Simple and complex spiking neurons: perspectives and analysis in a simple STDP scenario

Davide L Manna[1], Alex Vicente Sola[1], Paul Kirkland[1], Trevor Bihl[2], Gaetano Di Caterina[1]

[1] Neuromorphic Sensor Signal Processing Lab, Centre for Image and Signal Processing, Electrical and Electronic Engineering, University of Strathclyde, Glasgow, United Kingdom .

[2] Air Force Research Laboratory, Wright Patterson AFB, OH

E-mail: davide.manna@strath.ac.uk

Abstract. Spiking neural networks (SNNs) are largely inspired by biology and neuroscience and leverage ideas and theories to create fast and efficient learning systems. Spiking neuron models are adopted as core processing units in neuromorphic systems because they enable event-based processing. Among many neuron models, the integrate-and-fire (I&F) models are often adopted, with the simple Leaky I&F (LIF) being the most used. The reason for adopting such models is their efficiency and/or biological plausibility. Nevertheless, rigorous justification for adopting LIF over other neuron models for use in artificial learning systems has not yet been studied. This work considers various neuron models in the literature and then selects computational neuron models that are single-variable, efficient, and display different types of complexities. From this selection, we make a comparative study of three simple I&F neuron models, namely the LIF, the Quadratic I&F (QIF) and the Exponential I&F (EIF), to understand whether the use of more complex models increases the performance of the system and whether the choice of a neuron model can be directed by the task to be completed. Neuron models are tested within an SNN trained with Spike-Timing Dependent Plasticity (STDP) on a classification task on the N-MNIST and DVS Gestures datasets. Experimental results reveal that more complex neurons manifest the same ability as simpler ones to achieve high levels of accuracy on a simple dataset (N-MNIST), albeit requiring comparably more hyper-parameter tuning. However, when the data possess richer Spatio-temporal features, the QIF and EIF neuron models steadily achieve better results. This suggests that accurately selecting the model based on the richness of the feature spectrum of the data could improve the whole system's performance. Finally, the code implementing the spiking neurons in the SpykeTorch framework is made publicly available.

Spiking Neural Networks, STDP, Unsupervised Learning, Spiking Neurons, Temporal Features

1. Introduction

As technology in the Neuromorphic (NM) computing field keeps on advancing, so are the software methodologies and algorithms that can leverage the low-power, low-latency and event-driven properties that characterize NM [1]. Often, inspired by the success of conventional deep learning, this results in the development of Spiking Neural Networks (SNNs). When it comes to designing an SNN learning system for some machine learning task, researchers are faced with many decisions to make. Among these comes the choice of a particular neuron model. This specific aspect of the development of an SNN is an extremely sensitive one as spiking neurons are the core processing units of an NM system. To draw a parallel with the conventional Deep Learning (DL) research, spiking neurons can be thought of as being activation functions (such as the ReLU, ATAN etc), but holding an internal state. The dynamics of this state through time are governed by the differential equations that constitute the spiking neuron model.

Different neuron models exhibit different state dynamics. From a neuroscience point of view, these differences are immediately clear [2]. Some models are able to capture certain intrinsic behaviours of neurons, e.g. they can burst, chatter or fast-spike, while others cannot. Some models are also better at approximating subthreshold dynamics, thus possibly being more accurate representations of real neurons. However, it is still unclear how this ability translates into applicability in SNNs. There is in fact no definite answer onto whether certain types of neural dynamics can be beneficial to particular SNN applications, nor any common knowledge on the criteria that should drive the choice of such neurons in relation to such dynamics.

Within some specific contexts, the choice is constrained by the available hardware. As a matter of fact, several neuromorphic chips allow to only adopt the specific neuron model that the chip is able to emulate. BrainScaleS [3] for instance allows to only adopt the Adaptive Exponential Integrate-and-Fire neuron model; NeuroGrid [4] allows only an Adaptive Quadratic Integrate-and-Fire model ; Loihi [5], SyNAPSE [6, 7] and TrueNorth [8] allow only Leaky Integrate-and-Fire (LIF) based models; BiCoSS [9] allows using different models, including the LIF, Izhikevich's, and Hodgkin-Huxley's models, for large-scale SNNs and brain simulations. Until recently, the only chip allowing the implementation of any type of neuron model was SpiNNaker [10]. However, the recently released Loihi 2 adds to the list of chips with programmable neuron models [11], hence highlighting the importance of an accurate investigation on this matter.

It is thus interesting to look at what are the most suitable neurons models for SNN development. This can help to understand if the dynamics of the neurons relate, in some way, to the dynamics of the spatio-temporal features of the data.

Simple LIF neurons are the de-facto standard choice when it comes to SNN design [12–24]. When a rationale for this is provided, this choice is often attributed to the simplicity and, consequently, to the efficiency of the LIF neuron model. Whatever the case, such reasons hardly account for the accuracy performance of the task at hand, neither for the temporal feature representation capability of the model. Other works rely on more complex neuron models [25–28], attributing the choice to the biological plausibility or, once again, to efficiency. Neurons with different dynamics are present in areas of the brain with different functionalities [29–31]; however, the same does not apply to spiking neuron models in ML, where often the simplest neurons are used, leaving it unclear whether there are advantages or disadvantages relative to different types of NM data.

This work aims to answer the following research questions:

- Does the chosen neuron model influence the performance of an SNN?
- Should the choice of the neuron model be related with the data it will have to process?

Specifically, we are interested in understanding whether neuron models with different and more complex neuronal dynamics display any advantages over simpler (LIF) ones in an unsupervised learning context. Furthermore, we investigate whether such differences might exist depending on the task set to the network; hence we perform experiments using two different datasets. To do this, we first develop a basic experiment, which represents a simple yet efficient way to start a comparison between neuron models. We select neuron models so that they scale up in terms of complexity and spiking patterns, and evaluate their performance within the same neural network architectures on the N-MNIST dataset [32]. Then, we use the same neural network and train it on classification tasks taken from the DVS Gestures dataset [33], which displays a richer distribution of events. In order to perform the aforementioned experiments, we further contribute by enriching the SpykeTorch [34] framework with a new set of spiking neurons‡.

The rest of this paper is organized as follows: in Section 2 we provide some background information regarding the multitude of neuron models found in the literature and highlight some relevant related works; in Section 3 we present our experimentation pipeline in detail, focusing on the datasets, neural network design and learning paradigms; Sections 4 and 5 contain respectively the results obtained through our experiments and initiate an in-depth discussion on such results; Section 6 concludes the paper.

2. Background and Related Works

Spiking neuron models were first born in the field of neuroscience and neurophysiology, where mathematical models were developed to reproduce what was found by recording the activity of real neurons [35–37]. Early spiking neural networks resulted from studies aimed at understanding biological dynamics [38], or simulating areas of the brain and neuronal interactions [39–41]. The shift towards their use for computational tasks was gradual and was formalised afterwards [42, 43], with the focus generally being on the

‡ Code available at https://www.github.com/daevem/SpykeTorch-Extended

overall computational abilities of the network. When it comes to developing SNNs for ML applications, spiking neurons can be thought of as stateful activation functions. This means they retain a state of their value (the membrane potential) reached through previous inputs. They are thresholding functions, therefore allowing to only forward information upon the reaching of a set threshold .

In conventional DL, activation functions have been extensively studied due to their importance in the propagation of the information. Nonlinear functions such as the sigmoid and Tanh were introduced to break the linearity of multilayer Perceptrons [44,45]. Rectified Linear Units (ReLUs) substituted them to solve the vanishing gradient problem and allow deeper networks. Further variants [46–49] addressed other issues like the dying ReLU problem and helped to improve the performance of the networks [50–55]. Instead, in the context of SNNs and spiking neurons, it is hard to find works in the literature relative to the differences in the use of different neuron models in SNNs for NM and DL applications. To the best of our knowledge, the only work considering the role of spiking neurons in learning from a DL point of view is the one by Traub et al. [56]; however, they focus on the qualitative properties of Spike-Timing-Dependent Plasticity (STDP) and the mean firing rate of the system after training. Furthermore, they consider a non-NM dataset and a different set of neurons.

Most of the works on spiking neurons concentrate on the neurobiological aspects they expose. One of the most influential works in this matter is the one by Izhikevich [57], which compared several models of spiking neurons (LIF, LIF with adaptation, LIF-or-burst, resonate-and-fire, QIF, Izhikevich's, FitzHugh-Nagumo, Hindmarsh-Rose, Morris-Lecar, Wilson, Hodgkin-Huxley), outlining their ability to reproduce observed neuronal behaviours and the cost (in terms of floating-point operations) of implementing such neurons in software applications. Similar work was conducted in [58], but focusing on a smaller subset of neurons (LIF, Izhikevich's, FitzHugh-Nagumo, Wilson, Hodgkin-Huxley) and analysing their numerical stability. Although they closely study spiking neuron models, the two studies above concentrate on their computational costs and the intrinsic biological mechanics that each model can reproduce. However, they do not consider the effect of using spiking neurons with different dynamics in a DL system.
A number of other works concentrate on the efficacy of neuron models in representing observed cortical neurons firing patterns. One example is given by [59], where the authors make an exploratory analysis of how parameters influence Izhikevich's neurons in showing different spiking patters. Still regarding Izhikevich's neurons, Kumar et al. [60] estimate parameters that allow the neuron model to optimally reproduce a given spike train. Teeter et al. [61] use a generalized version of the LIF neuron model (GLIF) to understand whether more complex models allow to predict spike timing behaviors more closely; they conclude that this ability does not increase monotonically with the complexity, nor with the ability to reproduce sub-threshold dynamics. In [62] it is argued that integrate-and-fire (I&F) neuron models are good enough estimators of input spike trains when coupled with an adaptation variable. This is both quantitatively and qualitatively shown by the authors and provides a good ground to the adoption of this

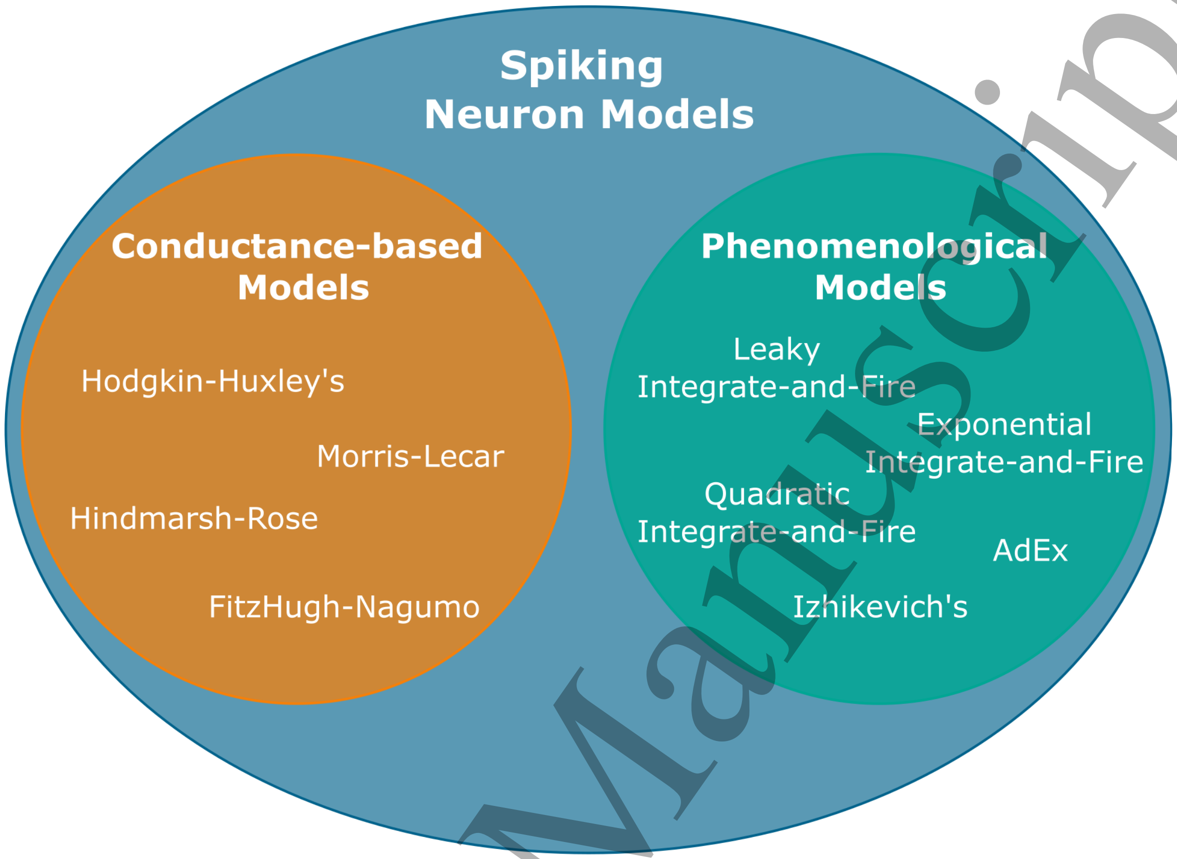


Figure 1: Venn Diagram of Some Spiking Neurons.

family of neuron models.
Finally, in [63] the authors compare different neuron models embedded in a liquid state machine (LSM), a particular type of reservoir computing network. They evaluate their model with two different input patterns and use Euclidean distance and entropy to estimate the ”separation” ability of the LSM, i.e. its ability to generate different response patterns for different stimuli. They test their LSM using six spiking neuron models (I&F, resonate-and-fire, FitzHugh-Nagumo, Hindmarsh-Rose, Morris-Lecar, and Izhikevich's) and perform experiments varying the density of the connections between the neurons. They found that the LSM failed to achieve satisfactory levels of separation only when using Izhikevich's neurons. Other models allowed better separation levels depending on the density of the connections. They conclude by postulating that, for LSM implementations, Morris-Lecar, resonate-and-fire, and Hindmarsh-Rose models are most suitable.

2.1. Neuron Models in the Literature

Neurons in the human brain differ by several characteristics, ranging from the type of neurotransmitters used, to the shape they have and the spiking patterns they expose. As a common ground, however, they all share a basic structure composed of a dendritic tree (input channels), an axon (output mean), and a soma (core) [2]. Dendrites (and Axons) can be further broken down into several blocks through which signals travel. In the literature, the neurons have been modelled with different levels of abstraction and detail. Works such as [64–68] focus on the definition and use of so-called multi-compartment neuron models. These kind of models try to account for each compartment of a neuron (dendrites, axon and soma) individually to more closely replicate biological evidence. However, a large number of models considers the neurons as dimensionless entities (point neurons), hence focussing on the modelization of the soma only. They can be roughly subdivided in two larger groups (see Figure 1), the bio-physical or conductance-based models and the event-based or integrate-and-fire models. In this work, we concentrate on the study of point neurons as they are largely used in SNNs for ML applications.

2.1.1. Conductance-based Models This class of neuron models is characterized by the fact that all the variables and parameters present in the model have a biophysical correspondence and are therefore measurable through experiments [69]. Among them, the Hodgkin-Huxley (HH) model is considered to be one of the most important in computational neuroscience and defines a system of 4 non-linear differential equation with four variables and a number of parameters. While further levels of complexity can be attained by including further variables in the model, this is not amenable to mathematical analysis. In fact, other simpler conductance-based models have been derived in the literature in order to ease the analysis, while still retaining biophysical plausibility. Some examples are the FitzHugh-Nagumo model [70, 71], the Hindmarsh-Rose [72] and the Morris-Lecar model [73]. Nevertheless, they still remain rather complex for what concerns analysis and computation, therefore this family of neuron models is often used only when studying single-cell or small population dynamics [57].

2.1.2. Phenomenological Models The family of Integrate-and-Fire or phenomenological neuron models comprises all those models that treat spikes as stereotypical events in time [2]. Therefore, each spike is completely described by the time at which it occurred, or was emitted. Integrate-and-fire models require at least two equations, one describing the dynamics of the membrane potential and the other one defining the action potential generation. Events are integrated over time and convey electrical charges that can cause excitation or inhibition of the membrane potential of the receiving neuron. Differently from the conductance-based models, the phenomenological ones are more indicated for the development of neural networks [57, 74, 75], thanks to their overall lower complexity and the lower number of parameters, which enable easier fitting.

The simplest model, apart from the perfect integrator, is the LIF [35]. The dynamics of the membrane potential are here described by the following linear differential equation:

$$\tau_m \frac{du}{dt} = -(u(t) - u_{rest}) + R \cdot I \tag{1}$$

where τ_m is the membrane time constant, $u(t)$ is the membrane potential as a function of time, u_{rest} is the resting potential of the membrane, R is a resistance and I is the incoming current. Here, the term $-(u(t) - u_{rest})$ accounts for the leakage of the membrane potential.

Although this model lacks the ability to describe most of the neuronal dynamics, it is the most common choice for the development of large scale neural networks, mostly because of its efficiency.

More complex I&F models attempt to account for some non-linear dynamics of neurons as a function of the value of their membrane potential in a certain moment in time. Two examples are given by the Exponential Integrate-and-Fire (EIF) model [76] and by the Quadratic Integrate-and-Fire neuron (QIF) or Theta neuron [69, 77]. As shown in Eq.(2), EIF model expands on the LIF model by including an exponential dependency on the current state of the membrane potential:

$$\tau_m \frac{du}{dt} = -(u(t) - u_{rest}) + \Delta_T \exp\left(\frac{u(t) - \Theta_{rh}}{\Delta_T}\right) + R \cdot I, \tag{2}$$

where Δ_T is a parameter determining the sharpness of the exponential curve and Θ_{rh} is the rheobase threshold. When $u > \Theta_{rh}$, the exponential term becomes prominent over the linear one, leading to an upswing of the curve that takes the membrane potential to infinity in finite time.

The QIF model, given by Eq.(3), employs a quadratic dependency from the membrane potential:

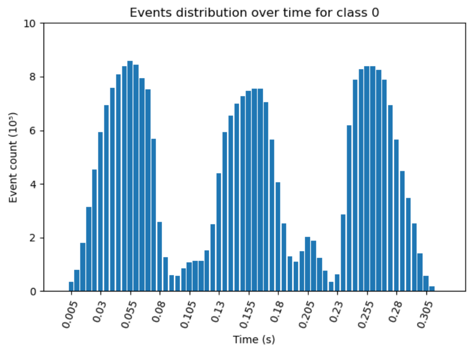
$$\tau_m \frac{du}{dt} = a_0(u(t) - u_c)(u(t) - u_{rest}) + R \cdot I, \tag{3}$$

where a_0 is a parameter of the model that regulates the magnitude of the dependency from the membrane potential and u_c is a cut-off threshold such that, when $I = 0$ and $u > u_c$, the membrane potential grows until the emission of a spike.

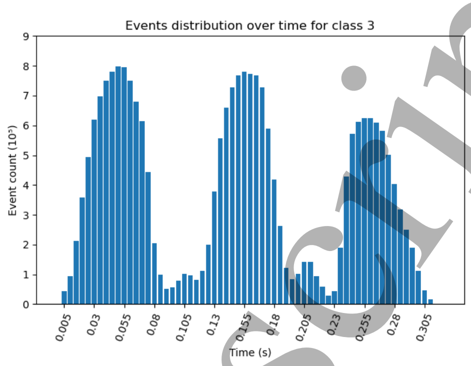
Both the QIF and the EIF bring in a further level of complexity with the inclusion of non-linear dependencies that affect both the computational costs and the ease of analysis, but allow for a more precise generation of spikes [2]. Additionally, a hidden cost lies in the use of two extra parameters in each of them.

2.1.3. Other Multi-Variable I&F Models With the inclusion of adaptation variables within the neuron model, it is possible to account for a larger number of spiking patterns and to render possible the manifestation of spike bursts, spike-adaptation responses and irregular spiking [2]. This comes at the cost of more differential equations in the model (one per variable) and two relevant examples are given by the Adaptive Exponential Integrate-and-Fire (AdEx) [78] neuron model, which builds on top of the EIF, and by Izhikevich's neuron model [79] which builds on top of the QIF.

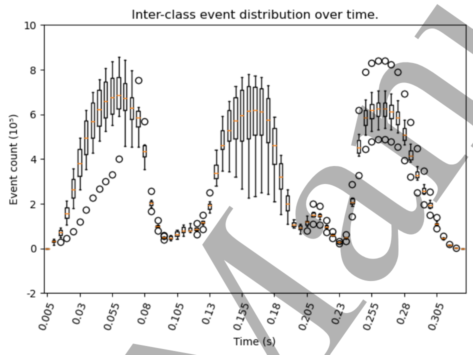
A number of neuron models have been theorized in the literature, all answering to different modelling needs or considering different aspects of the observed neuronal



(a)



(b)



(c)

Figure 2: Visualization of the number of events over time. Figure 2a and 2b for classes "0" and "3" are reported as representative. They depict the count of events for each time-step among samples of one class. Figure 2c reports the collective mean and variation of events throughout all the classes. As can be seen, events tend to appear always within the same time ranges for all the samples of all the classes in the dataset, thus highlighting the lack of temporal significance. Values on the y-axis are scaled by a factor of 10^5.

behaviours. The ones cited above are amongst the most relevant for what concerns this study and NM computing. In fact, as reported above, the LIF model is the most widely used in the development of SNN for NM applications, but at the same time, models like the AdEx and Izhikevich's have received a lot of attention in the literature. The EIF and the QIF are on one hand the baseline of the AdEx and Izhikevich's models respectively, and, on the other hand, a slightly more complex single-variable alternative to the LIF neuron model. As such, in this study, we will focus on these single-variable models.

3. Methods

We are interested in assessing the performance of different neuron models within the context of a Spiking Convolutional Neural Network (SCNN) trained with STDP. Since many factors could determine the outcome of the training, we begin by designing a simple experiment which involves the minor number of structural elements possible. This is done in order to limit the number of components that might impact the overall system performance. Therefore, we use a single-layer convolutional network in which spiking neurons are embedded right after the convolution operation on the input. The task set to the SCNN is a binary classification task, with the pairs of classes taken from the Neuromorphic MNIST dataset [32] and the DVS Gestures dataset [33], which contain event-based data samples. To develop the learning pipeline, we utilize SpykeTorch [34] as a base framework and build on top of it to include the elements required by this study, such as the diverse spiking neuron models.

3.1. Event-based Data

To assess the performance of our simple network, we select the two natively neuromorphic datasets mentioned above. We purposely discard other non-native NM datasets as they do not possess a temporal domain, nor data is originally event-based.

Data in the N-MNIST dataset is collected by recording MNIST digits shown on a screen using a moving DVS camera. Specifically, the camera makes the same 3 predefined movements for every sample, each lasting roughly 100 ms. In this way, although the dataset is built on top of a non-neuromorphic one, data samples in the dataset are natively event-based, rather than being converted from a static image. Figure 2 reports the count of events per time for two example classes and throughout all the classes of the dataset. By contrast, data samples in the DVS Gestures dataset (see Figure 3 for the inter-class distribution of events over time) are recorded using a fixed DVS camera in front of which participants move their arms according to instructions. Thus, 11 different classes of gestures are obtained, including for example arm rotation, waving or performing air guitar. The 11th class encodes "Other" random movements and is not considered in this work for simplicity.

Event data comes in the form of Address Event Representation (AER)-encoded files in which every sample is constituted by a sequence of events. Events are characterized by the specific time at which they occurred, by the location on the 2D plane and by the polarity (negative or positive light change). Similarly to [80], to make data usable by a 2D Convolutional Neural Network (CNN) we populate a 2D image using all the events that took place between time t and $t + dt$, allowing at most 1 event per (x, y) coordinates. For simplicity, we consider all events as being positive and use a batch size of 1. As a result, the network processes only 1 event map with a time resolution dt belonging to only 1 sample at a time. Finally, we characterize each event with a value of $\frac{1}{t_s}$ in line with [2]. This is done in order to preserve the amount of charge that a spike

Table 1. Table of hand-tuned parameters used for neurons. Each column represents a different parameter, as outlined in Section 2.1. The capacitance C is used instead of the resistance R by leveraging the equality $R = \frac{\tau_{rc}}{C}$. For every neuron, the time-step size used was 0.02, and the voltage threshold was recalculated every 100 samples.

Neurons	τ_m	u_{rest}	C	Δ_T	Θ_{rh}	a	u_c
LIF	0.2	0	0.1	-	-	-	-
EIF	0.2	0	0.1	1352	216	-	-
QIF	0.2	0	0.1	-	-	0.01	216

carries regardless of the time-step (t_s) size. No further pre-processing is applied to the data.

3.2. Spiking Neurons Implementation

The phenomenological family of neuron models is the best option when developing spiking neural networks, as outlined in Section 2.1. We specifically concentrate on three integrate and fire neurons, namely the LIF, the QIF and the EIF. The parameters used for these models in the same-parameter setting of experiments (see Section 4) are reported in Table 1. The LIF is the most widely used neuron that embeds a time dependency through the membrane potential leakage. The QIF and the EIF represent valid alternatives given their ability to best fit observed cortical neurons [2, 69]. Since they are single-variable models, they stand for a fairer comparison with the LIF, which is single-variable too. Indeed they all depend on the value of the membrane potential, but they all employ different types of dependencies from it. Furthermore, they are the base on which other more complex and popular neuron models are built on, respectively Izhikevich's neuron and the AdEx neuron model.

The SpykeTorch framework comes with a simple version of a I&F neuron model. To enable the use of these neuron models for our experiments, we expand on the framework and implement the above models by adapting the equations provided in [2]. State updates in the neurons are thus calculated on a per time-step basis, where each call to the neuron layer corresponds to an advancement of t_s time from the previously calculated update. Each neuron layer generates a number of neurons that reflects the size of the incoming post-synaptic currents, multiplied by the number of neuron populations that was specified at creation time. This allows for a more seamless inclusion in any point of an SNN. When a neuron in a population emits a spike, all the other neurons belonging to the same population are inhibited and put in a refractory state to promote learning in other populations. Each neuron model is implemented as a class inheriting from a parent Neuron class, in an object-oriented programming style. This differs from the original SpykeTorch implementation style, however, this approach was required for neurons that maintain an internal state. Besides, compatibility with the modules

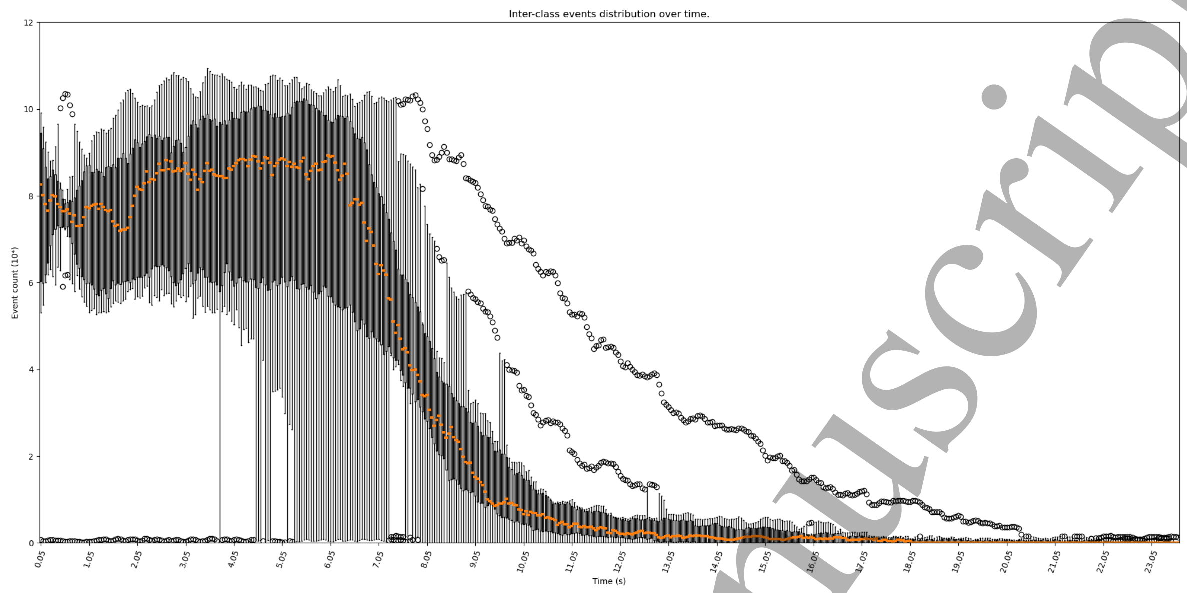


Figure 3: Visualization of the number of events over time in the DVS Gestures dataset. The events span across the whole time domain and are highly dense for about the first 7 seconds. At that point, they start to decrease in number, however, the tail of the curve continues for a long time. This is due to some of the gestures lasting longer than others.

and neurons in SpykeTorch is maintained. Further features and details are present in the actual implementation, however, these are not relevant to the current study and the authors point the reader to [81] and to the dedicated repository page for in-depth descriptions.

3.3. SCNN and Learning

We develop a simple feedforward convolutional network for our experiments, with a single convolutional layer that parses the input and connects it to the spiking neurons. Figure 4 illustrates the pipeline for a better understanding. The weights of the kernel can be thought of as being synapse strengths and the resulting feature map is the input to a set of spiking neurons arranged accordingly. In other words, the convolution represents the connectivity scheme for the spiking neurons. The use of a convolutional layer to connect inputs with the spiking neurons allows them to more easily learn spatial features. The weights of the convolutional layer are the only parameters being learnt by the network and no pooling nor normalization is applied.

We use the STDP presented and used in [82–84] as an unsupervised learning rule. By employing this kind of learning rule and, therefore, using it to evaluate neurons, we want to take a step closer to local feedforward learning, which is believed to be the key to exploit the potentials of neuromorphic engineering at its best [1, 85]. The mentioned

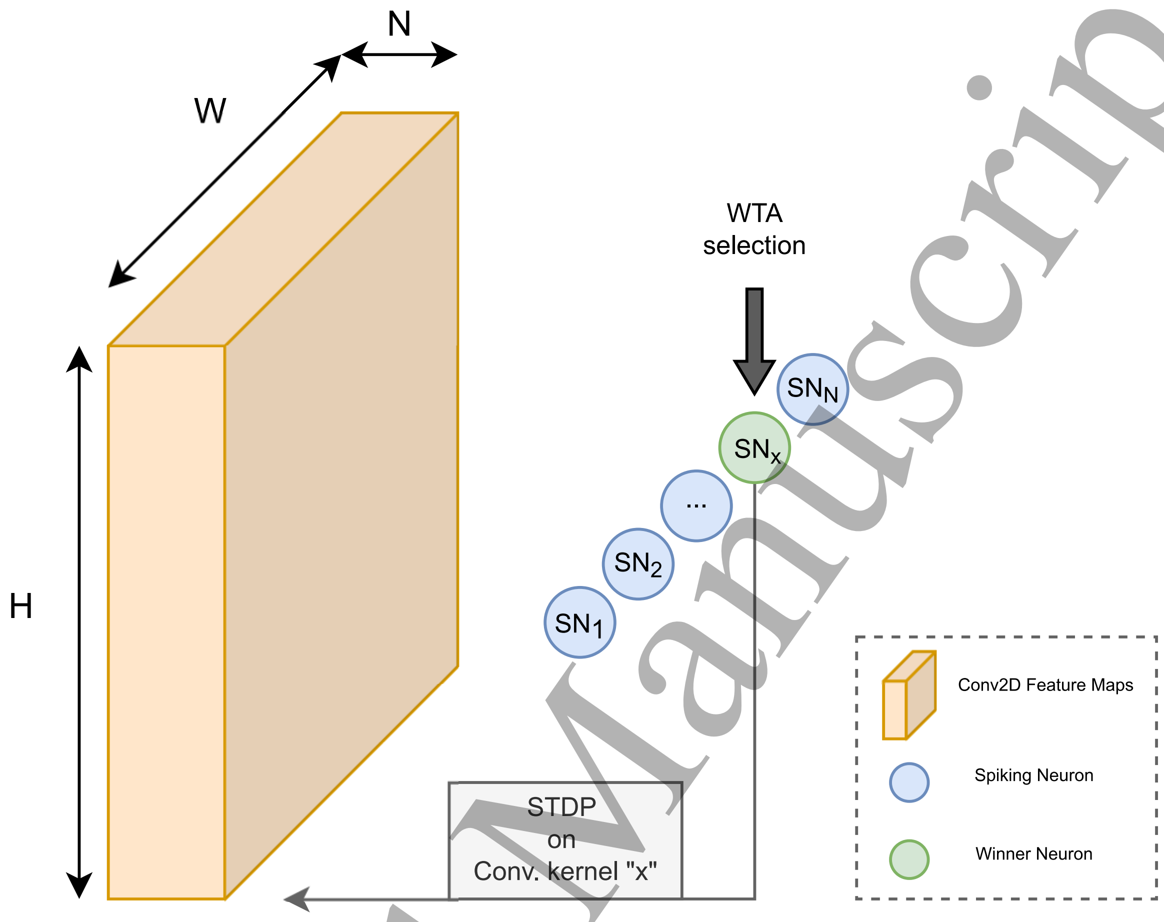


Figure 4: Diagram of the Learning Pipeline. A 2D convolution layer parses the input spike map and produces N feature maps width size WxH. Each value in each feature map is fed to a distinct neuron and the one spiking earliest is chosen as a winner by the WTA mechanism. STDP weight updates are then applied to the convolution kernel corresponding to that neuron. For ease of visualization, neuron populations are here represented as individual neurons, but each of them actually contains WxH neurons, i.e. like one feature map.

STDP rule applies the following weight updates:

$$\Delta W_{i,j} = \begin{cases} A^{+} \times (W_{i,j} - LB) \times (UB - W_{i,j}) & \text{if} \quad T_j \leq T_i, \\ A^{-} \times (W_{i,j} - LB) \times (UB - W_{i,j}) & \text{if} \quad T_j > T_i, \end{cases} \qquad (4)$$

where $W_{i,j}$ is the weight of the synapse connecting neuron j (pre-synaptic) to neuron i (post-synaptic), LB and UB are a lower and an upper bound value respectively, T_j is the timing of the spike emitted by neuron j, T_i the timing of the spike emitted by neuron i, and A^{+} and A^{-} are two parameters used to scale the weight update.

It is thus a system of two parabolic equations that are applied depending on whether Long Term Potentiation (LTP) or Long Term Depression (LTD) needs to take place.

One of the consequences of using this learning rule is that weight updates are self-regularized. In fact, the closer the weights get to the boundary values, the smaller the updates will be, allowing the weights to be refined more granularly as the learning proceeds. Another aspect that is important to highlight is in how this learning rule is applied. While the original theorization of Hebbian rules such as STDP states that the weight update should be proportional in value and sign on the time-difference between the post- and the pre-synaptic spikes, in a software implementation some approximations are required. Therefore, for each time-step of the execution, we apply LTP on weights connected to input locations where there has been a spike, and LTD in all the others. In order to promote competitive and differentiated feature learning, we also employ a k-Winners Take All (WTA) (with k=1) learning paradigm. WTA allows only k neurons per time-step to be eligible for STDP updates, specifically the ones firing sooner.

Finally, as an homeostatic mechanism to allow neurons to keep on firing despite the changes in their synaptic weights, we re-calculate individual neurons' thresholds according to Eq.(5). This for of adaptive thresholding is often required when employing STDP, and a common practice for SNNs [83].

$$V_{thresh} = \lambda \cdot R \cdot A \cdot \frac{t_s}{\tau_m} \cdot \overline{W} \cdot (W_k \cdot H_k) \cdot n_c, \tag{5}$$

and since $\tau_m = R \cdot C$, we can equivalently re-write:

$$V_{thresh} = \lambda \cdot A \cdot \frac{t_s}{C} \cdot \overline{W} \cdot (W_k \cdot H_k \cdot N_c), \tag{6}$$

where A is the amplitude of the spike, in our case assigned to be $A = \frac{1}{t_s}$, R is the resistance, C is the capacity, τ_m is the membrane time constant, $\overline{W}$ is the average value of the synaptic weights, W_k, H_k and N_c are the width, height and depth (number of channels) of the synaptic kernel, and λ is a regulation parameter that takes values in the range $[0, 1]$. In Eq.(6), the term $A \cdot \frac{t_s}{C} \cdot \overline{W}$, can be explained as being the average effect perceived on the membrane potential as a result of a single spike, whereas the second term, $(W_k \cdot H_k) \cdot n_c$, scales this effect to the size of the synaptic kernel. Therefore, Eq.(6) calculates what would be the average post-synaptic potential perceived in the case the input was dense with spikes. The parameter λ serves as a regulation of what percentage of this amount would be necessary to reach before emitting a spike.

3.4. Classification

Because we employ an unsupervised learning rule, labels are not used at any point during the learning of the weights. However, labels are needed to classify data samples, therefore we adopt a system similar to [86, 87] and count, for each neuron, the number of times it spiked in response to samples having a given label. At the end of the training phase, each neuron is assigned the label for which it spiked the most during training.
During the inference phase, for each data sample, a sequence of spikes is collected and

Table 2. Table of optimized parameters used for neurons. Each column represents a different parameter, as outlined in Section 2.1. The capacitance C is used instead of the resistance R by leveraging the equality $R = \frac{\tau_{rc}}{C}$. For every neuron, the time-step size used was 0.02, and the voltage threshold was recalculated every 100 samples. HC stands for Hand Clapping, while RHW stands for Right Hand Wave.

Neurons	0 vs 1							HC vs RHW						
	τ_m	u_{rest}	C	Δ_T	Θ_{rh}	a	u_c	τ_m	u_{rest}	C	Δ_T	Θ_{rh}	a	u_c
LIF	0.0602	0	0.2983	-	-	-	-	0.1435	0	0.3020	-	-	-	-
EIF	0.2578	0	0.2178	32	91.14	-	-	0.2389	0	0.2086	32	69.42	-	-
QIF	0.2804	0	0.2178	-	-	0.001	69.72	0.0621	0	0.1026	-	-	0.0393	275.95

weighted depending on the order they arrived. These are then summed and the label corresponding to the highest value is selected as the classification outcome.

3.5. Hyper-parameter Optimization

Defining a good set of parameters for a machine learning system is a non-trivial task. This often requires a lot of expertise and hand tuning and is greatly error-prone. To reduce the possibility of selecting sub-optimal parameters for neurons which would result in poor performance, we thus make use of an optimization system to find reasonably good parameters for our experiments. To this extent, we employ the BOHB optimization [88] using the HpBandSter library. This technique combines Bayesian Optimization (BO) and Hyperband (HB), a resource allocation and early stopping strategy. To use BOHB, we adapted our implementation so that it could be optimized using the HpBandSter library. More importantly, we defined the domains in which every parameter was allowed to vary. For example, the time-constant τ_m could be drawn from the interval $[0.06, 0.26]$. Moreover, because the optimization process could be task-specific, we performed separate optimizations for each different task. We summarize the final parameters in Table 2. As a result of this process, we can draw more robust conclusions about the performance of the neurons.

4. Results

We train the SCNN outlined in Section 3 using a subset of the N-MNIST and the DVS Gestures datasets. We randomly select 4 distinct couples of classes from each dataset and define, for each, a separate binary classification task. In this way, we aim to obtain more generalizable results and to reduce the dependency of the results on a particular coupling. Furthermore, the order in which data is presented to the SCNN might be influential in a system employing STDP as a learning rule. This is due to the fact that STDP rewards and builds on the inputs that are presented earlier. Therefore, to ensure independence from this behaviour of STDP, we repeat every experiment a total of 11 times per task.

Table 3. Table of results on the N-MNIST dataset. In each cell, the mean accuracy ± the standard deviation values are followed by the best accuracy found (after the comma). Values are rounded up to the closest second decimal value.

Neuron Model	0 vs 1	2 vs 9	3 vs 7	4 vs 8
LIF	0.77±0.11, 0.93	**0.74**±0.06, **0.79**	**0.73**±0.04, **0.78**	**0.57**±0.02, 0.59
EIF	**0.80**±0.12, **0.94**	0.64±0.07, 0.71	0.65±0.04, 0.71	0.55±0.04, **0.61**
QIF	0.57±0.03, 0.61	0.54±0.03, 0.58	0.51±0.02, 0.55	0.52±0.02, 0.55

4.1. Same Hyper-Parameters Training

We first conduct our experiments using hand-tuned hyper-parameters. These were found by a trial-and-error practice and represent a set of parameters that enabled learning for the task at hand. This means that neurons using these parameters were able to emit spikes and to have the weights adjusted in a way that enabled the learning of representations of the inputs. Where possible, we adopt the same hyper-parameters for all the neurons in all the experiments on each dataset. Since the QIF and the EIF models introduce 2 different hyper-parameters each, each of them undergoes a further hand-tuning. Results of the training sessions are shown in Table 3 for the N-MNIST-based tasks and in Table 4 for the DVS Gesture-based tasks. Here, for every task and neuron model, are reported the average and best test accuracies achieved, calculated according to Eq.(7):

$$accuracy = \frac{TP + TN}{TP + TN + FT + FN}, \tag{7}$$

where TP stands for true positive, TN for true negative, FT for false true and FN for false negative. On each column (task), the best average score and the absolute best score are highlighted in bold.

By examining the results on the N-MNIST-based tasks in Table 3, the LIF neuron model is found to perform better than the other two on average. Indeed, the EIF has higher average accuracy only on the 0 vs 1 task, whereas the QIF model fails to achieve accuracy levels high enough to match any of the other two counterparts.

Considering the results reported in Table 4, the situation differs slightly. The performance of both the LIF and the EIF neuron models, on average, decreases drastically, whereas the QIF maintains similar levels of accuracy as in the N-MNIST case. Nevertheless, both the EIF and QIF demonstrate superior classification abilities

Table 4. Table of results on the DVS Gestures dataset. In each cell, the mean accuracy ± the standard deviation values are followed by the best accuracy found (after the comma). Values are rounded up to the closest second decimal value. In the table, HC stands for Hand Clapping, RHW for Right Hand Wave, RACW for Right Arm Clockwise, AG for Air Guitar, RACCW for Right Arm Counter Clockwise, AR for Arm Roll, LACW for Left Arm Clockwise and AD for Air Drums.

Neuron Model	HC vs RHW	RACW vs AG	RHCW vs AR	LACW vs AD
LIF	0.53±0.06, 0.60	0.50±0.05, 0.56	0.54±0.13, 0.66	0.52±0.09, 0.69
EIF	**0.58±0.12, 0.77**	**0.50±0.04, 0.58**	0.53±0.11, 0.66	0.46±0.05, 0.52
QIF	0.57±0.10, 0.71	0.48±0.04, 0.52	**0.62±0.06, 0.67**	**0.53±0.09, 0.75**

throughout and their top accuracy levels often surpass those of the LIF model. These trends in the accuracy levels highlight two main aspects. Firstly, the complexity of the N-MNIST data is lower than that of the DVS Gestures dataset. Indeed, in both cases, the same neural network architecture and neuron models were employed, yet the accuracy in the DVS Gestures is on average considerably lower, hence highlighting the greater difficulty of the task. Data samples in the DVS Gestures dataset arguably have richer visual and temporal features which render the learning more difficult when compared to the N-MNIST. Secondly, the richer temporal diversity of the features might be better represented by means of neurons with richer voltage dynamics, such as the QIF and EIF. As reported by the experiments, in fact, these two are steadily better than the LIF models and even though in some instances one performs more poorly, the other still attains higher accuracy, possibly as a result of a better affinity to the temporal dynamics found in that particular task.

4.2. Optimized Hyper-Parameters Training

As a second set of experiments, we employ the optimization system outlined in Section 3.5 to obtain a set of hyper-parameters that is heuristically optimal for a specific scenario. The optimization is carried out on the "0 vs 1" and on the "HC vs RHW" tasks for each neuron model individually. We allow a total of 24 optimization iterations for each neuron and task, to not favor any experiment over the others. Results are reported in Table 5. Once again, after obtaining the optimized hyper-parameters, we train and evaluate each model a total of 11 times to increase the robustness of the results.

Table 5. Table of results using optimized hyper-parameters. In each cell, the mean accuracy ± the standard deviation values are followed by the best accuracy found (after the comma). Values are rounded up to the closest second decimal value. Optimization and evaluation is performed on one representative task per dataset only.

Neuron Model	0 vs 1	HC vs RHW
LIF	**0.95±0.02,** 0.982	0.53±0.05, 0.625
EIF	0.74±0.15, 0.93	**0.57±0.11, 0.67**
QIF	0.90±0.05, **0.985**	0.55±0.05, 0.625

Since the optimization is task-specific, we only evaluate our models on the two representative tasks they were optimized on. In the case of the "0 vs 1" task based on the N-MNIST, overall, the accuracy levels drastically increase. The LIF model accuracy grows by nearly 20 percentage points on average; however, the most striking increase is the accuracy of the QIF model, which gains 33 percentage points on average and 37.5 in the best case. This not only highlights the sensitivity of neuron models to their hyper-parameters, but also confirms that neurons with more complex dynamics can perform just as well as simpler ones.
In the case of the "HC vs RHW" task, instead, we see a slightly different trend. In the first place, the results are surprisingly worse than those obtained through hand-picked parameters. We hypothesize that this is because the optimization system required more iterations to find a good set of hyper-parameters. As stated above, we allowed the same number of optimization iterations as in the case of the N-MNIST dataset. However, given the higher complexity of the features present in the DVS Gestures dataset, it might have been necessary to allow more. Secondly, although still struggling to achieve higher accuracy levels, the SNNs employing QIF and EIF averagely outperform those with the LIF. This confirms the results obtained using the same hyper-parameters and strengthen the hypothesis that richer dynamics can be beneficial when employed on data with a richer set of temporal features.

Another point worth considering is the variability of the results obtained. Spanning from the N-MNIST-based tasks to the Gestures-based ones, the different neuron models demonstrate accuracy levels with a standard deviation of up to 15 percentage points. We hypothesize that this effect is caused by the order in which data is presented to the system in relation to the STDP learning rule which, as reported at the beginning of this section, is sensitive to such order. We also observe that the EIF model has higher fluctuations on average, which could possibly reflect a higher sensitivity to this effect.

4.3. Sensitivity to data presentation order

In the previous paragraph, we hinted at the possibility of EIF neuron model being particularly sensitive to the presentation order of the data as a result of being trained through STDP. Driven by this, in this section we perform a series of controlled experiments aimed at studying this possibility.

In order to assess whether any neuron is more sensitive to the order of the data, we must constrain some parameters of the experiment to ensure the final results do not depend on these. Since we use a homogeneous set of neurons (they all share the same parameterization), the only varying elements are the convolutional weights (randomly initialized), and the order of the data itself. Hence, consistently with our previous methodology, we design a set of 11 experiments per task and neuron model in which the initialization of weights was fixed using a seed for the random number generator. Further to this, we allow each neuron model to process data presented in the same order. In other words, we evaluate each neuron model using the same order of data before changing to an other order for a total of 11 times per task. By doing so, we rule out the possibility of any neuron model displaying certain sensitivity levels as a result of fortunate/unfortunate randomization of the data. As a measure of the network sensitivity to the presentation order of the data we utilize the standard deviation of the accuracy across all the experiments per each neuron and task. These are reported in Table 6 and Table 7. We report the average sensitivity with 95% confidence intervals for each neuron model in Figure 5.

We perform a one-way analysis of variance (ANOVA) to verify whether any neuron is significantly more sensitive than others. The test assumptions were checked. Levene's test was non-significant ($p = 0.256$), indicating that the assumption of homogeneity of variance was not violated. Normality was checked with a Q-Q Plot. No deviations were noted. We found no significant difference among the three neuron models in sensitivity to the order of the data, $F(2, 21) = 0.514$, $p = 0.605$, $\eta_p^2 = 0.047$. In the context of our experiments, these findings indicate that using either neuron model does not increase nor decrease the sensitivity to the presentation order of the data. Instead, such sensitivity is probably only due to the use of STDP as a learning rule.

5. Discussion

5.1. Implications of Using Different Neuron Models

The usage of spiking neuron models has some inherent implications on the machine learning pipeline from the implementation and the theoretical points of view.
Concerning the implementation, spiking neuron come with a whole set of hyper-parameters to tune. Considering the LIF, the simplest version requires a single parameter (the time constant or a leakage term), but other implementations might include up to 5 different parameters, such as the refractory period or the time-step size. If we switch to the QIF or the EIF, there are at least two new and non-optional

Table 6. Table of standard deviations on the N-MNIST dataset. Each cell reports the standard deviation calculated across all the experiments per each neuron and each task. Values are rounded to the closest third decimal. For each task, we highlighted in bold the highest values.

Neuron Model	0 vs 1	2 vs 9	3 vs 7	4 vs 8
LIF	**0.092**	0.063	**0.068**	**0.061**
EIF	0.054	**0.074**	0.056	0.024
QIF	0.076	0.069	0.065	0.021

Table 7. Table of standard deviations on the DVS Gestures dataset. Each cell reports the standard deviation calculated across all the experiments per each neuron and each task. Values are rounded to the closest third decimal. For each task, we highlighted in bold the highest values.

Neuron Model	HC vs RHW	RACW vs AG	RHCW vs AR	LACW vs AD
LIF	0.086	0.051	**0.122**	0.028
EIF	0.049	0.049	0.114	0.032
QIF	**0.088**	**0.079**	0.101	**0.074**

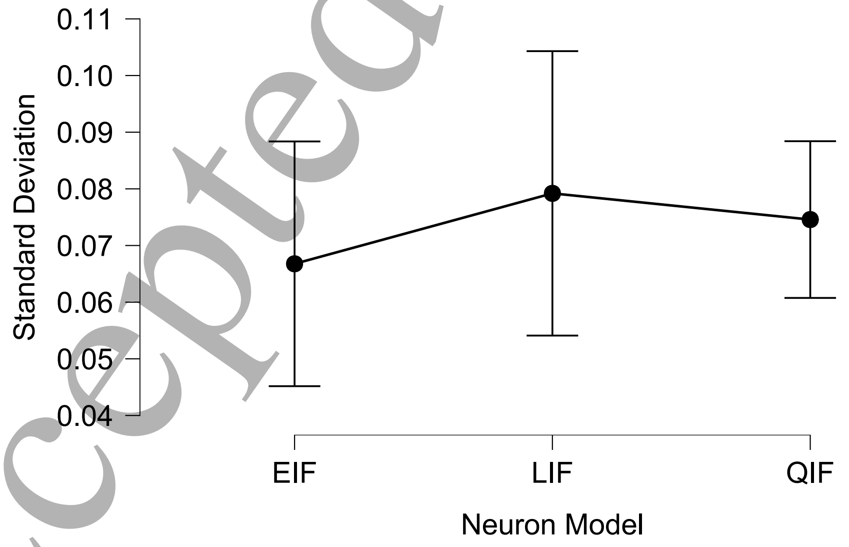
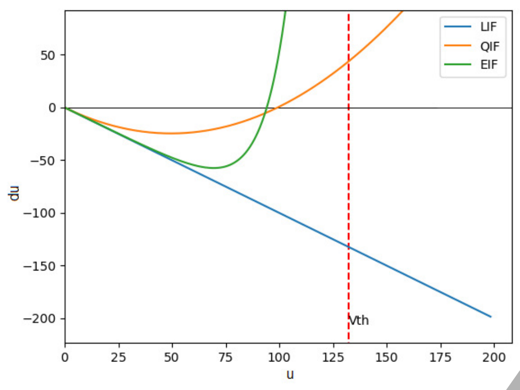


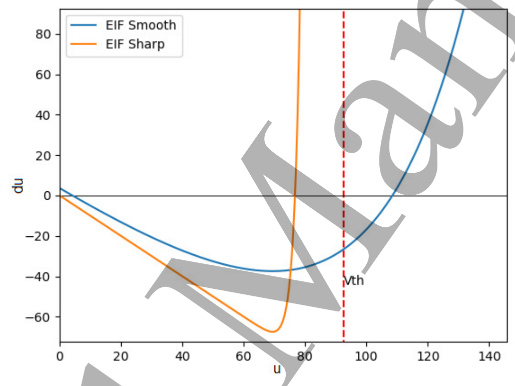
Figure 5: Plot of average standard deviations with 95% confidence intervals. The average was computed across all tasks per each neuron.

parameters to consider (see 2.1).

Determining a good set of hyper-parameters is a non-trivial task [75]. Although in the



(a)



(b)

Figure 6: Example of membrane potential dynamics of the spiking neuron models. In both figures, the y-axis represents the variation of the membrane potential du, while the x-axis represents the value of the membrane potential itself u. Figure 6a compares the three spiking neurons, whereas Figure 6b is an example of how varying the sharpness parameter Δ_T can affect the dynamics of the EIF.

NM field a lot of inspiration is taken from the human brain, it is not possible to simply assume that the same parameters that work in such a complex system would still be applicable in a simplification such as an SNN. We hence need to tweak parameters for our need or, alternatively, to define a parameter optimization strategy that does that heuristically in an automated way. However, also the latter solution often requires to make guesses about the domain in which parameters can vary and it requires a long time to compute. Furthermore, hyper-parameters can be correlated in some way, thus making both the hand-tuning and the automated optimization process more difficult. As a result, from an implementation point of view, using neuron models that require more hyper-parameters can significantly increase the usage complexity.

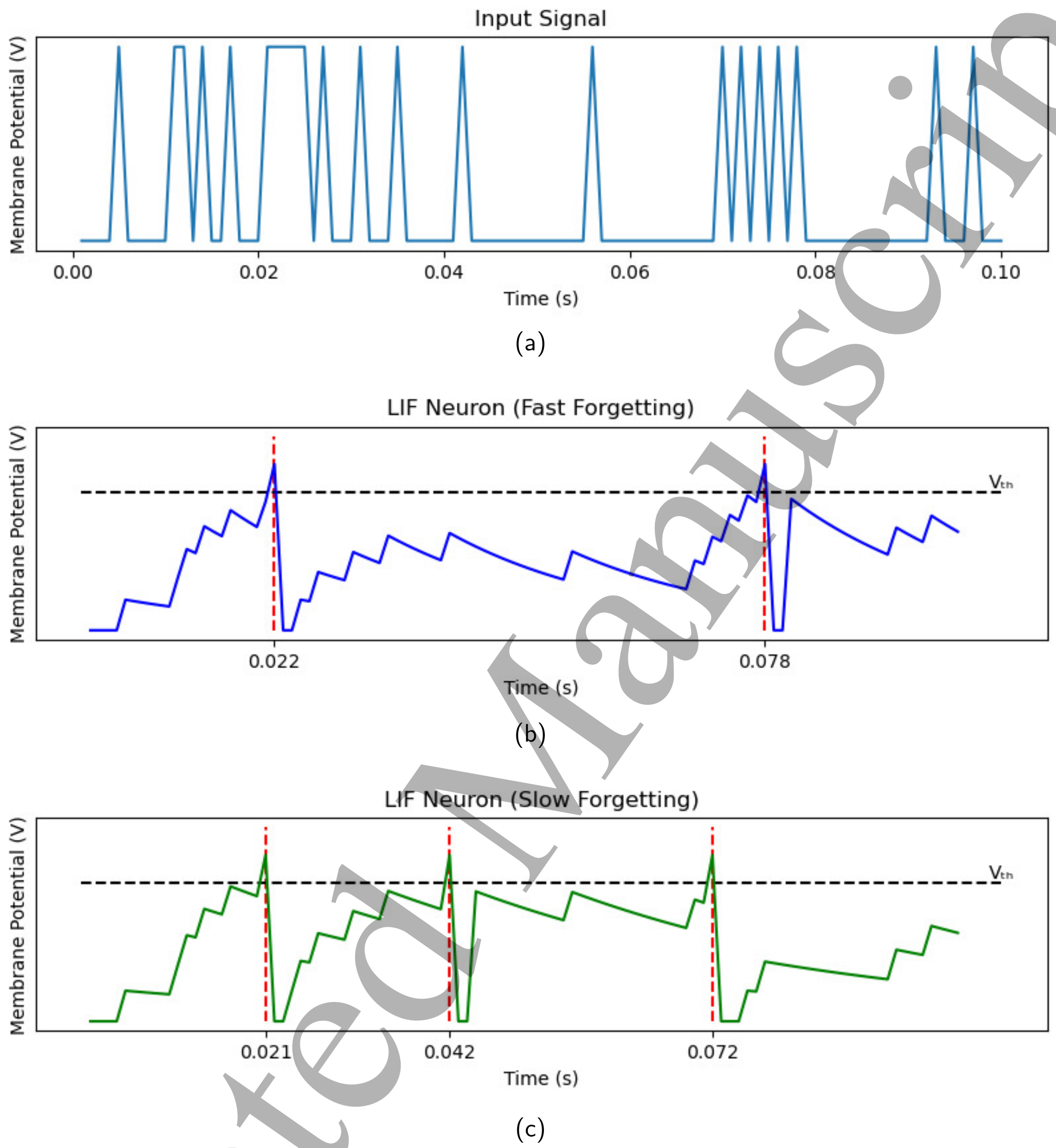


Figure 7: Example of Changing Hyper-Parameters in a LIF Neuron. The "Fast forgetting" neuron (smaller time constant τ_m) (Figure 7b) can only spike twice in response to the input (Figure 7a). The "Slow forgetting" one (Figure 7c) can fire three times and maintains a higher membrane potential throughout. Note that the "Slow forgetting" neuron also fires earlier, i.e. it requires less (close) spikes to reach the threshold. Both the neurons have a refractory period of 2 ms.

From a theoretical point of view, using different neurons or varying the parameters means to open to different non-linear dynamics and excitability patterns. Figure 6 and Figure 7 provide a visual understanding of these differences. In the LIF, the membrane potential updates depend linearly on the previous state of the membrane potential itself. The EIF manifests a similar relationship up to certain values of membrane potential (Θ_{rh}), after which the relationship assumes a more non-linear (exponential) aspect. The QIF loses any linear relationship in favor of a quadratic one. These dynamics play a role on the excitability (regions) of a neuron [2]. For instance, an EIF with a smooth exponential term (blue line in Figure 6b) will receive more mitigated updates throughout (slow-forgetting neuron), whereas an EIF with a sharp exponential term (orange line) will receive more negative updates up to the cutoff threshold Θ_{rh} and then highly positive ($+\infty$) ones, thus immediately reaching the firing threshold V_{th}. Hence, the second example would be a fast-forgetting neuron, but the cut-off threshold will act as an early firing threshold, as any subsequent update would bring the membrane potential up and above the actual firing threshold. A similar example is reported in Figure 7, where a change in the time-constant τ_m makes a LIF neuron forget faster or slower. This in turn has effects on the excitability of the neuron and its firing pattern.

The different firing abilities discussed above need to be considered within the context where the neurons are placed. If we consider the case of a homogeneous SCNN that is trained on a dataset in which the temporal distribution of events is similar for every sample, it might be pointless to consider having a wide range of excitability patters as more complex neurons have. Indeed, the increased amount of parameters would make it more difficult to find the right excitability that works well with that data. At the same time, they would likely come at a higher computational and power cost. Conversely, if the dataset is considerably diverse in terms of temporal distribution of events, it would arguably prove useful to have a broad range of excitability patterns to choose from. Therefore, an heterogeneous network of spiking neurons would likely be able to learn better or simply more features. In this context, employing more complex neurons with variable parameters can be significantly beneficial.

5.2. Temporal Features and Neuron Performance

In our experiments, we used a extremely simple homogeneous SCNN to perform a simple classification task on a simple subset of the N-MNIST and DVS Gesture datasets. The N-MNIST dataset, although natively event-based, is not a naturally dynamic dataset. The original data, the MNIST handwritten digits, are static images that do not contain temporal dynamical features. As such, the temporal features that are instead present in the N-MNIST are crafted. Furthermore, these dynamics are obtained by moving a DVS camera using the same sequence of movements with the same timing. Figure 2c shows that, as a result, the distribution of events throughout each sample present in the dataset is roughly the same. This means that the temporal features are not different from one another and are hence not discriminative of different classes of samples. Indeed, as discussed in the previous paragraph, using a homogeneous SCNN was enough to achieve reasonably high accuracy levels, despite the lack of diversity in the dynamics of the embedded neurons.

Concerning the performance in such a homogeneous settings, in our consideration of three single-variable neuron models we found that all of them have the ability to perform well. The difference however is in the cost of using one neuron rather than the other. From our experiments, we found that when hand-tuning parameters, the LIF neuron achieved averagely high accuracy levels, with EIF neuron being better at times. Using a set of optimized hyper-parameters, we observed a considerable improvement in the overall classification accuracy with the QIF achieving a 98.5% accuracy in the best case, thus surpassing its counterparts. The same QIF model performed rather poorly when using non-optimized parameters. As mentioned in Section 4.2, this highlights the fact that, despite the data displaying simple spatio-temporal features, more complex neurons are still able to perform well. The cost for achieving such results can, however, become rather high.

When employing Gestures data, the situations is slightly different. In this case, as depicted in Figure 3, there is no recurring distribution of events across different classes. Instead, events are distributed throughout the whole time domain. Each class of gestures has a distribution of events that varies with respect to the others, even more because of the fact that different actions require a different time to be executed. Thus, the temporal features in this dataset are more important and diverse. As a matter of fact, this is also shown by the results obtained using the same network as in the previous case. Here, although the performance gain is still modest, the EIF and QIF neurons steadily attain better classification accuracies than the LIF model. Since we used the same setting for all the experiments, this is likely traceable to the aforementioned differences in temporal features, which are now more diverse and complex than those in the N-MNIST dataset.

5.3. Temporal Features and Depth of the Network

The matter of temporal variety in the features being better represented by more complex dynamics opens up further questions as to their use in SNNs. Indeed, if we consider a hierarchical NN, the deepest layers normally learn more abstract representations, whereas the early layers typically learn to distinguish simple patterns, such as edges or corners [44]. This is easily conceivable when thinking about spatial features. For example, when a set of lines is recognized in the early layers of a CNN, these could later be understood to be a square, and further down the network as a house. Although it can be more difficult to imagine, when we include the time dimension, similar scenarios can arise where features relate temporally other than spatially. It thus seems straightforward that the temporal relationships might vary in complexity in different stages of the network depending on the task at hand.
We showed that the use of more complex neuron models improves the performance on more complex tasks at the level of one layer. When combining several of these layers in a hierarchical network more uses of their non-linear dynamics could arise, as they

would combine several spatio-temporal features built up in previous stages to understand compound featural patterns.

6. Conclusion

In this work, we have considered a simple unsupervised SCNN and analyzed the effect on the overall performance of changing the underlying neuron models. Because they were not previously present in the SpykeTorch framework, we implement the spiking neurons and make the code available. We firstly draw a set of 4 binary classification tasks using 4 couples of classes from the N-MNIST dataset on which we repeatedly train and evaluate our SCNN. Experimental results on these tasks show that all three neuron models can achieve top-level accuracies, albeit the more complex ones require more fine-tuning. In a second instance, we consider the DVS Gestures dataset, which exposes a richer set of features from both the visual and temporal points of view. In this case, the EIF and QIF steadily outperform the LIF on all 4 tasks drawn from this dataset. Further to this, we analyzed the sensitivity of the SCNN using each neuron model to the order of presentation of the data. Our analysis shows that none of them implies a statistically relevant difference in terms of sensitivity, and that such sensitivity is probably only relatable to the use of STDP, in our experiments. Collectively, our results show that accurately selecting the neuron model employed in an NM pipeline improves its performance, and that this selection should be driven by considering the complexity of the spatio-temporal features that the layer in the network will have to understand. Furthermore, it highlights that further research aimed at unveiling the role of the dynamics of neuron models in deep hierarchical learning would be highly beneficial to close the gap between conventional DL approaches and SNNs. Other future studies could consist of analysing further relationships between the neuron models and other components of the learning pipeline, such as the neural network architecture, and the learning rule. Furthermore, it would be interesting to investigate the result of using different spiking neurons on the synaptic efficacies when using STDP as a learning rule.

Acknowledgments

This work was supported by the US Air Force Office of Scientific Research under Grant for project FA8655-20-1- 7037. The contents were approved for public release under case AFRL-2022-2603 and they represent the views of only the authors and does not represent any views or positions of the Air Force Research Laboratory, US Department of Defense, or US Government. The authors declare that there is no conflict of interest.

References

[1] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A survey of neuromorphic computing and neural networks in hardware.

[2] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski. *Neuronal Dynamics*. Cambridge University Press, 2009.

[3] Johannes Schemmel, Sebastian Billaudelle, Phillip Dauer, and Johannes Weis. Accelerated analog neuromorphic computing. *ArXiv*, abs/2003.11996, 2020.

[4] Ben Varkey Benjamin, Peiran Gao, Emmett McQuinn, Swadesh Choudhary, Anand R. Chandrasekaran, Jean-Marie Bussat, Rodrigo Alvarez-Icaza, John V. Arthur, Paul A. Merolla, and Kwabena Boahen. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716, 2014.

[5] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[6] Paul Merolla, John Arthur, Filipp Akopyan, Nabil Imam, Rajit Manohar, and Dharmendra S. Modha. A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm. In *2011 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, 2011.

[7] Andrew S. Cassidy, Paul Merolla, John V. Arthur, Steve K. Esser, Bryan Jackson, Rodrigo Alvarez-Icaza, Pallab Datta, Jun Sawada, Theodore M. Wong, Vitaly Feldman, Arnon Amir, Daniel Ben-Dayan Rubin, Filipp Akopyan, Emmett McQuinn, William P. Risk, and Dharmendra S. Modha. Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–10, 2013.

[8] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, Brian Taba, Michael Beakes, Bernard Brezzo, Jente B. Kuang, Rajit Manohar, William P. Risk, Bryan Jackson, and Dharmendra S. Modha. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1537–1557, 2015.

[9] Shuangming Yang, Jiang Wang, Xinyu Hao, Huiyan Li, Xile Wei, Bin Deng, and Kenneth A. Loparo. BiCoSS: Toward large-scale cognition brain with multigranular neuromorphic architecture. *IEEE Transactions on Neural Networks and Learning Systems*, 33(7):2801–2815, jul 2022.

[10] Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[11] Garrick Orchard, E Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T Sommer, and Mike Davies. Efficient neuromorphic signal processing with loihi 2. In *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259. IEEE, 2021.

[12] Alex Vicente-Sola, Davide L. Manna, Paul Kirkland, Gaetano Di Caterina, and Trevor Bihl. Keys to accurate feature extraction using residual spiking neural networks. *arXiv*, abs/2111.05955, 2021.

[13] Syed Ahmed Aamir, Yannik Stradmann, Paul Muller, Christian Pehle, Andreas Hartel, Andreas Grubl, Johannes Schemmel, and Karlheinz Meier. An accelerated LIF neuronal network array for a large-scale mixed-signal neuromorphic architecture. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(12):4299–4312, dec 2018.

[14] Alan Diamond, Thomas Nowotny, and Michael Schmuker. Comparing neuromorphic solutions

in action: Implementing a bio-inspired solution to a benchmark classification task on three parallel-computing platforms. *Frontiers in Neuroscience*, 9, jan 2016.

[15] Ken E. Friedl, Aaron R. Voelker, Angelika Peer, and Chris Eliasmith. Human-inspired neurorobotic system for classifying surface textures by touch. *IEEE Robotics and Automation Letters*, 1(1):516–523, jan 2016.

[16] Eric Hunsberger and Chris Eliasmith. Spiking deep networks with lif neurons. October 2015.

[17] J. Göltz, L. Kriener, A. Baumbach, S. Billaudelle, O. Breitwieser, B. Cramer, D. Dold, A. F. Kungl, W. Senn, J. Schemmel, K. Meier, and M. A. Petrovici. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature Machine Intelligence*, 3(9):823–835, sep 2021.

[18] Milad Mozafari, Saeed Reza Kheradpisheh, Timothee Masquelier, Abbas Nowzari-Dalini, and Mohammad Ganjtabesh. First-spike-based visual categorization using reward-modulated STDP. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):6178–6190, dec 2018.

[19] Evangelos Stromatias, Daniel Neil, Francesco Galluppi, Michael Pfeiffer, Shih-Chii Liu, and Steve Furber. Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on SpiNNaker. In *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2015.

[20] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7):3227–3235, 2018.

[21] Soni Chaturvedi, AA Khurshid, and Meftah Boudjelal. Image segmentation using leaky integrate-and-fire model of spiking neural network. *International Journal of Wisdom Based Computing*, 2(1):21–28, 2012.

[22] Chunming Jiang, Le Yang, and Yilei Zhang. A spiking neural network with spike-timing-dependent plasticity for surface roughness analysis. *IEEE Sensors Journal*, 22(1):438–445, 2021.

[23] Shuangming Yang, Jiang Wang, Nan Zhang, Bin Deng, Yanwei Pang, and Mostafa Rahimi Azghadi. CerebelluMorphic: Large-scale neuromorphic model and architecture for supervised motor learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9):4398–4412, sep 2022.

[24] Alberto Patino-Saucedo, Horacio Rostro-González, Teresa Serrano-Gotarredona, and Bernabé Linares-Barranco. Event-driven implementation of deep spiking convolutional neural networks for supervised classification using the spinnaker neuromorphic platform. *Neural networks : the official journal of the International Neural Network Society*, 121:319–328, 2020.

[25] Tanguy Fardet, Mathieu Ballandras, Samuel Bottani, Stéphane Métens, and Pascal Monceau. Understanding the generation of network bursts by adaptive oscillatory neurons. *Frontiers in Neuroscience*, 12, feb 2018.

[26] Katayoon Taherkhani and Mahdi Aliyari Shoorehdeli. An artificial neural network based on izhikevich neuron model. In *2017 Iranian Conference on Electrical Engineering (ICEE)*. IEEE, may 2017.

[27] Soni Chaturvedi, Rutika N. Titre, and Neha Sondhiya. Review of handwritten pattern recognition of digits and special characters using feed forward neural network and izhikevich neural model. In *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*. IEEE, jan 2014.

[28] Roberto A. Vazquez. Training spiking neural models using cuckoo search algorithm. In *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, jun 2011.

[29] Henry Markram, Maria Toledo-Rodriguez, Yun Wang, Anirudh Gupta, Gilad Silberberg, and Caizhi Wu. Interneurons of the neocortical inhibitory system. *Nature Reviews Neuroscience*, 5(10):793–807, oct 2004.

[30] Barry W. Connors and Michael J. Gutnick. Intrinsic firing patterns of diverse neocortical neurons. *Trends in Neurosciences*, 13(3):99–104, mar 1990.

[31] Stephanie Dauth, Ben M. Maoz, Sean P. Sheehy, Matthew A. Hemphill, Tara Murty, Mary Kate Macedonia, Angie M. Greer, Bogdan Budnik, and Kevin Kit Parker. Neurons derived from different brain regions are inherently different in vitro: a novel multiregional brain-on-a-chip.

Journal of Neurophysiology, 117(3):1320–1341, 2017. PMID: 28031399.

[32] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, 9, nov 2015.

[33] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, Jeff Kusnitz, Michael Debole, Steve Esser, Tobi Delbruck, Myron Flickner, and Dharmendra Modha. A low power, fully event-based gesture recognition system. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7388–7397, 2017.

[34] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, and Timothée Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in Neuroscience*, 13, 2019.

[35] L. Lapicque. Recherches quantitatives sur l'excitation electrique des nerfs traitée comme une polarization. *Journal de Physiologie et Pathologie General*, 9:620–635, 1907.

[36] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, aug 1952.

[37] Wulfram Gerstner and Werner M. Kistler. *Spiking Neuron Models*. Cambridge University Press, aug 2002.

[38] Wulfram Gerstner, Raphael Ritz, and J Leo van Hemmen. A biologically motivated and analytically soluble model of collective oscillations in the cortex. *Biological cybernetics*, 68(4):363–374, 1993.

[39] Carl Van Vreeswijk and Haim Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724–1726, 1996.

[40] Mitsuo Kawato and Hiroaki Gomi. A computational model of four regions of the cerebellum based on feedback-error learning. *Biological cybernetics*, 68(2):95–103, 1992.

[41] D Zipser, B Kehoe, G Littlewort, and J Fuster. A spiking network model of short-term active memory. *Journal of Neuroscience*, 13(8):3406–3420, 1993.

[42] Wolfgang Maass. Lower bounds for the computational power of networks of spiking neurons. *Neural computation*, 8(1):1–40, 1996.

[43] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10(9):1659–1671, dec 1997.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[45] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[46] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.

[47] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[48] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv: Learning*, 2016.

[49] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *ArXiv*, abs/1710.05941, 2018.

[50] Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on mnist classification task. *arXiv preprint arXiv:1804.02763*, 2018.

[51] Steffen Eger, Paul Youssef, and Iryna Gurevych. Is it time to swish? comparing deep learning activation functions across nlp tasks. *arXiv preprint arXiv:1901.02671*, 2019.

[52] Bin Ding, Huimin Qian, and Jun Zhou. Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841, 2018.

[53] Qinghe Zheng, Mingqiang Yang, Xinyu Tian, Xiaochen Wang, and Deqiang Wang. Rethinking the role of activation functions in deep convolutional neural networks for image classification. *Engineering Letters*, 28(1), 2020.

[54] Mohit Goyal, Rajan Goyal, P. Venkatappa Reddy, and Brejesh Lall. *Activation Functions*, pages 1–30. Springer International Publishing, Cham, 2020.

[55] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136, 2020.

[56] Alexander Sboev, Roman Rybka, Alexey Serenko, Danila Vlasov, Nikolay Kudryashov, and Vyacheslav Demin. To the role of the choice of the neuron model in spiking network learning on base of spike-timing-dependent plasticity. *Procedia Computer Science*, 123:432–439, 2018.

[57] E.M. Izhikevich. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks*, 15(5):1063–1070, sep 2004.

[58] Lyle Long and Guoliang Fang. A review of biologically plausible neuron models for spiking neural networks. In *AIAA Infotech@Aerospace 2010*. American Institute of Aeronautics and Astronautics, apr 2010.

[59] Adam Barton, Eva Volna, and Martin Kotyrba. The application perspective of izhikevich spiking neural model – the initial experimental study. In *Recent Advances in Soft Computing*, pages 223–232. Springer International Publishing, aug 2018.

[60] Gautam Kumar, V. Aggarwal, N.v Thakor, Marc Schieber, and M.V. Kothare. Optimal parameter estimation of the izhikevich single neuron model using experimental inter-spike interval (isi) data. pages 3586 – 3591, 08 2010.

[61] Corinne Teeter, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Aaron Szafer, Nicholas Cain, Hongkui Zeng, Michael Hawrylycz, Christof Koch, and Stefan Mihalas. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature Communications*, 9(1), feb 2018.

[62] Renaud Jolivet, Alexander Rauch, Hans-rudolf Lüscher, and Wulfram Gerstner. Integrate-and-fire models with adaptation are good enough. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[63] Beata J. Grzyb, Eris Chinellato, Grzegorz M. Wojcik, and Wieslaw A. Kaminski. Which model to use for the liquid state machine? In *2009 International Joint Conference on Neural Networks*. IEEE, jun 2009.

[64] Roger D Traub, Robert K Wong, Richard Miles, and Hillary Michelson. A model of a ca3 hippocampal pyramidal neuron incorporating voltage-clamp data on intrinsic conductances. *Journal of neurophysiology*, 66(2):635–650, 1991.

[65] Geir Halnes, Sigita Augustinaite, Paul Heggelund, Gaute T. Einevoll, and Michele Migliore. A multi-compartment model for interneurons in the dorsal lateral geniculate nucleus. *PLoS Computational Biology*, 7(9):e1002160, sep 2011.

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

[67] Shuangming Yang, Tian Gao, Jiang Wang, Bin Deng, Mostafa Rahimi Azghadi, Tao Lei, and Bernabe Linares-Barranco. SAM: A unified self-adaptive multicompartmental spiking neuron model for learning with working memory. *Frontiers in Neuroscience*, 16, apr 2022.

[68] Shuangming Yang, Bernabe Linares-Barranco, and Badong Chen. Heterogeneous ensemble-based spike-driven few-shot online learning. *Frontiers in Neuroscience*, 16, may 2022.

[69] Eugene M. Izhikevich. *Dynamical Systems in Neuroscience*. MIT Press Ltd, January 2010.

[70] Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1(6):445–466, jul 1961.

[71] J. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, oct 1962.

[72] J. L. Hindmarsh and R. M. Rose. A model of neuronal bursting using three coupled first order differential equations. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 221(1222):87–102, mar 1984.

[73] C. Morris and H. Lecar. Voltage oscillations in the barnacle giant muscle fiber. *Biophysical Journal*, 35(1):193–213, jul 1981.

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

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

[76] Nicolas Fourcaud-Trocmé, David Hansel, Carl van Vreeswijk, and Nicolas Brunel. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *The Journal of Neuroscience*, 23(37):11628–11640, dec 2003.

[77] G. B. Ermentrout and N. Kopell. Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM Journal on Applied Mathematics*, 46(2):233–253, apr 1986.

[78] Romain Brette and Wulfram Gerstner. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637–3642, nov 2005.

[79] E.M. Izhikevich. Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, 14(6):1569–1572, nov 2003.

[80] Xiang Cheng, Yunzhe Hao, Jiaming Xu, and Bo Xu. Lisnn: Improving spiking neural networks with lateral interactions for robust object recognition. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 1519–1525. International Joint Conferences on Artificial Intelligence Organization, 7 2020.

[81] Davide L. Manna, Alex Vicente Sola, Paul Kirkland, Trevor J. Bihl, and Gaetano Di Caterina. Frameworks for snns: a review of data science-oriented sofware and an expansion of spyketorch. White Paper.

[82] Timothée Masquelier and Simon J Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS computational biology*, 3(2):e31, 2007.

[83] Paul Kirkland, Gaetano Di Caterina, John Soraghan, and George Matich. Spikeseg: Spiking segmentation via stdp saliency mapping. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[84] Paul Kirkland, Davide L. Manna, Alex Vicente-Sola, and Gaetano Di Caterina. Unsupervised spiking instance segmentation on event data using STDP. *arXiv*, abs/2111.05283, 2021.

[85] Aboozar Taherkhani, Ammar Belatreche, Yuhua Li, Georgina Cosma, Liam P. Maguire, and T.M. McGinnity. A review of learning in biologically plausible spiking neural networks. *Neural Networks*, 122:253–272, feb 2020.

[86] Peter Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in Computational Neuroscience*, 9, 2015.

[87] Laxmi R. Iyer and Arindam Basu. Unsupervised learning of event-based image recordings using spike-timing-dependent plasticity. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1840–1846, 2017.

[88] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *ICML*, 2018.