using Expectation Maximisation (EM) which is a powerful algorithm for maximum likelihood estimation.

The complexity of learning a GMM was until recently bound to be a function of the number of data points and the number of mixture components. As datasets grow larger, it is increasingly difficult to apply traditional clustering methods, and improvements in computational complexity become all the more necessary. Recent developments in convergence analysis of Gaussian mixture models trained with EM are generating renewed interest in the field [239, 240]. Novel algorithms for training mixture models with increased stability [241, 242] and efficiency through truncation and data summarisation methods [243, 244] pave the way for working with large event-based datasets recorded using neuromorphic cameras.

Neuromorphic vision sensors create spatially sparse representations of an image and bear several advantages over conventional cameras such as a very high dynamic range and a very low latency, making them suitable for fast motion tracking. Each pixel is separately updated in an asynchronous manner with microsecond temporal resolution, generating an event depending on luminance changes in a visual scene. This strategy leads to an enormous number of events. To illustrate this, N-MNIST, an event-based version of the popular MNIST handwritten digit classification dataset, comprises more than 250 million events over 60000 examples of 28×28 digits [223]. The latest generation of high definition neuromorphic sensors [72, 245] makes it even harder to find suitable feature extraction algorithms that work without being overwhelmed by the rate of data acquisition.

In this section, we present a method for efficient and stable EM-based learning of Gaussian mixtures suitable for large event-based datasets. The proposed algorithm is sub-linear in computational complexity with respect to the number of mixed Gaussian distributions. As GMMs are highly sensitive to initialisation [246], latest advances in seeding algorithms allow us to produce probably good initial clusters without compromising scalability to very large datasets [247, 248]. We apply our algorithm on lightweight coresets (lwcs) a data summarisation method, to further reduce algorithmic complexity [249, 250].

Our approach introduces a stochastic method for identifying a truncated approximation

of the posterior during learning. Truncated approximations [251] have been previously employed on latent variable models to identify subspaces of the posterior that contain most of the posterior mass [252, 253, 254, 255]. Empirical evidence has shown an improved performance in avoiding local optima during learning [256], however truncation is typically carried out using deterministic methods which are inherently more sensitive to the local optimum problem. To our knowledge, no work explores stochastic variants for truncation. Sampling from a random distribution is expected to allow the algorithm to get closer to the global optimum [257].

We first evaluate the efficiency, stability and performance of the algorithm on a variety of datasets that have been extensively used in the relevant literature for comparability purposes and to emphasize the versatility of the technique in handling different types of data. Datasets include CIFAR-10, an image dataset [258], SONG, a collection of audio features compiled from the Million Song dataset [259] and SUSY, a high energy physics dataset [260]. We extend the proposed algorithm to event-based data which has recently garnered interest in the computer vision and machine learning community. Our objective is to conciliate the asynchronous event-based representation with the classical frame representation by providing a common ground of processing. An in-depth analysis is done on POKER-DVS [225] in order to get a sense of the expected speedups with event-based datasets. After having validated the clustering performance of our algorithm, we run a classification task on popular datasets such as N-MNIST, N-Caltech101 [223], N-CARS, a car classification dataset [1], and GESTURE-DVS, a dataset tailored for gesture recognition systems [87]. We compare ourselves to popular event-based feature extraction methods which will be reviewed in the following section.

The proposed algorithm is more efficient in operations and time complexity compared to popular methods [261] and can be applied to a wide variety of tasks without compromising cluster quality and stability which is important for reproducibility and consistent predictions. Our approach is actually better at recovering the same clusters with different initialisations. Classifying large event-based datasets with our algorithm leads to near state-of-the-art accuracy and is competitive against most current event-based learning methods, considering particularly the significant speedups in feature extraction.

4.2.1 Event-based vision processing

a) Feature Extraction Methods

In very high frequency video, feature extraction techniques remain the dominant method of computer vision tasks. While efforts of adapting deep learning methods to event-based processing exist [94], naively running a neural network on the full size of the sensor at every event is too computationally intensive. Neuromorphic engineering has therefore focused on event-to-frame conversion strategies to ease the computational burden of working on an event-by-event basis, discarding the temporal dimension in the process [262, 73, 86, 263]. Spiking neural networks are a natural choice for dealing with event-based data. However, Hebbian-based methods cannot currently reach state-of-the-art performances in image classification tasks unless resorting to deep architectures and specialised hardware to leverage their full potentials [95, 97]. Event-by-event strategies specifically designed for neuromorphic systems such as HOTS [100, 264, 99] and HATS [1] work particularly

well on a smaller scale, but they can benefit immensely from a sub-linear complexity clustering algorithm on larger datasets. These methods rely on time surfaces which are descriptors reminiscent of local image patches that provide a spatio-temporal context in a neighbourhood centred around an event. This representation allows us to apply well-known machine learning algorithms on streams of events while preserving time information.

b) Clustering with time surfaces

As previously mentioned in section 1.3.2.a, time surfaces are built by applying an exponential decay on the most recently active pixels in a Moore spatial neighbourhood centred on each event [100]. The exponential kernel can be approximated by a piece-wise linear kernel to reduce computational complexity.

Working with time surfaces is fairly similar to working with image patches, where the usual luminance values are replaced by an explicit function of event times, and therefore we can use it in a typical patch-based pipeline. In this work, we derive visual features by clustering Time Surfaces and use them for classification. The clustering technique presented in this study is a simple and efficient method based on Gaussian Mixture Models, which is developed in detail in the next section. Providing a machine learning algorithm that bridges the gap between frame and event-based representations allows for a receptive-field based architecture from a purely event-based architecture while providing statistically optimal and robust features.

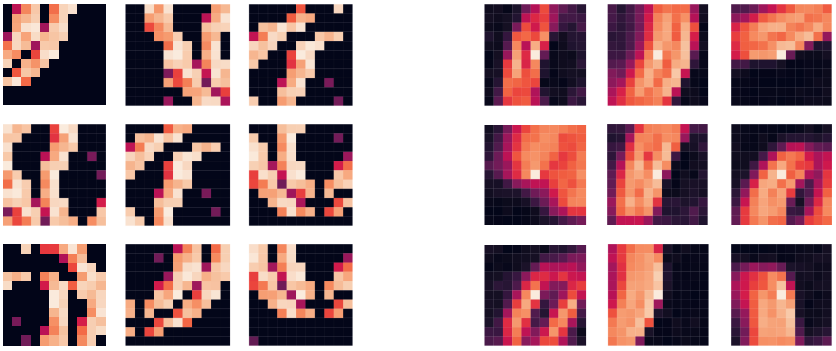


Fig. 4.1 *Examples of time surfaces (left) and cluster centres from the proposed GMM algorithm (right) on N-MNIST, an event-based handwritten digits dataset. The chosen time surfaces and cluster centres are unrelated to each other; however, we can see the smoothing effect of clustering with GMMs.*

4.2.2 Stochastic approximation of expectation maximisation on a Gaussian mixture model

A Gaussian mixture model assumes that each data point is distributed according to a random variable Y that follows one of a finite set of M multivariate Gaussian probability densities. More specifically, for a dataset $\mathcal{Y} = \left\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}\right\}$:

$$p\left(Y = \mathbf{y}^{(n)} | C = c; \theta\right) = \mathcal{N}\left(Y = \mathbf{y}^{(n)}; \mu_c; \Sigma_c\right) \tag{4.1}$$

$$p\left(C=c;\theta\right)=\alpha_c \tag{4.2}$$

where $\alpha_c \in [0,1]$, with $\sum_{c=1}^{M}\alpha_c = 1$, parametrises a distribution over the Gaussian densities and $\theta = \{\mu_{1:M}, \Sigma_{1:M}, \alpha_{1:M}\}$ denotes the model parameters. With the probabilistic model described in Eq. 4.1 and 4.2 and an assumption of independent identically distributed datapoints in $\mathcal{Y}$, the model log-likelihood becomes:

$$\mathcal{L}\left(\theta\right) = \quad \sum_n \log \sum_c \alpha_c \mathcal{N}\left(Y=\mathbf{y}^{(n)};\mu_c,\Sigma_c\right) \tag{4.3}$$

this is a very expressive model for data with a widely varying structure. However, identifying $\Sigma_{1:M}$ sets several challenges that are out of the scope for this study. Instead we will constrain the GMM by setting $\Sigma_c = \sigma\mathbb{1}, \forall c \in \{1,\ldots,M\}$ and rewriting the log-likelihood as:

$$\mathcal{L}\left(\theta\right) = \sum_{n=1}^{N}\log\sum_{c=1}^{M}\alpha_c\left(2\pi\sigma^2\right)^{-\frac{D}{2}}\exp\left(-\frac{d_c^{(n)}}{2\sigma^2}\right) \tag{4.4}$$

Under the assumed model constraints each Gaussian distribution is defined as

$$p\left(Y=\mathbf{y}^{(n)}|C=c;\theta\right) = \left(2\pi\sigma^2\right)^{-\frac{D}{2}}\exp\left(-\frac{d_c^{(n)}}{2\sigma^2}\right) \tag{4.5}$$

where $d_c^{(n)} = \left\|\mathbf{y}^{(n)}-\mu_c\right\|^2$ is the squared euclidean distance between the data point $\mathbf{y}^{(n)}$ and the mean of Gaussian indexed by c, and D is the number of observed variables. The Expectation Maximisation algorithm optimises a lower bound to the log-likelihood:

$$\mathcal{F}\left(\mathcal{Y},\theta\right) \triangleq \sum_n\sum_c p_c^{(n)}\log\left(\frac{\exp\left(-\frac{\|\mathbf{y}^{(n)}-\mu_c\|^2}{2\sigma_c^2}\right)}{\left(2\pi\sigma_c^2\right)^{\frac{D}{2}}}\alpha_c\right) + \sum_n \mathcal{H}\left(p_c^{(n)}\right) \tag{4.6}$$

where $p_c^{(n)}$ is a shorthand for the posterior distribution

$$p\left(C=c|Y=\mathbf{y}^{(n)};\hat{\theta}\right) = \frac{\exp\left(-d_c^{(n)}/2\sigma^2\right)\alpha_c}{\sum_{c'=1}^{M}\exp\left(-d_{c'}^{(n)}/2\sigma^2\right)\alpha_{c'}} \tag{4.7}$$

and $\mathcal{H}\left(p_c^{(n)}\right)$ denotes the entropy of the posterior distribution.

Exact EM is an iterative algorithm that optimises the likelihood by alternating two steps. The first step is to identify the distribution $p_c^{(n)}$ and is referred to as the E-step. The second step is called the M-step and amounts to maximising Eq. 4.6 with respect to θ using a gradient update as:

$$\mu_c = \frac{\sum_{n=1}^{N}p_c^{(n)}\mathbf{y}^{(n)}}{\sum_{n=1}^{N}p_c^{(n)}} \tag{4.8}$$

$$\sigma^2 = \frac{1}{DN} \sum_{n=1}^{N} \sum_{c=1}^{M} p_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{4.9}$$

$$\alpha_c = \frac{\sum_{n=1}^{N} p_c^{(n)}}{\sum_{n=1}^{N} \sum_{c'=1}^{M} p_{c'}^{(n)}} \tag{4.10}$$

Iterating the E-step and M-step of the EM algorithm until θ converges is the most popular method for fitting a GMM. The complexity of the EM algorithm on GMMs is $\mathcal{O}(DNM)$ making it already a very efficient algorithm.

In this work, we propose an approximation technique which avoids the dependency of the complexity on M by estimating the posterior, $p_c^{(n)}$, over a subset $\mathcal{K}^{(n)} \subset \{1, \ldots, M\}$, with $|\mathcal{K}^{(n)}| = H$ as:

$$q_c^{(n)} = \frac{\exp\left(-d_c^{(n)}/2\sigma^2\right) \alpha_c}{\sum_{c' \in \mathcal{K}^{(n)}} \exp\left(-d_{c'}^{(n)}/2\sigma^2\right) \alpha_{c'}} \delta\left(c \in \mathcal{K}^{(n)}\right) \tag{4.11}$$

The posterior for clusters $c \notin \mathcal{K}^{(n)}$, is never estimated and therefore $q_c^{(n)}$ allows for a much faster implementation than $p_c^{(n)}$. In terms of total variation, $d_{TV}\left(q_c^{(n)}, p_c^{(n)}\right) = \sum_{c \notin \mathcal{K}^{(n)}} p_c^{(n)} = 1 - \sum_{c \in \mathcal{K}^{(n)}} p_c^{(n)}$, the approximation is optimal when $\mathcal{K}^{(n)}$ holds the clusters that account for most of the posterior mass. In order to identify such a set $\mathcal{K}^{(n)}$, we start by initialising it with H clusters drawn without replacement as samples of the prior distribution $p(C; \theta)$. At each iteration, we update the set $\mathcal{K}^{(n)}$ by comparing its clusters with R new samples, also from the prior distribution without replacement, and maintaining the H best clusters.

$$\mathcal{K}^{(n)} = \begin{cases} \{c_{1:H} | c_i :\sim p(C)\} & \mathcal{K}^{(n)} = \emptyset \\ \left[\mathcal{K}^{(n)} \cup \left\{c_{1:R} | c_i :\sim p(C) \;\; \& \;\; c_i \notin \mathcal{K}^{(n)}\right\}\right]_{<_{d_c^{(n)}}}^{H} & \mathcal{K}^{(n)} \neq \emptyset \end{cases} \tag{4.12}$$

where $[\cdot]_{<_{d_c^{(n)}}}^{H}$ is an operator that keeps the H smallest elements of the set ordered by the distance $d_c^{(n)}$. The hyperparameters $\{H, R\}$ fully specify the approximation and enable control of the complexity of the algorithm directly as opposed to a precision estimate in typical approximations.

As in the EM algorithm, we can estimate the parameter updates by taking the gradient of the free-energy (Eq. 4.6) and setting it to 0.

$$\mu_c = \frac{\sum_{n=1}^{N} q_c^{(n)} \mathbf{y}^{(n)}}{\sum_{n=1}^{N} q_c^{(n)}} \tag{4.13}$$

$$\sigma^2 = \frac{1}{DN} \sum_{n=1}^{N} \sum_{c} q_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{4.14}$$

$$\alpha_c = \frac{\sum_{n=1}^{N} q_c^{(n)}}{\sum_{n=1}^{N}\sum_{c'=1}^{M} q_{c'}^{(n)}} \tag{4.15}$$

Iterating between estimating the approximate posterior (Eq. 4.11) and the parameter updates (Eq. 4.13–4.15) defines an algorithm we refer to as stochastic GMM, *S-GMM*. The complete procedure is detailed in Algorithm 1.

The complexity of the *S-GMM* algorithm compared to the exact EM algorithm for GMMs reduces from $\mathcal{O}(NMD)$ to $\mathcal{O}(N(R+H)D)$ in the E-step and $\mathcal{O}(NHD+NH)$ in the M-step. In section 4.2.3, we investigate applications of the algorithm where $R+H$ is orders of magnitude smaller than M and the model is successfully fitted to the data.

S-GMM starts with a randomly initialised $\mathcal{K}^{(n)}$ that is intended to store the centres more likely to represent that data point. The set $\mathcal{K}^{(n)}$ is updated randomly, using samples from the prior, at the beginning of each M-step with R new centres. After estimating the distances between the centres in $\mathcal{K}^{(n)}$ and the newly sampled centres, we only maintain the H centres in $\mathcal{K}^{(n)}$ nearest to the data point. The approximate posterior $q_c^{(n)}$ is estimated over the set $\mathcal{K}^{(n)}$ and used to update the centres in the M-step. Since $q_c^{(n)}$ is inversely proportional to the centres' distance from the data point $\mathbf{y}^{(n)}$, far away centres would have a very low - exponentially decreasing - value for $q_c^{(n)}$.

Algorithm 1 Stochastic Gaussian Mixture Model (S-GMM)

Require: Data set $\mathcal{Y}$, # of centres M, H, R
initialise $\mu_{1:M}$, σ, $\mathcal{K}^{(n)} = \emptyset$ for all n;
repeat
  $\tilde{\mu}_{1:M} \leftarrow 0$, $\tilde{q}_{1:M} \leftarrow 0$, $\tilde{\sigma}^2 \leftarrow 0$
  for $n \in \{1, \ldots, N\}$ **do**
    $\mathcal{K}^{(n)} = \mathcal{K}^{(n)} \cup \left\{ c_{1,\ldots,R} | c_i :\sim p(C) \text{ and } c_i \notin \mathcal{K}^{(n)} \right\}$
    for $c \in \mathcal{K}^{(n)}$ **do**
      $d_c^{(n)} = \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2$
    end for
    $\mathcal{K}^{(n)} = \left\{ c | c : \text{ with the } H \text{ smallest } d_c^{(n)} \right\}$
    for $c = \mathcal{K}_c^{(n)}$ **do**
      $q_c^{(n)} = \frac{\exp\left(-d_c^{(n)}/2\sigma^2\right)\alpha_c}{\sum_{c' \in \mathcal{K}^{(n)}} \exp\left(-d_{c'}^{(n)}/2\sigma^2\right)\alpha_{c'}}$
    end for
  end for
  Calculate $\mu_{1:M}$, $\alpha_{1:M}$ and σ^2 (Eq. 4.13, 4.15, 4.14)
until $\theta = \{\mu_{1:M}, \alpha_{1:M}, \sigma^2\}$ and has converged

In order to compare algorithm 1 with popular typical clustering techniques, e.g. k-means and approximations [247, 249], we may optionally constraint the algorithm to accept

a uniform prior over the clusters, i.e. $p(C=c) = \frac{1}{M}, \forall c \in \{1,\dots,M\}$, throughout the learning process. The GMM data model for a uniform prior and the same isotropic variance for all centres is effectively identical to the k-means clustering algorithm and allows for a fairer comparison between our approximation and the broader literature. The fixed uniform prior variant of *S-GMM* will be referred to as *u-S-GMM*.

a) Implementation details

Initialisation During the first epoch of the proposed algorithm, $\mathcal{K}^{(n)}$ is initialised using prior samples and Gaussian centres $\mu_{1:C}$ are initialised with the AFK-MC^2 [248] method. This algorithm samples an initial centre $\mu_1 \in \mathcal{Y}$ uniformly at random and then uses it to derive the proposal distribution $g(\mathbf{y}|\mu_1)$. A Markov chain of length m is used to iteratively sample sufficiently distinct new centres $\mu_{2:M}$ from the data. AFK-MC^2 requires one pass of complexity $\mathcal{O}(ND)$ through the data to define the proposal distribution $g(\mathbf{y})$, and the centres are sampled with a complexity of $\mathcal{O}(m(M-1)^2D)$. The complete process is described in Algo. 2

Lightweight coresets To further improve computational efficiency we can optionally use lightweight coresets (lwcs) [250, 265, 266]. Coresets are smaller representative datasets, $\mathcal{Y}' = \{(\mathbf{y}_1, w_1), \dots, (\mathbf{y}_{N'}, w_{N'})\}$, of a full dataset, $\mathcal{Y}$, in which each data point is individually weighted depending on its significance in describing the original data. The objective on a coreset is adjusted to account for the weights on each data point:

$$\mathcal{F}(\mathcal{Y}, \theta) \triangleq \sum_n w_n \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log \left(\frac{\exp\left(-\frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{2\sigma_c^2}\right)}{(2\pi\sigma_c^2)^{\frac{D}{2}}} \alpha_c \right) + \sum_n w_n \mathcal{H}\left(q_c^{(n)}\right) \tag{4.16}$$

Since the parameter updates are gradient-based updates of Eq. 4.16, the weights $w_{1:N'}$ are a multiplicative constant on the parameters and therefore the parameter updates become:

$$\mu_c = \frac{\sum_{n=1}^{N'} w_n q_c^{(n)} \mathbf{y}^{(n)}}{\sum_{n=1}^{N'} w_n q_c^{(n)}} \tag{4.17}$$

$$\sigma^2 = \frac{1}{D \sum_{i=1}^{N'} w_i} \sum_{n=1}^{N'} \sum_c w_n q_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{4.18}$$

$$\alpha_c = \frac{\sum_{n=1}^{N'} w_n q_c^{(n)}}{\sum_{n=1}^{N'} \sum_{c'=1}^{M} w_n q_{c'}^{(n)}} \tag{4.19}$$

These updates can replace Eq. 4.13, 4.15, and 4.14 in algorithm 1 to allow applications on a coreset $\mathcal{Y}'$. Working on coresets introduces an error in the approximation that has been analysed rigorously in earlier work [250]. Constructing the coreset requires two iterations of complexity $\mathcal{O}(ND)$ over the data but reduces the complexity of S-GMM to $\mathcal{O}(N'(R+H)D)$ in the E-step and $\mathcal{O}(N'HD + N'H)$ in the M-step. The complexity of AFK-MC^2 is also reduced since the proposal distribution is defined on the coreset with complexity $\mathcal{O}(N'D)$.

Algorithm 2 ASSUMPTION-FREE K-MC^2 (AFK-MC^2)

Require: Data set $\mathcal{Y}$, # of centres M, chain length m

$\mu_1 \leftarrow$ Point uniformly sampled from $\mathcal{Y}$ for all n;
for all $\mathbf{y} \in \mathcal{Y}$ **do**
  $g(\mathbf{y}) \leftarrow \frac{1}{2} \|\mathbf{y} - \mu_1\|^2 / \sum_{\mathbf{y}' \in \mathcal{Y}} \|\mathbf{y}' - \mu_1\|^2 + \frac{1}{2n}$
end for
for $c = 2, 3, \ldots, M$ **do**
  $\mathbf{y} \leftarrow$ Point sampled from $\mathcal{Y}$ using $g(\mathbf{y})$
  $d_{\mathbf{y}} \leftarrow \min \|\mathbf{y} - \{\mu_{1:c-1}\}\|^2$
  for $l = 2, 3, \ldots, m$ **do**
    $\mathbf{x} \leftarrow$ Point sampled from $\mathcal{Y}$ using $g(\mathbf{x})$
    $d_{\mathbf{x}} \leftarrow \min \|\mathbf{x} - \{\mu_{1:c-1}\}\|^2$
    if $\frac{d_{\mathbf{x}} g(\mathbf{x})}{d_{\mathbf{y}} g(\mathbf{y})} > \text{Unif}(0, 1)$ **then**
      $\mathbf{y} \leftarrow \mathbf{x}$, $d_{\mathbf{y}} \leftarrow d_{\mathbf{x}}$
    end if
  end for
  $\mu_c \leftarrow \mathbf{y}$
end for

4.2.3 Experiments and results

In this section, we evaluate the performance of S-GMM (algorithm 1) and its uniform variant (u-S-GMM) on three tasks in comparison to standard k-means initialised with the k-means++ method [246]. The k-means baseline was programmed to have an equal structure as the proposed algorithms. We first assess convergence on an artificial dataset with known ground truth [267]. We then move on to a clustering analysis on a variety of datasets originating from different fields such as computer vision [225, 258], high energy physics [260], and music metadata [259] to demonstrate general machine learning applicability of our contribution. We extend the analysis to POKER-DVS, a 4-class event-based dataset that is small enough to be able to run the k-means baseline without any memory and runtime issues. Finally, we test our algorithm on a real-world classification task using very large vision datasets collected from neuromorphic cameras and compare to feature extraction methods specifically designed for dealing with events.

We initialise the cluster centres using the AFK-MC^2 algorithm with a Markov chain length $m = 5$. As a *stopping criterion* we terminate the iterative EM process when the increase in free energy, according to Eq. 4.16, is less than $\epsilon = 10^{-4}$. All experiments were run on a C++ implementation that relies on the Blaze vectorisation library [268], and numerical results are reported as an average over 5 trials unless otherwise stated.

As a reminder, M denotes the number of cluster centres, N' denotes the coreset size, H denotes the size of the truncated subspace and R is the number of new samples in each iteration.

a) Artificial data

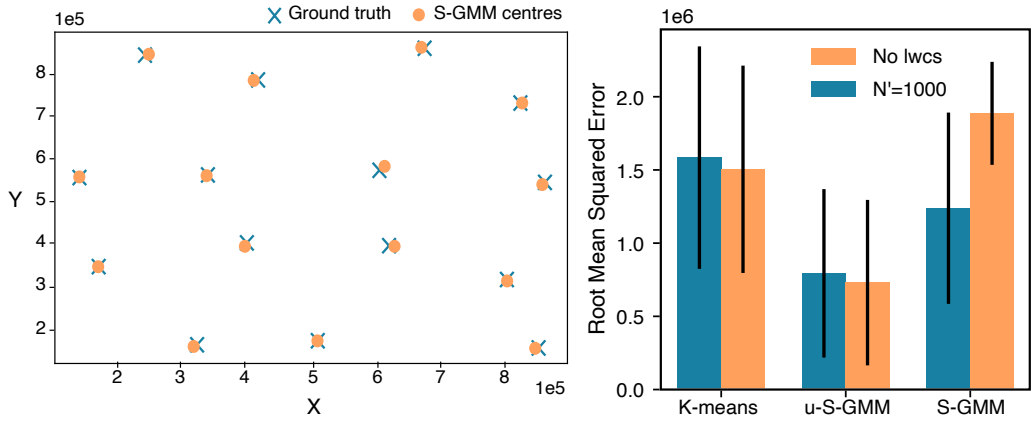


Fig. 4.2 *Validation on a two-dimensional artificial dataset.* ***(Left)*** *S-GMM successfully recovers all 15 Gaussian centres from a standard clustering dataset with* $N = 5000$ *samples,* $H = 3$*,* $R = 6$ *and* $N' = 1000$*.* ***(Right)*** *Using coresets both S-GMM and u-S-GMM outperform k-means across* 100 *trials when comparing the learned centres to the ground truth.*

In this section, we explore the convergence properties of our truncated stochastic approach on an artificial dataset with $N = 5000$ datapoints and 15 Gaussian centres [267].

In Fig. 4.2 we measure the root mean squared error (RMSE) between learned centres and the ground truth. S-GMM and u-S-GMM are able to recover the cluster centres better than k-means across 100 trials. Results suggest the stochastic approach is better at avoiding locally optimal solutions. Additionally, working with a truncated set can improve numeric stability by ignoring low probability clusters.

As the number of data points increases, S-GMM becomes more prone to local optima, as seen in Fig. 4.2 (Right). This behaviour is difficult to rigorously analyse. Our intuition from the experiments suggest that in early iterations poorly initialised clusters might never be assigned a data point. This would lead to zero values in the prior and the cluster will never be used by the algorithm again, even if it is necessary elsewhere. When fewer data points are available, the parameter estimates are noisier and that make it easier to avoid this behaviour. It is important to note this was not reproducible on any other datasets explored in this study. This is perhaps due to the increased variability of real data.

Choosing the optimal number of cluster centres M typically involves running an algorithm multiple times for different values of M and examining the clustering results. Learning prior distributions with the S-GMM algorithm simplifies this problem: we select a maximum number of centres M and the algorithm prunes unnecessary ones by reducing their probability. Fig. 4.3 demonstrates this process on the artificial dataset. We compare the final prior distributions for $M = 15$, the exact number of Gaussian centres in the dataset, and $M = 20$. The observed sharp decay on priors α beyond the 15^{th} cluster suggests S-GMM was able to determine the optimal number of centres in this case. As a drawback, if M is too low, the probability of important clusters might be reduced, affecting the

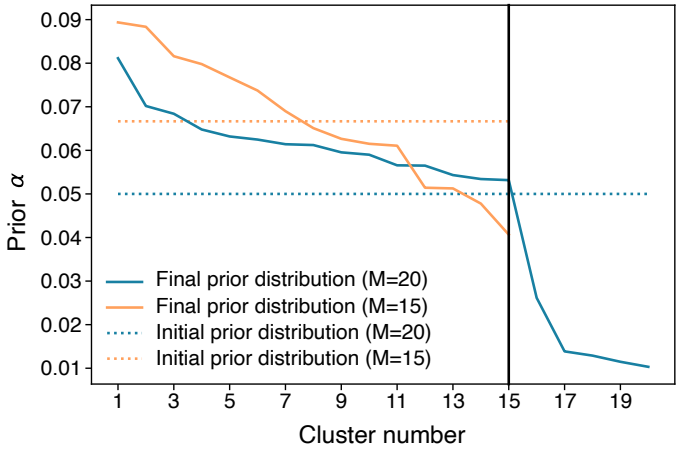


Fig. 4.3 *Prior distributions with S-GMM on an artificial dataset with* 15 *Gaussian centres. By choosing a maximum number of clusters such that* $M > 15$*, the algorithm is able to determine the optimal number of clusters by decaying priors* α *for unnecessary cluster centres (blue curve). However, if the number of clusters is too low, the probability of important clusters might be reduced (orange curve).*

clustering performance. The initial prior distributions shown as dotted lines illustrate the behaviour of an unweighted clustering algorithm such as u-S-GMM and K-means.

b) Clustering analysis

For a more descriptive analysis, we run our algorithms on a series of well-known clustering datasets. The accuracy of the algorithms is measured using the relative quantisation error with K-means as our baseline according to:

$$\eta = \frac{(Q_{\text{algorithm}} - Q_{\text{k-means}})}{Q_{\text{k-means}}} \tag{4.20}$$

where $Q_{\text{algorithm}} = \sum_{n=1}^{N} \min_c \left\| \mu_c - \mathbf{y}^{(n)} \right\|^2$ is the quantisation error of the Gaussian Mixture Model trained with the respective algorithm. K-means initialisation is done with K-means++ [246]. With this in mind, we performed clustering analysis for the following algorithms:

- k-means as a baseline
- k-means on coresets (k-means + lwcs)
- S-GMM on coresets
- u-S-GMM on coresets

Results are summarised in Table 4.1. The S-GMM and u-S-GMM algorithms are parameterised with $H = 5$ and $R = 10$ and they are allowed to execute a different number of iterations until they satisfy the *stopping criterion*. Since the truncated algorithms are going through fewer clusters in each iteration, convergence requires more iterations than K-means. Nevertheless, iterations are significantly cheaper in terms of distance evaluations

Table 4.1: Relative quantisation error and distance evaluation speedup

Dataset	Algorithm		Relative Quantisation Error η	Distance Evaluation Speedup	Iterations
	name	N'			
CIFAR-10	k-means	-	$0.0 \pm 0.03\%$	$\times 1.01 \pm 0.07$	18.2 ± 1.47
$N_{\text{train}} = 50,000$	k-means + lwcs	2^{15}	$2.0 \pm 0.02\%$	$\times 6.13 \pm 0.43$	$\mathbf{13.4 \pm 1.02}$
$N_{\text{test}} = 10,000$	u-S-GMM	2^{15}	$\mathbf{1.0 \pm 0.03}\%$	$\times 26.55 \pm 1.12$	110.4 ± 4.59
$D = 3072$	S-GMM	2^{15}	$\mathbf{1.0 \pm 0.01}\%$	$\times \mathbf{35.54 \pm 1.13}$	82.4 ± 2.58
$M = 500$					
POKER-DVS	k-means	-	$0.0 \pm 0.12\%$	$\times 1.02 \pm 0.16$	57.6 ± 9.39
$N_{\text{train}} = 143,569$	k-means + lwcs	2^{12}	$17.0 \pm 0.19\%$	$\times 160.7 \pm 29.93$	$\mathbf{12.2 \pm 2.71}$
$N_{\text{test}} = 63,847$	u-S-GMM	2^{12}	$\mathbf{14.0 \pm 0.35}\%$	$\times 406.89 \pm 46.5$	170.6 ± 21.6
$D = 121$	S-GMM	2^{12}	$17.0 \pm 0.59\%$	$\times \mathbf{863.1 \pm 150.46}$	81.6 ± 13.92
$M = 500$					
SONG	k-means	-	$0.0 \pm 0.07\%$	$\times 1.0 \pm 0.0$	11.0 ± 0.0
$N = 515,345$	k-means + lwcs	2^{13}	$29.0 \pm 0.24\%$	$\times 121.91 \pm 7.15$	$\mathbf{5.2 \pm 0.4}$
$D = 90$	u-S-GMM	2^{13}	$\mathbf{28.0 \pm 0.22}\%$	$\times 3076.15 \pm 496.31$	66.8 ± 9.3
$M = 4000$	S-GMM	2^{13}	$29.0 \pm 0.26\%$	$\times \mathbf{3468.57 \pm 318.85}$	58.4 ± 5.39
SUSY	k-means	-	$0.0 \pm 0.02\%$	$\times 1.0 \pm 0.01$	46.6 ± 0.49
$N = 5,000,000$	k-means + lwcs	2^{16}	$7.0 \pm 0.6\%$	$\times 18.43 \pm 0.74$	$\mathbf{35.8 \pm 1.17}$
$D = 18$	u-S-GMM	2^{16}	$\mathbf{6.0 \pm 0.6}\%$	$\times 135.98 \pm 2.84$	494.8 ± 11.02
$M = 2000$	S-GMM	2^{16}	$10.0 \pm 0.24\%$	$\times \mathbf{332.67 \pm 6.42}$	199.33 ± 12.36

Note: preferred algorithm for each metric in **bold**.

$d_c^{(n)}$ leading to the observed increase in efficiency. u-S-GMM consistently has the lowest relative quantisation error η across all datasets, whereas S-GMM reaches increased average speedups in terms of distance evaluations. Learning priors leads to a faster convergence at the cost of some accuracy.

Fig. 4.4 further investigates the performance of the algorithms with respect to different coreset sizes. Columns on the left relate the relative quantisation error to the number of distance evaluations on a logarithmic scale compared to the k-means baseline. The size of each marker indicates the size of the coresets N'. Columns on the right highlight the increase in performance associated with larger coresets. Marker sizes on these plots represent distance evaluations. In all cases the proposed algorithms cluster data with a low relative quantisation error to the baseline for the least amount of distance evaluations. S-GMM appears to be less accurate on some datasets possibly due to M being the same across both stochastic algorithms for better comparability, instead of selecting a maximum number of clusters M_{max} such that $M_{max} > M$ which results in unimportant clusters being pruned. With the S-GMM algorithm, if the chosen number of clusters is too low, relevant cluster centres can be lost. Interestingly enough, on the SONG dataset with $M = 4000$ clusters, k-means on coresets (black line in Fig. 4.4) has a lower relative quantisation error. This issue is completely mitigated by increasing the number of new samples at every iteration to $R = 40$, which implies that S-GMM and u-S-GMM converge to a local minimum if the truncated space is sufficiently small (Fig.4.5). Estimating the posterior with a slightly larger R at each EM step is therefore beneficial on complicated datasets with a lot of clusters.

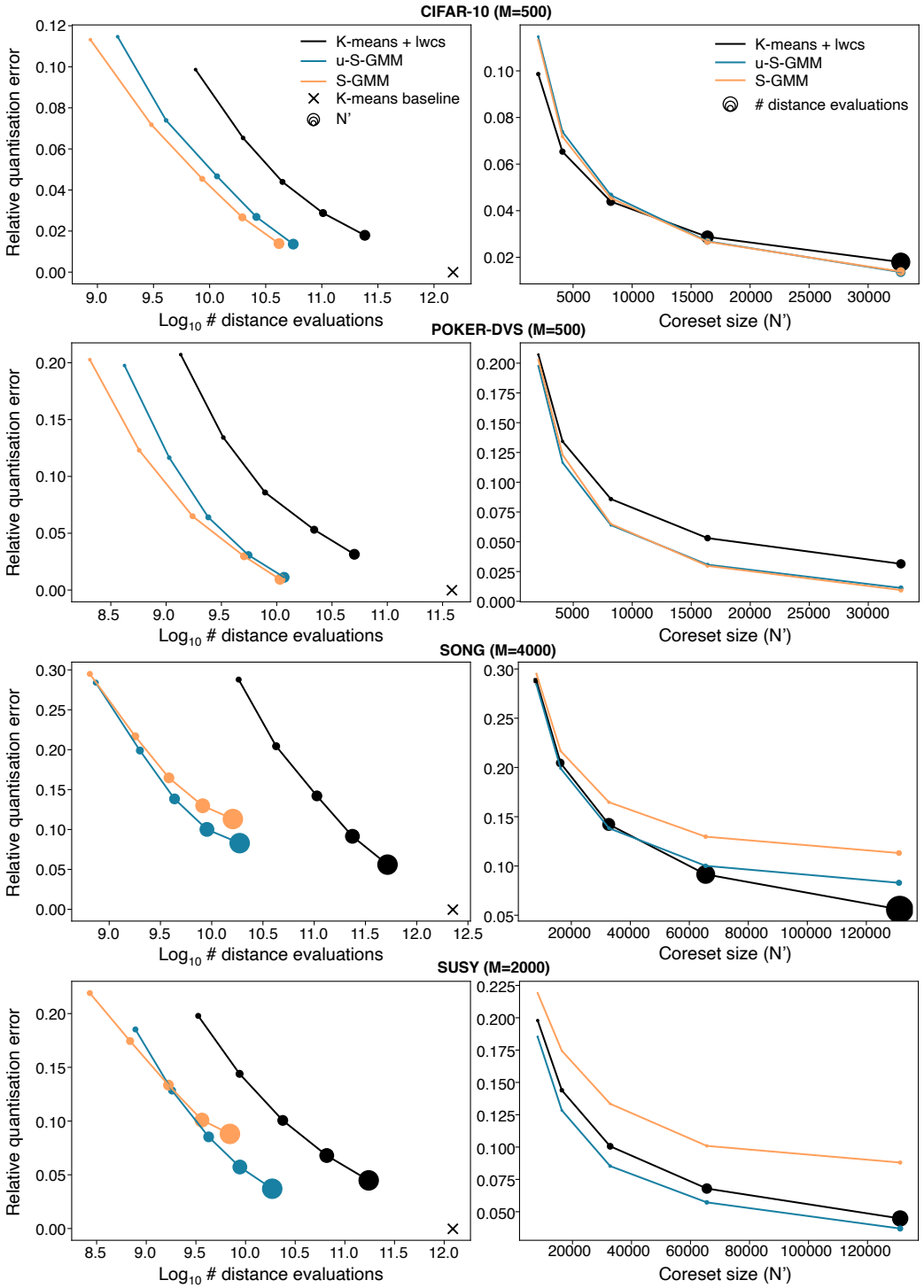


Fig. 4.4 *Number of distance evaluations vs relative quantisation error on increasingly large coreset sizes N' for CIFAR-10, POKER-DVS, SONG and SUSY.* **(Left)** *Relative quantisation error according to the amount of distance evaluations on a logarithmic scale.* **(Right)** *Relative quantisation error for different coreset sizes.*

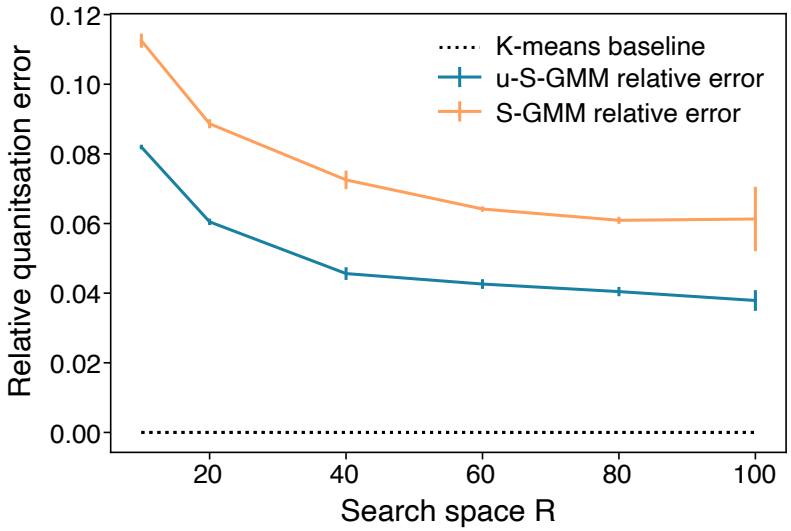


Fig. 4.5 *Relative quantisation error evolution on SONG ($M = 4000$, $H = 5$, $N' = 2^{17}$) with increasingly large R. On datasets with a large number of clusters, sampling from a larger truncated search space can reduce the relative error. For this particular dataset, $R = 40$ offers a good compromise and the error is reduced by more than 8% compared to $R = 10$.*

Complexity Fig. 4.6 shows the scaling behaviour of our algorithms with an increasing number of clusters M on CIFAR-10 (top) and POKER-DVS (bottom), with M ranging from 200 to 1400 clusters and hyperparameters set to $H = 5$ and $R = 10$. On the left, we show the operations complexity, measured according to the amount of distance evaluations $d_c^{(n)}$, and on the right, the time complexity in seconds. For a fair comparison with K-means, we wanted to use an equal number of datapoints by avoiding coresets altogether in this experiment. Results suggest K-means is linear in both time and operations complexity while the proposed algorithms scale sub-linearly, with S-GMM being the most efficient. POKER-DVS is a relatively small dataset both in terms of the number of datapoints, and dimensionality, so the significance of a discrete distribution (S-GMM) over uniform sampling (u-S-GMM) appears in the runtime. For larger datasets with higher dimensions such as CIFAR-10, this difference is negligible.

Stability We assess the proposed algorithms' ability to recover the same clusters using different initialisations. We cluster the CIFAR-10 dataset 5 times with hyperparameters set to $M = 500$, $H = 5$, and $R = 10$, and compare the learned centres of each pair of runs using a normalised RMSE. Fig. 4.7 shows the mean and standard deviation between errors. Both S-GMM and the uniform variant show improvements in stability over K-means which is important for reproducibility. Repeating this experiment on POKER-DVS with $M = 500$ yielded a similar trend as with CIFAR-10.

Hyperparameter selection When selecting optimal values for the hyperparameters, we set $H = 5$ under the assumption that clusters with lower probabilities have a negligible effect as the probability values for the exact posterior decay exponentially. For both

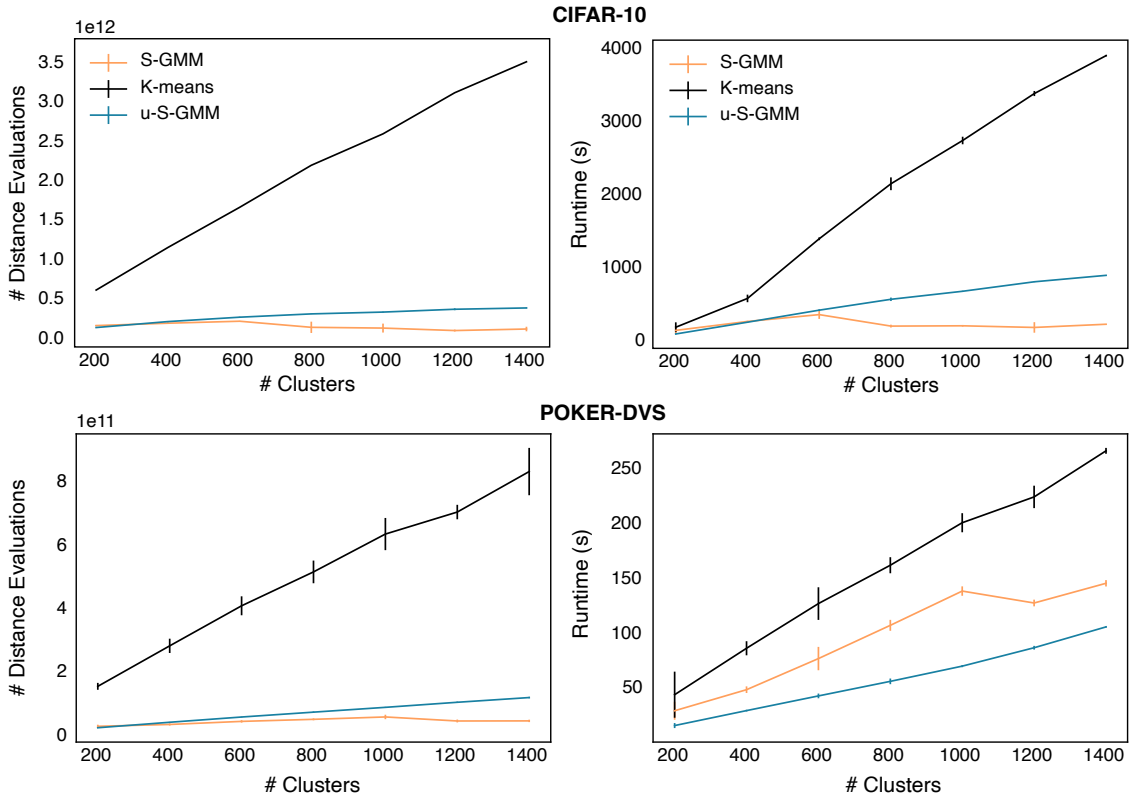


Fig. 4.6 *Operations and Time Complexity analysis on CIFAR-10 and POKER-DVS for an increasing number of clusters. For a fair comparison with K-means, all algorithms were measured without any coresets on the full datasets.* **(Left)** *Operations complexity measure according to the number of distance evaluations.* **(Right)** *Time complexity in seconds.*

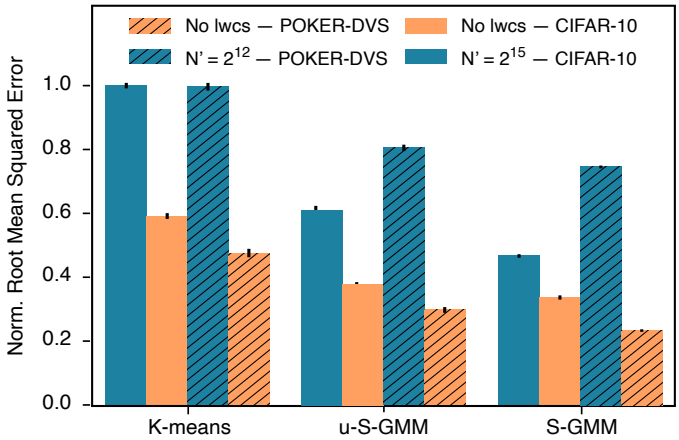


Fig. 4.7 *Stability on CIFAR-10 (solid bars) and POKER-DVS (dashed bars) over 5 trials with $H = 5$ and $R = 10$. S-GMM and u-S-GMM appear to be more stable than K-means in recovering the same clusters with different initialisation.*

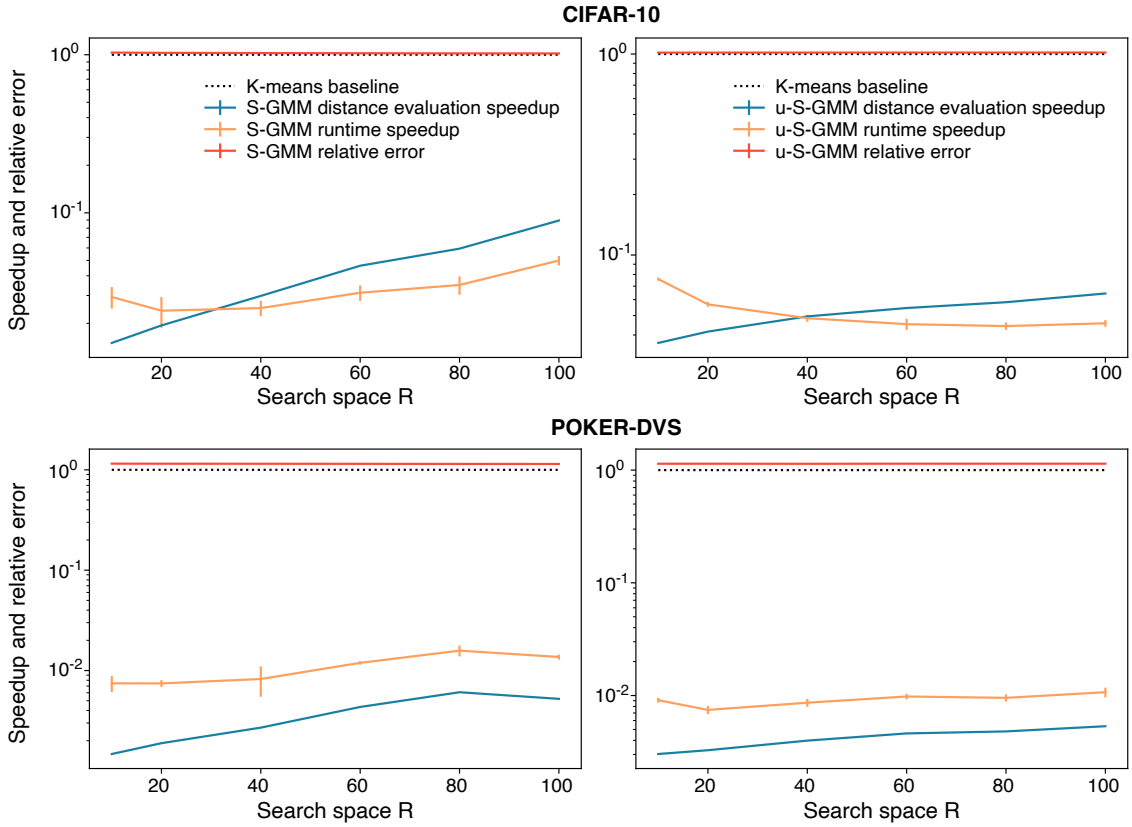


Fig. 4.8 *R hyperparameter search on CIFAR-10 ($M = 500$, $H = 5$, $N' = 2^{15}$) and POKER-DVS ($M = 500$, $H = 5$, $N' = 2^{12}$).* **(Left)** *hyperparameter search for S-GMM.* **(Right)** *hyperparameter search for u-S-GMM.* **(Both)** *Relative error, distance speedup and runtime speedup are presented on a logarithmic scale relative to the K-means baseline shown as a dotted line. For datasets with a low number of clusters, increasing R has a negligible effect on the error, however, values of R ranging from* 10 *and* 40 *seem to be optimal in terms of runtime and distance evaluations.*

algorithms, selecting values of R in the range $[10, 40]$ brings about speedups in terms of runtime and distance evaluations without affecting the relative error compared to the K-means baseline as seen on CIFAR-10 and POKER-DVS in Fig.4.8, with $M = 500$ and $N' = 2^{15}$. As previously mentioned however, for datasets with a larger number of clusters such as SONG, the error of the model increases when R is too low (Fig. 4.5). On CIFAR-10, despite a faster convergence, S-GMM becomes less efficient than u-S-GMM on larger values of R due to the added complexity of prior updates. POKER-DVS behaves similarly across the search space R. To be in the same scale as the distance evaluation speedup and runtime speed, the *relative error* is different from the *relative quantisation error* η used in previous numerical experiments, and is calculated as such:

$$\text{relative error} = \frac{Q_{\text{algorithm}}}{Q_{\text{k-means}}} \tag{4.21}$$

4.2.4 Event-based classification

Table 4.2: Classification accuracy on very large event-based datasets

	POKER-DVS	N-CARS	N-MNIST	GESTURE-DVS	N-CALTECH101
Unsupervised Feature Extraction					
S-GMM	100%	85.69%	98.43%	86.67%	55.42%
u-S-GMM	100%	85.92%	98.42%	87.42%	55.93%
web-scale K-means	100%	84.76%	98.18%	N.A.	N.A.
H-FIRST [224]	91.6%	56.1%	71.2%	N.A.	5.4%
HOTS [100]	95 – 100%	62.4%	80.8%	N.A.	21%
Gabor-SNN	N.A.	78.9%	83.7%	N.A.	19.6%
HATS [1]	N.A.	90.2%	99.1%	N.A.	64.2%
Supervised Feature Extraction					
SLAYER [94]	N.A.	N.A.	99.2%	93.64%	N.A.
BLS [269]	N.A.	92.9%	98.3%	N.A.	72.2%
ResNet-34 [8]	N.A.	90.9%	98.4%	N.A.	69.1%

We benchmark the S-GMM and u-S-GMM algorithms on a series of classification tasks using very large-scale event-based datasets from neuromorphic vision sensors [29]. Datasets include N-MNIST, a handwriting recognition dataset [223], N-CARS, a realistic car classification dataset [1], GESTURE-DVS [87] a dataset tailored for low-power gesture recognition systems, and finally, N-Caltech101 [223], an event-based version of Caltech101 [270].

In the case of handwriting recognition for example, a single image of MNIST will produce an average of 4176 events in one recording of N-MNIST, for a total of more than 250 million events in the training set alone. In a first step, we create time surfaces for all data points according to Eq. 1.1. We experimentally set time surface side length $r = 5$, decay factor $\tau = 80000\mu s$ and $p = 1 \,\forall\, p \in \{-1, 1\}$. This results in an image patch of dimension $D = r \times r = 25$ used as an input to our clustering algorithm.

After clustering the time surfaces, we create spatial histograms [271] of assigned clusters describing the local statistics of a visual scene, and we concatenate them into a histogram used to train a logistic regression classifier. The histogram is standardised so that each dimension has zero mean and unit variance. This pre-processing step is important to get a faster convergence. In Table 4.2 we provide an overview of the classification results on popular event-based datasets using S-GMM and u-S-GMM, in comparison to supervised and unsupervised state-of-the-art methods specifically designed to deal with event-based datasets such as HOTS [100], HATS [1] and H-First [224] among others. We also run N-CARS and N-MNIST on web-scale K-means [261] which is a popular approximation for the the k-means algorithm, as the standard k-means cannot be applied on this dataset in a reasonable time. As in previous sections, classification results on our algorithms are reported as an average over 5 trials; the rest is taken from the literature [94, 1, 100, 224, 269].

We set the hyperparameters to $M = 1000, N' = 2^{16}, H = 5$ and $R = 40$, effectively

avoiding any data-specific parameter tuning to showcase the low parameterisation required to get close to the state of the art in event-based classification. Results are compiled in table 4.2. Both proposed algorithms have a similar accuracy on most datasets, and show improvements over web-scale k-means. These results are in agreement with the clustering analysis which showed better local optima avoidance when working with a stochastic approximation to the EM algorithm over a truncated subspace. The proposed general machine learning algorithm is better at classifying event-based datasets than most task-specific unsupervised feature extraction methods with the exception of HATS which needs to have access to all past events to build each average time surface as evidenced by the use of a very large decay constant. This process is costly and not scalable to very large streams of events. Furthermore, HATS requires significant parametrisation efforts moving between datasets, compared to our method.

We repeated the classification task on N-MNIST, this time with a linear kernel on the time surfaces. S-GMM achieves an accuracy of 98.24% compared to 98.43% with an exponential kernel. Working with a linear kernel eliminates costly exponential calculations at every event, leading to further speedups in the overall classification pipeline with a negligible impact on the final results.

4.3 A sampling-based approach for efficient learning in a Gaussian Mixture Model

In this section, we propose another method for EM-based learning that uses a stochastic truncated approximation [251] of the posterior in the E-step. This method is more involved but it pushes the efficiency even further than the S-GMM algorithm, making it competitive against the state-of-the art in deterministic truncated approaches [254, 243]. Instead of sampling from a uniform distribution to identify the truncated space, we draw samples from a proposal distribution that is based on the truncated subspace of the previous iteration and favours clusters near the optimal cluster of the previous truncated posterior. Our algorithm integrates recent developments in initialisation methods [247, 248] and can be applied on coresets [249, 250] to maintain comparable performance to state-of-the-art approaches.

Truncated approximations have been used in the past for multiple-cause models [252, 253, 254, 256] to achieve efficient training in discrete latent variable models. Stochastic approximations on a truncated space [255, 272] focusing on deep learning models have also been proposed. Compared to deterministic approaches, we expect a stochastic model to avoid well-known local optima issues [257] related to EM-based learning for GMMs, thereby improving clustering performance.

The approach we take in this work utilises a similarity matrix over the clusters in order to identify clusters with a higher probability of being near a datapoint without having to evaluate its distance to all clusters. Estimating the similarity matrix relies on posterior approximations from earlier iterations and does not require excessive computation.

In the numerical experiments section, we evaluate the performance of the algorithm and compare it to the current state of the art [243] in terms of efficiency, stability and

accurate cluster recovery. We focus on three types of tasks: **(i)** clustering on an artificial data, **(ii)** clustering on popular datasets, **(iii)** and feature extraction for classification on event-based data. In the artificial data section, we evaluate our algorithm in terms of extracting the ground truth and we compare it to k-means to observe an improved performance. We then apply the algorithm on four different datasets that are widely used by the clustering community, namely, KDD, a protein homology dataset [273], CIFAR-10, an image dataset [258], SONG, a dataset of music metadata [259] and SUSY, a high energy physics dataset [260]. Finally, we present an unsupervised feature extraction application on event-based vision data [1, 274, 264, 100]. Event-based vision sensors produce a high amount of datapoints to represent visual information in a manner that resembles the retina. Classical classification pipelines for computer vision are not applicable at that scale. Results show that our algorithm sets the state of the art in terms of efficiency without compromising stability. Our method can be applied on a wide variety of tasks while maintaining a competitive clustering performance.

4.3.1 EM with sparsely sampled clusters for GMMs

In the interest of clarity and completeness, and to understand the differences with the previously proposed S-GMM algorithm, some of the concepts seen in section 4.2.2 are reiterated and explained differently.

Consider a dataset of N data points, $\mathcal{Y} = \left\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}\right\}$. Each datapoint $\mathbf{y}^{(n)}$ is treated as an instance of a random variable Y that follows one of M possible Gaussian distributions $p\left(Y|C=c;\theta\right) = \mathcal{N}\left(Y;\mu_c,\sigma\mathbb{1}\right)$ with a prior probability distribution $p\left(C\right) = \frac{1}{M}$, where C takes values in $\{1\ldots,M\}$ and $\theta = \{\mu_{1:M},\sigma\}$ denotes the set of model parameters. We can learn the optimal parameters, $\theta = \{\mu_{1,\ldots,M},\sigma\}$, by maximising the data log-likelihood $\mathcal{L}\left(\theta\right) \triangleq \log p\left(Y=\mathcal{Y}|\theta\right)$ using the EM algorithm. The EM algorithm optimises the free-energy which is a lower bound to the log-likelihood:

$$\mathcal{F}\left(\mathcal{Y},\theta\right) \triangleq \sum_n \sum_c p_c^{(n)} \log p\left(C=c, Y=\mathbf{y}^{(n)}|\theta\right) + \sum_n \mathcal{H}\left(p_c^{(n)}\right) \tag{4.22}$$

$$= \sum_n \sum_c p_c^{(n)} \log \frac{p\left(C=c|Y=\mathbf{y}^{(n)},\theta\right)}{p_c^{(n)}} + \sum_n \log p\left(Y=\mathbf{y}^{(n)}|\theta\right) \tag{4.23}$$

where $\mathcal{H}\left(p_c^{(n)}\right)$ denotes the Shannon entropy of the distribution $p_c^{(n)}$. The distribution $p_c^{(n)}$ is typically set to be the posterior distribution $p\left(C=c|Y=\mathbf{y}^{(n)};\hat{\theta}\right)$, as it sets the first term of Eq. 4.23 to 0 and the free-energy to be equal to the log-likelihood*. Under the assumed model constraints each Gaussian distribution is defined as

$$p\left(Y=\mathbf{y}^{(n)}|C=c;\theta\right) = \left(2\pi\sigma^2\right)^{-\frac{D}{2}} \exp(-\frac{d_c^{(n)}}{2\sigma^2})$$

where $d_c^{(n)} = \left\|\mathbf{y}^{(n)} - \mu_c\right\|^2$ is the squared euclidean distance between the datapoint $\mathbf{y}^{(n)}$ and the mean of Gaussian indexed by c, and D is the number of observed variables.

*The first term of Eq. 2 is the negative KL-divergence between the distribution $p_c^{(n)}$ and the exact posterior, $p\left(C=c|Y=\mathbf{y}^{(n)},\theta\right)$.

Exact EM is an iterative algorithm that optimises the likelihood by alternating between two steps. The first step is to identify the distribution $p_c^{(n)}$ that sets the lower bound in Eq. 4.23 to be equal to the log-likelihood. That is $p_c^{(n)}$ has to be equal to the posterior in order to set the KL-divergence in Eq. 4.23 to be equal to 0. For the Gaussian Mixture Model in this work that would be:

$$p_c^{(n)} = \frac{\exp\left(-d_c^{(n)}/2\sigma^2\right)}{\sum_{c'=1}^{M}\exp\left(-d_{c'}^{(n)}/2\sigma^2\right)} \tag{4.24}$$

The second step is called the M-step and amounts to maximising Eq. 4.22 with respect to θ using a gradient update as:

$$\mu_c = \frac{\sum_{n=1}^{N} p_c^{(n)}\mathbf{y}^{(n)}}{\sum_{n=1}^{N} p_c^{(n)}} \tag{4.25}$$

$$\sigma^2 = \frac{1}{DN}\sum_{n=1}^{N}\sum_{c=1}^{M} p_c^{(n)}\left\|\mathbf{y}^{(n)} - \mu_c\right\|^2 \tag{4.26}$$

Iterating the E-step and M-step of the EM algorithm until θ converges is the most popular method for fitting a GMM.

The E-step requires estimating the differences between the mean of all gaussians and all the datapoints. Thus, the complexity of the E-step is $\mathcal{O}(DNM)$ making it a very efficient algorithm. However, one can notice that the E-step, Eq. 4.24, is a softmax function which produces mostly 0 values. Here, we focus on a method to avoid estimating the softmax over all dimensions since it leads to redundant computation.

In order to avoid the dependency of the complexity on M we use an approximation $q_c^{(n)}$ of the posterior, $p_c^{(n)}$, over a subset $\mathcal{K}^{(n)} \subset \{1,\dots,M\}$, with $|\mathcal{K}^{(n)}| = H$ as:

$$q_c^{(n)} = \frac{\exp\left(-d_c^{(n)}/2\sigma^2\right)}{\sum_{c'\in\mathcal{K}^{(n)}}\exp\left(-d_{c'}^{(n)}/2\sigma^2\right)}\delta\left(c \in \mathcal{K}^{(n)}\right) \tag{4.27}$$

where $\delta\left(c \in \mathcal{K}^{(n)}\right)$ is the Kronecker delta. In other words, we assume that clusters outside $\mathcal{K}^{(n)}$ have a probability of 0 for datapoint $\mathbf{y}^{(n)}$, and therefore are not estimated. Using $q_c^{(n)}$ instead of $p_c^{(n)}$, modifies the exact EM algorithm by not setting the KL-divergence to 0 at the E-step. However, we can derive an algorithm that monotonically increases the free-energy by identifying a $q_c^{(n)}$ that decreases the KL-divergence at each E-step.

Proposition 1. *Let $\mathcal{K}^{(n)}$ be a set of cluster indices, and $\mathcal{K}'^{(n)} = \mathcal{K}^{(n)} \setminus \{i\} \cup \{j\}$, where $i \in \mathcal{K}^{(n)}$, $j \notin \mathcal{K}^{(n)}$. Then $KL[q^{(n)}\|p^{(n)}] < KL[q'^{(n)}\|p^{(n)}]$ if and only if $d_i^{(n)} < d_j^{(n)}$.*

Proof. Since all the Gaussians are equiprobable $d_i^{(n)} < d_j^{(n)} \Rightarrow p_i^{(n)} > p_j^{(n)}$. Note that

$\lim_{x\to 0} x \log x = 0$. It follows that:

$$
\begin{aligned}
& KL[q^{(n)} \| p^{(n)}] < KL[q'^{(n)} \| p^{(n)}] \\
\Leftrightarrow & \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log \frac{q_c^{(n)}}{p_c^{(n)}} < \sum_{c \in \mathcal{K}'^{(n)}} {q'}_c^{(n)} \log \frac{{q'}_c^{(n)}}{p_c^{(n)}} \\
\Leftrightarrow & \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log \frac{p_c^{(n)} / \sum_{c' \in \mathcal{K}^{(n)}} p_{c'}^{(n)}}{p_c^{(n)}} < \sum_{c \in \mathcal{K}'^{(n)}} {q'}_c^{(n)} \log \frac{p_c^{(n)} / \sum_{c' \in \mathcal{K}'^{(n)}} p_{c'}^{(n)}}{p_c^{(n)}} \\
\Leftrightarrow & \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log \sum_{c' \in \mathcal{K}^{(n)}} p_{c'}^{(n)} > \sum_{c \in \mathcal{K}'^{(n)}} {q'}_c^{(n)} \log \sum_{c' \in \mathcal{K}'^{(n)}} p_{c'}^{(n)} \\
\Leftrightarrow & \log \sum_{c' \in \mathcal{K}^{(n)}} p_{c'}^{(n)} > \log \sum_{c' \in \mathcal{K}'^{(n)}} p_{c'}^{(n)} \\
\Leftrightarrow & p_i^{(n)} > p_j^{(n)}
\end{aligned}
$$

□

Proposition 1 shows that in order to decrease the KL-divergence at each E-step we only need to iteratively renew the clusters in $\mathcal{K}^{(n)}$ with clusters that are closer to the data-point, $\mathbf{y}^{(n)}$. The M-step can be modified to utilise expectation values over $q_c^{(n)}$ instead of $p_c^{(n)}$ and maintain monotonic convergence [275].

To identify the Gaussians in $\mathcal{K}^{(n)}$, we start by selecting H clusters uniformly at random. At each iteration, we update $\mathcal{K}^{(n)}$ by using R randomly sampled Gaussians in the vicinity of the one that is nearest to the datapoint $\mathbf{y}^{(n)}$. To efficiently identify the Gaussians centred near a datapoint, we define a distribution $p\left(C_t | C_{t-1} = \bar{c}_n; S\right)$, where $\bar{c}_n = \arg\min_c \left\{ d_c^{(n)} | c \in \mathcal{K}^{(n)} \right\}$. The parameter $S \in \mathbb{R}^{M \times M}$ denotes a similarity matrix among the clusters that assigns higher values, $S_{i,j}$, to cluster pairs, $\{i, j\}$, that are likely to be close to the same datapoints, as in Eq. 4.33. The iterative update of $\mathcal{K}^{(n)}$ is defined as:

$$
\bar{\mathcal{K}}_t^{(n)} = \mathcal{K}_{t-1}^{(n)} \cup \left\{ c_{1:R} | c_i \sim p\left(C_t | C_{t-1} = \bar{c}_n\right) \wedge c_i \notin \mathcal{K}_{t-1}^{(n)} \right\} \tag{4.28}
$$

$$
\mathcal{K}_t^{(n)} = \left\{ c | c \in \bar{\mathcal{K}}_t^{(n)} \text{ with the } H \text{ smallest } d_c^{(n)} \right\} \tag{4.29}
$$

where t denotes the EM iteration. $p\left(C_t | C_{t-1} = \bar{c}_n; S\right)$ is the distribution that is given by the normalised row of a cluster similarity matrix S after setting the probabilities corresponding to $\mathcal{K}^{(n)}$ to 0:

$$
p\left(C_t = c | C_{t-1} = \bar{c}_n; S\right) = \frac{S_{\bar{c}_n, c}}{\sum_{c' \in \mathcal{K}^{(n)}} S_{\bar{c}_n, c'}} \delta\left(c \notin \mathcal{K}^{(n)}\right) \tag{4.30}
$$

i.e. the distribution at time t is given by the row defined by the cluster $\bar{c}_n$ that had the minimal distance with the datapoint $\mathbf{y}^{(n)}$ at time $t-1$.

Algorithm 3 Data Similarity Gaussian Mixture Model (D-GMM)

Require: Dataset $\mathcal{X}$, # of centres M

initialise $\mu_{1:M}$, σ, $\mathcal{K}^{(n)}$ and $S = 0$ for all n;
repeat
  $\mathcal{J} = \{1, \dots, N\}$
  for $n \in \mathcal{J}$ **do**
    $\bar{c}_n = \arg\min_c \left\{ \|\mathbf{y}^{(n)} - \mu_c\|^2 | c \in \mathcal{K}^{(n)} \right\}$
    $p\left(C_t = c | C_{t-1} = \bar{c}_n; S\right) := \begin{cases} \frac{S_{\bar{c}_n, c}}{\sum_{c'} S_{\bar{c}_n, c'}} & \mathcal{K}^{(n)} \neq \emptyset \\ \frac{1}{M} & \mathcal{K}^{(n)} = \emptyset \end{cases}$
    $\mathcal{K}^{(n)} = \mathcal{K}^{(n)} \cup \left\{ c_{1:R} | c_i \sim p\left(C_t | C_{t-1} = \bar{c}_n; S\right) \text{ and } c_i \notin \mathcal{K}^{(n)} \right\}$
    for $c \in \mathcal{K}^{(n)}$ **do**
      $d_c^{(n)} = \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2$
    end for
    $\mathcal{K}^{(n)} = \left\{ c | c : \text{ with the } H \text{ smallest } d_c^{(n)} \right\}$
  end for
  Calculate $\mu_{1:M}$,σ^2, and S using Eqs. 4.31–4.33
until $\mu_{1:M}$ and σ^2 have converged

The parameter updates, from Eq. 4.26, are adapted to the approximate posterior.

$$\mu_c = \sum_{n=1}^{N} q_c^{(n)} \mathbf{y}^{(n)} / \sum_{n=1}^{N} q_c^{(n)} \tag{4.31}$$

$$\sigma^2 = \frac{1}{DN} \sum_{n=1}^{N} \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{4.32}$$

The similarity matrix is defined based on the distances $d_c^{(n)}$ under the assumption that nearby clusters have small distances to similar datapoints

$$S_{i,j} = \frac{1}{N} \sum_{n=1}^{N} \exp(-\left(d_i^{(n)} + d_j^{(n)}\right)) \delta\left(\{i, j\} \subset \mathcal{K}^{(n)}\right) \tag{4.33}$$

Eq. 4.33 produces a symmetric positive definite matrix that is used to sample datapoints near the optimal at each step of the process, with a simple reduction operation over pre-computed values. Iterating between Eq. 4.27 and the parameter updates, Eqs. 4.31-4.33, details an algorithm that we call Data Similarity GMM (D-GMM), Algo. 3, due to the similarity matrix being based on a data voting process. The complexity of an E-step of the *D-GMM* algorithm reduces compared to an E-step of the exact EM algorithm for GMMs from $\mathcal{O}(NMD)$ to $\mathcal{O}(N(R+H)D)$, where typically $R + H << M$. For the M-step, the complexity becomes $\mathcal{O}(NHD + NH^2)$ from $\mathcal{O}(NMD)$, however, as we will show in the experiments' section, $H^2 << M$ to be sufficient for most applications.

a) Implementation details

As before, during the first epoch of the proposed algorithm the sets $\mathcal{K}^{(n)}$ are initialised using prior samples. During the first epoch of the proposed algorithm the sets $\mathcal{K}^{(n)}$ are initialised using prior samples. The centres of the gaussians, $\mu_{1:C}$, are initialised using the AFK-MC^2 [248] initialisation method (Algo. 2). After an epoch has passed, the $\mathcal{K}^{(n)}$ is updated as in algorithm 3.

We can also use coresets (lwcs) to further improve computational efficiency [250, 265, 266]. The lower bound to the log-likelihood becomes:

$$\mathcal{F}\left(\mathcal{Y}', \theta\right) \triangleq \sum_{n} w_n \sum_{c \in \mathcal{K}^{(n)}} q_c^{(n)} \log p\left(C = c, Y = \mathbf{y}^{(n)} | \theta\right) + \sum_{n} w_n \mathcal{H}\left(q_c^{(n)}\right) \tag{4.34}$$

Since the parameter updates are gradient-based updates of Eq. 4.34, the weights $w_{1:N'}$ are a multiplicative constant on the parameters and therefore the parameter updates become:

$$\mu_c = \sum_{n=1}^{N'} w_n q_c^{(n)} \mathbf{y}^{(n)} / \sum_{n=1}^{N'} w_n q_c^{(n)} \tag{4.35}$$

$$\sigma^2 = \frac{1}{DN'} \sum_{n=1}^{N'} \sum_{c} w_n q_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{4.36}$$

$$S_{i,j} = \frac{1}{N'} \sum_{n=1}^{N'} w_n \exp(-\left(d_i^{(n)} + d_j^{(n)}\right)) \delta\left(\{i, j\} \subset \mathcal{K}^{(n)}\right) \tag{4.37}$$

These updates can replace Eq. 4.31, 4.32 and 4.33 in algorithm 3 to allow applications on a coreset $\mathcal{Y}'$. Working on coresets introduces an error in the approximation that has been analysed rigorously in earlier work [250]. Working on coreset reduces the complexity of D-GMM to $\mathcal{O}\left(N'\left(R + H\right)D\right)$ for the E-step and $\mathcal{O}\left(N'HD + N'H^2\right)$ for the M-step.

4.3.2 Experiments and results

We evaluate the performance of the algorithm experimentally on three classes of tasks. First, we examine convergence on artificial data where the ground truth is known. We proceed with a comparison against state-of-the-art algorithms such as vc-GMM [243, 244] on popular clustering datasets. Lastly, to demonstrate the emerging necessity of Machine Learning algorithms developed with high efficiency requirements, we present an application to a dataset extracted from event-driven visions sensors. In this regime standard k-means becomes extremely slow and we resort to efficient variants of the standard algorithm. Lastly, we present a real world application of clustering on event-driven vision sensors for feature extraction [261].

For all tasks, the Gaussian centres are initialised using the AFK-MC^2 algorithm with $m = 5$. Furthermore, we follow the same **convergence protocol** as in [243] and terminate the algorithm when the free-energy increment following Eq. 4.34 is less than $\epsilon = 10^{-3}$. Unless stated otherwise, we evaluate the stability of the results on 5 repetitions for all experiments, unless otherwise stated.

For clarity, below is a reminder for the hyperparameter notations:

- M denotes the number of centres
- N' denotes the coreset size
- H and C' denote the size of the truncated subspace for D-GMM and vc-GMM respectively
- R and G are the search space hyperparameters for D-GMM and vc-GMM respectively

When choosing the truncation hyperparameters H (C'), we consider that the probability values of the exact posterior decays exponentially and accordingly set $H = 5$ ($C' = 5$) under the assumption that lower probability values will be negligible. We follow the same rationale for the truncation updates R (G). We use various configurations for M and N' so we can compare with the state of the art.

a) Artificial data

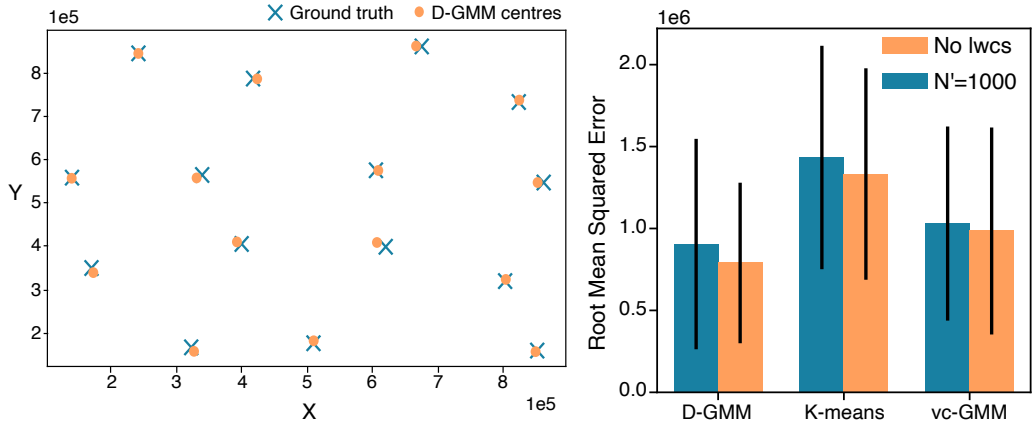


Fig. 4.9 *Validation on a two-dimensional artificial dataset.* ***(Left)*** *D-GMM recovers all* 15 *centres with* $N = 5000$, $H = 3$, $R = 6$ *and* $N' = 1000$. ***(Right)*** *D-GMM outperforms other methods across* 100 *trials and using coresets does not significantly increase the RMSE compared to the full dataset.*

In this section, we present a convergence analysis on artificial data [267] with $N = 5000$ data points and 15 Gaussian centres. Fig. 4.9 on the right shows the RMSE between the learned centres of the algorithms and the ground truth centres with and without coresets. We compare our algorithm, D-GMM, against vc-GMM, and standard k-means, setting the hyperparameters to $M = 15$, $N' = 1000$, $H = C' = 3$ and $R = G = 6$. The results suggest that both truncated algorithms are able to recover the centres as well as the exact algorithm. The slight improvement (below a standard deviation) might be attributed to the fact that a truncated approximation will "hard–code" very low probabilities to 0 which may enhance numerical stability. With the D-GMM algorithm, the stochastic behaviour might also have an effect on avoiding locally optimal solutions. In Fig. 4.9 on the left, we present an example of a run where the centres were successfully recovered.

b) Clustering analysis

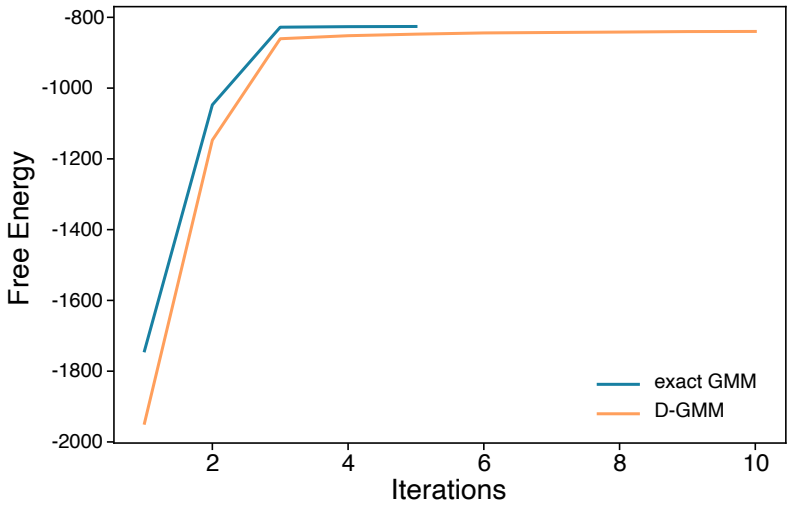


Fig. 4.10 *Free-energy evolution of D-GMM on CIFAR-10 across iterations compared to the exact algorithm. We can see that our approximation is slower to converge, however, each iteration is considerably more efficient.*

For a more detailed comparison with the state of the art, we consider a series of well-known clustering datasets. Tab. 4.3 details a comparison between k-means, vc-GMM [243, 244], and D-GMM. We use the k-means algorithm on the full dataset to define a baseline for the centres. The accuracy of the algorithms is measured using the relative quantisation error according to Eq. 4.20. Since D-GMM and vc-GMM are going through fewer clusters per datapoint in each iteration, convergence is slower for these two algorithms (Fig. 4.10). However, the efficiency of the algorithm is determined by the overall number of distance evaluations. In the last two columns of Tab. 4.3, we present the average number of iterations to convergence as well as the average speedup in terms of distance evaluations, $d_c^{(n)}$, relative to k-means. The results show a clear speedup for D-GMM in most cases and comparable relative quantisation error. Fig. 4.11 presents a comparison between the efficient clustering algorithms with increasing coreset size. K-means on the full dataset is presented as baseline. The size of each marker in Fig. 4.11 represents the size of the coreset. We find that in most cases D-GMM clusters data with a low relative quantisation error to the baseline for the least amount of distance evaluations.

Complexity The approximation method we use is focused on avoiding distance evaluations, $d_c^{(n)}$ with all available clusters. Therefore, it is very efficient in problems where a high number of clusters is expected to be present in the dataset. Fig. 4.12 (Top) shows the scaling behaviour of our algorithm with an increasing number of clusters M on the CIFAR-10 dataset, with M ranging from 100 to 1500 cluster centres. The distance evaluations for each algorithm are normalised by the minimum value across all M and presented in a log-log plot which indicates the power of the relationship between operation complexity and number of clusters. We normalised both axes for an easier visualisation of the complexity. As expected, the scaling behaviour of k-means is linear to the number of

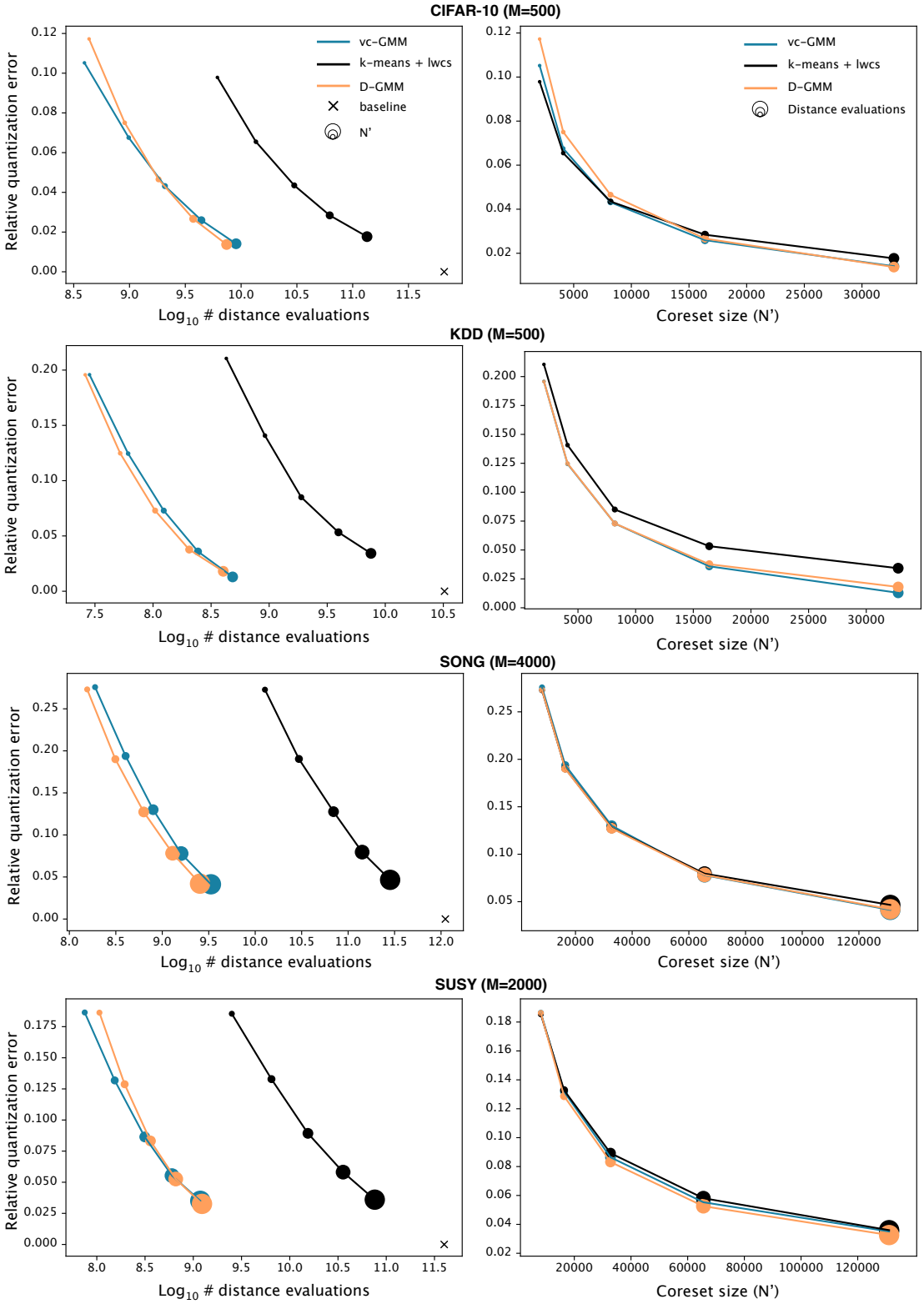
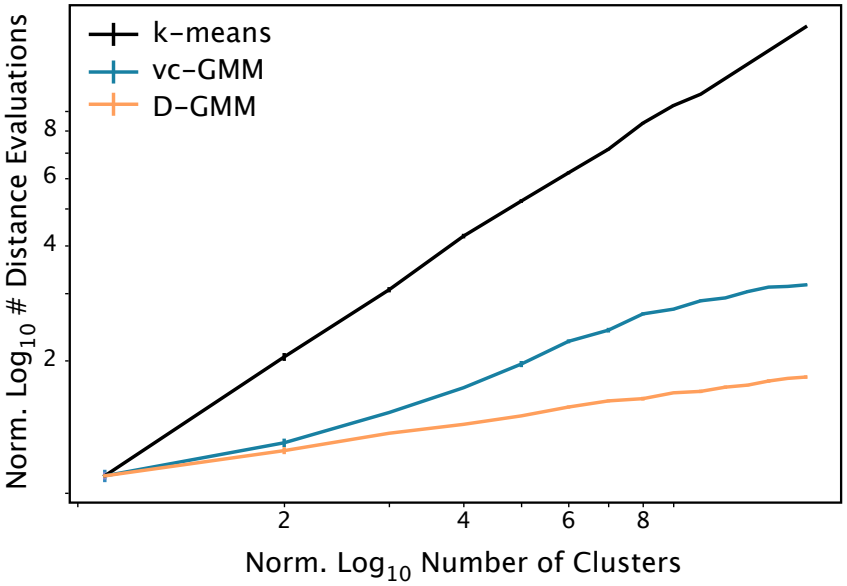


Fig. 4.11 *Distance evaluations vs relative quantisation error on increasingly large datasets.*



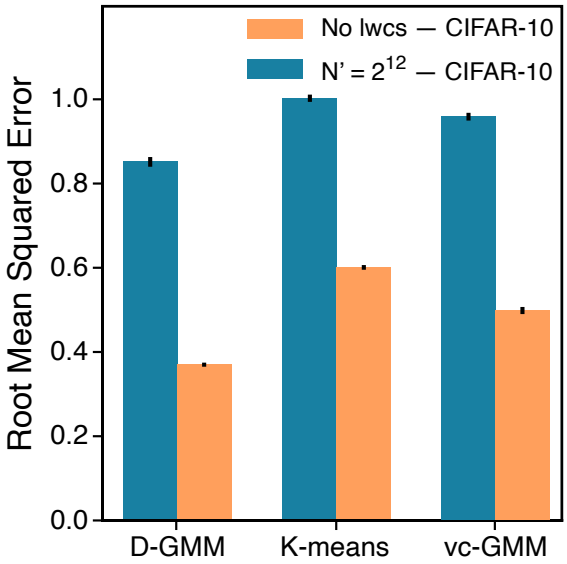


Fig. 4.12 *Operations complexity and stability on CIFAR-10 over* **100** *trials.* ***(Top)*** *Normalised log distance evaluations for an increasing M, with* 100 *trials for each M.* ***(Bottom)*** *RMSE between the learned centres ($M = 500$).* ***(Both)*** *Error bars denote* 1 *standard deviation.*

Table 4.3: Relative quantisation error and distance evaluation speedup

Dataset	Algorithm		Relative Quantisation Error η	Distance Evaluation Speedup	Iterations
	name	N'			
KDD $N = 145,751$ $D = 74$ $M = 500$	k-means	-	$0.0 \pm 0.7\%$	$\times 1.0 \pm 0.0$	5.0 ± 0.0
	k-means + lwcs	2^{12}	$14.0 \pm 0.5\%$	$\times 35.1 \pm 0.0$	$\mathbf{5.0 \pm 0.0}$
	vc-GMM	2^{12}	$\mathbf{12.0 \pm 0.4}\%$	$\times 533.1 \pm 36.5$	11.7 ± 1.0
	D-GMM	2^{12}	$\mathbf{12.0 \pm 1.0}\%$	$\times \mathbf{622.1 \pm 28.0}$	17.0 ± 0.7
CIFAR-10 $N_{\text{train}} = 50,000$ $N_{\text{test}} = 10,000$ $D = 3,072$ $M = 500$	k-means	-	$0.0 \pm 0.0\%$	$\times 1.0 \pm 0.0$	7.6 ± 0.4
	k-means + lwcs	2^{12}	$\mathbf{7.0 \pm 0.1}\%$	$\times 48.9 \pm 3.5$	$\mathbf{5.4 \pm 0.4}$
	vc-GMM	2^{12}	$\mathbf{7.0 \pm 0.0}\%$	$\times 674.7 \pm 45.8$	11.8 ± 0.8
	D-GMM	2^{12}	$8.0 \pm 0.0\%$	$\times \mathbf{731.5 \pm 41.9}$	21.4 ± 1.2
SONG $N = 515,345$ $D = 90$ $M = 4000$	k-means	-	$0.0 \pm 0.0\%$	$\times 1.0 \pm 0.0$	5.0 ± 0.0
	k-means + lwcs	2^{16}	$8.0 \pm 0.0\%$	$\times 7.8 \pm 0.0$	$\mathbf{5.0 \pm 0.0}$
	vc-GMM	2^{16}	$8.0 \pm 0.1\%$	$\times 698.2 \pm 0.7$	12.0 ± 0.0
	D-GMM	2^{16}	$8.0 \pm 0.2\%$	$\times \mathbf{862.1 \pm 18.3}$	21.7 ± 0.4
SUSY $N = 5,000,000$ $D = 18$ $M = 2000$	k-means	-	$0.0 \pm 0.0\%$	$\times 1.0 \pm 0.0$	14.7 ± 0.4
	k-means + lwcs	2^{16}	$6.0 \pm 0.1\%$	$\times 11.1 \pm 0.4$	$\mathbf{14.1 \pm 0.5}$
	vc-GMM	2^{16}	$6.0 \pm 0.1\%$	$\times \mathbf{663.1 \pm 17.1}$	25.4 ± 0.6
	D-GMM	2^{16}	$\mathbf{5.0 \pm 0.1}\%$	$\times 605.7 \pm 11.1$	55.6 ± 1.0

Note: preferred algorithm for each metric in **bold**.

clusters while the approximations are sub-linear. D-GMM is the most efficient algorithm in terms of distance evaluations as the number of cluster centres increases.

Stability We test the ability of the algorithm to recover the same clusters using different initialisation. We first run the clustering algorithm on CIFAR-10 without using coresets, with hyperparameters set to $M = 500$, $H = 5$, and $R = 5$, and we compare the recovered centres of every distinct pair of runs using the l_2-norm between the centres after reshuffling them. We then repeat the same experiment with $N' = 2^{12}$ to assess the stability of the algorithm when coresets are used. The average and standard deviation between all errors are plotted in Fig. 4.12 (Bottom). D-GMM appears to be more stable than both k-means and vc-GMM, confirming the results obtained on the artificial dataset.

4.3.3 Large scale feature extraction for classification

To test D-GMM in a more challenging setting, we use it for feature extraction on event-based vision datasets. Neuromorphic vision sensors [29] have independent pixels with a tendency to generate a lot of events with a high temporal resolution. The resulting datasets have a significantly higher number of data-points than standard vision datasets.

As features for D-GMM, we build time surfaces from events, following the same protocol outline in section 4.2.4. After clustering the time surfaces, we train a logistic regression classifier with a standardised spatial histogram of clusters. Tab. 4.4 summarises classifica-

tion results for D-GMM on popular event-based datasets, compared to vc-GMM [243] and the previously proposed S-GMM and u-S-GMM algorithms (section 4.2).

Setting the hyperparameters to $M = 1000$, $H = C' = 5$, $N' = 2^{16}$ and $R = G = 40$ for comparability with results from the previous section, the classification performance of the D-GMM is roughly equivalent to other approximation methods on most datasets, while maintaining a clear advantage in terms of computational efficiency as reported in Tab. 4.1. The best performances however, are reached by the u-S-GMM algorithm. On complicated datasets such as N-Caltech101, the stochastic approximations proposed in this chapter consistently perform better than the deterministic vc-GMM algorithm.

Table 4.4: Average classification accuracy on very large event-based datasets over 5 trials.

	POKER-DVS	**N-CARS**	**N-MNIST**	**GESTURE-DVS**	**N-CALTECH101**
D-GMM	100%	85.56%	98.42%	86.97%	55.34%
S-GMM	100%	85.69%	98.43%	86.67%	55.42%
u-S-GMM	100%	85.92%	98.42%	87.42%	55.93%
vc-GMM	100%	85.65%	98.41%	87.12%	55.01%

4.4 Discussion

We have presented two novel EM-based training algorithms for the Gaussian Mixture Model. Our algorithms considerably increase computational efficiency compared to k-means by calculating the posterior over a data-specific subset of clusters. In the case of the S-GMM algorithm (and the uniform prior variant), the subset is iteratively refined in the E-step by sampling from a uniform distribution. That way, we do not add complicated extraneous computation like other efficient approximations [243] which allows for simple implementation in a variety of widely used software frameworks. Learning prior distributions simplifies the parametrisation of our model by reducing the probability of unnecessary clusters. The method is based on the exact EM for GMMs with no additional assumptions as to the geometry of the data or the cluster centres and yet, to our knowledge, this approximation has not been previously addressed in the relevant literature.

As for the D-GMM algorithm, the subset is iteratively refined by sampling in the neighbourhood of the best performing cluster at each EM iteration. To identify the neighbourhood of each cluster we propose a similarity matrix based on earlier computed distances between the clusters and datapoints, thus avoiding additional complexity. We compare this algorithm to vc-GMM [243] which is, to our knowledge, the most efficient GMM algorithm currently available. In terms of computational complexity, D-GMM is more efficient in most cases, improving both with an increasing number of datapoints and with an increasing number of clusters compared to vc-GMM. Typical approximations of the exponential function, and probably some other elementary operations, can still be explored for further improvements to the implementation.

To maximise the potential of the proposed algorithms we have combined them with lightweight coresets and the AFK-MC^2 initialisation [250, 248], which are state-of-the-art methods in the literature for GMM centre initialisation and data pre-processing respectively.

The resulting algorithms are highly efficient and competitive in a variety of tasks, and show increased stability in recovering cluster centres compared to other approaches. The lack of dependency on sequential processing allows for straightforward parallelisation in both a CPU and GPU environment.

The speedup in operations and time complexity is particularly interesting in the context of neuromorphic computer vision applications with very large datasets. Working with neuromorphic cameras poses a unique challenge as algorithms are expected to maintain the platform's low power requirements while datasets grow in size. Our computationally efficient clustering algorithms can exploit the extremely sparse nature of event-based data by extracting the spatio-temporal neighbourhood around each event [100] and combining these features in a bag-of-features approach for classification with a simple linear model. Processing events by mean of time surfaces, rather than frames, allows for finer details in a machine learning algorithm, such as precise timing of light variation, and more generally, information about scene dynamics that are typically lost in processing with standard frame-based sensors. Our algorithms outperform most state-of-the-art unsupervised methods and are even competitive against some supervised methods on datasets such as N-MNIST, particularly when taking into consideration the reduced energy demands and faster computation.

For compatibility with neuromorphic systems, online variants of the proposed algorithms can be imagined [276]. Previous studies on spike-based EM even shows potential ways to approximate our algorithms on a spiking neural network via the spike-timing-dependent learning rule [277]. Accordingly, the ULPEC memristor-based computing system (chapter 3) could benefit from the findings in this chapter.

Chapter 5

General discussion

This work set out to explore the importance of precise timing for energy-efficient machine learning. In addition to the increased privacy and security, moving away from cloud computing and on-device inference towards on-device learning is particularly beneficial for tasks requiring fast processing and low latency. To that end, various strategies are actively being developed to reduce the computational burden of machine learning algorithms. The field of neuromorphic engineering shows great promise in resource-constrained environments by going beyond the classical Von Neumann architecture [278, 24], towards systems and computational techniques inspired by biology. Event-based vision sensors are inherently energy-efficient by eliminating redundancy in the data and generating sparse streams of events with high temporal resolution instead of dense frames. This paradigm shift opens the door for novel machine learning applications that were simply not possible before such as: ultra-low latency object detection [1]; spatio-temporal pattern recognition under extreme lighting conditions [279]; and on-device real-time gesture recognition [31].

Instead of assessing event-based learning algorithms on the same tasks as artificial neural networks, it is important to find scenarios where precise timing actually makes sense. We explore in chapter 2, a simple yet elegant solution to discrete time series classification compared to methods based on conventional machine learning [280, 281], by directly leveraging conduction delays in a spiking neural network. To this end, we implement **Hummus**, a spiking neural network simulator that allows us to work with both synaptic weights and conduction delays in an event-based manner (see Appendix A). As an added benefit, the simulator is interfaced with a popular machine learning framework, allowing us to build complex neuron models for online classification.

The myelin plasticity model is an unsupervised delay learning rule for spiking neural networks, that speeds up learning by avoiding the strict dependency on a fully-connected architecture as seen in the relevant literature [172, 175]. We validate the myelin plasticity model on a touch localisation task with real data gathered from an array of eight spatially-separated piezoelectric sensors. In this particular context, delay learning is significantly more efficient than the alternative unsupervised approach explored in [189] without compromising the results. The myelin plasticity model can be made to work on more complicated data by including a bio-inspired weight update rule that assigns importance

to synapses and extracts spatio-temporal patterns from noisy signals.

Spiking neural networks need to be simulated on massively parallel neuromorphic architectures such as SpiNNaker [107] and the Intel Loihi chip [25] as they are inefficient on simple CPUs. Implementations of delay learning models on neuromorphic hardware have been previously tested [282] and were shown to be effective at tasks such as real-time robot path planning [283]. Neuromorphic computing systems improve speed and power efficiency by limiting data movement through tightly-coupled memory or in-memory computing architectures. These systems can be improved even further by using memristors of CMOS technology [284]. Memristors were originally designed for non-volatile memory applications. That however, has proven challenging and is a field of active research [285]. While looking for alternative use cases, scientists thought of leveraging the resistance switching properties of memristors for learning applications [201, 229]. In collaboration with partners on the European ULPEC project, we explore various architectures and learning strategies for a hardware spiking neural network with memristive synapses that operate with ultra-low power and latency. An event-based vision sensor is combined with a memristor crossbar array connected to a layer of analog CMOS neurons. We deliver a first attempt at using ferroelectric memristors for unsupervised learning in the context of automated driving.

So far, memristors have been held back by reliability, speed and endurance issues. In contrast to molecular and ionic thin film memristors, ferroelectric memristors are not plagued by such issues. Resistance switching is a purely electronic process and does not rely on deep structural changes such as the formation or breakdown of conductive filaments. We briefly explore the feasibility of the myelin plasticity model on a memristive crossbar array before moving on to alternative energy-efficient algorithms. Current waveforms being injected into analog neurons need to be shifted by an adjustable amount of time in order to implement learnable delays. The absence of a memory in an analog neuron architecture makes this task difficult. A mixed-signal approach is therefore required for memristor-based delay learning.

Instead of adding complexity to our neuromorphic computing system, we turn to synaptic plasticity mechanisms that are directly compatible with memristors. Studies show that the resistance switching properties of memristors can be leveraged to emulate spike-timing-dependent plasticity (STDP) [201, 213], a biological mechanism commonly associated with learning [219]. More specifically, memristor crossbar arrays behave similarly to an STDP-based spiking neural network as seen in chapter 3. The crossbar design inherently imposes a full connectivity scheme that is not ideal in data-intensive scenarios. Considering event-based vision sensors are able to generate millions of events per second, we propose a sparsely-connected crossbar architecture that:

- improves energy-efficiency by reducing the number of memristive synapses that need to be simultaneously programmed
- improves classification performance by taking advantage of the local statistics in a natural scene

When labelled datasets are available [1, 202], the classification performance of the memristive neural network can be improved by shifting to supervised learning methods. A

reward-modulated STDP [119] can be easily implemented on the ULPEC neuromorphic computing system without affecting its overall design.

Memristor-based computing systems are not without drawbacks. The weight precision, which heavily affects classification performance, is determined by the number of resistance states that can be stabilised. We address this issue by implementing a time-invariant STDP that quantises weights via square waveforms. However, and depending on the physical properties of the memristive device, the weight precision of the network could be further limited to one or two bits. We can mitigate this issue through low precision STDP rules and threshold adaptation mechanisms [286], but these strategies are never really competitive on realistic classification tasks.

A better solution is to move away from Hebbian learning, towards state-of-the-art multilayer spiking neural networks trained with backpropagation-based methods [128, 92, 93, 129]. A mixed signal implementation of quantisation-aware training can even be explored when the weight precision of memristive spiking neural networks is too low [287, 288]. Despite the expected increase in power consumption, a memristive computing system with backpropagation-based learning would still be more efficient than other hardware solutions with equivalent classification performance, particularly during inference.

To reduce the energy bottleneck, and for compatibility with neuromorphic systems, backpropagation-based spiking neural networks address two major concerns with the classical algorithm. First, the requirement for symmetrical weights between forward and backward passes, also known as the weight transport problem [289], is avoided by using fixed random weights in backward connections, as first outlined in [290]. Second, deep layers are trained using local errors to bypass the update-locking problem where a full forward pass is required before updating the parameters [93].

Event-based and frame-based methods are not necessarily mutually exclusive and can actually complement each other. For instance, multi-sensor devices need to be able to work with data derived from different sources. Versatile solutions that take advantage of decades of machine learning research to handle both frames and events, are well sought after. When working with massive streams of events, a typical pipeline involves reducing the computational burden by building intensity images from events or working at fixed time intervals [291], and using conventional machine learning afterwards. This is largely due to the time complexity of event-by-event algorithms [100, 1, 98] which quickly becomes prohibitive after a certain number of events. This issue is not well addressed in the literature as the neuromorphic community relies on small datasets and event-based vision sensors with low spatial resolutions. With the advent of high definition neuromorphic cameras [26, 72, 27], event-by-event algorithms need to be rethought.

With this in mind, we devise two sublinear complexity clustering algorithms, and a C++ machine learning framework for stochastic learning of Gaussian mixtures. The framework, **Peregrine**, is open source and can be used by the community to quickly cluster very large datasets without requiring much memory. The proposed algorithms set the state of the art in terms of computational-efficiency and improve stability compared to similar clustering methods [261, 243]. We demonstrate the effectiveness of these algorithms in extremely demanding settings by applying them on large event-based datasets to obtain

features for classification using a simple linear classifier. Event-based algorithms that construct descriptors reminiscent of local image patches [100, 1, 98, 99] currently rely on such bag-of-feature approaches for classification tasks.

The state of the art in image classification and several other processing tasks is given by deep learning [7, 8]. For autonomous systems such as self-driving cars, it is often interesting to have a verifiable model. Gaussian mixture models coupled with a convex logistic regression classifier have been used in the past [292] and are considerably simpler to interpret than deep learning models. Recent advances in the field [293, 294, 295, 296] support our approach and suggest that combining well-designed feature extractors with a single linear layer of classification leads to a rather small difference in accuracy compared to convolutional neural networks.

As part of an online learning system, our efficient clustering methods can be implemented on field-programmable gate arrays [297], and modified to use incremental parameter updates. Taking it a step further, we can even use our findings to improve the computational efficiency of popular event-by-event algorithms. For instance, the iterative online clustering method introduced in HOTS [100, 31] can be refactored to work on truncated sets (chapter 4) instead of having to compute all distances at every event, paving the way for online clustering algorithms suitable for event-based datasets with a very large number of centres and datapoints.

In closing, co-designing hardware and software seems to be a necessary step for on-device learning. Neuromorphic systems and more specifically memristor crossbar arrays are thought to be an ideal solution to the machine learning bottleneck [298], particularly when coupled with temporal learning rules such as synaptic plasticity. These systems are still in the research phase and a couple of issues need to addressed before they can reach maturity: **(i)** the memristor fabrication **(ii)** the scalability of the system; and **(iii)** the performance of event-based learning rules. Nevertheless, we can already reach reasonable accuracies on handwritten digit recognition tasks. When a system is tailored for a very specific task, it is better to bypass on-device learning and train state-of-the-art artificial neural networks instead, which are then transferred to memristive systems for very efficient inference [126]. The neuromorphic community tends to greatly focus on classification tasks where supervised learning is clearly advantageous. However, machine learning offers opportunities beyond classification. Clustering methods are often used in unsupervised settings such as spike sorting, image segmentation and even for medical diagnosis to find common patterns between patients [299]. Unsupervised learning rules for spiking neural networks are not yet competitive with conventional unsupervised machine learning methods. Bridging the gap between both approaches, stochastic approximations of Gaussian mixture models allow us to efficiently handle a wide variety of unlabelled datasets including massive streams of events. These techniques can be optimised to work in resource-constrained environments.

Appendix A

Hummus: event-based spiking neural network simulator

Every experiment described in chapters 2 and 3 were done using *Hummus*, an open source spiking neural network simulator coded using the C++17 standard, and built first and foremost for neuromorphic computer vision and pattern recognition tasks. Hummus was born out of the inflexibility of other simulators to adapt according the needs of the neuromorphic engineering field, whether by placing too many constraints for biological realism, or by limiting compatibility with the event-based paradigm for performance reasons. We wanted to easily explore precise timing-based learning rules that make use of conduction delays in addition to synaptic weights, and work with neurons that include non-linear event-based current dynamics without having to delve into endless lines of code. In that regard, Hummus was developed with two goals in mind: **flexibility** and **simplicity**.

Through polymorphic classes (Fig. A.1), we made it easy to extend the simulator with new add-ons, learning rules, neuron models or synapse types in a completely separate header file without having to make change to the main code. Each polymorphic class has a set of virtual methods acting as messages that can be used throughout the network for different purposes such as indicating when a neuron spikes or when a learning rule is ready to be activated. Furthermore, we interfaced the simulator with the Sepia I/O library which allows us to directly use the output of event-based cameras in a spiking neural network. Fig. A.2 demonstrates the simplicity of setting up an SNN for classifying the output of event-based cameras. Indeed, all the complexity is hidden in the back end of the simulator and this code snippet only showcases a few of the available methods to set up an SNN but reservoir computing, recurrent connections and convolutional and pooling layers can all be easily initialised.

As an added option, the simulator is interfaced with libtorch, the C++ front end of the PyTorch machine learning framework. We've already implemented a logistic regression-based online classification neuron model in the simulator to improve pattern classification performance. It is important to note that the simulator is a work in progress and performance improvements need to be made through parallelisation and vectorisation C++

libraries.

The simulator is able to run in a clock-based mode where neurons are updated at fixed intervals, and in an event-based mode where neurons are asynchronously updated with incoming spikes. In that regard, we implemented a fully event-based leaky integrate and fire neuron with complex current dynamics and synapses that can have adjustable conduction delays. The clock-based mode is particularly useful for neuron models with a membrane equation that does not have an analytical solution, such as the model using for the myelin plasticity delay learning rule presented in chapter 2. In this case, the solution is approximated with a numerical solution that relies on updating the neurons at fixed time intervals.

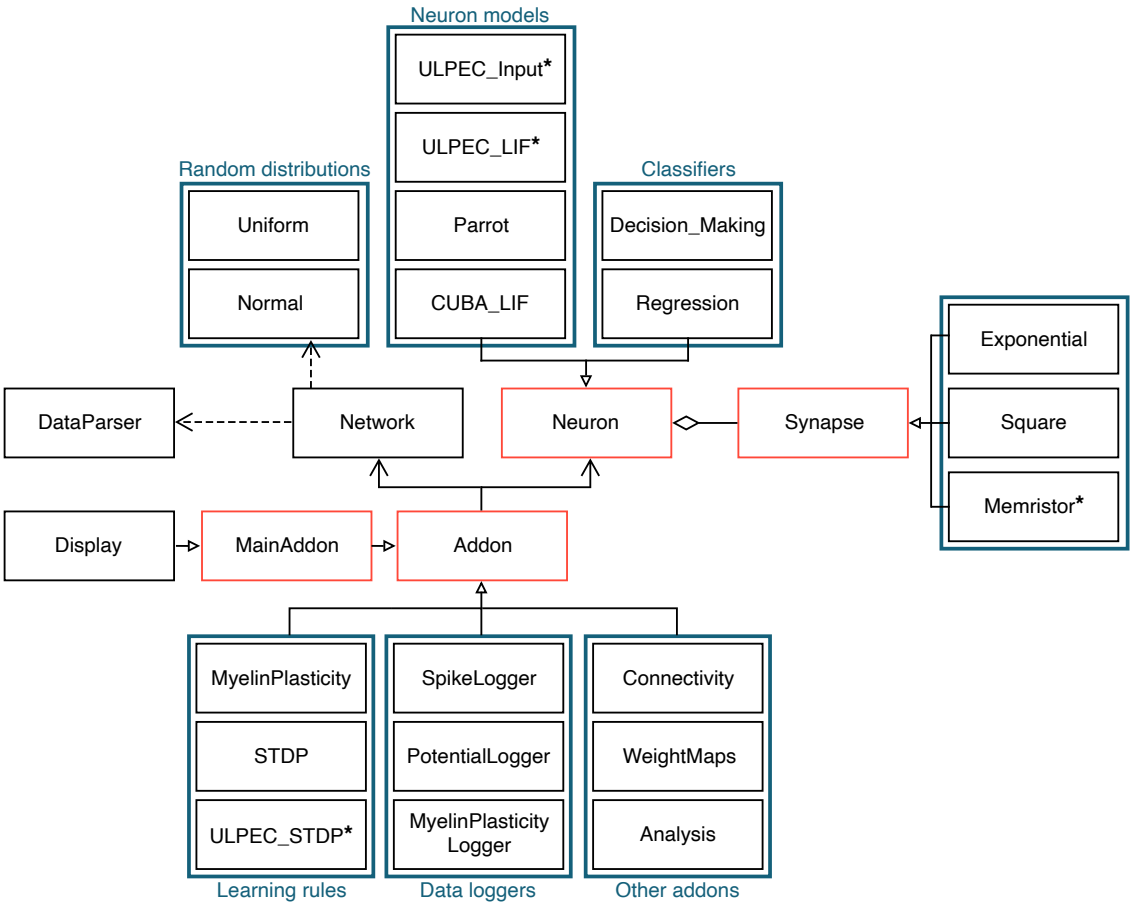


Fig. A.1 *UML diagram for the Hummus spiking neural network simulator coded in C++17. Polymorphic classes that can easily be extended are shown in red. The random distribution classes are used to initialise synaptic weights and delays in methods of the Network class that connect layers of neurons. The only external dependencies are Qt5 when the GUI is used (Display class), and PyTorch's C++ frontend when a logistic regression classifier is needed (Regression class). Classes marked by an asterisk are specific to the memristor-based visual data processing system seen in chapter 3.*

```cpp
#include <iostream>
#include "../source/core.hpp"
#include "../source/neurons/parrot.hpp"
#include "../source/neurons/cuba_lif.hpp"
#include "../source/neurons/regression.hpp"
#include "../source/learning_rules/stdp.hpp"
#include "../source/addons/analysis.hpp"

int main(int argc, char** argv) {
    using namespace hummus;

    // load data
    DataParser parser;
    auto training_dataset = parser.load_data("path/to/dataset/train");
    auto test_dataset     = parser.load_data("path/to/dataset/test");

    // initialise add-ons and STDP learning rule
    Network network;
    auto& analysis = network.make_addon<Analysis>(test_dataset.labels);
    auto& stdp     = network.make_addon<STDP>;

    // LIF layer parameters
    int n          = 100 // number of neurons
    int tau_ref    = 3; // refractory period (ms)
    float C        = 200; // capacitance (pF)
    float Gl       = 10; // leak conductanec (nS)
    bool WTA       = true; // enables winner-takes-all mechanism

    // STDP-based 2-layer spiking neural network with all to all connected synapses
    auto input  = network.make_grid<Parrot>(28, 28); // 2D input layer of 784 neurons
    auto output = network.make_layer<CUBA_LIF>(n, {&stdp}, tau_ref, C, Gl, WTA);
    network.all_to_all<Square>(input, output, Uniform(0, 1));

    // pytorch logistic regression classifier trained on output spikes
    auto classifier = network.make_logistic_regression<Regression>(training_dataset,
                                                                   test_dataset);
    // run network and print classification accuracy
    network.run_es_database(training_dataset.files, test_dataset.files);
    analysis.accuracy();

    return 0;
}
```

Fig. A.2 *Code snippet showcasing the simplicity of Hummus in pattern recognition tasks. In 12 lines we can* ***(i)*** *read training and test datasets originating from event-based cameras (Event Stream format),* ***(ii)*** *initialise a 2-layer network with all-to-all connected STDP synapses characterised by square-shaped excitatory postsynaptic currents (EPSC) and a winner-take-all scheme on the output layer,* ***(iii)*** *use a PyTorch logistic regression classifier, trained on the output spikes, as an extra layer of the SNN for online classification purposes, and finally* ***(iv)*** *we can directly get the classification accuracy once running is complete. For simplicity, we use default values for all functions but everything can be parametrised.*

Appendix B

Solution to the LIF membrane potential equation

The leaky integrate and fire neuron used by the myelin plasticity delay learning rule presented in chapter 2, is described by the following differential equation:

$$\tau_m \cdot \frac{dV(t)}{dt} = E_L - V(t) + R_m \cdot I(t) \tag{B.1}$$

Where E_L is the resting membrane potential, R_m indicates the membrane resistance and $I(t)$ is the input current at time t. In order to get an exponential excitatory post-synaptic current (EPSC) shape, $I(t)$ takes the form:

$$I\left(t\right) = I_{inj} \cdot \sum_i w_i \cdot e^{-\frac{t-s_i}{\tau_{syn}}} \cdot \mathcal{H}\left(t\right) \tag{B.2}$$

where I_{inj} is the injected current every time a neuron fires, w_i indicates the synaptic weight of synapse i, τ_{syn} is the synaptic time constant, and $\mathcal{H}(t)$ is the Heaviside step function which represents the discontinuous behaviour of a spike. For a single synapse i and by focusing on the range $[s_i, t]$ where $\mathcal{H}(t) = 1$:

$$I_i(t) = I_{inj} \cdot w_i \cdot e^{-\frac{t-s_i}{\tau_{syn}}} \tag{B.3}$$

In order to solve the differential equation, we start with the standard form below:

$$\frac{dV_i(t)}{dt} + \frac{1}{\tau_m} \cdot V_i(t) = \frac{E_L + R_m \cdot I_{inj} \cdot w_i \cdot e^{-\frac{t-s_i}{\tau_{syn}}}}{\tau_m} \tag{B.4}$$

We set $P(t) = \frac{1}{\tau_m}$ and $Q(t) = \frac{E_L + R_m \cdot I_{inj} \cdot w_i \cdot e^{-\frac{t-s_i}{\tau_{syn}}}}{\tau_m}$. We can now determine the integrating factor $\lambda(t)$ such that:

$$\begin{aligned}\lambda(t) &= e^{\int_{s_i}^{t} P(\bar{t})d\bar{t}} \\ &= e^{\frac{t-s_i}{\tau_m}}\end{aligned} \tag{B.5}$$

Assuming initial conditions such that $V_i(s_i) = E_L$ the general solution for the differential equation takes the form:

$$\begin{aligned}V_i(t) &= \frac{1}{\lambda(t)}\left[\int_{s_i}^{t}\lambda(\bar{t})\cdot Q(\bar{t})d\bar{t} + C\right] \\ &= e^{-\frac{t-s_i}{\tau_m}}\left[\int_{s_i}^{t} e^{\frac{\bar{t}-s_i}{\tau_m}}\cdot\frac{E_L + R_m\cdot I_{inj}\cdot w_i\cdot e^{-\frac{\bar{t}-s_i}{\tau_{syn}}}}{\tau_m}d\bar{t} + C\right] \\ &= e^{-\frac{t-s_i}{\tau_m}}\left[\frac{E_L}{\tau_m}\cdot e^{-\frac{s_i}{\tau_m}}\int_{s_i}^{t} e^{\frac{\bar{t}}{\tau_m}}d\bar{t} + \frac{R_m\cdot I_{inj}\cdot w_i}{\tau_m}\int_{s_i}^{t} e^{\frac{\bar{t}-s_i}{\tau_m}}\cdot e^{-\frac{\bar{t}-s_i}{\tau_{syn}}}d\bar{t} + C\right] \\ &= e^{-\frac{t-s_i}{\tau_m}}\Big[E_L\cdot e^{-\frac{s_i}{\tau_m}}(e^{\frac{t}{\tau_m}} - e^{\frac{s_i}{\tau_m}}) + \frac{R_m\cdot I_{inj}\cdot w_i}{\tau_m}\cdot e^{\frac{s_i}{\tau_{syn}}}\cdot e^{-\frac{s_i}{\tau_m}} \\ &\qquad\qquad \int_{s_i}^{t} e^{\frac{\bar{t}}{\tau_m}}\cdot e^{-\frac{\bar{t}}{\tau_{syn}}}d\bar{t} + C\Big] \\ &= e^{-\frac{t-s_i}{\tau_m}}\Big[E_L\cdot e^{-\frac{s_i}{\tau_m}}(e^{\frac{t}{\tau_m}} - e^{\frac{s_i}{\tau_m}}) + \frac{R_m\cdot I_{inj}\cdot w_i\cdot\tau_{syn}}{\tau_m - \tau_{syn}}\cdot e^{\frac{s_i}{\tau_{syn}}}\cdot e^{-\frac{s_i}{\tau_m}} \\ &\qquad\qquad (e^{\frac{s_i}{\tau_m}}\cdot e^{-\frac{s_i}{\tau_{syn}}} - e^{\frac{t}{\tau_m}}\cdot e^{-\frac{t}{\tau_{syn}}}) + C\Big] \\ &= E_L - E_L\cdot e^{-\frac{t-s_i}{\tau_m}} + \frac{R_m\cdot I_{inj}\cdot w_i\cdot\tau_{syn}}{\tau_m - \tau_{syn}}\cdot(e^{-\frac{t-s_i}{\tau_m}} - e^{-\frac{t-s_i}{\tau_{syn}}}) + Ce^{-\frac{t-s_i}{\tau_m}}\end{aligned} \tag{B.6}$$

Solving for $V_i(s_i)$:

$$C = E_L \tag{B.7}$$

The solution to the membrane differential equation becomes:

$$V_i(t) = E_L + \frac{R_m\cdot I_{inj}\cdot w_i\cdot\tau_{syn}}{\tau_m - \tau_{syn}}\cdot(e^{-\frac{t-s_i}{\tau_m}} - e^{-\frac{t-s_i}{\tau_{syn}}}) \tag{B.8}$$

Appendix C

Deriving GMM parameter updates

In the Gaussian mixture models (GMM) presented in section 4.2, for a dataset $\mathcal{Y} = \left\{\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N)}\right\}$ with M Gaussian distributions, each one is defined as:

$$p\left(Y = \mathbf{y}^{(n)} | C = c; \theta\right) = (2\pi\sigma^2)^{\frac{D}{2}} \exp\left(-\frac{\left\|\mathbf{y}^{(n)} - \mu_c\right\|^2}{2\sigma^2}\right) \tag{C.1}$$

where $\theta = \{\mu_{1:M}, \Sigma_{1:M}, \alpha_{1:M}\}$ indicates the model parameters. The prior distribution α_c is defined as:

$$p\left(C = c; \theta\right) = \alpha_c \tag{C.2}$$

where $\sum_c \alpha_c = 1$. The variational Gaussian mixture models presented in section 4.2 rely on the Expectation Maximisation algorithm to optimise the free energy which is a lower bound to the log likelihood:

$$\mathcal{F}\left(\mathcal{Y}, \theta\right) \triangleq \sum_{n=1}^{N} \sum_{c \in \mathcal{K}^{(n)}} p_c^{(n)} \log p\left(Y = \mathbf{y}^{(n)} | C = c; \theta\right) p\left(C = c | \theta\right) + \sum_{n=1}^{N} \mathcal{H}\left(p_c^{(n)}\right) \tag{C.3}$$

$$\triangleq \sum_{n=1}^{N} \sum_{c \in \mathcal{K}^{(n)}} p_c^{(n)} \log \left(\frac{1}{(2\pi\sigma_c^2)^{\frac{D}{2}}} \exp\left(-\frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{2\sigma_c^2}\right) \alpha_c\right) + \sum_{n=1}^{N} \mathcal{H}\left(p_c^{(n)}\right) \tag{C.4}$$

$$\tag{C.5}$$

where $p_c^{(n)}$ is the posterior distribution and $\mathcal{H}\left(p_c^{(n)}\right)$ indicates the entropy.

In order to reduce the complexity of our GMM algorithms we use an approximation technique that avoids calculating the full posterior distribution. Instead, we estimate $p_c^{(n)}$ over a subset $\mathcal{K}^{(n)} \subset \{1, \ldots, M\}$, with $|\mathcal{K}^{(n)}| = H$. The approximation $q_c^{(n)}$ to the full posterior $p_c^{(n)}$ becomes:

$$q_c^{(n)} = \frac{\left(2\pi\sigma_c^2\right)^{-D/2} \exp\left(-d_c^{(n)}/2\sigma_c^2\right) \alpha_c}{\sum_{c' \in \mathcal{K}^{(n)}} \left(2\pi\sigma_{c'}^2\right)^{-D/2} \exp\left(-d_{c'}^{(n)}/2\sigma_{c'}^2\right) \alpha_{c'}} \delta\left(c \in \mathcal{K}^{(n)}\right) \tag{C.6}$$

Reminder on posterior probabilities:

$$\begin{aligned} p(x,y) &= p(y,x) \Leftrightarrow \\ p(x|y)p(y) &= p(y|x)p(x) \Leftrightarrow \\ p(x|y) &= \frac{p(y|x)p(x)}{p(y)} \Leftrightarrow \\ p(x|y) &= \frac{p(y|x)p(x)}{\sum_x p(x,y)} \Leftrightarrow \\ p(x|y) &= \frac{p(y|x)p(x)}{\sum_x p(y|x)p(x)} \end{aligned}$$

where x is the cluster centre and y is a datapoint. $p(y|x)$ defines the Gaussian of a particular cluster and $p(x)$ is the probability of that cluster.

The model parameters are updated according to the following equations:

$$\mu_c = \frac{\sum_{n=1}^{N} q_c^{(n)} \mathbf{y}^{(n)}}{\sum_{n=1}^{N} q_c^{(n)}} \tag{C.7}$$

$$\sigma_c^2 = \frac{1}{D \sum_n q_c^{(n)}} \sum_{n=1}^{N} q_c^{(n)} \left\| \mathbf{y}^{(n)} - \mu_c \right\|^2 \tag{C.8}$$

$$\alpha_c = \frac{\sum_{n=1}^{N} q_c^{(n)}}{\sum_{n=1}^{N} \sum_{c'=1}^{M} q_{c'}^{(n)}} \tag{C.9}$$

C.1 Deriving the variance σ^2

We differentiate the lower bound to the log likelihood with respect to σ_c:

$$\frac{\partial}{\partial \sigma_c} \mathcal{F}(\mathcal{Y}, \theta) = \sum_{n=1}^{N} \frac{\partial}{\partial \sigma_c} \left\langle -\frac{D}{2} \log(2\pi\sigma_c^2) - \frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{2\sigma_c^2} \right\rangle_q \tag{C.10}$$

$$= \sum_{n=1}^{N} \left(-q_c^{(n)} \frac{D}{2} \frac{2\pi\sigma_c}{\pi\sigma_c^2} + q_c^{(n)} \frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{\sigma_c^3} \right) \tag{C.11}$$

$$= -\sum_{n=1}^{N} q_c^{(n)} \frac{D}{\sigma_c} + \sum_{n=1}^{N} q_c^{(n)} \frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{\sigma_c^3} \tag{C.12}$$

where $<>_q$ denotes expectation with respect to $q_c^{(n)}$.

We now compare the gradient of the free energy to 0:

$$-\sum_{n=1}^{N} q_c^{(n)} \frac{D}{\sigma_c} + \sum_{n=1}^{N} q_c^{(n)} \frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{\sigma_c^3} = 0 \tag{C.13}$$

$$\Leftrightarrow \sum_{n=1}^{N} q_c^{(n)} \frac{D}{\sigma_c} = \sum_{n=1}^{N} q_c^{(n)} \frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{\sigma_c^3} \tag{C.14}$$

$$\Leftrightarrow \frac{\sigma_c^3}{\sigma_c} \sum_{n=1}^{N} q_c^{(n)} D = \sum_{n=1}^{N} q_c^{(n)} \|\mathbf{y}^{(n)} - \mu_c\|^2 \tag{C.15}$$

$$\Leftrightarrow \sigma_c^2 = \frac{1}{D \sum_{n=1}^{N} q_c^{(n)}} \sum_{n=1}^{N} q_c^{(n)} \|\mathbf{y}^{(n)} - \mu_c\|^2 \tag{C.16}$$

$$\tag{C.17}$$

C.2 Deriving the mean μ

Similarly, we start by differentiating the free energy with respect to μ_c:

$$\frac{\partial}{\partial \mu_c} \mathcal{F}(\mathcal{Y}, \theta) = -\sum_{n=1}^{N} \frac{\partial}{\partial \mu_c} \left\langle \frac{\|\mathbf{y}^{(n)} - \mu_c\|^2}{2\sigma_c^2} \right\rangle_q = \tag{C.18}$$

$$= -\sum_{n=1}^{N} q_c^{(n)} \frac{2\left(\mathbf{y}^{(n)} - \mu_c\right)}{2\sigma_c^2} \tag{C.19}$$

$$= -\sum_{n=1}^{N} q_c^{(n)} \frac{\mathbf{y}^{(n)}}{\sigma_c^2} + \sum_{n=1}^{N} q_c^{(n)} \frac{\mu_c}{\sigma_c^2} \tag{C.20}$$

We solve the equation by setting it to 0:

$$-\sum_{n=1}^{N} q_c^{(n)} \frac{\mathbf{y}^{(n)}}{\sigma_c^2} + \sum_{n=1}^{N} q_c^{(n)} \frac{\mu_c}{\sigma_c^2} = 0 \tag{C.21}$$

$$\Leftrightarrow \sum_{n=1}^{N} q_c^{(n)} \frac{\mu_c}{\sigma_c^2} = \sum_{n=1}^{N} q_c^{(n)} \frac{\mathbf{y}^{(n)}}{\sigma_c^2} \tag{C.22}$$

$$\Leftrightarrow \sum_{n=1}^{N} q_c^{(n)} \mu_c = \sum_{n=1}^{N} q_c^{(n)} \mathbf{y}^{(n)} \tag{C.23}$$

$$\Leftrightarrow \mu_c = \frac{\sum_{n=1}^{N} q_c^{(n)} \mathbf{y}^{(n)}}{\sum_{n=1}^{N} q_c^{(n)}} \tag{C.24}$$

C.3 Deriving the prior distribution α

To derive the prior update, we constrain the free energy gradient such that $\sum_{c'=1}^{M} \alpha_{c'} = 1$ using the method of Lagrange multipliers:

$$\frac{\partial}{\partial \alpha_c}\left(\mathcal{F}(\mathcal{Y}, \theta) + \lambda\left(\sum_{c'=1}^{M} \alpha_{c'} - 1\right)\right) = \sum_{n=1}^{N} \frac{\partial}{\partial \alpha_c} \langle \log \alpha_c \rangle_q + \lambda \frac{\partial}{\partial \alpha_c}\left(\sum_{c'=1}^{M} \alpha_{c'} - 1\right) \tag{C.25}$$

$$= \sum_{n=1}^{N} \frac{q_c^{(n)}}{\alpha_c} + \lambda \tag{C.26}$$

We now set this equation to 0:

$$\sum_{n=1}^{N} \frac{q_c^{(n)}}{\alpha_c} + \lambda = 0 \tag{C.27}$$

$$\Leftrightarrow \alpha_c = -\frac{\sum_{n=1}^{N} q_c^{(n)}}{\lambda} \tag{C.28}$$

Knowing that $\sum_{c'=1}^{M} \alpha_{c'} = 1$, we can solve for λ:

$$\sum_{c'=1}^{M} \alpha_{c'} = 1 \tag{C.29}$$

$$\Leftrightarrow \sum_{c'=1}^{M} -\frac{\sum_{n=1}^{N} q_{c'}^{N}}{\lambda} = 1 \tag{C.30}$$

$$\Leftrightarrow \lambda = -\sum_{c'=1}^{M} \sum_{n=1}^{N} q_{c'}^{N} \tag{C.31}$$

The prior update equation becomes:

$$\alpha_c = \frac{\sum_{n=1}^{N} q_c^{(n)}}{\sum_{c'=1}^{M} \sum_{n=1}^{N} q_{c'}^{N}} \tag{C.32}$$

Mean μ and variance σ^2 updates for section 4.3 are derived following the same steps, but the algorithm uses uniform priors where: $p(C = c) = \alpha_c = \frac{1}{M}, \forall c \in \{1, \ldots, M\}$.

Bibliography

[1] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "Hats: Histograms of averaged time surfaces for robust event-based object classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1731–1740. v, 9, 10, 11, 14, 32, 33, 48, 50, 51, 63, 77, 79, 91, 92, 93, 94

[2] L. G. Roberts, "Machine perception of three-dimensional solids," Ph.D. dissertation, Massachusetts Institute of Technology, 1963. xi

[3] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. xi

[4] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962. xi, 4, 51

[5] K. Fukushima, "A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, pp. 193–202, 1980. xi, 4, 51

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. xi, 5, 9

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. xi, 5, 9, 14, 51, 94

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. xi, 77, 94

[9] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017. xi

[10] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015. xi

[11] G. Marcus, "Deep learning: A critical appraisal," *arXiv preprint arXiv:1801.00631*, 2018. xi

[12] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *arXiv preprint arXiv:1906.02243*, 2019. xi

[13] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, p. 9, 2016. xi

[14] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018. xi

[15] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov *et al.*, "Benchmarking tinyml systems: Challenges and direction," *arXiv preprint arXiv:2003.04821*, 2020. xi

[16] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, pp. 1–12. xi, 14

[17] S. Rivas-Gomez, A. J. Pena, D. Moloney, E. Laure, and S. Markidis, "Exploring the vision processing unit as co-processor for inference," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2018, pp. 589–598. xi

[18] W. Maass, C. H. Papadimitriou, S. Vempala, and R. Legenstein, *Brain Computation: A Computer Science Perspective*. Cham: Springer International Publishing, 2019, pp. 184–199. [Online]. Available: https://doi.org/10.1007/978-3-319-91908-9_11 xi, 1

[19] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike Timing Dependent Plasticity Finds the Start of Repeating Patterns in Continuous Spike Trains," *PLoS ONE*, vol. 3, no. 1, pp. e1377–9, Jan. 2008. xi, 13, 14

[20] R. Brette, "Computing with neural synchrony," *PLoS Comput Biol*, vol. 8, no. 6, p. e1002561, 2012. xi

[21] C. A. Mead and M. A. Mahowald, "A silicon model of early visual processing," *Neural networks*, vol. 1, no. 1, pp. 91–97, 1988. xii, 7

[22] M. A. Mahowald, "Silicon retina with adaptive photoreceptors," in *Visual information processing: from neurons to chips*, vol. 1473. International Society for Optics and Photonics, 1991, pp. 52–58. xii, 7

[23] M. Mahowald, "Vlsi analogs of neuronal visual processing: a synthesis of form and function," Ph.D. dissertation, California Institute of Technology Pasadena, 1992. xii, 7

[24] T. Delbruck, Y. Hu, and Z. He, "V2e: From video frames to realistic dvs event camera streams," *arXiv preprint arXiv:2006.07722*, 2020. xii, 8, 91

[25] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018. xii, 12, 14, 92

[26] J. Huang, M. Guo, and S. Chen, "A dynamic vision sensor with direct logarithmic output and full-frame picture-on-demand," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4. xii, xiii, 9, 11, 31, 61, 93

[27] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff, H. Takahashi, H. Wakabayashi, Y. Oike, and C. Posch, "5.10 a 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86µm pixels, 1.066geps readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 112–114. xii, xiii, 9, 11, 61, 93

[28] S.-C. Liu and A. van Schaik, *Neuromorphic Sensors, Cochlea*. New York, NY: Springer New York, 2013, pp. 1–5. [Online]. Available: https://doi.org/10.1007/978-1-4614-7320-6_118-1 xii

[29] C. Posch, D. Matolin, and R. Wohlgenannt, "A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, 2010. xii, 9, 32, 37, 77, 88

[30] T. Wunderlich, A. F. Kungl, E. Müller, A. Hartel, Y. Stradmann, S. A. Aamir, A. Grübl, A. Heimbrecht, K. Schreiber, D. Stöckel *et al.*, "Demonstrating advantages of neuromorphic computation: a pilot study," *Frontiers in neuroscience*, vol. 13, p. 260, 2019. xii

[31] J.-M. Maro, S.-H. Ieng, and R. Benosman, "Event-based gesture recognition with dynamic background suppression using smartphone computational capabilities," *Frontiers in Neuroscience*, vol. 14, p. 275, 2020. xii, 10, 11, 14, 91, 94

[32] R. Hornig, M. Dapper, E. Le Joliff, R. Hill, K. Ishaque, C. Posch, R. Benosman, Y. LeMer, J.-A. Sahel, and S. Picaud, "Pixium vision: first clinical results and innovative developments," in *Artificial Vision*. Springer, 2017, pp. 99–113. xii, 15, 16

[33] F. Galluppi, D. Pruneau, J. Chavas, X. Lagorce, C. Posch, G. Chenegros, G. Cordurié, C. Galle, N. Oddo, and R. Benosman, "A stimulation platform for optogenetic and bionic vision restoration," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4. xii, 16, 17

[34] P.-H. Prévot, K. Gehere, F. Arcizet, H. Akolkar, M. A. Khoei, K. Blaize, O. Oubari, P. Daye, M. Lanoë, M. Valet, S. Dalouz, P. Langlois, E. Esposito, V. Forster, E. Dubus, N. Wattiez, E. Brazhnikova, C. Nouvel-Jaillard, Y. LeMer, J. Demilly, C.-M. Fovet, P. Hantraye, M. Weissenburger, H. Lorach, E. Bouillet, M. Deterre, R. Hornig, G. Buc, J.-A. Sahel, G. Chenegros, P. Pouget, R. Benosman, and S. Picaud, "Behavioural responses to a photovoltaic subretinal prosthesis implanted in non-human primates," *Nature Biomedical Engineering*, vol. 4, no. 2, pp. 172–180, Feb. 2020. [Online]. Available: http://www.nature.com/articles/s41551-019-0484-2 xii, 9, 16, 18

[35] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997. xii, 12, 14

[36] M. P. Mattson, "Superior pattern processing is the essence of the evolved human brain," *Frontiers in neuroscience*, vol. 8, p. 265, 2014. 1

[37] B. Duchaine, L. Cosmides, and J. Tooby, "Evolutionary psychology and the brain," *Current opinion in neurobiology*, vol. 11, no. 2, pp. 225–230, 2001. 1

[38] T. Gollisch and M. Meister, "Eye smarter than scientists believed: neural computations in circuits of the retina," *Neuron*, vol. 65, no. 2, pp. 150–164, 2010. 1, 3, 14, 16

[39] G. Schwartz and M. J. Berry 2nd, "Sophisticated temporal pattern recognition in retinal ganglion cells," *Journal of neurophysiology*, vol. 99, no. 4, pp. 1787–1798, 2008. 1

[40] C. Stangor, J. Walinga *et al.*, *Introduction to Psychology-1st Canadian Edition*. BCcampus, 2018. 2, 3, 4

[41] M. D. Binder, N. Hirokawa, and U. Windhorst, *Encyclopedia of neuroscience*. Springer Berlin, Germany, 2009, vol. 3166. 2

[42] H. Kolb, E. Fernandez, and R. Nelson, *Photoreceptors–Webvision: The Organization of the Retina and Visual System*. University of Utah Health Sciences Center, 1995. 2

[43] D. H. Hubel and T. N. Wiesel, "Integrative action in the cat's lateral geniculate body," *The Journal of Physiology*, vol. 155, no. 2, p. 385, 1961. 3, 7

[44] T. Ichinose, B. Fyk-Kolodziej, and J. Cohn, "Roles of on cone bipolar cell subtypes in temporal coding in the mouse retina," *Journal of Neuroscience*, vol. 34, no. 26, pp. 8761–8771, 2014. 3

[45] S. Thorpe, D. Fize, and C. Marlot, "Speed of processing in the human visual system," *nature*, vol. 381, no. 6582, pp. 520–522, 1996. 3

[46] S. Thorpe, A. Delorme, and R. Van Rullen, "Spike-based strategies for rapid processing," *Neural networks*, vol. 14, no. 6-7, pp. 715–725, 2001. 3, 25

[47] G. D. Field and E. Chichilnisky, "Information processing in the primate retina: circuitry and coding," *Annu. Rev. Neurosci.*, vol. 30, pp. 1–30, 2007. 3

[48] W. A. MacKay, *Neuro 101: Neurophysiology without tears*. Sefalotek, 2010. 4

[49] M. A. Goodale, A. D. Milner *et al.*, "Separate visual pathways for perception and action," *Trends in neurosciences*, 1992. 4

[50] L. G. Ungerleider and J. V. Haxby, "'what' and 'where' in the human brain," *Current opinion in neurobiology*, vol. 4, no. 2, pp. 157–165, 1994. 4

[51] D. Van Essen and E. DeYoe, "Concurrent processing in the primate visual cortex. u ms gazzaniga (ed.), the cognitive neuroscience," 1995. 4

[52] C. J. Perry and M. Fallah, "Feature integration and object representations along the dorsal stream visual hierarchy," *Frontiers in computational neuroscience*, vol. 8, p. 84, 2014. 4

[53] E. Freud, D. C. Plaut, and M. Behrmann, "'what' is happening in the dorsal visual pathway," *Trends in Cognitive Sciences*, vol. 20, no. 10, pp. 773–784, 2016. 4

[54] J. Wilson and S. M. Sherman, "Receptive-field characteristics of neurons in cat striate cortex: changes with visual field eccentricity," *Journal of Neurophysiology*, vol. 39, no. 3, pp. 512–533, 1976. 5

[55] J. P. Van Kleef, S. L. Cloherty, and M. R. Ibbotson, "Complex cell receptive fields: evidence for a hierarchical mechanism," *The Journal of physiology*, vol. 588, no. 18, pp. 3457–3470, 2010. 5

[56] J. Mantas, "An overview of character recognition methodologies," *Pattern recognition*, vol. 19, no. 6, pp. 425–430, 1986. 5

[57] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2107–2116. 7

[58] Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1856–1860. 7

[59] C. R. Pereira, D. R. Pereira, F. A. Silva, J. P. Masieiro, S. A. Weber, C. Hook, and J. P. Papa, "A new computer vision-based approach to aid the diagnosis of parkinson's disease," *Computer Methods and Programs in Biomedicine*, vol. 136, pp. 79–88, 2016. 7

[60] E. R. Fossum, "Active pixel sensors: Are ccds dinosaurs?" in *Charge-Coupled Devices and Solid State Optical Sensors III*, vol. 1900. International Society for Optics and Photonics, 1993, pp. 2–14. 7

[61] T. Delbruck, "Silicon retina with correlation-based, velocity-tuned pixels," *IEEE Transactions on neural networks*, vol. 4, no. 3, pp. 529–541, 1993. 7

[62] R. Etienne-Cummings and J. Van der Spiegel, "Neuromorphic vision sensors," *Sensors and Actuators A: Physical*, vol. 56, no. 1-2, pp. 19–29, 1996. 7

[63] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x 128 120 db 15μ s latency asynchronous temporal contrast vision sensor," *IEEE journal of solid-state circuits*, vol. 43, pp. 566–576, 2008. 7, 8

[64] M. A. Sivilotti, "Wiring considerations in analog vlsi systems, with application to field-programmable networks." 1991. 7

[65] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019. 8, 61

[66] X. Berthelon, G. Chenegros, T. Finateu, S.-H. Ieng, and R. Benosman, "Effects of cooling on the snr and contrast detection of a low-light event-based camera," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 12, no. 6, pp. 1467–1474, 2018. 9

[67] S. Mafrica, S. Godiot, M. Menouni, M. Boyron, F. Expert, R. Juston, N. Marchand, F. Ruffier, and S. Viollet, "A bio-inspired analog silicon retina with michaelis-menten auto-adaptive pixels sensitive to small and large changes in light," *Optics express*, vol. 23, no. 5, pp. 5614–5635, 2015. 9

[68] T. Serrano-Gotarredona and B. Linares-Barranco, "A 128 × 128 1.5contrast sensitivity 0.9dynamic vision sensor using transimpedance preamplifiers," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 827–838, 2013. 9

[69] M. Yang, S. Liu, and T. Delbruck, "A dynamic vision sensor with 1temporal contrast sensitivity and in-pixel asynchronous delta modulator for event encoding," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 9, pp. 2149–2160, 2015. 9

[70] C. Brandli, R. Berner, M. Yang, S. Liu, and T. Delbruck, "A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014. 9

[71] B. Son, Y. Suh, S. Kim, H. Jung, J. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu, "4.1 a 640×480 dynamic vision sensor with a 9µm pixel and 300meps address-event representation," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, 2017, pp. 66–67. 9

[72] S. Chen and M. Guo, "Live demonstration: Celex-v: a 1m pixel multi-mode event-based sensor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0. 9, 31, 61, 62, 93

[73] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5419–5427. 9, 14, 63

[74] M. B. Milde, O. J. Bertrand, R. Benosmanz, M. Egelhaaf, and E. Chicca, "Bioinspired event-driven collision avoidance algorithm based on optic flow," in *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2015, pp. 1–7. 9, 12

[75] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986. 9

[76] R. Ghosh, A. Gupta, A. Nakagawa, A. Soares, and N. Thakor, "Spatiotemporal filtering for event-based action recognition," *arXiv preprint arXiv:1903.07067*, 2019. 9

[77] D. Gehrig, A. Loquercio, K. G. Derpanis, and D. Scaramuzza, "End-to-end learning of representations for asynchronous event-based data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5633–5643. 9

[78] S. Gao, G. Guo, H. Huang, X. Cheng, and C. P. Chen, "An end-to-end broad learning system for event-based object classification," *IEEE Access*, vol. 8, pp. 45 974–45 984, 2020. 9

[79] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu, "Space-time event clouds for gesture recognition: from rgb cameras to event cameras," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1826–1835. 9

[80] Y. Sekikawa, K. Hara, and H. Saito, "Eventnet: Asynchronous recursive event processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3887–3896. 9

[81] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660. 9

[82] Y. Bi, A. Chadha, A. Abbas, E. Bourtsoulatze, and Y. Andreopoulos, "Graph-based object classification for neuromorphic vision sensing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 491–501. 9

[83] J. Kogler, C. Sulzbachner, and W. Kubinger, "Bio-inspired stereo vision system with silicon retina imagers," in *International Conference on Computer Vision Systems*. Springer, 2009, pp. 174–183. 9

[84] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "Ev-flownet: Self-supervised optical flow estimation for event-based cameras," *arXiv preprint arXiv:1802.06898*, 2018. 9

[85] A. Chadha, Y. Bi, A. Abbas, and Y. Andreopoulos, "Neuromorphic vision sensing for cnn-based action recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 7968–7972. 9

[86] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "Events-to-video: Bringing modern computer vision to event cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3857–3866. 9, 63

[87] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7243–7252. 9, 63, 77

[88] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous convolutional networks for object detection in neuromorphic cameras," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0. 9

[89] B. R. Pradhan, Y. Bethi, S. Narayanan, A. Chakraborty, and C. S. Thakur, "N-har: A neuromorphic event-based human activity recognition system using memory surfaces," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5. 9

[90] S. Afshar, T. J. Hamilton, J. Tapson, A. van Schaik, and G. Cohen, "Investigation of event-based surfaces for high-speed detection, unsupervised feature extraction, and object recognition," *Frontiers in neuroscience*, vol. 12, p. 1047, 2019. 10, 11

[91] G. Orchard, X. Lagorce, C. Posch, S. B. Furber, R. Benosman, and F. Galluppi, "Real-time event-driven spiking neural network object recognition on the spinnaker platform," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2015, pp. 2413–2416. 10

[92] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *bioRxiv*, p. 738385, 2020. 10, 13, 93

[93] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (decolle)," *Frontiers in Neuroscience*, vol. 14, p. 424, 2020. 10, 13, 60, 93

[94] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, 2018, pp. 1412–1421. 10, 13, 20, 63, 77

[95] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016. 10, 13, 63

[96] L. R. Iyer and A. Basu, "Unsupervised learning of event-based image recordings using spike-timing-dependent plasticity," *International Joint Conference on Neural Networks (IJCNN)*, 2017. 10, 13, 38, 45, 46, 50, 51

[97] J. C. Thiele, O. Bichler, and A. Dupret, "A timescale invariant stdp-based spiking deep network for unsupervised online feature extraction from event-based sensor data," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8. 10, 13, 51, 63

[98] B. Ramesh, H. Yang, G. M. Orchard, N. A. Le Thi, S. Zhang, and C. Xiang, "Dart: distribution aware retinal transform for event-based cameras," *IEEE transactions on pattern analysis and machine intelligence*, 2019. 10, 11, 93, 94

[99] S. Afshar, N. Ralph, Y. Xu, J. Tapson, A. v. Schaik, and G. Cohen, "Event-based feature extraction using adaptive selection thresholds," *Sensors*, vol. 20, no. 6, p. 1600, 2020. 10, 11, 63, 94

[100] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "Hots: a hierarchy of event-based time-surfaces for pattern recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1346–1359, 2016. 10, 11, 48, 57, 61, 63, 64, 77, 79, 90, 93, 94

[101] L. U. Perrinet, "Role of homeostasis in learning sparse representations," *Neural computation*, vol. 22, no. 7, pp. 1812–1836, 2010. 11

[102] G. G. Turrigiano and S. B. Nelson, "Homeostatic plasticity in the developing nervous system," *Nature reviews neuroscience*, vol. 5, no. 2, pp. 97–107, 2004. 11

[103] R. Matungka, "Studies on log-polar transform for image registration and improvements using adaptive sampling and logarithmic spiral," Ph.D. dissertation, The Ohio State University, 2009. 11

[104] Prophesee. [Online]. Available: https://www.prophesee.ai 12

[105] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014. 12

[106] S. Furber and S. Temple, "Neural systems engineering," in *Computational intelligence: A compendium*. Springer, 2008, pp. 763–796. 12

[107] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014. 12, 14, 92

[108] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, p. 774, 2018. 12

[109] L. Vannucci, A. Ambrosano, N. Cauli, U. Albanese, E. Falotico, S. Ulbrich, L. Pfotzer, G. Hinkel, O. Denninger, D. Peppicelli *et al.*, "A visual tracking model implemented on the icub robot as a use case for a novel neurorobotic toolkit integrating brain and physics simulation," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 1179–1184. 12

[110] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–8. 12

[111] L. Miró-Amarante, F. Gómez-Rodríguez, A. Jiménez-Fernández, and G. Jiménez-Moreno, "A spiking neural network for real-time Spanish vowel phonemes recognition," *Neurocomputing*, vol. 226, no. C, pp. 249–261, Feb. 2017. 12

[112] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J. M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris, M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A. P. Davison, S. El Boustani, and A. Destexhe, "Simulation of networks of spiking neurons: A review of tools and strategies," *Journal of Computational Neuroscience*, vol. 23, no. 3, pp. 349–398, Jul. 2007. 12

[113] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers in Computational Neuroscience*, vol. 9, no. 429, 2015. 12, 13, 51

[114] E. M. Izhikevich, "Polychronization: Computation with spikes," *Neural Computation*, vol. 18, no. 2, pp. 245–282, Feb. 2006. 13, 19, 20, 30

[115] A. Tavanaei, T. Masquelier, and A. Maida, "Representation learning using event-based stdp," *Neural Networks*, vol. 105, pp. 294–303, 2018. 13

[116] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "Stdp and stdp variations with memristors for spiking neuromorphic learning systems," *Frontiers in neuroscience*, vol. 7, p. 2, 2013. 13

[117] P. Lewden, A. F. Vincent, C. Meyer, J. Tomas, S. Siami, and S. Saïghi, "Hardware spiking neural networks: Slow tasks resilient learning with longer term-memory bits," in *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2019, pp. 1–4. 13, 32, 37, 38, 42, 45, 46, 54

[118] N. Frémaux and W. Gerstner, "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules," *Frontiers in neural circuits*, vol. 9, p. 85, 2016. 13

[119] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, "First-spike-based visual categorization using reward-modulated stdp," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 6178–6190, 2018. 13, 14, 93

[120] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Stdp-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, 2018. 13, 51

[121] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015. 13

[122] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in neuroscience*, vol. 11, p. 682, 2017. 13

[123] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019. 13

[124] L. Zhang, S. Zhou, T. Zhi, Z. Du, and Y. Chen, "Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1319–1326. 13

[125] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers in Neuroscience*, vol. 14, 2020. 13

[126] A. Kugele, T. Pfeil, M. Pfeiffer, and E. Chicca, "Efficient processing of spatio-temporal data streams with spiking neural networks," *Frontiers in Neuroscience*, vol. 14, p. 439, 2020. 13, 94

[127] V. Fischer, J. Köhler, and T. Pfeil, "The streaming rollout of deep networks-towards fully model-parallel execution," in *Advances in Neural Information Processing Systems*, 2018, pp. 4039–4050. 13

[128] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019. 13, 93

[129] S. R. Kheradpisheh and T. Masquelier, "S4nn: temporal backpropagation for spiking neural networks with one spike per neuron," *International Journal of Neural Systems*, vol. 30, no. 6, p. 2050027, 2020. 13, 51, 93

[130] S. R. Kheradpisheh, M. Mirsadeghi, and T. Masquelier, "Bs4nn: Binarized spiking neural networks with temporal coding and learning," *arXiv preprint arXiv:2007.04039*, 2020. 13

[131] H. Mostafa, V. Ramesh, and G. Cauwenberghs, "Deep supervised learning using local errors," *Frontiers in neuroscience*, vol. 12, p. 608, 2018. 13

[132] M. Payvand, M. E. Fouda, F. Kurdahi, A. M. Eltawil, and E. O. Neftci, "On-chip error-triggered learning of multi-layer memristive spiking neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2020. 13, 60

[133] M. N. Shadlen and W. T. Newsome, "The variable discharge of cortical neurons: implications for connectivity, computation, and information coding," *Journal of neuroscience*, vol. 18, no. 10, pp. 3870–3896, 1998. 13

[134] S. Thorpe and J. Gautrais, "Rank order coding," in *Computational neuroscience*. Springer, 1998, pp. 113–118. 14

[135] J. Putney, R. Conn, and S. Sponberg, "Precise timing is ubiquitous, consistent, and coordinated across a comprehensive, spike-resolved flight motor program," *Proceedings of the National Academy of Sciences*, vol. 116, no. 52, pp. 26 951–26 960, 2019. 14

[136] J. Luo, S. Macias, T. V. Ness, G. T. Einevoll, K. Zhang, and C. F. Moss, "Neural timing of stimulus events with microsecond precision," *PLoS biology*, vol. 16, no. 10, p. e2006422, 2018. 14

[137] M. J. Berry, D. K. Warland, and M. Meister, "The structure and precision of retinal spike trains," *Proceedings of the National Academy of Sciences*, vol. 94, no. 10, pp. 5411–5416, 1997. 14, 25

[138] P. Reinagel and R. C. Reid, "Temporal coding of visual information in the thalamus," *Journal of Neuroscience*, vol. 20, no. 14, pp. 5392–5400, 2000. 14, 25

[139] G. T. Buracas, A. M. Zador, M. R. DeWeese, and T. D. Albright, "Efficient discrimination of temporal patterns by motion-sensitive neurons in primate visual cortex," *Neuron*, vol. 20, no. 5, pp. 959–969, 1998. 14, 25

[140] Ł. Kuśmierz, T. Isomura, and T. Toyoizumi, "Learning with three factors: modulating hebbian plasticity with errors," *Current opinion in neurobiology*, vol. 46, pp. 170–177, 2017. 14

[141] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018. 14, 60

[142] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, 2015, pp. 161–170. 14

[143] Edge tpu. [Online]. Available: https://coral.ai/technology/ 14, 31

[144] Movidius vpu. [Online]. Available: https://www.intel.com/content/www/us/en/products/processors/movidius-vpu.html 14, 31

[145] B. Rajendran, A. Sebastian, M. Schmuker, N. Srinivasa, and E. Eleftheriou, "Low-power neuromorphic hardware for signal processing applications: A review of architectural and system-level design approaches," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 97–110, 2019. 14

[146] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in neuroscience*, vol. 9, p. 141, 2015. 14

[147] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014. 14, 31, 32

[148] C. Frenkel, M. Lefebvre, J. Legat, and D. Bol, "A 0.086-mm 2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos." *IEEE transactions on biomedical circuits and systems*, vol. 13, no. 1, pp. 145–158, 2019. 14

[149] Akida neural processor. [Online]. Available: https://brainchipinc.com/technology/ 14

[150] K. Meier, "A mixed-signal universal neuromorphic computing system," in *2015 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2015, pp. 4–6. 14

[151] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014. 15

[152] V. Wang and A. E. Kuriyan, "Optoelectronic devices for vision restoration," *Current Ophthalmology Reports*, vol. 8, no. 2, pp. 69–77, 2020. 15

[153] R. P. Danis, J. A. Lavine, and A. Domalpally, "Geographic atrophy in patients with advanced dry age-related macular degeneration: current challenges and future prospects," *Clinical Ophthalmology (Auckland, NZ)*, vol. 9, p. 2159, 2015. 16

[154] P. Degenaar, N. Grossman, M. A. Memon, J. Burrone, M. Dawson, E. Drakakis, M. Neil, and K. Nikolic, "Optobionic vision—a new genetically enhanced light on retinal prosthesis," *Journal of neural engineering*, vol. 6, no. 3, p. 035007, 2009. 16

[155] S. Kime, F. Galluppi, X. Lagorce, R. B. Benosman, and J. Lorenceau, "Psychophysical assessment of perceptual performance with varying display frame rates," *Journal of Display Technology*, vol. 12, no. 11, pp. 1372–1382, 2016. 16, 17

[156] M. Poujade, "Apport des dispositifs de restauration de la vision et de la résolution temporelle," Ph.D. dissertation, Sorbonne Université, 2019. 16, 17

[157] E. Chichilnisky and R. Kalmar, "Temporal resolution of ensemble visual motion signals in primate retina," *Journal of Neuroscience*, vol. 23, no. 17, pp. 6681–6689, 2003. 16

[158] A. Kusnyerik, M. Resch, H. J. Kiss, and J. Nemeth, "Vision restoration with implants," in *Mobility of Visually Impaired People*. Springer, 2018, pp. 617–630. 16

[159] R. Gütig and H. Sompolinsky, "The tempotron: a neuron that learns spike timing–based decisions," *Nature Neuroscience*, vol. 9, pp. 420–428, 2006. 19, 20

[160] G. M. Wittenberg and S. S. H. Wang, "Malleability of Spike-Timing-Dependent Plasticity at the CA3-CA1 Synapse," *Journal of Neuroscience*, vol. 26, no. 24, pp. 6610–6617, Jun. 2006. 19

[161] H. A. Swadlow and S. G. Waxman, "Axonal conduction delays," *Scholarpedia*, vol. 7, no. 6, p. 1451, 2012. 19

[162] C. E. Carr and M. Konishi, "A Circuit for Detection of Interaural Time Differences in the Brain-Stem of the Barn Owl," *Journal of Neuroscience*, vol. 10, no. 10, pp. 3227–3246, Oct. 1990. 19, 25

[163] J. C. Middlebrooks, A. E. Clock, L. Xu, and D. M. Green, "A panoramic code for sound location by cortical neurons," *Science*, vol. 264, no. 5160, pp. 842–844, May 1994. 19

[164] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic aps and epsps," *Science*, vol. 275, no. 5297, pp. 213–215, 1997. 19

[165] R. D. Fields, "A new mechanism of nervous system plasticity: activity-dependent myelination," *Nature Reviews Neuroscience*, vol. 16, no. 12, pp. 756–767, Nov. 2015. 19

[166] S. Koudelka, M. G. Voas, R. G. Almeida, M. Baraban, J. Soetaert, M. P. Meyer, W. S. Talbot, and D. A. Lyons, "Individual Neuronal Subtypes Exhibit Diversity in CNS Myelination Mediated by Synaptic Vesicle Release," *Current Biology*, vol. 26, no. 11, pp. 1447–1455, Jun. 2016. 19

[167] G. Bonetto, Y. Kamen, K. A. Evans, and R. T. Káradóttir, "Unraveling myelin plasticity," *Frontiers in Cellular Neuroscience*, vol. 14, p. 156, 2020. 19

[168] P. Baldi and A. F. Atiya, "How Delays Affect Neural Dynamics and Learning," *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 612–621, Jul. 1994. 19

[169] W. Maass, "On the relevance of time in neural computation and learning," *Theoretical Computer Science*, vol. 261, no. 1, pp. 157 – 178, 2001, eighth

International Workshop on Algorithmic Learning Theory. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397500001377 19

[170] C. W. Eurich, K. Pawelzik, U. Ernst, J. D. Cowan, and J. G. Milton, "Dynamics of self-organized delay adaptation," *Physical Review Letters*, vol. 82, no. 7, pp. 1594–1597, 1999. 20

[171] C. W. Eurich, K. Pawelzik, U. Ernst, A. Thiel, J. D. Cowan, and J. G. Milton, "Delay adaptation in the nervous system," *Neurocomputing*, vol. 32, pp. 741–748, Jun. 2000. 20

[172] S. Hussain, A. Basu, M. Wang, and T. J. Hamilton, "Deltron: Neuromorphic architectures for delay based learning," in *2012 IEEE Asia Pacific Conference on Circuits and Systems*. IEEE, 2012, pp. 304–307. 20, 26, 29, 91

[173] H. Paugam-Moisy, R. Martinez, and S. Bengio, "Delay learning and polychronization for reservoir computing," *Neurocomputing*, vol. 71, no. 7-9, pp. 1143–1158, 2008. 20

[174] R. Wang, G. Cohen, K. Stiefel, T. Hamilton, J. Tapson, and A. van Schaik, "An FPGA implementation of a polychronous spiking neural network with delay adaptation," *Frontiers in Neuroscience*, vol. 7, 2013. 20, 29

[175] T. Matsubara, "Spike Timing-Dependent Conduction Delay Learning Model Classifying Spatio-Temporal Spike Patterns," *Front. Comput. Neurosci.*, vol. 11, no. 104, pp. 1–16, Jun. 2017. 20, 29, 30, 91

[176] R. M. Wang, T. J. Hamilton, J. C. Tapson, and A. van Schaik, "A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks," *Frontiers in Neuroscience*, vol. 9, p. 180, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2015.00180 20

[177] E. R. Kandel, J. H. Schwartz, T. M. Jessell, D. of Biochemistry, M. B. T. Jessell, S. Siegelbaum, and A. Hudspeth, *Principles of neural science*. McGraw-hill New York, 2000, vol. 4. 25

[178] P. Dayan and L. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Cambridge, MA, USA: MIT Press, 2001. 25

[179] A. Dean, "The variability of discharge of simple cells in the cat striate cortex," *Experimental Brain Research*, vol. 44, no. 4, pp. 437–440, 1981. 25

[180] G. Buzsáki, R. Llinas, W. Singer, A. Berthoz, and Y. Christen, *Temporal coding in the brain*. Springer Science & Business Media, 2012. 25

[181] W. Gerstner, R. Kempter, J. L. Van Hemmen, and H. Wagner, "A neuronal learning rule for sub-millisecond temporal coding," *Nature*, vol. 383, no. 6595, pp. 76–78, 1996. 25

[182] R. Laje and D. V. Buonomano, "Robust timing and motor patterns by taming chaos in recurrent neural networks," *Nature neuroscience*, vol. 16, no. 7, p. 925, 2013. 25

[183] W. Maass, "Fast sigmoidal networks via spiking neurons," *Neural Computation*, vol. 9, no. 2, pp. 279–304, 1997. 25

[184] N. Iannella and A. D. Back, "A spiking neural network architecture for nonlinear function approximation," *Neural networks*, vol. 14, no. 6-7, pp. 933–939, 2001. 25

[185] Z. Mainen and T. Sejnowski, "Reliability of spike timing in neocortical neurons," *Science*, vol. 268, no. 5216, pp. 1503–1506, 1995. 25

[186] A. Goel and D. V. Buonomano, "Temporal interval learning in cortical cultures is encoded in intrinsic network dynamics," *Neuron*, vol. 91, no. 2, pp. 320–327, 2016. 25

[187] M. Wehr and A. M. Zador, "Balanced inhibition underlies tuning and sharpens spike timing in auditory cortex," *Nature*, vol. 426, no. 6965, p. 442, 2003. 25

[188] P. H. Brownell, "Compressional and surface waves in sand: Used by desert scorpions to locate prey," *Science*, vol. 197, no. 4302, pp. 479–482, 1977. 25

[189] G. Haessig, M. B. Milde, P. V. Aceituno, O. Oubari, J. C. Knight, A. van Schaik, R. B. Benosman, and G. Indiveri, "Event-Based Computation for Touch Localization Based on Precise Spike Timing," *Frontiers in Neuroscience*, vol. 14, p. 420, May 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00420/full 25, 26, 27, 28, 29, 91

[190] R. M. Wang, T. J. Hamilton, J. C. Tapson, and A. van Schaik, "A neuromorphic implementation of multiple spike-timing synaptic plasticity rules for large-scale neural networks," *Frontiers in neuroscience*, vol. 9, p. 180, 2015. 29

[191] J. F. Hunzinger, V. H. Chan, and R. C. Froemke, "Learning complex temporal patterns with resource-dependent spike timing-dependent plasticity," *Journal of Neurophysiology*, vol. 108, no. 2, pp. 551–566, 2012. 30

[192] J. Barrios-Avilés, T. Iakymchuk, J. Samaniego, L. D. Medus, and A. Rosado-Muñoz, "Movement detection with event-based cameras: Comparison with frame-based cameras in robot object tracking using powerlink communication," *Electronics*, vol. 7, no. 11, p. 304, 2018. 31

[193] A. Marcireau, S. H. Ieng, and R. B. Benosman, "Sepia, tarsier and chameleon: a modular c++ framework for event-based computer vision." *Frontiers in Neuroscience*, vol. 13, p. 1338, 2019. 31

[194] J. Backus, "Can programming be liberated from the von neumann style? a functional style and its algebra of programs," *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, 1978. 31

[195] J. C. Knight and T. Nowotny, "Gpus outperform current hpc and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model," *Frontiers in neuroscience*, vol. 12, p. 941, 2018. 31

[196] R. Brette and D. F. Goodman, "Simulating spiking neural networks on gpu," *Network: Computation in Neural Systems*, vol. 23, no. 4, pp. 167–182, 2012. 31

[197] M. Stimberg, D. F. Goodman, and T. Nowotny, "Brian2genn: accelerating spiking neural network simulations with graphics hardware," *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020. 31

[198] J. B. Aimone, O. Parekh, and W. Severa, "Neural computing for scientific computing applications: more than just machine learning," in *Proceedings of the Neuromorphic Computing Symposium*, 2017, pp. 1–6. 32

[199] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971. 32, 33, 40

[200] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008. 32, 33, 60

[201] S. Boyn, J. Grollier, G. Lecerf, B. Xu, N. Locatelli, S. Fusil, S. Girod, C. Carrétéro, K. Garcia, S. Xavier *et al.*, "Learning through ferroelectric domain dynamics in solid-state synapses," *Nature communications*, vol. 8, no. 1, pp. 1–7, 2017. 32, 33, 35, 38, 54, 60, 92

[202] P. de Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi, "A large scale event-based detection dataset for automotive," *arXiv*, pp. arXiv–2001, 2020. 32, 92

[203] P. Mazumder, S.-M. Kang, and R. Waser, "Memristors: devices, models, and applications," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1911–1919, 2012. 33

[204] A. Chanthbouala, V. Garcia, R. O. Cherifi, K. Bouzehouane, S. Fusil, X. Moya, S. Xavier, H. Yamada, C. Deranlot, N. D. Mathur *et al.*, "A ferroelectric memristor," *Nature materials*, vol. 11, no. 10, pp. 860–864, 2012. 33, 34, 35, 54

[205] L. Chua, "Resistance switching memories are memristors," *Applied Physics A*, vol. 102, no. 4, pp. 765–783, 2011. 33

[206] M. A. Zidan, J. P. Strachan, and W. D. Lu, "The future of electronics based on memristive systems," *Nature Electronics*, vol. 1, no. 1, pp. 22–29, 2018. 33

[207] N. Nithya and K. Paramasivam, "A comprehensive study on the characteristics, complex materials and applications of memristor," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2020, pp. 171–176. 33

[208] L. Wang, C. Yang, J. Wen, S. Gai, and Y. Peng, "Overview of emerging memristor families from resistive memristor to spintronic memristor," *Journal of Materials Science: Materials in Electronics*, vol. 26, no. 7, pp. 4618–4628, 2015. 33

[209] H. An, K. Bai, and Y. Yi, "The roadmap to realize memristive three-dimensional neuromorphic computing system," *Advances in Memristor Neural Networks-Modeling and Applications*, pp. 25–44, 2018. 33

[210] J. J. Yang, M.-X. Zhang, J. P. Strachan, F. Miao, M. D. Pickett, R. D. Kelley, G. Medeiros-Ribeiro, and R. S. Williams, "High switching endurance in tao x memristive devices," *Applied Physics Letters*, vol. 97, no. 23, p. 232102, 2010. 34

[211] W. Banerjee, "Challenges and applications of emerging nonvolatile memory devices," *Electronics*, vol. 9, no. 6, p. 1029, 2020. 34

[212] S. Boyn, S. Girod, V. Garcia, S. Fusil, S. Xavier, C. Deranlot, H. Yamada, C. Carrétéro, E. Jacquet, M. Bibes *et al.*, "High-performance ferroelectric memory based on fully patterned tunnel junctions," *Applied Physics Letters*, vol. 104, no. 5, p. 052909, 2014. 34, 35, 54

[213] L. Chen, T.-Y. Wang, Y.-W. Dai, M.-Y. Cha, H. Zhu, Q.-Q. Sun, S.-J. Ding, P. Zhou, L. Chua, and D. W. Zhang, "Ultra-low power hf 0.5 zr 0.5 o 2 based ferroelectric tunnel junction synapses for hardware neural network applications," *Nanoscale*, vol. 10, no. 33, pp. 15 826–15 833, 2018. 34, 92

[214] N. Locatelli, V. Cros, and J. Grollier, "Spin-torque building blocks," *Nature materials*, vol. 13, no. 1, pp. 11–20, 2014. 34

[215] H. Yamada, V. Garcia, S. Fusil, S. Boyn, M. Marinova, A. Gloter, S. Xavier, J. Grollier, E. Jacquet, C. Carrétéro *et al.*, "Giant electroresistance of super-tetragonal bifeo3-based ferroelectric tunnel junctions," *ACS nano*, vol. 7, no. 6, pp. 5385–5390, 2013. 35

[216] R. Guo, Z. Wang, S. Zeng, K. Han, L. Huang, D. G. Schlom, T. Venkatesan, J. Chen *et al.*, "Functional ferroelectric tunnel junctions on silicon," *Scientific reports*, vol. 5, p. 12576, 2015. 35

[217] M. Abuwasib, C. R. Serrao, L. Stan, S. Salahuddin, and S. R. Bakaul, "Tunneling electroresistance effects in epitaxial complex oxides on silicon," *Applied Physics Letters*, vol. 116, no. 3, p. 032902, 2020. 35

[218] G. Lecerf, J. Tomas, S. Boyn, S. Girod, A. Mangalore, J. Grollier, and S. Saïghi, "Silicon neuron dedicated to memristive spiking neural networks," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2014, pp. 1568–1571. 37, 54

[219] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Competitive STDP-Based Spike Pattern Learning," *Neural Computation*, vol. 21, no. 5, pp. 1259–1276, May 2009. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.2008.06-08-804 38, 92

[220] A. Page, S. P. T. Oates, and T. Mohsenin, "An ultra low power feature extraction and classification system for wearable seizure detection," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2015, pp. 7111–7114. 42, 49

[221] D. Cox, "Some procedures connected with the logistic qualitative response curve. in (fn david, ed.) research papers in statistics: Essays in honour of j. neyman's 70th birthday," 1966. 43

[222] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006. 43

[223] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015. 45, 51, 62, 63, 77

[224] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "Hfirst: a temporal approach to object recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 10, pp. 2028–2040, 2015. 48, 77

[225] T. Serrano-Gotarredona and B. Linares-Barranco, "Poker-dvs and mnist-dvs. their history, how they were made, and other details," *Frontiers in neuroscience*, vol. 9, p. 481, 2015. 48, 63, 69

[226] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988. 51

[227] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996. 51

[228] E. P. Simoncelli and B. A. Olshausen, "Natural image statistics and neural representation," *Annual Review of Neuroscience*, vol. 24, no. 1, pp. 1193–1216, 2001, pMID: 11520932. [Online]. Available: https://doi.org/10.1146/annurev.neuro.24.1.1193 51

[229] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, no. 1, pp. 1–13, 2020. 60, 92

[230] G. Finocchio, M. Di Ventra, K. Y. Camsari, K. Everschor-Sitte, P. K. Amiri, and Z. Zeng, "The promise of spintronics for unconventional computing," *arXiv preprint arXiv:1910.07176*, 2019. 60

[231] W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, "Integer factorization using stochastic magnetic tunnel junctions," *Nature*, vol. 573, no. 7774, pp. 390–393, 2019. 60

[232] G. Gallego, J. E. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza, "Event-based, 6-dof camera tracking from photometric depth maps," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 10, pp. 2402–2412, 2017. 61

[233] M. Fujimoto and Y. Riki, "Robust speech recognition in additive and channel noise environments using gmm and em algorithm," in *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, 2004, pp. I–941. 62

[234] T. Kurata, T. Okuma, M. Kourogi, and K. Sakaue, "The hand mouse: Gmm hand-color classification and mean shift tracking," in *Proceedings IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*. IEEE, 2001, pp. 119–124. 62

[235] D. Zhou and H. Zhang, "Modified gmm background modeling and optical flow for detection of moving objects," in *2005 IEEE international conference on systems, man and cybernetics*, vol. 3. IEEE, 2005, pp. 2224–2229. 62

[236] J. VanderPlas, A. J. Connolly, Ž. Ivezić, and A. Gray, "Introduction to astroml: Machine learning for astrophysics," in *2012 conference on intelligent data understanding*. IEEE, 2012, pp. 47–54. 62

[237] F. Lerch, A. Ultsch, and J. Lötsch, "Distribution optimization: An evolutionary algorithm to separate gaussian mixtures," *Scientific Reports*, vol. 10, no. 1, pp. 1–10, 2020. 62

[238] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 2. IEEE, 1999, pp. 246–252. 62

[239] J. Xu, D. J. Hsu, and A. Maleki, "Global analysis of expectation maximization for mixtures of two gaussians," in *Advances in Neural Information Processing Systems*, 2016, pp. 2676–2684. 62

[240] A. Moitra and G. Valiant, "Settling the polynomial learnability of mixtures of gaussians," in *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 2010, pp. 93–102. 62

[241] S. Kolouri, G. K. Rohde, and H. Hoffmann, "Sliced wasserstein distance for learning gaussian mixture models," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 62

[242] R. Hosseini and S. Sra, "An alternative to em for gaussian mixture models: Batch and stochastic riemannian optimization," *Mathematical Programming*, pp. 1–37, 2019. 62

[243] F. Hirschberger, D. Forster, and J. Lücke, "Accelerated training of large-scale gaussian mixtures by a merger of sublinear approaches," *stat*, vol. 1050, p. 12, 2019. 62, 78, 83, 85, 89, 93

[244] D. Forster and J. Lücke, "Can clustering scale sublinearly with its clusters? a variational em acceleration of gmms and k-means," *arXiv preprint arXiv:1711.03431*, 2017. 62, 83, 85

[245] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff *et al.*, "5.10 a 1280× 720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86 μm pixels, 1.066 geps readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 112–114. 62

[246] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006. 62, 69, 71

[247] O. Bachem, M. Lucic, S. H. Hassani, and A. Krause, "Approximate k-means++ in sublinear time," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 62, 67, 78

[248] O. Bachem, M. Lucic, H. Hassani, and A. Krause, "Fast and provably good seedings for k-means," in *Advances in neural information processing systems*, 2016, pp. 55–63. 62, 68, 78, 83, 89

[249] S. Har-Peled and S. Mazumdar, "On coresets for k-means and k-median clustering," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 291–300. 62, 67, 78

[250] O. Bachem, M. Lucic, and A. Krause, "Scalable k-means clustering via lightweight coresets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1119–1127. 62, 68, 78, 83, 89

[251] J. Lücke and J. Eggert, "Expectation truncation and the benefits of preselection in training generative models," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2855–2900, 2010. 63, 78

[252] G. Exarchakis and J. Lücke, "Discrete sparse coding," *Neural computation*, vol. 29, no. 11, pp. 2979–3013, 2017. 63, 78

[253] Z. Dai, G. Exarchakis, and J. Lücke, "What are the invariant occlusive components of image patches? a probabilistic generative approach," in *Advances in neural information processing systems*, 2013, pp. 243–251. 63, 78

[254] D. Forster and J. Lücke, "Truncated variational em for semi-supervised neural simpletrons," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3769–3776. 63, 78

[255] J. Lücke, Z. Dai, and G. Exarchakis, "Truncated variational sampling for 'black box'optimization of generative models," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 467–478. 63, 78

[256] G. Exarchakis, M. Henniges, J. Eggert, and J. Lücke, "Ternary sparse coding," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2012, pp. 204–212. 63, 78

[257] C. Jin, Y. Zhang, S. Balakrishnan, M. J. Wainwright, and M. I. Jordan, "Local maxima in the likelihood of gaussian mixture models: Structural results and algorithmic consequences," in *Advances in neural information processing systems*, 2016, pp. 4116–4124. 63, 78

[258] A. Krizhevsky, "Learning multiple layers of features from tiny images," *University of Toronto*, 05 2012. 63, 69, 79

[259] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011. 63, 69, 79

[260] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature communications*, vol. 5, p. 4308, 2014. 63, 69, 79

[261] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 1177–1178. 63, 77, 83, 93

[262] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, "Feature detection and tracking with the dynamic and active-pixel vision sensor (davis)," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2016, pp. 1–7. 63

[263] C. Scheerlinck, H. Rebecq, D. Gehrig, N. Barnes, R. Mahony, and D. Scaramuzza, "Fast image reconstruction with an event camera," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 156–163. 63

[264] G. Haessig and R. Benosman, "A sparse coding multi-scale precise-timing machine learning algorithm for neuromorphic event-based sensors," in *Micro-and Nanotechnology Sensors, Systems, and Applications X*, vol. 10639. International Society for Optics and Photonics, 2018, p. 106391U. 63, 79

[265] D. Feldman, M. Faulkner, and A. Krause, "Scalable training of mixture models via coresets," in *Advances in neural information processing systems*, 2011, pp. 2142–2150. 68, 83

[266] M. Lucic, M. Faulkner, A. Krause, and D. Feldman, "Training gaussian mixture models at scale via coresets," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5885–5909, 2017. 68, 83

[267] P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition*, vol. 39, no. 5, pp. 761–765, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2005.09.012 69, 70, 84

[268] Blaze library. [Online]. Available: https://bitbucket.org/blaze-lib/blaze/src/master/ 69

[269] S. Gao, G. Guo, H. Huang, X. Cheng, and C. L. P. Chen, "An end-to-end broad learning system for event-based object classification," *IEEE Access*, vol. 8, pp. 45 974–45 984, 2020. 77

[270] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *2004 conference on computer vision and pattern recognition workshop*. IEEE, 2004, pp. 178–178. 77

[271] H. Zhang and D. Zhao, "Spatial histogram features for face detection in color images," in *Pacific-Rim Conference on Multimedia*. Springer, 2004, pp. 377–384. 77

[272] E. Guiraud, J. Drefs, and J. Lücke, "Evolutionary expectation maximization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 442–449. 78

[273] R. Caruana, T. Joachims, and L. Backstrom, "Kdd-cup 2004: results and analysis," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 2, pp. 95–108, 2004. 79

[274] X. Clady, J.-M. Maro, S. Barré, and R. B. Benosman, "A motion-based feature for event-based pattern recognition," *Frontiers in neuroscience*, vol. 10, p. 594, 2017. 79

[275] R. M. Neal and G. E. Hinton, "A view of the em algorithm that justifies incremental, sparse, and other variants," in *Learning in graphical models*. Springer, 1998, pp. 355–368. 81

[276] P. Liang and D. Klein, "Online em for unsupervised models," in *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, 2009, pp. 611–619. 90

[277] B. Nessler, M. Pfeiffer, L. Buesing, and W. Maass, "Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity," *PLoS Comput Biol*, vol. 9, no. 4, p. e1003037, 2013. 90

[278] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990. 91

[279] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128 x 128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, feb 2008. 91

[280] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of intelligent and robotic systems*, vol. 31, no. 1-3, pp. 91–103, 2001. 91

[281] R. Adhikari, "A neural network based linear ensemble framework for time series forecasting," *Neurocomputing*, vol. 157, pp. 231–242, 2015. 91

[282] R. M. Wang, T. J. Hamilton, J. Tapson, and A. van Schaik, "A mixed-signal implementation of a polychronous spiking neural network with delay adaptation," *Frontiers in neuroscience*, vol. 8, p. 51, 2014. 92

[283] T. Hwu, A. Y. Wang, N. Oros, and J. L. Krichmar, "Adaptive robot path planning using a spiking neuron algorithm with axonal delays," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 2, pp. 126–137, 2017. 92

[284] C. E. Graves, C. Li, X. Sheng, D. Miller, J. Ignowski, L. Kiyama, and J. P. Strachan, "In-memory computing with memristor content addressable memories for pattern matching," *Advanced Materials*, vol. 32, no. 37, p. 2003437, 2020. 92

[285] M. A. Lastras-Montaño and K.-T. Cheng, "Resistive random-access memory based on ratioed memristors," *Nature Electronics*, vol. 1, no. 8, pp. 466–472, 2018. 92

[286] A. Yousefzadeh, E. Stromatias, M. Soto, T. Serrano-Gotarredona, and B. Linares-Barranco, "On practical issues for stochastic stdp hardware with 1-bit synaptic weights," *Frontiers in neuroscience*, vol. 12, p. 665, 2018. 93

[287] C. Song, B. Liu, W. Wen, H. Li, and Y. Chen, "A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system,"

in *2017 IEEE 6th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*. IEEE, 2017, pp. 1–6. 93

[288] Q. Yang, H. Li, and Q. Wu, "A quantized training method to enhance accuracy of reram-based neuromorphic systems," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5. 93

[289] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognitive science*, vol. 11, no. 1, pp. 23–63, 1987. 93

[290] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, no. 1, pp. 1–10, 2016. 93

[291] E. Perot, P. de Tournemire, D. Nitti, J. Masci, and A. Sironi, "Learning to detect objects with a 1 megapixel event camera," *arXiv preprint arXiv:2009.13436*, 2020. 93

[292] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223. 94

[293] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Advances in neural information processing systems*, 2014, pp. 2654–2662. 94

[294] E. Oyallon, E. Belilovsky, and S. Zagoruyko, "Scaling the scattering transform: Deep hybrid networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5618–5627. 94

[295] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019. 94

[296] W. Brendel and M. Bethge, "Approximating cnns with bag-of-local-features models works surprisingly well on imagenet," *arXiv preprint arXiv:1904.00760*, 2019. 94

[297] M. Neggazi, M. Bengherabi, Z. Boulkenafet, and A. Amira, "An efficient fpga implementation of gaussian mixture models based classifier: Application to face recognition," in *2013 8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA)*. IEEE, 2013, pp. 367–371. 94

[298] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigne, "Spiking neural networks hardware implementations and challenges: A survey," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 2, pp. 1–35, 2019. 94

[299] H. Alashwal, M. El Halaby, J. J. Crouse, A. Abdalla, and A. A. Moustafa, "The application of unsupervised clustering methods to alzheimer's disease," *Frontiers in computational neuroscience*, vol. 13, p. 31, 2019. 94