Artificial Intelligence

Radboud University Nijmegen

Neuron Model Complexity

Exploiting the richness of single neuron dynamics in spiking neural networks

Author:
Steffen Ricklin
s1009136

Supervisors:
Dr. Renato Duarte[a,b]
Dr. Matthieu Gilson[c,d]

Shared second readers:
Dr. Nasir Ahmad[a,b]
Dr. Mahyar Shahsavari[a,b]

September 25, 2023

a - Artificial Intelligence, Radboud University, The Netherlands
b - Donders Institute for Brain, Cognition and Behaviour, The Netherlands
c - Faculty of Medicine, Aix-Marseille University, France
d - Institut de Neuroscience de la Timone, Aix-Marseille University, France

Abstract

Training Spiking Neural Networks in a supervised fashion using backpropagation has been problematic to outright impossible until very recently. A practical solution to the problem of the non-differentiable partial derivative of the neurons' spikes has been found by approximating it by a smooth surrogate gradient. This method is called Surrogate Gradient Learning and has been used in most studies in Spiking Neural Networks built upon the fairly simple Leaky Integrate-and-Fire (LIF) neuron model. Such networks can be trained to reach high performances comparable with state-of-the-art Artificial Neural Networks. However, the LIF's repertoire of firing patters is limited compared to observations in biological neurons. The question arises whether a more biological Spiking Neural Network can also be trained. In this research project, we thus investigated what influence more complex neuron models, that allow for richer sub-threshold dynamics and different firing regimes, have on the performance of the models and the requirements on training these networks. We hypothesized that neuron models with sub-threshold dynamics have an advantage over the basic Leaky Integrate-and-Fire model if the data used has a temporal component. Our results showed that these models can in fact be used in the same fashion as Leaky Integrate-and-Fire models to train networks relying on Surrogate Gradient Learning. On the one hand, networks that used the Izhikevich or Adaptive Exponential Integrate-and-Fire neuron model achieved almost as good performance as the LIF (although slightly worse). On the other hand, the Izhikevich and Adaptive Exponential exhibited a better capability of generalization than the Leaky Integrate-and-Fire, especially when trained on a dataset with a temporal component.

Acknowledgements

First and foremost, I want to express my sincere gratitude to my supervisors, Matthieu and Renato, for their extended supervision and support, and who first proposed the project idea. The results presented in my thesis would be impossible without your supervision. Next, I want to thank Kai for proofreading and giving his valuable feedback on the written thesis. Finally, my heartfelt gratitude goes towards my family and my friends, especially Peter, for their never-ending support throughout this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this research project, we studied the impact of using biologically plausible neuron models in Spiking Neural Networks in combination with a recent learning method. Previous research with focus on that learning method included the use of a standard but simplistic and less biologically plausible neuron model. In the global search for more energy-efficient methods of machine learning research is again and again returning to the human brain, the most energy-efficient biological supercomputer known to us, to take inspiration from insights in cognition and learning. It was thus a logical next step to extend the research on this recent learning method by applying it to more biologically plausible neuron models.

The first section of this chapter focuses on giving the reader a brief overview of previous research, with focus on supervised learning, Spiking Neural Networks (SNNs) and the above mentioned learning method, and states the motivation behind the research that was conducted in this project (Sec. 1.1). It is then followed by an introduction to the neuron models that our research focused on (Sec. 1.2) . Finally, Sec. 1.3 gives an outline of the structure and contents of the remaining chapters.

1.1 Background and Motivation

The understanding of cognitive functions is a fundamental question in neuroscience. In particular, a long-standing line of research has investigated how neurons collectively perform operations to implement complex functions [51]. One of the most studied cognitive function at the "microscopic" level for neuronal networks (considering them individually) is classification, which aims to map input stimuli to output patterns that represent categories or classes. Input stimuli can be either static like images in the context of visual recognition or time series like sounds in the context of auditory recognition. In deep learning, specific architectures to connect neurons and training algorithms have been designed to implement efficient input-output mappings [54].

The recent success of deep learning arises from very large datasets and the widespread use of auto-differentiation [30]. However, these very powerful models involve a huge number of parameters and thus consume large amounts of energy to be trained, but also to use them as they typically require supercomputers. Although these models take inspiration from insights we gained from studying the human brain, they are nowhere near the low energy consumption and high learning efficiency of the brain [57]. A key difference here lies in the computational units that represent neurons in such networks: broadly speaking, we can discriminate between artificial neural networks (ANNs) that rely on analog neurons, like the perceptron [6] which processes continuous variables and spiking neural networks (SNNs) which rely on simulated neurons which process events (action potentials in biology) [39][20]. A claimed advantage of SNNs is that this spiking behavior approximates real-life neuronal systems more closely than ANNs [42]. Moreover, from an engineering perspective, a strong advantage is that they are much more energy-efficient than analog neurons and thus hold promises to design embedded "intelligent" systems for autonomous robot agents dealing with real-life problems [45][21], like spatial navigation in a natural environment [52][53].

1.1.1 Supervised Learning and Backpropagation

Auto-differentiation methods to train ANNs rely on the error backpropagation (BP) algorithm [47], which has been proposed decades ago to optimize connectivity weights between the neuron layers in feedforward ANNs. BP is done by efficiently applying the chain rule on partial derivatives (backward step) of every forward computational step on the network's classification error with respect to the input data. These partial derivatives are then used to adjust the connection weights according to their contribution to the error, solving the credit assignment problem [47][30]. Every forward computation is therefore required to be differentiable in order to compute the partial derivatives. Nowadays, auto-differentiation is automatized in framework like PyTorch [38] or TensorFlow [25].

Artificial neurons like the perceptron [16] are known to work reasonably well. However, they are over-simplistic models of how the brain's neurons actually communicate: in the form of spikes. Over the course of the past two to three decades many more models have been developed that approximate the spiking nature of biological neurons [39][20][42]. Amongst others, Wang *et al.* [54] give an overview of the many approaches one could take to model SNNs. Basically, it is possible to simulate a SNN in discrete time to use the powerful tools developed for ANNs. A typical layout of the computations involved to train a (recurrent) SNN is shown and explained in Fig. 1.1. Auto-differentiation is widely applicable to various types of SNNs, including recurrent; however, as a first step, we focused on strictly feedforward SNNs here.

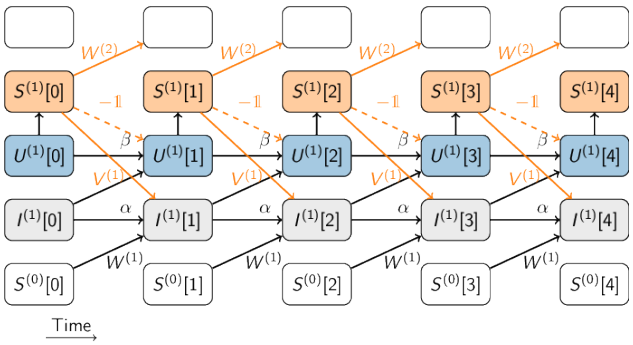

Figure 1.1: Spiking Neural Network graph. The graph shows the state variables of the network going through different time steps from left ($t = 0$) to right ($t = 4$) [60]. The binary vector $S^1[t]$ represents the spikes in layer 1 at time t, whereas the binary vector $S^1[0]$ represents the input layer of the network. The vectors I^1 and U^1 respectively describe the current and voltage in layer 1. The weight matrix $W^{(l)}$ corresponds to forward connections and modulates the contribution of each input spike of layer 0 to the current $I^{(1)}$ in the next time step. Likewise, the weight matrix $V^{(l)}$ corresponds to the recurrent connections. The parameters α and β are involved in the update of U and I from a time step to the next. The discretization of the time steps is described in more detail in the SpyTorch tutorial 1 [60] [61]. Image source: https://github.com/fzenke/spytorch.

There are various domains of research of SNNs, each having their own benefits and drawbacks [54][55][13]. The focus of this research project will be on single-layer

learning of networks with varying neuron models and network architectures by using BP. The use of spiking neuron models was a problem in the past because backpropagation through an action potential was impossible due to the non-differentiable nature of spikes. Neftci *et al.* [35] summarized several approaches to solve this problem for supervised learning; one of the presented solutions was Surrogate Gradient Learning (SGL).

1.1.2 Surrogate Gradient Learning

Spiking neurons are non-differentiable because of the discontinuity of the voltage when a spike is elicited. Non-differentiable computations corrupt the training of a network's connection weights because the necessary gradients cannot be computed. Depending on the framework and configuration of the network training process, the training either fails and stops completely or only the weights up to the non-differentiable are updated according to the flow of the gradients. SGL circumvents this problem by replacing the derivative of the spiking non-linearity by a *surrogate* smooth function [61].

So far, SGL has mainly been shown to work with SNNs of integrate-and-fire neurons, leaky integrate-and-fire (LIF) neurons to be precise [35][61]. There are few exceptions, e.g. the resonate-and-fire neuron that has been used in combination with a similar method to replace and approximate gradients at spike time [2][56]. The LIF neuron model provides a neuronal model with spiking behavior, but its simple sub-threshold dynamics and firing mechanism can only partially explain the diversity of spiking patterns biological neurons show [48][33].

In this research project, we examined how SGL works with different mathematical neuronal models that exhibit rich spiking behaviors as is observed in biology [48][33]. The focus here is thus on a systematic study of what happens when different types of neurons and varying firing regimes are used for the same task.

1.1.3 Motivation

This project was motivated by the understanding of neuronal computations in the brain, focusing on classification as one of the fundamental operations performed by neuronal networks in the brain. This line of research has been active for many decades [51][19][27].

The good cost-benefit trade-off of the LIF model has lead to its widespread use in the research of SNNs. The benefits of the model are its computational efficiency and remarkably well approximation of expected spiking behaviors which even enables the solving of more complex problems. The drawbacks are that the model is very simplistic compared to more biologically plausible models that allow for features like sub-threshold or spike adaptation. The LIF model therefore lacks the range of spiking behaviors that those complex models and the human brain show. Hence, we argue that it is important to show that neuron models that offer different spiking behaviors can be used as well as the LIF neuron in SNNs that implement SGL.

Impact and Importance

A better insight into when to use particular neuron models will enable us to build more powerful and efficient spiking neural networks, better suited to extract specific input patterns in spike trains. We aimed to identify which neuron models offer an interesting cost-benefit trade-off for a certain task like classification. We hypothesized that networks with models that display more complex behaviors result in better performances with an equal number of neurons per layer or similar performances with less neurons per layer which ultimately makes the networks more efficient.

Similar to the above point, models with a richer variety of spiking behaviors than the LIF model may allow for a sparser spiking activity to achieve similar performances due to their more complex sub-threshold dynamics. Sparser spiking activity is relevant in keeping a network energy efficient without trying to lose important information [41][29][19].

The study of SNNs in terms of computational power for classification tasks has only started in the past decade. Previous research has to the best of our knowledge mainly focused on versions of the LIF neuron model and not a comparison between it and other neuron models. The interplay between the (local) neuronal dynamics and the (global) network function is not well understood for spiking neurons and is the core of our study. This also has important implications for the design of neuromorphic hardware [58][5][52].

1.2 Choice of Neuronal Models

This section gives an overview about the dynamics of the different spiking neuron models that were chosen and implemented here. Within the scope of this project the focus was set on using voltage-based models of neurons that extend the dynamics and emerging properties of simple Integrate-and-Fire models like the LIF by a substantial amount. Over the last three decades many neuron models haven been developed [42][55]. In this project, we implemented three of those models which are described in the following paragraphs. Note that complex models such as the Hodgkin-Huxley model which is a biologically very realistic conductance-based model were out of the scope of the project because they cannot be easily realized with the SGL method.

1.2.1 LIF

The Leaky Integrate-and-Fire (LIF) neuron model is a fundamental model of an individual neuron that has been used for many decades [28][10] and is widely used in computational neuroscience due to its computational efficiency and simplicity, also in the context of SNNs with auto-differentiation [35].

The LIF model consists of a membrane potential (for the soma, or neuronal cell body) and a threshold to produce output spikes that propagate via the axon toward target neurons. In essence, the membrane potential integrates pre-synaptic (from source neurons) currents by summing them over time; when it exceeds the threshold, an output spike is fired and the membrane potential is reset to a lower value. The continuous update equation of the membrane potential and the formulation of its

reset are given in Eq. (1.1) and Eq. (1.2) respectively.

$$\tau_{mem}\frac{dV}{dt} = -(V(t) - V_{rest}) + RI(t) \quad (1.1)$$

$$V(t) \geq 1 \rightarrow V(t) = 0 \quad (1.2)$$

Here τ_{mem} is the membrane time constant, $V(t)$ is the membrane potential, V_{rest} is the resting potential, R is the input resistance and $I(t)$ is the input current [18][35].

1.2.2 Izhikevich

Compared to the LIF neuron which has linear sub-threshold integration of synaptic inputs, the Izhikevich neuron involves another variable in addition to the membrane potential, which allows for intrinsic adaptation as well as a variety of firing patterns [23]. This makes the Izhikevich neuron much more versatile than the LIF neuron to reproduce a variety of different firing patterns (Fig. 1.2) that have been observed in the biology for different types of neurons. For example, fast-spiking (FS) and low-threshold spiking (LTS) neurons are associated with inhibitory neurons, whereas regular spiking (RS), intrinsically bursting (IB) and chattering (CH) neurons are more associated with excitatory neurons [23]. The linear LIF neuron is only able to produce a firing pattern that is close to the RS firing pattern of the Izhikevich neuron.

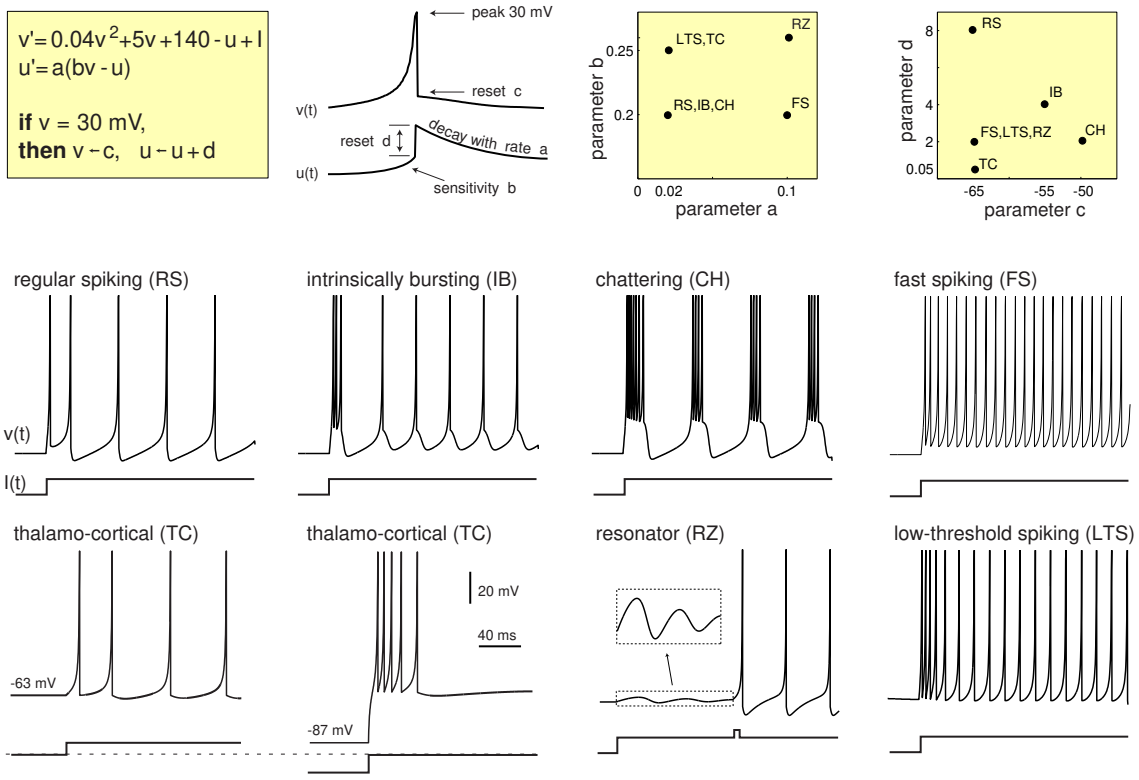

Figure 1.2: depicts the different spike patterns the Izhikevich neuron can produce. *Electronic version of the figure and reproduction permissions are freely available at www.izhikevich.com* (also see www.izhikevich.org/publications/spikes.htm).

The update equations for the membrane potential v and the membrane recovery variable u are shown in Eq. (1.3) to Eq. (1.5).

$$\dot{v} = I + v^2 - u \tag{1.3}$$
$$\dot{u} = a(bv - u) \tag{1.4}$$
$$v \geq 1 \rightarrow v = c,\ u = u + d \tag{1.5}$$

Here, I is the input current and u is the membrane recovery variable which accounts for activations of K^+ ionic currents and inactivation of Na^+ ionic currents. The constants a to d define certain properties which result in different spike pattern: a is the timescale of u, i.e. the smaller a gets the slower u's recovery, whereas b represents the sensitivity to sub-threshold fluctuations of v. c and d are the reset potentials of v and u respectively.

1.2.3 AdEx

The second and final model that we used to explore and extend the use of SGL is the Adaptive Exponential Integrate-And-Fire (AdEX) model [9] which has been used in a lot of studies [14][40][4] because of it more realistic biological dynamics and extends the features of the LIF and Izhikevich models [34][17].

1.3 Outlook

This thesis project aimed for a proof of concept that addresses the following questions:

- can biologically plausible homogeneous networks be trained using SGL?
- can temporal relationships in data be better classified by such/ networks?
- can variability in firing regimes impede or boost network learning and performance?

The structure of the remainder of this thesis will be outlined here. Chapter 2 follows the current chapter and explains the methods used to conduct and analyze the experiments of this project as well as the implementation of the SNNs and the discretized form of the three neuron models mentioned above. Next, chapter 3 covers the experiments (including datasets) and their results. For each dataset three experiments were conducted: in the first experiment a hyperparameter search for each neuron model aimed to optimize performance and determine influential factors for training (Sec. 3.1.1, 3.2.1). Experiment two investigated whether network training resulted in changed connection weights that move a neuron model out of its intended firing regime (Sec. 3.1.2, 3.1.3). And last but not least, in the third experiment the suite of used firing regimes was extended by additional (non-regular) firing regimes. With that, we aimed to study the regimes' influence on SNN performance (Sec. 3.2.2, 3.2.3). Chapter 4 closes this thesis off with a discussion on and a conclusion about the reported results as well as a perspective on suggested future steps.

Chapter 2

Methods

We explored the performance of the different neuron models in a variety of datasets consisting of spike trains fed to the network. The presented datasets in this thesis are spike trains generated from static images (handwritten digits) from the MNIST dataset [31] and the Spoken Heidelberg digits (SHD) dataset [11], as well as the synthetic Random Manifold (RandMan) dataset in Appendix Sec. A.2.

In this chapter, we present the methods used to conduct the computational experiments for the MNIST (Sec. 3.1) and SHD (Sec. 3.2) datasets. From more broad to more specific, these methods are comprised of the architecture of the SNNs (Sec. 2.1), the hardware setup on which the experiments ran (Sec. 2.2), the implementation of SGL (Sec. 2.3), followed by the descriptions of the three neuron models in their discrete form (Sec. 2.4) and the metrics used to analyze the training quality of a SNN and its neuron population statistics (Sec. 2.5). Furthermore, we describe the input encoding for the MNIST dataset (Sec. 2.6).

2.1 Network Architecture

Implementation and training of the network was done using PyTorch [38]. The Adam optimizer [26] was used for the optimization step after computing the gradients.

All implemented and trained SNNs consisted of an input layer, a single hidden layer (HL) and an readout (or output) layer. A single hidden layer was sufficient for the fundamental type of research of this project. The number of nodes in the input layer, i.e. its size, was dependent on the number of features of the dataset (see start of section 3.1 and Tab. 3.4).

The input 'neurons' were only transmitting the input spikes to the HL and the output layer, whereas the HL and output neurons simulated actual neuron models.

2.1.1 Spiking Hidden-Layer and Non-Spiking Readout-Layer

The size of the HL influences the capabilities of a neural network drastically: too small and the network can not learn its task and underfits; too large and the network is prone to overfit on the training data. Thus, it was sensible to optimize the HL size in a hyperparameter tuning experiment (Sec. 2.3.2). For all successive experiments the HL size was set to 800 neurons. The HL was comprised of spiking neurons of the models that are compared in this thesis. In other words, either LIF, Izhikevich or AdEx neurons were used as neurons in the HL. Every network only had homogeneous HLs. This means the layer contained only one type of neuron, with every neuron configured to operate in the same firing regime. Mixed-regime layers were out of scope for this project.

The output layer size was determined by the number of classes the network was supposed to classify. For all datasets the data of 10 classes was selected which means that the output layer had 10 output neurons. The output layer is used for making the classification prediction to which class the given stimuli in the input layer belonged to. In this layer, all neurons were Leaky Integrate neurons without spiking behavior. The predicted class will be the neuron with the highest activity over the full simulation time [60][61].

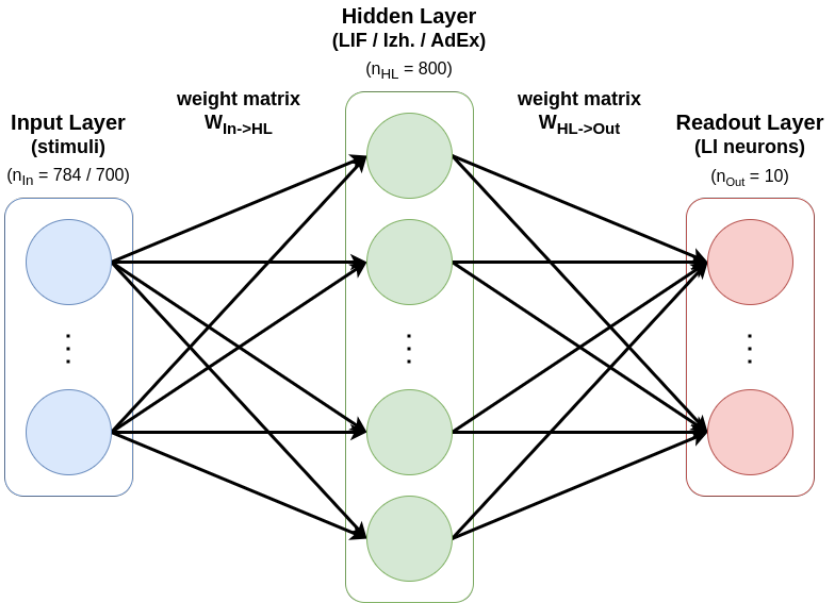

Figure 2.1: Schematic overview of the SNN architecture that we used. The input layer has $n_{In} = 784$ and $n_{In=700}$ spike trains for the MNIST and SHD datasets respectively. The final networks consisted of $n_{HL} = 800$ neurons in the hidden layer, for each neuron type. The Readout layer size was $n_{Out} = 10$ neurons for the 10 classes.

Neurons in the network had full (all-to-all) connectivity. With one HL, each networks had two matrices of connection weights, $W_{In \to HL}$ and $W_{HL \to Out}$. Training aimed to optimize connection weights in both weight matrices.

The full network architecture is depicted in Fig. 2.1 as schematic overview, including its layer sizes.

2.1.2 Weight Initialization

Initialization of the weights was done using traditional Kaiming He initialization [22][60] which was sufficient for the single layer implemented SNNs. An alternative would have been to use Fluctuation Driven Initialization which was developed to enable more data-specific initialization of LIF SNNs that use SGL and resolve the problem of vanishing surrogate gradients in larger networks [46].

The weights $\mathbf{W}$ are sampled from a Gaussian distribution as shown in Eq. (2.1)

$$\mathbf{W} \sim \mathcal{N}(\mu, \sigma), \quad \text{with } \mu = 0, \sigma = \frac{\Delta_w}{{s_{layer}}^2} \tag{2.1}$$

where Δ_w represents the initial weight scale and s_{layer} the size of the pre-synaptic or incoming layer for the respective weight matrix, i.e. the size of the input layer for $W_{In \to HL}$ and the size of the HL for $W_{HL \to Out}$.

For enabling the network to learn at least a few neurons must elicit spikes. A 'silent' SNN does not learn which could be compared to ANNs that is initialized with only equal connection weights. Conversely, a network of too many neurons with extremely high spiking activity will not be able to learn. It is thus necessary

to select a suitable scaling of the initial weights using Δ_w. Because different neuron models are sensitive to input spikes in varying degrees as well as to different kind of data, Δ_w was trained as a hyperparameter in the hyperparameter optimization experiments.

2.2 Hardware

All networks that are discussed in this report were each trained on a GPU cluster with 10 *Quadro RTX 6000* GPUs. Each of these GPUs had a memory of 22.5 GiB. Every network only required a single GPU each so that some simulation were done in parallel.

Training times per SNN (50 epochs, 800 HL neurons) were shortest for the SHD dataset with on average 6 *min* of training for each neuron model. SNNs trained on RandMan data took around 29 *min* for LIF, 30 *min* for Izhikevich and 25 *min* for AdEx SNNs. Training of SNNs on the MNIST data (largest dataset) took the longest with on average 1:55 *h* for LIF, 2:05 *h* for Izhikevich and 2:16 *h* for AdEx SNNs.

Training of the network was done using batches. The number of samples in a batch, the batch size, affects concepts such as overfitting of the SNN on the train data and generalization to new input samples. Therefore, the batch size was included in the hyperparameter optimization experiments (Sec. 3.1.1 and 3.2.1) as well.

2.3 Surrogate Gradient Learning

Eq. (2.2) shows the surrogate gradient function *SuperSpike* for the gradients of spikes which was adapted from Zenke *et al.* [60][59][35].

$$\frac{\partial f}{\partial x} = \frac{\partial g}{\partial f} \frac{1}{(\Delta_{SG} \cdot |x| + 1)^2} \tag{2.2}$$

Here, the gradient $\frac{\partial g}{\partial f}$ of the previous computation (first term) is multiplied (chain rule) with the gradient of the spike (second term). Variable x is the input from the forward pass which was computed as $V[t] - V_{thresh}$ and is thus positive for $V[t] > V_{thresh}$ when spikes occur. Variable Δ_{SG} determines the scale of the surrogate gradient (SG scale): the larger Δ_{SG} the smaller the gradient.

The surrogate gradient computation was designed for the LIF neuron which is normalized to operate between $0 \leq V[t] \leq 1$, for which Zenke *et al.* [61] determined $\Delta_{SG} = 100$ to work well. The other neuron models were not normalized to this range. Consequently, it was reasonable to assume that the Izhikevich and AdEx required smaller SG scales. In small feedforward SNNs the SG scale is not as influential as in SNNs with recurrent networks [61]. However, this was only tested for LIF SNNs. Therefore, we included the SG scale as another hyperparameter for the hyperparameter optimization experiments to find suitable SG scales for the Izhikevich and AdEx models.

2.3.1 Loss function

SGL is shown to work well for different loss functions as well as input paradigms and data sets [61]. Here, we used the *LogSoftmax* and the negative log-likelihood (*NLLLoss*) methods from PyTorch [38] as a loss function to compute the cross entropy classification loss which was also used in SpyTorch [60][61].

Biological neurons are shown to exhibit relatively sparse spiking [36]. To mirror this behavior, we added a loss term for activity regularization [61] of neurons to the total loss computation. It has been shown that adding the activity regularization to the loss function can achieve sparse activation while maintaining similar functional performance of the model [61]. The implemented loss is the L2 loss, further called L2 penalty, as implemented in Zenke *et al.* [60] which enforces SNNs to use lower average number of spikes per neuron. The influence of the L2 penalty could vary per neuron model. Therefore, the L2 penalty was another hyperparameter to explore.

2.3.2 Hyperparameter Tuning

We conducted a hyperparameter optimization experiment for each dataset and neuron model due to the many hyperparameters mentioned above that may impact the performance of SGL (Sec. 3.1.1 and 3.2.1). The hyperparameter optimization was implemented using the framework Optuna [1]. Optuna is a black-box optimizer and is explained in further detail in section 3.1.1. The search was configured for the hyperparameters HL size, batch size, learning rate, L2 penalty, initial weight scale and SG scale because of previously mentioned reason.

The main goal of the hyperparameter optimization was to find out whether different neuron models can be trained with similar hyperparameters.

2.4 Discretized Neuron Models

Simulations of any real-world continuous systems must be done in discrete time steps on regular (digital) computing systems. Consequently, the simulation of the described neuron models in section 1.2 must be done in discrete time. Here follow descriptions of the discrete form of the LIF, Izhikevich and AdEx model. Following the approach developed for the LIF model [35][61], we derived similar equations for the other neuron models to be trained with auto-differentiation (e.g. SGL) in networks for classification.

Generally, for all models it holds that the update of the synaptic input current I_{syn} is formulated by Eq. (2.3)

$$I_{syn}[t+1] = I_{syn}[t] \cdot e^{-\frac{dt}{\tau_{syn}}} + I_{in}[t] \tag{2.3}$$

which computes the next time step $(t+1)$ for the I_{syn}. The exponential term determines the decay of the membrane current. Here, the time constant for the synaptic exponential decay is $\tau_{syn} = 5 \cdot 10^{-3}$. For all neuron models it further held that the simulation time resolution was set to $dt = 10^{-3}$. The summed input current $I_{in}[t]$ of weighted pre-synaptic currents connects pre-synaptic neurons to the post-synaptic neuron.

2.4.1 LIF

Here, the LIF model relies on a discrete-time implementation (Sec. 1.2.1), which has been proposed for simulation using modern auto-differentiation frameworks like PyTorch or Keras [38], [61][25].

Equation (2.4) describes the neuron's membrane voltage update ($V[t+1]$).

$$V[t+1] \quad = \quad e^{-\frac{dt}{\tau_{mem}}} \cdot V[t] + I_{syn}[t] \tag{2.4}$$

Here, the exponential term describes the exponential decay of the membrane voltage over time, with a time resolution of $dt = 10^{-3}$ seconds and the membrane time constant $\tau_{mem} = 10^{-2}$. These values are adapted from Zenke *et al.* [60]. The time resolution dt was the same for all neuron models.

If the membrane voltage reaches a threshold of $V_{thresh} = 1\ mV$ then V is reset to 0, as described by Eq. (2.5).

$$V[t] \quad = \quad \begin{cases} V[t] & \text{if } V[t] < V_{thresh} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

For the LIF it holds that, without any input, the resting potential V_{rest} of the neuron is equal to the after-spike reset potential V_{thresh}.

2.4.2 Izhikevich

Compared to the LIF model, the Izhikevich neuron involves two dynamic variables, as can be seen in Eq. (1.3) and Eq. (1.4), so there are two update equations.

The voltage update of the next time step $V[t+1]$ is described by Eq. (2.6) and the adaptation update $U[t+1]$ by Eq. (2.7).

$$V[t+1] \quad = \quad V[t] + dt\ (0.04 \cdot V[t]^2 + 5 \cdot V[t] + 140 - U[t] + I_{syn}[t]) \tag{2.6}$$
$$U[t+1] \quad = \quad U[t] + dt\ (a\ (b \cdot V[t] - U[t])) \tag{2.7}$$

The coefficients of the terms were obtained from the specific Izhikevich model shown prior in Fig. 1.2. Together with matching values for the parameters a to d (Tab. 2.1) specific behaviors of the neuron model can be triggered to elicit firing regimes similar to that of cortical neurons [23]. In Eq. (2.7) the parameters a and b represent the adaptation time-scale and sensitivity respectively.

If the membrane potential reaches the threshold $V_{thresh} = 30mV$ (for all firing regimes), then $V[t+1]$ and $U[t+1]$ are reset as shown in Eq. (2.8) and Eq. (2.9) respectively.

$$V[t] \quad = \quad c \qquad \text{if } V[t] \geq V_{tresh} \tag{2.8}$$
$$U[t] \quad = \quad U[t] + d \quad \text{if } V[t] \geq V_{tresh} \tag{2.9}$$

Parameters c and d thus represent the after-spike reset value for the membrane potential and the increase in after-spike adaptation respectively.

For the purposes of this project, we used the four firing regimes RS (regular spiking), FS (fast spiking), IB (intrinsically bursting) and CH (chattering). These four

	parameter			
regime	a	b	c	d
RS	0.02	0.2	-65	8
FS	0.10	0.2	-65	2
IB	0.02	0.2	-55	4
CH	0.02	0.2	-50	2

Table 2.1: Parameter values for the four implemented Izhikevich firing regimes, with a being the time-scale of U in ms, b the sensitivity of U, c the reset potential of V in mV and d the after-spike increase of U.

seemed sufficiently different in terms of spiking behaviour to compare their possible influences on training SNNs. Applying the parameter values given in Izhikevich [23] to Eq. (2.6) until Eq. (2.9), the implemented Izhikevich model was able to reproduce the chosen firing regimes as shown in Fig. 1.2. The parameter values are also reported in Tab. 2.1. From the table it is clear that all regimes use the adaptation variable U because neither the parameters a, b nor d are zero. Here, we want to highlight that this means that all regimes have voltage- and spike-dependent adaptation, even the RS regime which was later compared to the non-adapting LIF model.

2.4.3 AdEx

Similar to the Izhikevich model, the AdEx has two update equation due to two dynamic variables [9][34]. These are the membrane potential V and the adaptation variable W.

The voltage update of the next time step $V[t+1]$ is described by Eq. (2.10) and the adaptation update $W[t+1]$ by Eq. (2.11).

$$V[t+1] = V[t] + \frac{dt}{\tau_{mem}} \Big(V_{rest} - V[t] + \Delta_T \cdot e^{\frac{V[t]-\theta_{rh}}{\Delta_T}} - W[t] + I_{syn}[t]\Big) \quad (2.10)$$

$$W[t+1] = W[t] + \frac{dt}{\tau_{adap}} \left(a \left(V[t] - V_{rest}\right) - W[t]\right) \quad (2.11)$$

Here $\Delta_T = 2.0$ defines the sharpness of the exponential decay for the exponential term. Respectively, τ_{mem} and τ_{adap} are the time constants of the membrane potential and the adaptation variable. The rheobase threshold at which the AdEx starts the exponential process of initiating the spike is given by $\theta_{rh} = -50\ mV$ for all AdEx regimes. Also, the potential at resting state $V_{rest} = -70\ mV$ for all AdEx regimes. Parameter values that differed per firing regime are reported in Tab. 2.2 and were taken from table 6.1 of Gerstner *et al.* [18].

The implemented reset at spike time for the membrane potential $V[t]$ is given in Eq. (2.12) and the reset for the adaptation variable $U[t]$ is given in Eq. (2.13).

$$V[t] = V_{reset} \quad \text{if } V[t] \geq V_{tresh} \quad (2.12)$$

$$W[t] = W[t] + b \quad \text{if } V[t] \geq V_{tresh} \quad (2.13)$$

The values of the after-spike reset V_{reset} and adaptation increment b depend on the firing regime.

	parameter				
regime	a	b	τ_{mem}	τ_{adap}	V_{reset}
TO	0.0	60	20.0	30	-55
AD	0.0	5	20.0	100	-55
BU	-0.5	7	5.0	100	-46
IB	0.5	7	5.0	100	-51
IR	-0.5	7	9.9	100	-46

Table 2.2: AdEx firing regime parameter settings, with a being the adaptation voltage coupling in nS, b the spike triggered adaptation increment in pA, τ_{mem} the membrane time constant in ms, τ_{adap} the adaptation time constant in ms and V_{reset} the reset potential in mV. Source of the chosen parameter values is table 6.1 in Gerstner *et al.* [18].

The AdEx experiments used the regimes TO (tonic), AD (adapting), BU (bursting), IB (intrinsically bursting) and IR (irregular) (Tab. 2.2). Similar to the RS-Izhikevich regime, the TO-AdEx regime has spike-triggered adaptation enabled which is not given for the LIF model. This is an important remark because the LIF is compared in the experiments especially to the RS-Izhikevich and TO-AdEx regimes.

From now on, for the sake of brevity, whenever the default LIF firing behavior, the RS-Izhikevich regime and the TO-AdEX regime are addressed together, we will refer to all of their firing behavior as regular spiking (even though their neural dynamics differ).

2.5 Spiking Metrics

Sections 3.1.2 and 3.2.2 explore the quantitative metrics of neuronal firing behaviors inside the implemented SNNs. Further motivation behind choosing the metrics, that are explained here in this section, can be found in section 3.1.2. Here, we want to give an overview over the used metrics and how they were computed.

Metrics of importance were the average spike count, firing rate, inter-spike interval (ISI) and coefficient of variation of ISIs (CV ISI) for each neuron in the hidden layer.

The mean spike counts were computed from a spike matrix that was filled during simulation time (within a single epoch). More specifically, as shown in Eq. (2.14),

$$\mu_S = \frac{1}{B}\sum^{B}\sum^{T}\mathbf{S} = \frac{|\mathbf{S}|}{B} \tag{2.14}$$

the mean spike count μ_S of the spike matrix $\mathbf{S}$ was obtained by first taking the sum over the simulation time window T, followed by taking the mean over all samples in a batch of batch size B. The number of spikes are defined as $|\mathbf{S}|$.

For the remaining three metrics, in terms of code, that same spike matrix was used to create *SpikeList* objects (list of *SpikeTrain* objects) for each HL neuron using parts from the *Functional Neural Architecture* (FNA) [12] repository. Among many more methods, the spike lists featured methods to compute the relevant metrics.

The firing rate of a neuron n is defined as the number of spikes $|S_n|$ within time window T divided by T, as shown in Eq. (2.15) [18].

$$r_n = \frac{|S_n|}{T} \tag{2.15}$$

Inter-spike intervals are defined as exactly that, the time that passed between one spike and the next [37]. For each batch sample, the spike lists needed to be merged into a single spike list. During the merging process, the time windows of the spike lists were sliced to be between the spike time of the spike train with the earliest spike and the spike train with the latest spike. This procedure was intended to reduce larger ISIs caused by spike-intervals between the last spike of one sample and the first spike of the next sample.

The CV ISI is a measure of how regular or irregular spikes in a spike train are elicited [18][44]. It is defined as the ratio of the standard deviation $\sigma_{\mathbf{s}}$ and the mean $\mu_{\mathbf{s}}$ of the ISIs $\mathbf{s}$ of a spike train (Eq. (2.16)) [18].

$$C_V = \frac{\sigma_{isi}}{\mu_{isi}} \tag{2.16}$$

A CVs > 1 would describe more irregular firing than a spike train whose spikes were generated by a Poisson process using the same firing rate [18].

2.6 Input Spiking Activity and Datasets

This section focuses on the type of inputs that SNNs require. Here, the problem that the input format of one of the datasets poses is addressed. As mentioned at the start of this chapter, the datasets itself are described in more detail in sections 3.1 (MNIST) and section 3.2 (SHD).

HL neurons of the SNNs, as implemented here, take weighted spike trains as input signals. Therefore, the representation of the input data (samples) in the network's input layer must be in the form of spike trains for each neuron. For the SHD data, for which the samples consist of spike trains, nothing needs to be adapted. However, the MNIST data samples are gray-scale image arrays. That means that each samples is a two-dimensional array with gray-scale values from 0 to 255 for each pixel. Since this is the wrong input representation for the SNNs, the MNIST data must be transformed into spike trains for each input neuron. The relation here is pixel to neuron and gray-scale to spike train. The following subsection describes the encoding scheme that was used to achieve this transformation.

2.6.1 Input Encoding of MNIST data

The MNIST data was encoded using a spike-latency encoding scheme (or ISI encoding [3]) [8][7]. This was achieved by first normalizing the gray-scale values to be in the range $[0, 1]$ in which the LIF's membrane potential fluctuates. Next, the time it would take for a current-based LIF neuron to spike was computed if the normalized gray-scale values were to represent current values and reach a certain threshold value (0.2 in this case). The time-to-first-spike values were then used to create a spike

train for each neuron. These spike trains contained only a single spike per neuron if the threshold was reached and no spikes otherwise. This procedure was adapted from SpyTorch [60] and is used in Zenke *et al.* [61].

Chapter 3

Experiments

As stated in Sec. 1.1, the main research goal of this thesis was to test whether the AdEx and Izhikevich neuron models achieve similar classification performances as compared to the LIF model. On the one hand, we hypothesized that, thanks to their higher complexity, the AdEx and Izhikevich models may generate more complex input-output mappings, hence solving the problems of predicting the classes more effectively. On the other hand, this complexity may impair the training of the parameters. To test this hypothesis, we studied the different neuron models' performances in terms of classification accuracy and neuron firing regimes quantified by usual statistics (Sec. 2.5).

During this research project, we trained the models on three different datasets and compared their results. However, only two of those are discussed in the main part of this report. The results of training the models on the well-known MNIST dataset (Sec. 3.1) and the biologically more relevant Spoken Heidelberg Digits (SHD) [11] (Sec. 3.2) dataset are reported here. The results of networks trained on the third dataset, generated data of smooth Random Manifolds [61], were moved to Appendix Sec. A.2 because the dataset as configured turned out to be too difficult to successfully train the models which did not generate any further meaningful insights.

For each dataset, we ran three computational experiments. The first experiment per dataset aimed at confirming the general ability that Izhikevich and AdEx SNNs can be trained using SGL. In particular, the training involved several hyperparameters that are known to affect the resulting classification performance, so these experiments studied their influence (as well as compare suitable parameter ranges across neuron models). To do this, we relied on Optuna which is a black-box optimizer suitable to identify hyperparameter configurations, in a more efficient manner than a simple grid search [1]. These experiments are reported in Sec. 3.1.1 and Sec. 3.2.1. For the subsequent experiments on a given dataset, the best set of hyperparameters was then used.

The second experiment (Sec. 3.1.2 and 3.2.2) investigated the change of the neurons' firing behaviors from before and after the training of the network and how training influenced the network as a whole. As explained before in Sec. 2.4, both the AdEx and the Izhikevich model are configurable to reproduce specific firing regimes [23][34]. We want to highlight that in the first and second experiment the AdEx and the Izhikevich model were limited to the regular firing regime only. To explore the other regimes in that detail was left for future work and was beyond the scope of the present project.

However, the influence of the firing regimes on the resulting classification performance was an important question that is addressed in the third and final experiment (Sec. 3.1.3 and 3.2.3). There, we compared the influence of the (initial) firing regime on the training results, taking into account the hidden layer of our SNN. Note that this experiment has been conducted with the same hyperparameter configuration as the regular firing regime, i.e. without dedicated hyperparameter search. Again, the reason for that was the limited scope of the project.

The results of these experiments are first reported and discussed for the MNIST dataset (Sec. 3.1), followed by the results of the SHD experiments (Sec. 3.2). Sec. 3.1 contains detailed explanations of the experimental setups which also hold for the same types of computational experiments that are reported in Sec. 3.2.

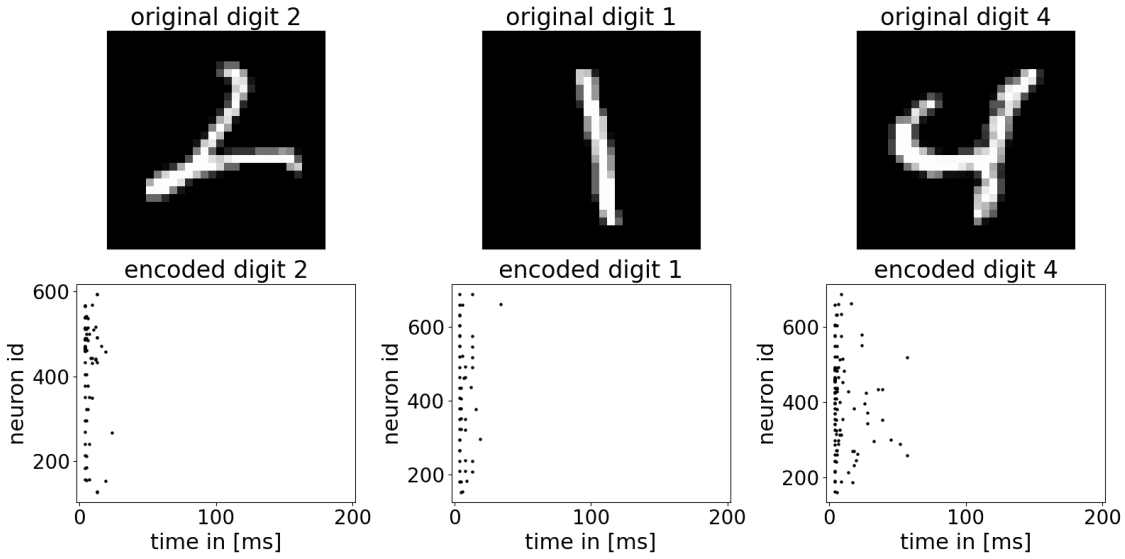

Figure 3.1: Three randomly selected samples of original MNIST inputs as images (top) and their respective latency encoded inputs shown as raster plots (bottom).

As explained before (Sec. 2.1.1), the experiments were limited to feedforward networks with a single hidden layer, without recurrent connections.

3.1 MNIST

The MNIST dataset is a collection of images of handwritten digits from zero to nine [31]. Since then, the MNIST dataset has grown to be a common benchmark dataset for image processing systems. MNIST was also used by Zenke *et al.* [61] in their studies on the robustness of LIF SNNs which makes MNIST a relevant dataset for this research project as well.

The dataset consists of 60k train samples and 10k test samples and ten classes, each class representing one of the ten digits. Each digit is a 28x28 black-and-white image.

In order to make the dataset usable for SNNs, the dataset was adapted into a spiking version of itself by applying a latency spike encoding scheme on it which has previously been described in Sec. 2.6.1. Fig. 3.1 shows three examples of both the original inputs and their respective encoded inputs. As you can see, the simulation time was 200 ms but the images were encoding using less time. The simulation of 200 ms gave the neuron models enough time to process the given stimuli. However, that caused longer periods of no stimuli for the network and thus more silent neurons towards the end of the simulation time. The resulting larger ISIs of the neurons were mitigated due to the previously mentioned slicing of spike trains (spike trains of same neurons in a batch were merged).

3.1.1 Hyperparameter Tuning

Hyperparameters for training the SNNs consisted of parameters for the SNN architecture (number of units in hidden layer), the training procedure (learning rate,

parameter	learning rate	weight scale	SG scale	batch size	hidden layer size	L2 penalty
range	$[10^{-4}, 10^{-1}]$	$[10^{0}, 10^{3}]$*	$[10^{1}, 10^{3}]$	$[128, 512]$	$[128, 1024]$	$[10^{-6}, 10^{-3}]$

Table 3.1: Optuna hyperparameter ranges for the optimization studies. For each data set and each neuron model type the same ranges were used. One exception applies here: the initial weight scale of the AdEx is explored in a range of $[10^{0}, 10^{5}]$.

L2 regularization, initial weight scaling) as well as data organization during the training (batch size). Hyperparameters are known to have a strong influence on the training outcome. Therefore, we aimed in this experiment to get a global view on suitable ranges for each hyperparameter for which the trained connection weights in the SNNs converge in a fast manner toward a "good" minimum in the loss landscape (even though it may be a local minimum). Furthermore, the hyperparameter optimization studies provided a general insight about how easy it is to train a neuron model in terms of the broadness of the acceptable hyperparameter range. Each pair of dataset and neuron model were likely to have a different set of suitable hyperparameters. Consequently, we ran three hyperparameter searches here and another three for the SHD dataset in Sec. 3.2.1.

Experimental Setup

The Optuna optimization objective was configured to maximize the test accuracy of a model. Maximizing the classification accuracy of the SNNs seemed reasonably and straight forward at that time.

Every study contained 200 trials and every trial was trained for at most 200 epochs. To reduce computation time, we enabled the pruning feature offered by Optuna to prune unpromising trials early on. Optuna used the *Hyperband* algorithm [32] by which the Optuna decides after every epoch whether pruning should occur. Pruned trials are caused by hyperparameter configurations that were found to be unsuccessful (too slow loss improvement with respect to the number of remaining epochs). Through the *Hyperband* algorithm which internally uses the *SuccessiveHalving* algorithm [24] exponentially more trials use more promising hyperparameter configurations [32]. Additionally, independent of the Optuna pruning, we implemented an early stopping (ES) procedure to prune trials that did not improve their classification error anymore after 10 epochs. The difference between Optuna's pruning and our ES procedure lies in the fact that pruning mainly occurred within the first few epochs of hyperparameter optimization and the need to stop networks whose training loss did not further decrease. Furthermore, the Optuna pruning focused on training the hyperparameter configuration efficiently with respect to the test accuracy whereas ES focused on training the SNN with respect to the train loss and stop not improving networks.

For all optimization studies the same value ranges applied for the hyperparameters mentioned above and are shown in Tab. 3.1. One exception is that the explored initial weight scale range of the AdEx is larger with $[1, 10^{5}]$ than the default range.

Optuna Results

The results are presented in the same manner for all datasets and neuron models. For a general explanation of the scatter plots take Fig. 3.2 as an example. The figure contains six scatter plots, one for each hyperparameter optimized by Optuna. Each scatter plot displays the relevant hyperparameter on the x-axis against a trained SNNs' classification accuracy on the y-axis. Each data point represents either the train or test accuracy of a SNN model (blue and orange, respectively). For comparison, the red dotted line indicates where a network model would perform at chance level.

Before describing the hyperparameter optimization results, we would like to emphasize two points. First, sub-optimal hyperparameter configurations may come from one or several hyperparameters, which includes non-trivial interactions between them. Conversely, a hyperparameter value that systematically yields low accuracy indicates a dominating effect of that hyperparameter independently from the others. Second and more importantly, many data points with low accuracies correspond to pruned trials by Optuna for which the performance has not yet converged to its maximum (but heuristically deemed as not sufficiently good for a number of training epochs by Optuna); in other words, this means that the reported accuracy may be lower than for the same number of training epochs for the SNN.

In the following, we focus on a more in-depth example of reporting the results for the LIF and the MNIST data and the summary of the tendencies across neuron models for the MNIST dataset. In-depth reporting of Izhikevich and AdEx results is available in appendix A.1.1.

Optuna results of LIF hyperparameter optimization. In general, the Optuna study with LIF-SNNs trained on the MNIST dataset (Fig. 3.2) resulted in peak accuracies of $acc_{train} \approx 99.99\%$ and $acc_{test} \approx 98.20\%$ for the train and test set respectively. The mean absolute difference between the train and test accuracies was around 1.63% for all trials for which the test accuracy was $\geq 20\%$. For the hyperparameters we made the following observations: (1) for the full search space of the number of hidden units most trials reach top accuracies. There is no obvious best hidden layer size for the LIF model. Some trials show worse accuracies but those are most likely caused by other hyperparameters that were set to a sub-optimal value. (2) The same observation can be made about the plot displaying the batch size search space. For all batch sizes the majority of SNNs reached top accuracies. Optuna seems to have focused more on batch sizes smaller than $3 \cdot 10^2$ because there were more trials for those values. However, for higher values we did not observe an obvious drop in network performances. (3) The learning rate plot shows that that the SNN models were not able to reach the top accuracies for higher learning rates which in fact decreased for learning rates $\geq 10^{-2}$. (4) The L2 regularization penalty plot displays similar results to sub-plots (1) and (2) with no discernible values that caused the SNNs to only reach sub-optimal accuracies. (5) In the weight scale sub-plot we can see that the top-accuracy were only reached if $2 \leq W_s \leq 15$. While the plot only shows one data point at a weight scale of 1 there are on fact 18 trials that used that value and ended up with accuracies at chance level. You can also see that by comparing the number of data points at chance level in the other sub plots with the number of data points with those accuracy levels

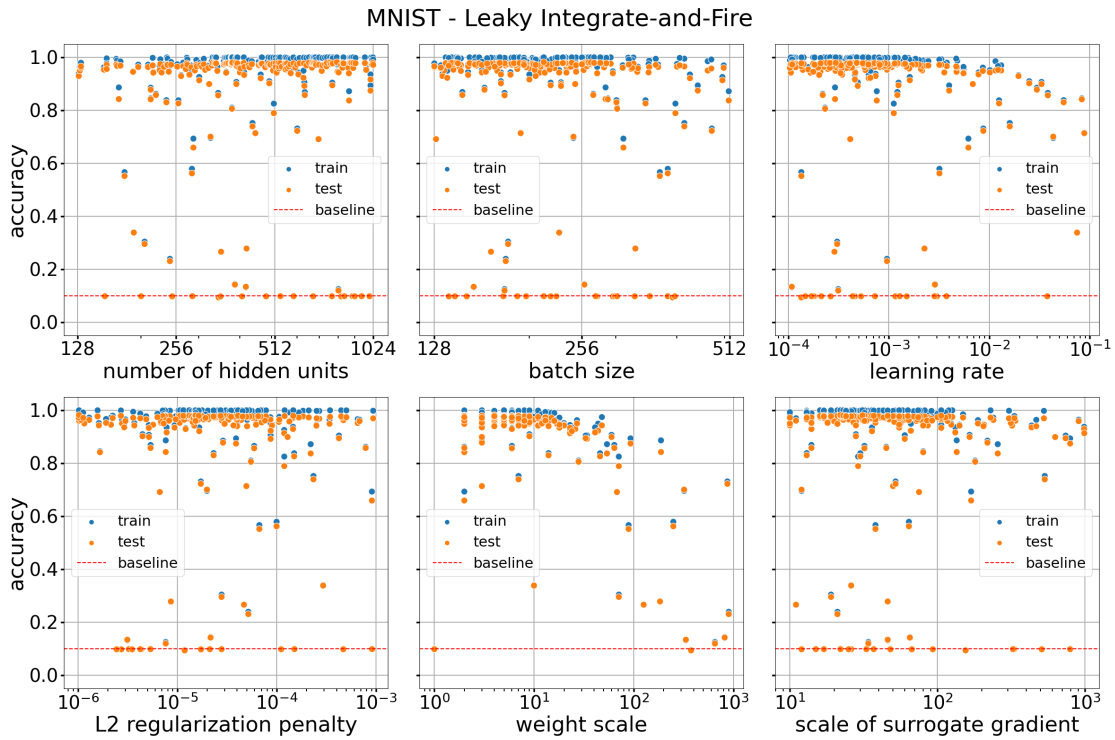

Figure 3.2: Results of the Optuna hyperparameter optimization study in which **LIF**-SNNs are trained and tested on the **MNIST** dataset. Each sub-figure shows a hyperparameter plotted against the accuracies of SNNs after training. From left to right, top to bottom the hyperparameters are: number of hidden units, batch size, learning rate, L2 penalty, weight scale and the scale of the surrogate gradient. The blue and orange dots represent the model accuracies for the training and test sets, respectively. The red horizontal dotted line indicates the chance level with an accuracy of 0.1 (for 10 classes). Peak accuracies reached $acc_{train} \approx 99.99\%$ and $acc_{test} \approx 98.20\%$ for the train and test sets, respectively.

shown in the weight scale subplot. This indicates that several trials are placed on the same weight scale - accuracy coordinates. (6) for the sixth and last hyperparameter, the SG scale, its plot does not show a clear range of values for which the models' accuracies decreased but Optuna showed a preference to select lower valued scales instead of larger ones. The remaining two hyperparameters, i.e. the learning rate and the weight scale display less stable results for their ranges than the other hyperparameters. For the learning rate, accuracies tended to decrease if the learning rate became larger than 10^{-2}. Similarly if weight scales were set to be equal to or larger than 15 the networks' classification accuracy decreased as well. On closer inspection of the data, it became apparent that if the weight scale was 1, that then the network performed at chance level whereas weight scales of $2 \leq W \leq 15$ led to top accuracies. In summary, LIF-SNNs were robust to the search space of the chosen hyperparameters. In Fig. 3.2 we saw that for the explored range of hidden layer, batch size, L2 penalty and the SG scale the networks reached training and test accuracies close to 98.2%. The learning rate and weight scale did show to have an impact on network performances if their values are too large.

Summary for all neuron models. The results above and in similar detail in appendix A.1.1 showed that with the right set of hyperparameters comparatively high classification accuracies can be achieved. For all three neuron models accuracies $> 90\%$ were reached. The highest test accuracies were reported by the LIF SNNs (98.20%) which are on a similar level as reported by Zenke *et al.* [61]. SNNs that used the Izhikevich model landed the second highest test accuracies with 96.68% (Fig. A.1) while networks that used the AdEx model reached the third highest test accuracies with 94.13% (Fig. A.2). The results also show that the LIF in general was more robust to larger ranges of various hyperparameters but all three of the neuron models were sensitive to smaller or larger values for the learning rate and the initial weight scale. Specifically, the Izhikevich and AdEx networks depended on narrower value ranges for the initial weight scale. More precisely, too low initial weight scales led to many "bad" trials. In those cases, Optuna lost many trials that it could have used to focus on other hyperparameters' search space.

From the results we see that the MNIST dataset was relatively easy to learn. The classification accuracies are close to being perfect.

Final hyperparameter values Based on the top ten to top five trials for each Optuna study, we looked at the average and median values for six hyperparameters and chose reasonable values for them. For the sake of comparability, it was feasible to select the same HL and batch sizes per neuron model. The L2 penalty was set to its minimum search space value because of its apparent insignificant influence on SNN training. The learning rate, weight scale as well as the SG scale can be seen as being specific to each neuron model for each data set. Considering these factors, we selected the individually suitable values for those hyperparameters as shown in Tab. 3.2. These values were used in both subsequent experiments.

3.1.2 Firing Regime Drift

The previous experiment showed that the Izhikevich and AdEx neuron models can be trained similarly well as the LIF neuron model on the MNIST data (Sec. 3.1.1).

Dataset	Model	Hyperparameter					
		learning rate	weight scale	SG scale	batch size	HL size	L2 penalty
MNIST	LIF	0.0003	3	25	256	800	10^{-6}
	Izh.	0.003	50	30			
	AdEx	0.055	850	65			

Table 3.2: Chosen hyperparameter for MNIST and neuron model pairs. The abbreviations *scale of SG* and *HL size* refer to the scale of the surrogate gradient during the backpropagation step and the size of or number of neurons in the hidden layer respectively. The weight scale refers to the scaling of the network weights and applies to all weight matrices.

SGL allows for learning in other neuron models than the LIF and they can reach high accuracies (especially test accuracy). Beyond the classification accuracy, a more biological question is whether the spiking regime changes during training for the Izhikevich's and AdEx's models. This is important with respect to biological plausibility. To test this, we studied the statistics of the neuron population in the hidden layer of neurons in the regular spiking regime.

The change of synaptic weight during SNN learning may change the firing regime as we are also dealing with inputs that are encoded to work with the LIF model and do not necessarily match how one would encode inputs for the Izhikevich or AdEx. This experiment was designed to investigate whether the neurons in their respective regular spiking regimes do in fact stay in that firing regime. Therefore, studying the spiking behavior of the neurons that build up the SNNs is all the more important.

Experimental Setup

This computational experiment focused on several metrics that reflect the effects of learning on the hidden layer. The hidden layer is key to form representations of the inputs that are the used by the output layer to make the classification decision.

First, the quality of learning can be quantified by how much the train and test loss converged in a number of training epochs (50), as well as the presence of strong fluctuations in its curvature. Second, we looked into the changes in the two weight distributions, those connecting the input layer to the hidden layer ($W_{In \to HL}$) and those connecting the hidden layer to the output layer ($W_{HL \to Out}$). Third and lastly, the possible alteration of the firing regime by the training was evaluated by comparing the firing rate and spike count of the response of hidden unit neurons to each stimulus, as well as the inter-spike-interval (ISI) and coefficient of variation of ISI (CV-ISI) distributions for validating that the neurons stay in their respective regular firing regime. These spiking metrics were already explained before in Sec. 2.5.

For each neuron model one network was trained to collect the relevant results. Three trials in total of networks that reach high classification accuracy was enough to gain the necessary insights in how the neuron types behave after training. Each pair has been trained using the hyperparameters reported in Tab. 3.2. All neuron models within this experiment were set to behave in their default regular firing regime.

Results

Fig. 3.2 shows an example of the results explained below. The top row depicts the train and test losses over the training epochs on the left hand side and the network's weights before and after learning in the middle and on the right hand side. The left weight plot displays the weights of the weight matrix $W_{In \to HL}$ between the input layer and the hidden layer whereas the right weight plot displays weight matrix $W_{HL \to Out}$ which contains the connection strengths from the hidden layer to the output layer. The plots in the bottom row display the spike metrics described earlier from before and after training. From left to right these are the mean firing rates, mean spike counts, ISIs and CV-ISIs per neuron.

The three networks trained on the MNIST dataset reached high accuracy values as was expected from the hyperparameter search results. The LIF had the highest values for the train and test set of 100% and 98.23% respectively, whereas the Izhikevich and AdEx reached values of 96.75% and 95.61%, and of 91.93% and 91.56% respectively.

In the following, we report the results of the individual networks with the same approach as in the previous Sec. 3.1.1. Similar to the previous experiment, the reporting of results for the LIF network is more detailed whereas the rest of the results will be summarized for the sake of brevity.

LIF The train and test losses in Fig. 3.3 show that the training of the LIF-network's weights converged after around 20 epochs. The test losses were higher than the train losses but they kept following the train losses with approximately the same difference. Around epoch 29 there was a sudden increase in both losses after which the network quickly returned to previous loss levels. This might have occurred due to the surrogate gradient not being precise which may cause an unstable loss curve. There was a shift in the weights during network training from (close to) zero-valued weights to larger values. While both weight matrices show this trend the distribution of $W_{HL \to Out}$ changed more and has thicker tails. After training, the hidden layer neurons had reduced mean firing rates while also the amount of neurons with close to zero firing rates was reduced. Notably, there were no more single neurons with average firing rates larger than 20 Hz after training. The mean spike count plot also shows that there were a lot more non-silent neurons after training than before training. This can also be related to the increased amount of non-zero weights in $W_{In \to HL}$. More non-zero weights mean a higher chance for neurons to elicit spikes due to positively amplified inputs. The reported ISIs were larger after training with most intervals being close to 200 ms, very few close to 0 ms and a few spike trains reported larger than 600 ms intervals. Please note however, that for the ISI and CV ISI plots the amount of actual data points underlying the distributions was different for the pre and post training conditions. This was because the amount of spikes differed before and after training; and generally, a spike train can only compute an ISI if it contains at least two spikes. For this reason, the description of Fig. 3.3 reports number of samples without spikes, prior and post training. Last but not least, the mean of the CV ISI distribution increased from approximately 0.75 prior training to 1.1 post training. Both distributions before and after training indicate a rather regular firing behavior because their averages are close to 1. Still, the LIF neurons fired more irregular after training than before.

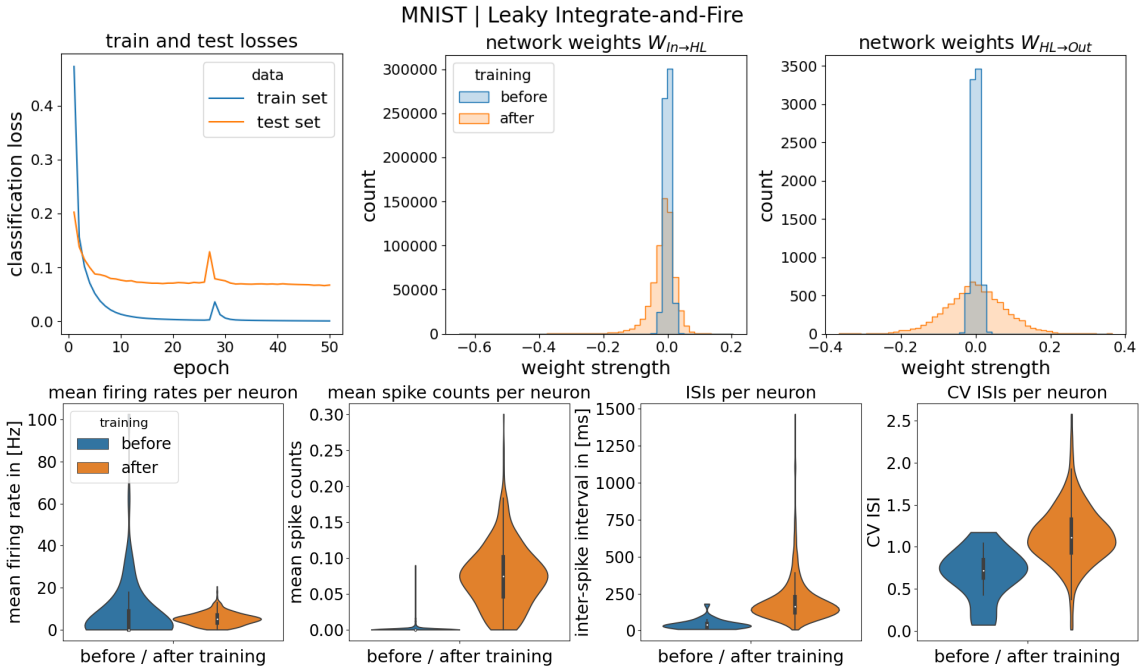

Figure 3.3: Network and neuronal statistics of an LIF network trained on MNIST data. The trial reached a train and test accuracy of 100% and 98.23% respectively. The top left panel indicates the loss values during the optimization. The same color coding is used for all panels with blue and orange for the train and test sets, respectively. The top middle panel indicates the weight distribution for the connections from the inputs to the hidden layer, and the top right panel for connections from the hidden layer to the outputs. The bottom left and middle left panels respectively show the spiking rates and spike counts of the neurons during the response to stimuli (averaged over a batch of 256 stimuli and across 800 neurons). The bottom middle right panel indicate the distributions of inter-spike intervals (ISIs); not that silent neurons are excluded here. The bottom right panel indicates the corresponding coefficient of variation (CV) of the ISIs for neurons that fire more than 3 spikes during the response, as an indication for the firing regime. Samples without spikes before training: $\approx 64\%$. After training: 0%.

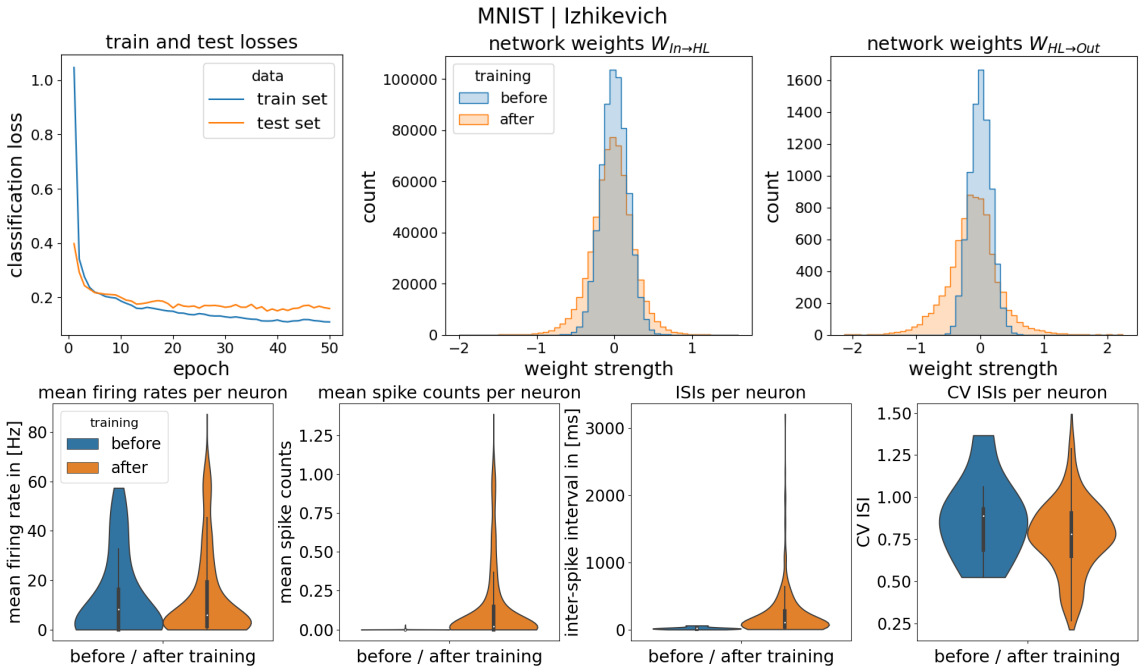

Figure 3.4: Network and neuronal statistics of an Izhikevich network trained on MNIST data. The trial reached a train and test accuracy of 96.75% and 95.61% respectively. Samples without spikes before training: $\approx$ 83%. After training: 0%. For a detailed description of all figure panels, see Fig. 3.3.

Izhikevich The results from the Izhikevich network in Fig. 3.4 indicate that the network did not completely converge because the train loss was still slightly decreasing after 50 epochs. However, the test loss did not decrease any further. The changes in weights do seem to be larger for the LIF than for the Izhikevich and AdEx network. Especially the AdEx weight distributions changed less in relation to the weight size while still reaching a test accuracy of 91.56%. The Izhikevich network started with more zero-spike samples (83%) than the LIF (64%) but managed to reduce that amount to 0%. The mean firing rates did slightly increase for a few neurons and was comparatively higher with a maximum of around 80 Hz compared to maximally 20 Hz of the LIF. Because the number of samples without spikes was greatly reduced and more sparse spiking neurons remained after training, there were more neurons with high ISIs. The irregularity of spiking did not change much after training. With average CVs smaller than 1.0, the firing behavior was rather regular and if at all, one could say that there was a minor shift in the CV ISI post training distribution towards more regular spiking.

AdEx The results for the AdEx in Fig. 3.5 indicate that the network converged. Here the test loss was remarkably close to the train loss during training. The train and test accuracy values were 91.93% and 91.56% respectively. The weight distributions changed less than to the LIF weight changes. While the AdEx started with many high firing rate neurons, training showed to have an effect on reducing the mean firing rates significantly. Still, many sparse spiking neurons remained with only a few neurons that had more spikes on average. The ISI and CV ISI before training have a very flat distribution because of too few samples that caused spikes. However, the network learned from those few samples to reach high accuracies with

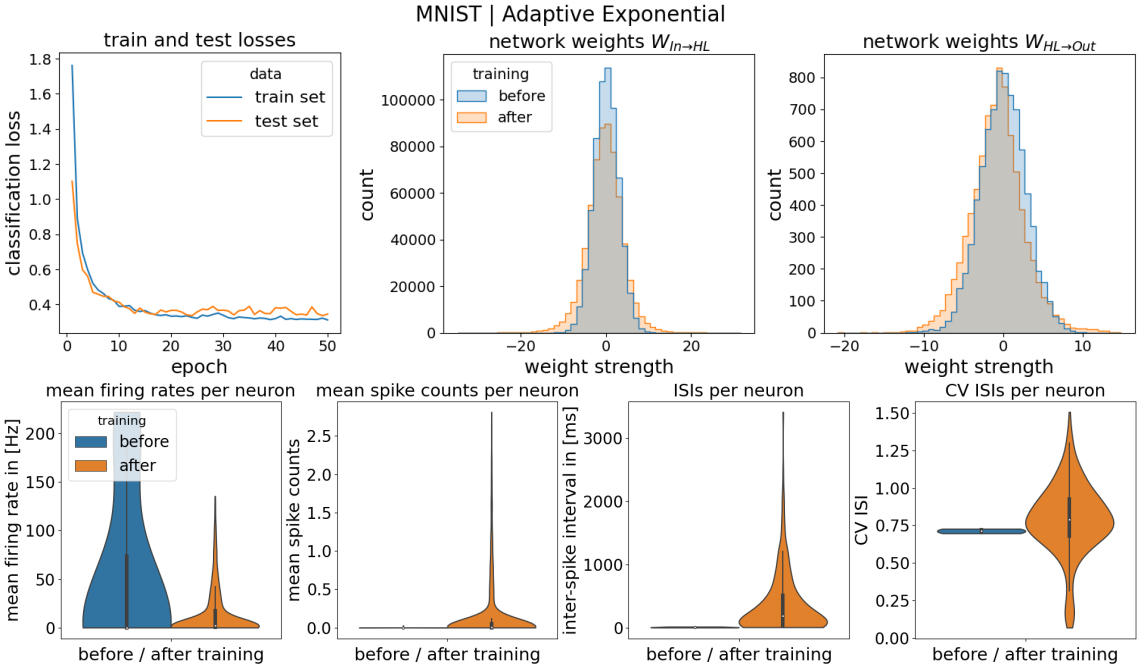

Figure 3.5: Network and neuronal statistics of an AdEx network trained on MNIST data. The trial reached a train and test accuracy of 91.93% and 91.56% respectively. Samples without spikes before training: $\approx 93\%$. After training: 0%. For a detailed description of all figure panels, see Fig. 3.3.

a sparse spiking network. Most neurons showed regular firing behavior, with very few displaying irregular and other very regular, deterministic firing behaviors.

The test loss for both Izhikevich and AdEx was remarkably closer to the train loss compared to the difference in losses for the LIF network (Fig. 3.3). The Izhikevich and AdEx networks did seem to reach high classification accuracies without large weight changes compared to the LIF. While the LIF had overall the lowest mean firing rates, its neurons showed more irregular firing behavior and higher counts of mean spikes per neuron. The Izhikevich and AdEx neuron models were capable of reaching similarly high classification accuracies as the LIF while displaying sparser and more regular spiking activity.

3.1.3 Firing Regime Comparisons

From the closer inspection of network and neuronal statistics (Sec. 3.1.2), we moved on to train SNNs that use other firing regimes than we have previously used. The aim of this experiment was to show that the range of firing regimes the Izhikevich and AdEx dynamics provide can in fact be leveraged by using SGL. Another goal was to find advantages and disadvantages other firing regimes offer compared to the RS-Izhikevich and TO-AdEx regimes. For completeness and a better comparison, we still included the regular firing regimes that we have seen before here as well. Thus, the studied set of Izhikevich firing regimes included next to the regular spiking (RS) the fast spiking (FS), intrinsically bursting (IB) and chattering (CH) regimes. Additional firing regimes of the AdEx were, next to tonic firing (TO): adaptive (AD), bursting (BU), intrinsically bursting (IB) and irregular firing (IR).

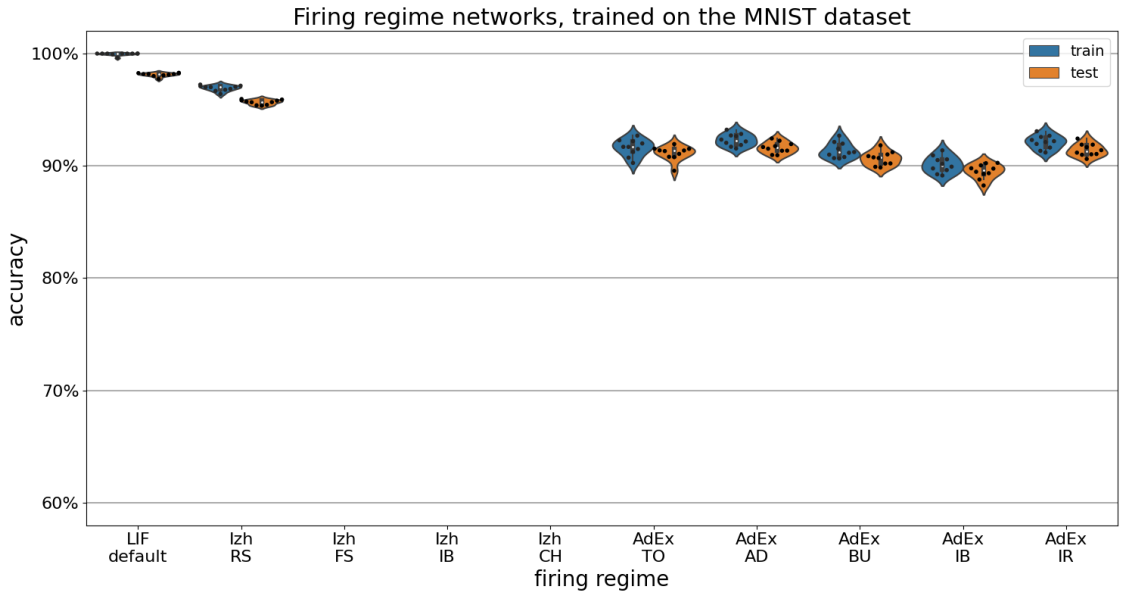

Figure 3.6: Test accuracies of various firing regimes, trained on the MNIST data set. Each regime was trained 10 times. Note that data points below 0.6 test accuracies were cut off here to increase visibility of the narrow violin plots.

Experimental Setup

Again, the hyperparameter values reported in Tab. 3.2 were used for the respective neuron models. These were obtained from optimization studies on SNNs in the regular spiking regime. Here, we want to remind the reader that these hyperparameter values were used for the other firing regimes as well which means that the non-regular firing regimes were potentially not trained in their actual optimal hyperparameter spaces. From above selection of firing regimes, there were consequently 10 unique SNN configurations: one for the LIF model, four for the Izhikevich regimes and five for the AdEx regimes. The performance of those networks is reported in terms of test classification accuracy. Each SNN configuration was trained ten times which allowed for enough possible variation in the training accuracies.

Results

Fig. 3.6 shows the results in form of swarm plots (black dots) that are plotted on top of violin plots. Please note that some networks returned very similar accuracy values which is why the swarm plots that are plotted on top the violin plots do not always show all ten trained networks per regime. The results in that figure confirm the high train and test accuracies of the regular firing regimes for all three models which we have previously seen in Sec. 3.1.1 and Sec. 3.1.2. Explicit values of the mean and standard deviation of each firing regime network are reported in Tab. 3.3. The non-regular Izhikevich regimes FS, IB and CH failed in all their attempts to train their respective network's connection weights. However, accuracies below 60% were cut off to increase the visibility of the violin plots of the successfully trained networks. Therefore, the results of the FS, IB and CH Izhikevich networks are not visible in Fig. 3.6. All non-regular AdEx regimes however did result in accuracies of the train and test set that were on a similar level as the regular (TO) firing regime.

	LIF	**Izhikevich**				**AdEx**				
regime	**-**	**RS**	FS	IB	CH	TO	**AD**	BU	IB	IR
median	**98.2**	**95.7**	9.8	9.8	9.8	91.3	**91.6**	90.7	89.6	91.3
mean	**98.1**	**95.7**	9.8	9.8	9.8	91.1	**91.6**	90.7	89.5	91.4
std	**0.2**	**0.2**	0.0	0.0	0.0	0.6	**0.5**	0.6	0.6	0.5

Table 3.3: Percentage-wise mean, median and standard deviation of the test accuracies of ten trained SNNs for each firing regime. The SNNs were trained on the MNIST dataset. The regimes with the highest test accuracy per neuron model are highlighted in bold.

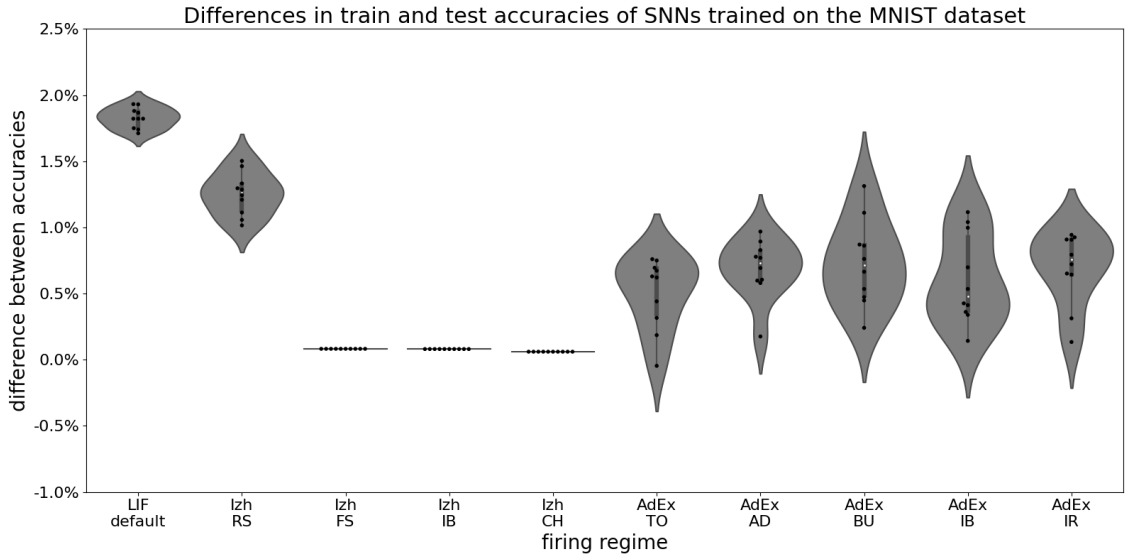

Figure 3.7: Test accuracies of various firing regimes, trained on the MNIST data set. Each regime was trained 10 times.

In fact, the regimes AD and IR returned on average slightly higher test accuracies than the TO regime. Generally, the highest test (and train) accuracy values were reached by the LIF model ($\approx 98.1\%$), while the Izhikevich RS regime reached the second highest ($\approx 95.7\%$) and the AdEx in the AD the third highest set of test accuracy values ($\approx 91.6\%$). Notably, the standard deviation of the classification for both train and test set is very small for all firing regimes.

3.2 Spoken Heidelberg Digits

For the second series of simulations, we trained the SNN models on the Spoken Heidelberg Digits (SHD) dataset [11]. The dataset was created as a ready-made benchmark dataset for training SNNs which is in contrast to the MNIST dataset that first needs to be transformed into SNN-suitable input encodings [11].

As the name suggests, the data are audio recordings of spoken digits. Cramer *et al.* [11] created the dataset from converting audio data to spike data from recordings of twelve speakers saying the digits zero to nine. The ten digits were spoken in the English and German language which resulted in twenty classes. The train and test sets contain recordings of ten and two distinct speakers respectively.

	classes	input units	samples train	samples test
original	20	700	8156	2264
English only	10	700	4011	1079

Table 3.4: Specifications of the SHD dataset [11], including the number of classes, input units (channels) and samples. The experiments discussed of this project only used the 10 English classes out of the available 20 English and German classes.

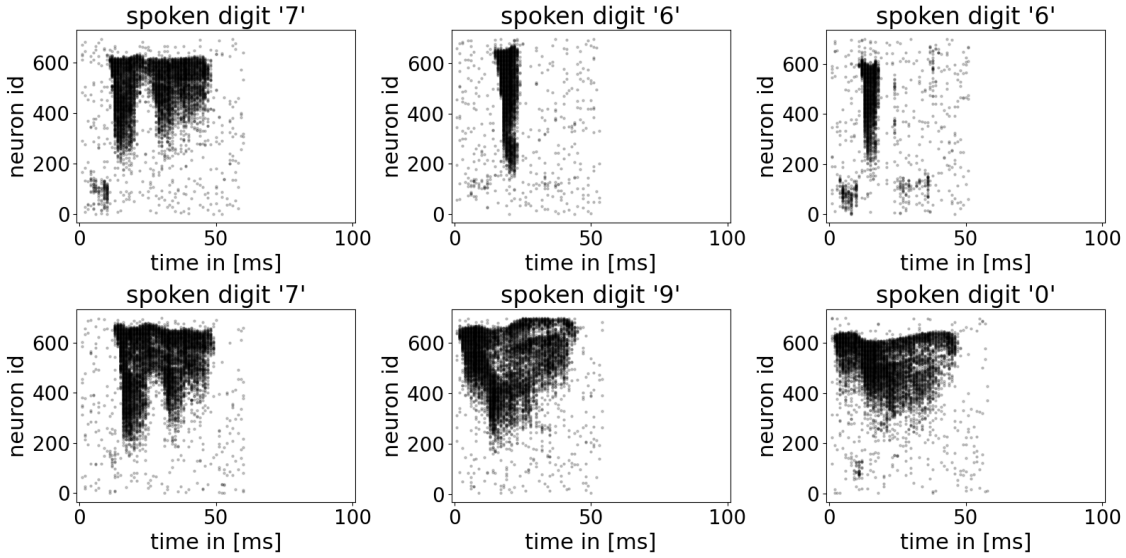

Figure 3.8: Six randomly selected samples of original SHD input stimuli. The input to the SNNs was extended to $100ms$ because of the varying length of the samples and to give the neuron in the network time to propagate all information from input to output layer.

The experiments reported here, only used the ten classes that represent the English spoken digits. The reasoning behind this was comparability to the ten MNIST classes. With ten classes in all used datasets the results were easier to compare with one another. The details and size of the resulting dataset are shown in Tab. 3.4.

Cramer *et al.* [11] converted audio data to spike data by passing the audio data through their artificial cochlea model called *Lauscher*. *Lauscher* approximates the generation of spikes of the "inner ear and the ascending auditory pathway" as explained by Cramer *et al.* [11].

Through the means of approximating neurological signals the SHD data is biologically closer to incoming signals of neurons than the spiking MNIST data is. Therefore, the SHD dataset seems to be biologically more relevant than the spiking MNIST data when it comes to the comparison of the neuron models in this research project.

Fig. 3.8 shows three samples of both the original inputs and their respective encoded inputs. As can be seen in the figure, the SHD dataset can be considered more biologically realistic than the MNIST dataset. In the following, we investigated whether for the SHD data the same conclusions for the neuron models apply (in

Dataset	Model	Hyperparameter					
		learning rate	weight scale	SG scale	batch size	HL size	L2 penalty
SHD	LIF	0.0002	1.5	20	180	800	10^{-6}
	Izh.	0.001	4	20			
	AdEx	0.004	35	20			

Table 3.5: Chosen hyperparameter for SHD and neuron model pairs. The term *SG scale* refers to the scale of the surrogate gradient during the backpropagation step, while *HL size* refers to the size of or number of neurons in the hidden layer. The weight scale refers to the scaling of the network weights and applies to all weight matrices.

terms of learning and of the resulting classification performance).

3.2.1 Hyperparameter Optimization Studies

A hyperparameter search is necessary for the SNN models trained on the SHD data as well because of the same reasons given in Sec. 3.1.1. Again, we used Optuna to find suitable sets of hyperparameters for each neuron model. The specific experimental setup was the same as for the MNIST dataset (Sec. 3.1.1).

Optuna Results

In the following, we summarized the results of this experiment. A more in-depth description along with figures of the results per neuron model is given in Appendix A.1.2.

The hyperparameter optimization study was successful in finding hyperparameter values that allowed every neuron model to train SNNs with relatively high accuracies. Again, the best performing networks came from the LIF, followed by the Izhikevich and lastly the AdEx model. The best LIF trials reached close to perfect train accuracy values of around 99% but only reached peak test accuracy values of around 85% (Fig. A.3). The Izhikevich SNN models showed in fact average train accuracies of 75% but higher test accuracies around 83% (Fig. A.4). A similar observation comes from the AdEx networks for which peak accuracies were around 62% for the train set and 65% for the test set (Fig. A.5). For the LIF, the difference between the train and test set accuracy values is larger than for the other two models.

The results show that training the SNNs depends on the right set of hyperparameters, more so for the AdEx and Izhikevich SNNs and less so for LIF SNNs. The LIF is more robust to a wider range of hyperparameter values for all hyperparameters. Choosing the right weight scale and learning rate is essential for all models but especially so for the AdEx model which showed a small range of viable values for both hyperparameters. More hidden units and a smaller batch size were more relevant for learning with the Izhikevich and AdEx networks compared to the LIF networks. The same holds for lower L2 penalties and the SG scale for all three models.

Final hyperparameter values From those results the hyperparameter values presented in Tab. 3.5 were chosen. Similar to the MNIST results, we selected the

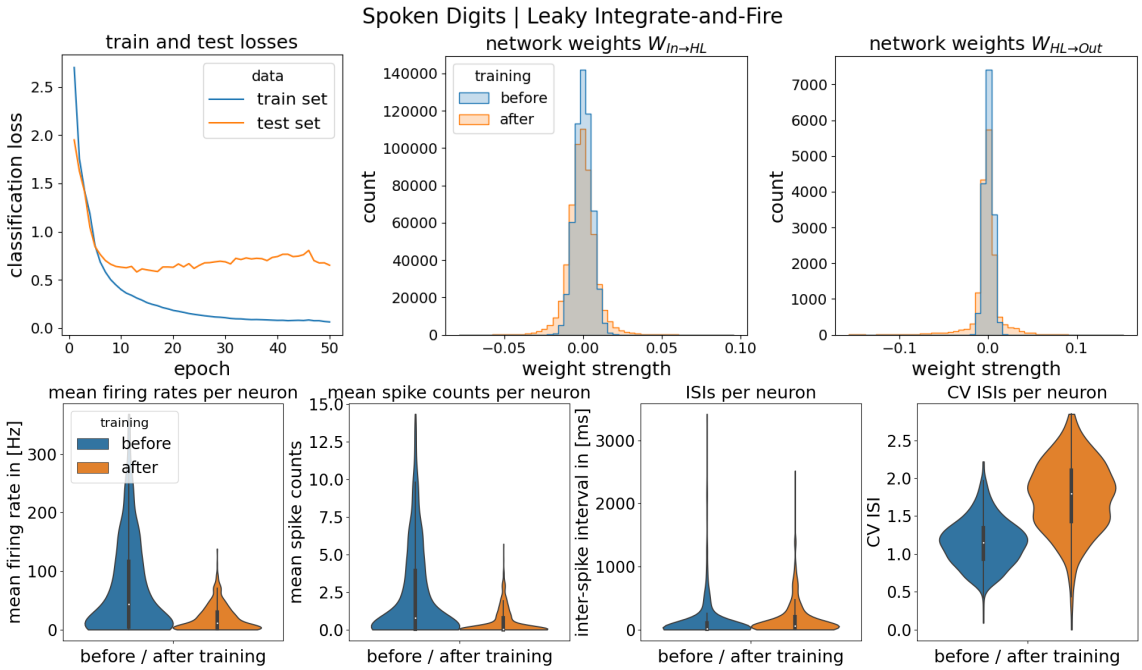

Figure 3.9: Network and neuronal statistics of a LIF network trained on SHD data. The trial reached a train and test accuracy of 99.70% and 80.67% respectively. Samples without spikes before and after training: 0%. For a detailed description of all figure panels, see Fig. 3.3.

same HL and batch sizes for each neuron model. A low L2 penalty did lead to better performing networks for each neuron model. The learning rate, weight scale as well as the SG scale can be seen as being specific to each neuron model which is why we selected suitable values for those hyperparameters. Based on top ten to top five trials per neuron model, we chose reasonable values from comparing the average and median values of the learning rate, weight scale and SG scale. These settings were then used in the subsequent computational experiments.

3.2.2 Firing Regime Drift

Let us move on to inspect the network and spike metrics of the regular firing regime networks that were trained on the SHD dataset. Here, the same reasoning and explanations for the experiment applied that we have already given in Sec. 3.1.2 for the MNIST dataset. Again, the same experimental setup applied that is described in Sec. 3.1.2.

Results

In the following, we presented the results of the single trials for each neuron model.

LIF The trained LIF network reached a train and test accuracy of around 99.7% and 80.67% respectively. Fig. 3.9 shows the network losses, its connection weights as well as its spike metrics. While the network's train loss did not improve much more after 40 epochs, the test loss deviated from the train loss after a few epochs and increased on average over the course of training. The test loss started to drop

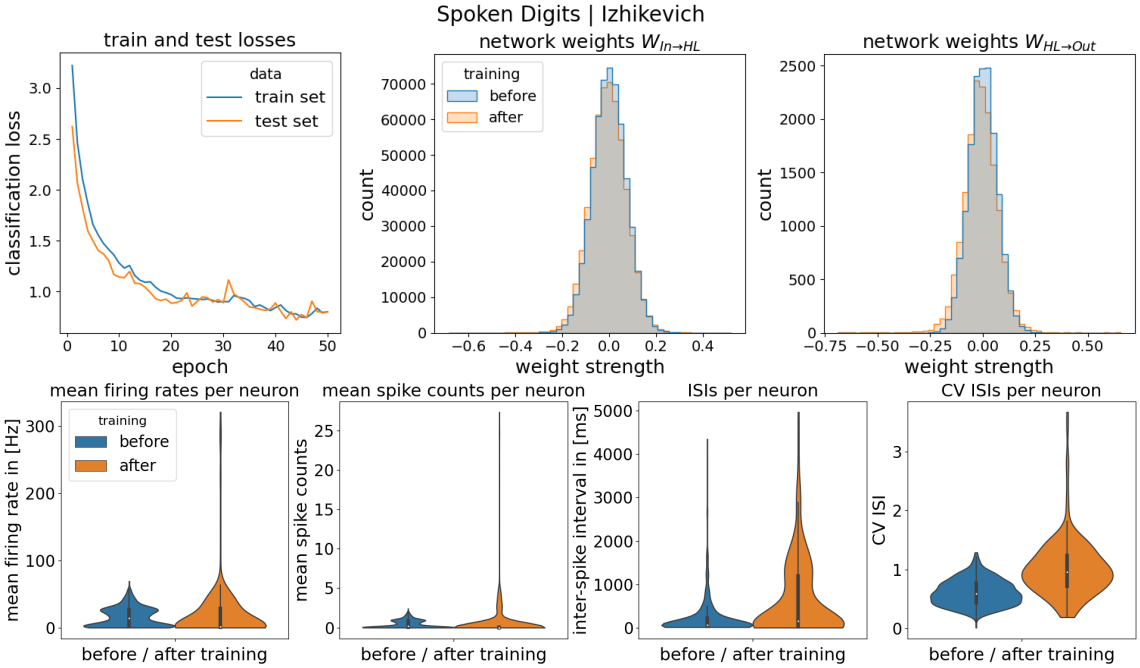

Figure 3.10: Network and neuronal statistics of an Izhikevich network trained on SHD data. The trial reached a train and test accuracy of 79.14% and 77.22% respectively. Samples without spikes before and after training: 0%. For a detailed description of all figure panels, see Fig. 3.3.

slightly shortly before epoch 50. The divergence in the losses and the worsening test loss indicate that the LIF model overfit on the train set. The weight distributions of both weight matrices show a larger spread after training. Looking at the reduced mean firing rates and spike counts after training, it is clear that the network showed sparse spiking behavior with no neuron eliciting more than 6 spikes on average. The ISIs did not change much except for a slight decrease for the maximum ISI from around $3500ms$ to $2500ms$. Furthermore, the CV ISI distribution after training changed noticeably to higher CV ISI values on average. This indicates that the neurons fired more irregularly, with a mean CV of around 1.75.

Izhikevich The trained Izhikevich network reached a train and test accuracy of around 79.14% and 77.22% respectively. Fig. 3.10 shows the network losses, its connection weights as well as its spike metrics. The network seems to have mostly converged after 50 epochs of training. The test loss was very close to the train loss every epoch. This is reflected in the close train and test accuracy values. The weight distributions have not changed much after network training for the Izhikevich model which already had rather broad weight distributions pre and post training when compared to the LIF (and the AdEx). This is also visible in the spike metrics plot. Here, the plot shows that the spiking activity increased after training but not by much. After training, there were few neurons with a high firing rate and thus a higher spike count whereas there were none before training. Compared with the other neuron models, the Izhikevich model started in a sparse spiking regime and stayed there. The network also had increased ISIs, with slightly increased irregularity in the neurons' firing behavior. However, the mean of the CV distribution was still around regular levels of firing.

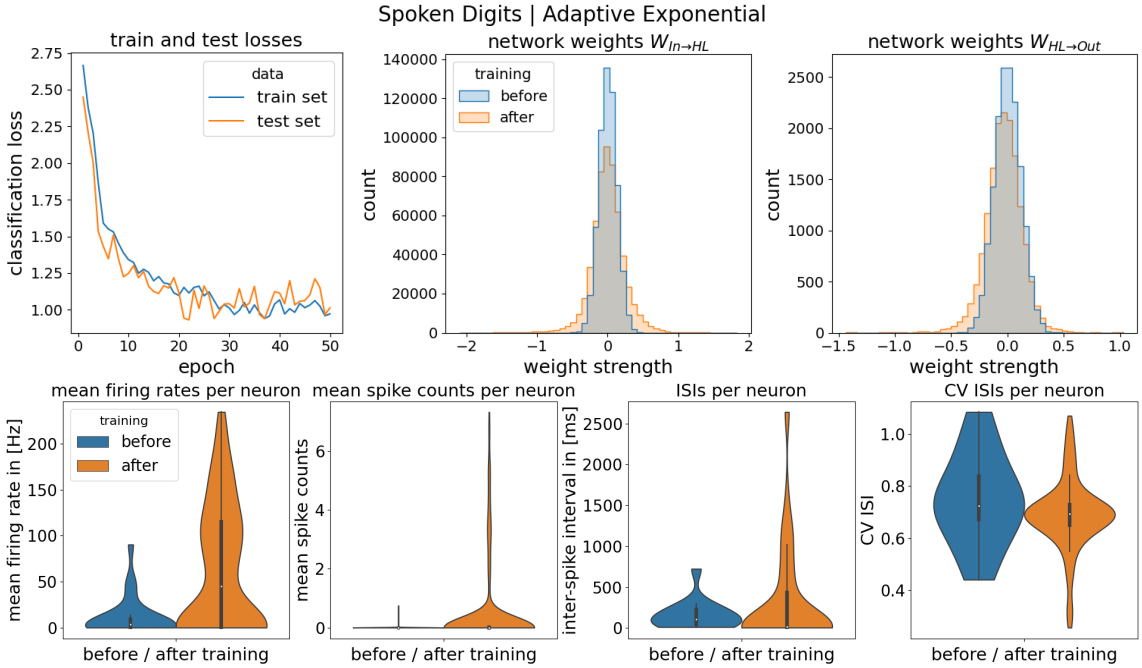

Figure 3.11: Network and neuronal statistics of an AdEx network trained on SHD data. The trial reached a train and test accuracy of 63.08% and 65% respectively. Samples without spikes before training: $\approx$ 23%. After training: $\approx$ 0%. For a detailed description of all figure panels, see Fig. 3.3.

AdEx The trained AdEx network reached a train and test accuracy of around 63.08% and 65.00% respectively. Fig. 3.11 shows the network losses, its connection weights as well as its spike metrics. Looking at the loss which is somewhat unstable, the network seems to have converged in the last twenty epochs. Similar to the Izhikevich model, the AdEx test loss followed the train loss very closely. However, the test loss was very unstable compared to its smoothness in the other models. The weight distributions show a change after training without any noteworthy salience. After training, the AdEx neurons increased their firing rates, mean spikes counts as well as ISIs per neuron. The network thus deviated from a sparse spiking behavior. Looking at the CV ISI, there were a few neurons that started spiking more regularly than others but most neurons moved towards a CV ISI of 0.7. Overall, the mean spiking regularity did not change much during training.

To summarize the main similarities and differences: the LIF and the Izhikevich reached a similarly high test accuracy (80.67% and 77.22% respectively) with a stark difference in train accuracy between one another. The AdEx model reached a good but not as high test accuracy (65.0%). Moreover, the train losses of the Izhikevich and AdEx models converged on higher values compared to the LIF model's train loss. However, the test losses and the test accuracies of both models were notably closer to the train losses and accuracies than was the case for the LIF model. The LIF and Izhikevich models operated or learned to operate within a sparse spiking mode and tended to fire spikes more irregularly, whereas the AdEx model diverged more from a sparse spiking mode and displayed more regular firing behaviors.

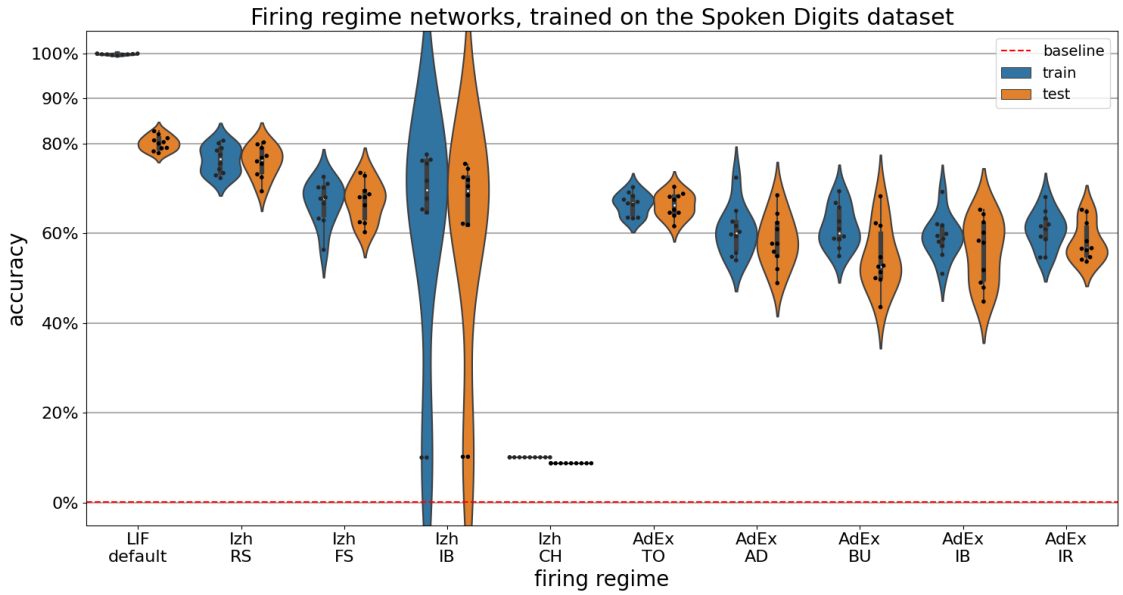

Figure 3.12: Test accuracies of various firing regimes, trained on the SHD data set. For each regime we trained the networks 10 times.

	LIF	**Izhikevich**				**AdEx**				
regime	**-**	**RS**	FS	IB	CH	**TO**	AD	BU	IB	IR
median	**80.1**	**76.4**	68.0	69.4	8.8	**66.3**	57.7	52.7	58.1	56.6
mean	**80.1**	**75.9**	67.1	57.7	8.8	**66.3**	58.3	54.7	56.2	58.2
std	**1.6**	**3.5**	4.4	25.5	0.0	**2.7**	5.9	7.3	7.3	4.3

Table 3.6: Percentage-wise mean, median and standard deviation of the test accuracies of ten trained SNNs for each firing regime. The SNNs were trained on the SHD dataset. The regimes with the highest test accuracy per neuron model are highlighted in bold.

3.2.3 Firing Regime Comparisons

Next, the final experiment explored the performance of the neuron models when other firing regimes than the regular spiking regime were used. Here, the same reasoning and explanations for the experiment applied that we have already given in Sec. 3.1.3 for the MNIST dataset. The same experimental setup as described in Sec. 3.1.3 applied here as well. We trained ten different networks: one, four and five different networks for the LIF, Izhikevich and AdEx networks respectively. Each network was trained ten times for 50 epochs.

Results

Fig. 3.12 shows the distribution of train and test accuracy values for each network. Tab. 3.6 reports the median, mean and standard deviation of the mean of the test accuracies of all tested regimes. While the LIF networks obtained the best accuracies of all networks, Fig. 3.12 clearly shows that the LIF networks overfit on the train set with train accuracies around 100% and obtained test accuracy values that were on average 20% lower with a mean of 80.1%.

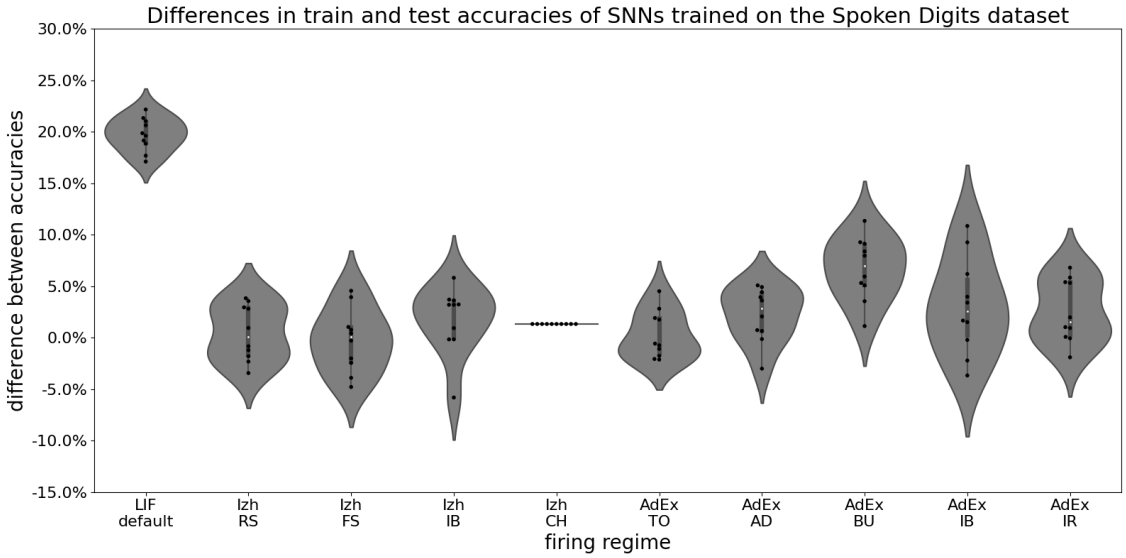

Figure 3.13: Test accuracies of various firing regimes, trained on the SHD data set. Each regime was trained 10 times.

The Izhikevich networks obtained comparatively high accuracies: peak train and test accuracy values of the Izhikevich RS regime were even as high as the mean LIF test accuracy. Out of the additional Izhikevich regimes, the FS and IB performed well while the CH networks were not trainable at all. In two out of ten cases the IB networks were not trainable also. Compared to the MNIST results (Sec. 3.1.3) for which none of the additional firing regimes worked, this was a substantial improvement. If we look at the best mean test accuracy, the FS regime performed best (67.1%) out of the additional Izhikevich regimes. However, if we look at the median test accuracy which was less influenced by the two outliers, the IB networks did perform better (69.4%) than the FS networks. Thus, if we disregard the IB outliers the IB performed better than the FS regime.

Out of all AdEx regimes, the TO regime reached the highest average accuracy (66.3%) while also displaying the smallest deviations from its mean accuracy. This means that the AdEx TO on average did not perform better than the Izhikevich regimes (apart from the CH regime). Other AdEx regimes performed on average slightly worse than the TO regime. Non-TO AdEx regimes did however perform on similar levels. IB and AD networks performed best out of Non-TO networks if we look at the median (58.1%) and mean (58.3%) test accuracy values respectively.

We close the results chapter by inspecting the differences in train and test accuracies more closely. In Fig. 3.12 we have already seen that for all Izhikevich and AdEx regimes the test accuracy values were on similarly high level as their respective train accuracy values. The LIF networks however showed a stark difference between its accuracies. Fig. 3.13 depicts these differences more clearly while Tab. 3.7 reports the median, mean and standard deviation of accuracy differences for each regime. This figure and table make the difference between the LIF on the one hand and the Izhikevich and AdEx on the other hand more obvious: While the LIF overfit on the train set, the Izhikevich and AdEx regimes generalized surprisingly well on the

	LIF	**Izhikevich**				**AdEx**				
regime	**-**	**RS**	FS	IB	CH	TO	AD	BU	IB	IR
median	**19.7**	**0.1**	0.1	3.2	1.3	**-0.7**	2.8	6.9	2.5	1.5
mean	**19.7**	**0.4**	-0.3	1.7	1.3	**0.3**	2.2	6.7	3.1	2.5
std	**1.6**	**2.7**	3.1	3.3	0.0	**2.3**	2.6	3.1	4.7	3.0

Table 3.7: Percentage-wise median, mean and standard deviation of the mean differences of the test accuracies for each firing regime. The SNNs were trained on the SHD dataset. The regimes in bold represent the regimes with the smallest differences per neuron model.

unseen test set. In fact, for the RS and FS Izhikevich SNNs and the TO, AD, IB and IR AdEx SNNs, some of their training runs reached even higher test accuracies than their respective train accuracies.

Chapter 4

Discussion

We start the discussion of the results with a brief summary on the important findings.

First of all, for all neuron models the regular firing regime networks were trainable for each dataset. However, the LIF neuron reached the highest average train and test accuracy values for all datasets. The Izhikevich networks in RS regime were able to reach on average better results than AdEx networks in the TO regime. Still, all three neuron models reached accuracies above 90%.

Second, we found that for each datasets all three neuron models depended on a suitable initial weight scale to for successful training of the networks (Sec. 3.1.1, 3.2.1). The Izhikevich and even more so the AdEx model showed to operate in more narrow initial weight scale ranges than the LIF. In contrast, the LIF overall operates well without strict hyperparameter tuning.

Third, the results showed that learning enforced a sparse spiking paradigm (Sec. 3.1.2, 3.2.2). Except for one case (AdEx trained on SHD), all models showed signs of operating in a sparse spiking regime. Compared to the LIF model, the Izhikevich and AdEx model both featured a small selection of neurons that displayed higher than average spiking activities. These seemed to be very few but influential neurons. Interestingly, when comparing the network performances and the CV ISIs there seems to be a correlation between high performing networks and sparser but more irregular firing. It does seem that the training reduced the number of spikes in the Izhikevich hidden layer (similar to the LIF) which enforced sparse coding in the hidden layer of the network. This has been studied before [41][57][50][43][58][19] and in a way transforms the inputs into activity patterns that do not overlap, so they are "orthogonal" and for a basis that makes the readout learning easy [49].

Fourth, while for the MNIST dataset the FS, IB and CH Izhikevich regimes were not trainable (Sec. 3.1.3), only the CH Izhikevich regime was not trainable for the SHD dataset (Sec. 3.2.3). The reason for this might actually be unsuitable sets of hyperparameters for these 'failed' regimes. The CH might require a precise hyperparameter tuning, but it might also be that it cannot be trained at all.

Fifth, compared to the MNIST accuracies, the SHD accuracies were lower for each neuron model. This means that the SHD dataset was more difficult to train on. This was likely caused by fewer samples in the (train) data set on the one hand and perhaps by more complex patterns and temporal dependencies within the spike trains on the other hand.

And finally, while it is not very apparent in the results of the MNIST regime comparisons (Sec. 3.1.3), the SHD results indicate that the LIF model is more prone to overfit on the train set than the Izhikevich and AdEx models (Sec. 3.2.3). Especially for the latter dataset, the results of the Izhikevich and AdEx networks indicate strong generalization capabilities towards new unseen data.

Additional experiments

During this research project, more computational experiments were conducted than reported. The results of these experiments are excluded from the main body of this thesis because they did not greatly inform the conclusions about the main research

goal. However, they still returned results that could be of value for future research and are thus summarized in the following:

In earlier simulations, the connection weights of $W_{IN \to HL}$ were fixed (frozen) to investigate the capability of the readout layer to learn patterns from the Izhikevich and AdEx compared to the LIF model. The results showed that the readout layer alone was capable of delivering surprisingly good classification performances. Naturally, the classification accuracy was lower than the accuracies reported in chapter 3. Interestingly, when all firing regimes were compared, there were more apparent inter-dataset differences (MNIST-SHD) for Izhikevich regimes (as we have seen before), whereas there were more intra-dataset differences for the AdEx regimes (between regimes, for each dataset). The latter observation indicates that the supposed flexibility of diverse firing regimes involved in shaping input-output mappings for the classification task could only be used when both input and output connectivities (weights matrices) are trained. This holds for at least the AdEx regimes.

Additional experiments explored the performance of the Izhikevich and AdEx neuron models on a third dataset. The results of those trials that were trained on RandMan data are reported in Appendix Sec. A.2. They were not included in the main report because the results did not further inform the goals of the research project. The comparison of the various firing regimes (Sec. A.2.3) showed that the LIF SNNs overfit and that the Izhikevich and AdEx SNNs failed to train and reach good accuracies for both train and test set. A summary of these and other RandMan results is still included in the appendix for completeness.

4.1 Conclusion

In this research project, we have shown that the Izhikevich and AdEx neuron models can be used in addition to the LIF neuron model to train SNNs that use SGL.

On the one hand, the Izhikevich and AdEx networks did not perform better than the LIF model in terms of test accuracy. In fact, they performed slightly worse in that regard. On the other hand, the Izhikevich and AdEx performed better in terms of generalization, especially if trained on the biologically more realistic SHD dataset. This observation held for all firing regimes of both neuron models. A direct comparison of how the generalization capabilities of both models behaves when the classes of the SHD are increased from ten to twenty classes would be an interesting next step.

The trained networks remained in a sparse spiking paradigm. It should therefore be possible to reduce the hidden layer size and create more efficient networks. Previous research already used fewer neurons in the hidden layer of LIF SNNs while reaching close-to-perfect accuracy [61]. Here, we chose larger hidden layers because the hyperparameter search indicated the Izhikevich and AdEx SNNs would require more neurons. Because of the latter's shown sparseness of spikes, future work may show that using fewer neurons works similarly well.

Furthermore, Optuna's optimization objective was set to solely maximize the train accuracy. Additionally, one could set Optuna or other hyperparameter optimization techniques to optimize towards sparse spiking networks, similar to the adding of the

L2 penalty to the loss function.

Even though the non-regular firing regimes did not show any advantage over using the regular firing regimes for the classification, they showed that a variety of neuronal firing regimes are compatible with SGL training. RS and TO regimes, for which the hyperparameters were optimized, reached the highest performances compared to their respective neuron model's other regimes that did not benefit from an individual hyperparameter optimization. However, most non-regular regimes obtained as similarly high test accuracies as the regular firing regimes. Future work could find that individual hyperparameter tuning for the other non-regular regimes further improves their performances. The tuning may depend on the data for classification, as can be seen for the FS and IB Izhikevich regimes that were not trainable for the MNIST dataset but did well on the SHD data. Additionally, IB Izhikevich networks did not always train successfully on the SHD data. In any case, this opens the way to train neuronal network with various regimes, for example to simulate excitatory-inhibitory neuron interactions as is observed in the biology [15].

The networks in this research project were limited to single layer networks with strictly feedforward connections to study the fundamental impact of using more complex neuron models when applying SGL. However, future research may reveal further differences between firing regimes by adding recurrent connections or increase the number of hidden layers, to better match biological neuronal networks [57]. Additional layers and recurrent connections have already been shown to improve performances of SNNs that implement SGL [61]. Therefore, the assumption is not far off that larger networks with recurrent connection reveal more insights into advantages and disadvantages of certain firing regimes.

The results are thus a proof of concept that such biologically plausible homogeneous networks can be trained to perform operations on spike trains with structured variability similar to experimental data, which can in principle be extended beyond classification.

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[2] B. AlKhamissi, M. ElNokrashy, and D. Bernal-Casas, "Deep spiking neural networks with resonate-and-fire neurons," *arXiv preprint arXiv:2109.08234*, 2021.

[3] D. Auge, J. Hille, E. Mueller, and A. Knoll, "A survey of encoding techniques for signal processing in spiking neural networks," *Neural Processing Letters*, vol. 53, no. 6, pp. 4693–4710, 2021.

[4] R. Berner, T. Gross, C. Kuehn, J. Kurths, and S. Yanchuk, "Adaptive dynamical networks," *arXiv preprint arXiv:2304.05652*, 2023.

[5] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," *Frontiers in neurorobotics*, vol. 12, p. 35, 2018.

[6] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 4. Springer, 2006, vol. 4.

[7] L. Bonilla, J. Gautrais, S. Thorpe, and T. Masquelier, "Analyzing time-to-first-spike coding schemes: A theoretical approach," *Frontiers in Neuroscience*, vol. 16, p. 971 937, 2022.

[8] R. Brette, "Computing with neural synchrony," *PLoS computational biology*, vol. 8, no. 6, e1002561, 2012.

[9] R. Brette and W. Gerstner, "Adaptive exponential integrate-and-fire model as an effective description of neuronal activity," *Journal of neurophysiology*, vol. 94, no. 5, pp. 3637–3642, 2005.

[10] A. N. Burkitt, "A review of the integrate-and-fire neuron model: I. homogeneous synaptic input," *Biological cybernetics*, vol. 95, pp. 1–19, 2006.

[11] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, 2020. [Online]. Available: `https://zenkelab.org/resources/spiking-heidelberg-datasets-shd/`.

[12] R. Duarte, B. Zajzon, and T. Schulte to Brinke. "Functional neural architecture github repository (private)." (2023), [Online]. Available: `https://github.com/rcfduarte/func-neurarch/`.

[13] J. K. Eshraghian, M. Ward, E. O. Neftci, *et al.*, "Training spiking neural networks using lessons from deep learning," *Proceedings of the IEEE*, 2023.

[14] C. Frenkel, D. Bol, and G. Indiveri, "Bottom-up and top-down neural processing systems design: Neuromorphic intelligence as the convergence of natural and artificial intelligence," *arXiv preprint arXiv:2106.01288*, 2021.

[15] R. C. Froemke, "Plasticity of cortical excitatory-inhibitory balance," *Annual review of neuroscience*, vol. 38, pp. 195–219, 2015.

[16] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[17] W. Gerstner and R. Brette. "Adaptive exponential integrate-and-fire model." (2009), [Online]. Available: `http://www.scholarpedia.org/article/Adaptive_exponential_integrate-and-fire_model`.

[18] 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.

[19] A. Grimaldi, A. Gruel, C. Besnainou, J.-N. Jérémie, J. Martinet, and L. U. Perrinet, "Precise spiking motifs in neurobiological and neuromorphic data," *Brain Sciences*, vol. 13, no. 1, p. 68, 2022.

[20] A. Grüning and S. M. Bohte, "Spiking neural networks: Principles and challenges.," in *ESANN*, Bruges, 2014.

[21] H. Hagras, A. Pounds-Cornish, M. Colley, V. Callaghan, and G. Clarke, "Evolving spiking neural network controllers for autonomous robots," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, IEEE, vol. 5, 2004, pp. 4620–4626.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[23] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[24] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Artificial intelligence and statistics*, PMLR, 2016, pp. 240–248.

[25] N. Ketkar and N. Ketkar, "Introduction to keras," *Deep learning with python: a hands-on introduction*, pp. 97–111, 2017.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] V. Koren, G. Bondanelli, and S. Panzeri, "Computational methods to study information processing in neural circuits," *Computational and Structural Biotechnology Journal*, 2023.

[28] L. Lapicque, "Recherches quantitatives sur l'excitation electrique des nerfs," *J Physiol Paris*, vol. 9, pp. 620–635, 1907.

[29] S. B. Laughlin, "Energy as a constraint on the coding and processing of sensory information," *Current opinion in neurobiology*, vol. 11, no. 4, pp. 475–480, 2001.

[30] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[31] 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.

[32] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The journal of machine learning research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[33] Y. Mochizuki, T. Onaga, H. Shimazaki, *et al.*, "Similarity in neuronal firing regimes across mammalian species," *Journal of Neuroscience*, vol. 36, no. 21, pp. 5736–5747, 2016.

[34] R. Naud, N. Marcille, C. Clopath, and W. Gerstner, "Firing patterns in the adaptive exponential integrate-and-fire model," *Biological cybernetics*, vol. 99, pp. 335–347, 2008.

[35] 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.

[36] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs," *Current opinion in neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[37] S. Ostojic, "Interspike interval distributions of spiking neurons driven by fluctuating inputs," *Journal of neurophysiology*, vol. 106, no. 1, pp. 361–373, 2011.

[38] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[39] H. Paugam-Moisy and S. M. Bohte, "Computing with spiking neuron networks.," *Handbook of natural computing*, vol. 1, pp. 1–47, 2012.

[40] A. Payeur, J. Guerguiev, F. Zenke, B. A. Richards, and R. Naud, "Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits," *Nature neuroscience*, vol. 24, no. 7, pp. 1010–1019, 2021.

[41] L. Perrinet, "On efficient sparse spike coding schemes for learning natural scenes in the primary visual cortex," *BMC Neuroscience*, vol. 8, no. 2, pp. 1–1, 2007.

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

[43] A. Pitti, M. Quoy, S. Boucenna, and C. Lavandier, "Brain-inspired model for early vocal learning and correspondence matching using free-energy optimization," *PLoS Computational Biology*, vol. 17, no. 2, e1008566, 2021.

[44] L. Ramlow and B. Lindner, "Interspike interval correlations in neuron models with adaptation and correlated noise," *PLoS computational biology*, vol. 17, no. 8, e1009261, 2021.

[45] D. Roggen, S. Hofmann, Y. Thoma, and D. Floreano, "Hardware spiking neural network with run-time reconfigurable connectivity in an autonomous robot," in *NASA/DoD Conference on Evolvable Hardware, 2003. Proceedings.*, IEEE, 2003, pp. 189–198.

[46] J. Rossbroich, J. Gygax, and F. Zenke, "Fluctuation-driven initialization for spiking neural network training," *Neuromorphic Computing and Engineering*, vol. 2, no. 4, p. 044 016, 2022.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[48] S. Shinomoto, K. Shima, and J. Tanji, "Differences in spiking patterns among cortical neurons," *Neural computation*, vol. 15, no. 12, pp. 2823–2842, 2003.

[49] S. R. Shomali, S. N. Rasuli, M. N. Ahmadabadi, and H. Shimazaki, "Uncovering hidden network architecture from spiking activities using an exact statistical input-output relation of neurons," *Communications Biology*, vol. 6, no. 1, p. 169, 2023.

[50] R. A. Silver, "Neuronal arithmetic," *Nature Reviews Neuroscience*, vol. 11, no. 7, pp. 474–489, 2010.

[51] M. Van Gerven, "Computational foundations of natural intelligence," *Frontiers in computational neuroscience*, p. 112, 2017.

[52] A. Viale, A. Marchisio, M. Martina, G. Masera, and M. Shafique, "Carsnn: An efficient spiking neural network for event-based autonomous cars on the loihi neuromorphic research processor," in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, pp. 1–10.

[53] A. Viale, A. Marchisio, M. Martina, G. Masera, and M. Shafique, "Lanesnns: Spiking neural networks for lane detection on the loihi neuromorphic processor," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022, pp. 79–86.

[54] X. Wang, X. Lin, and X. Dang, "Supervised learning in spiking neural networks: A review of algorithms and evaluations," *Neural Networks*, vol. 125, pp. 258–280, 2020.

[55] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, "Spiking neural networks and their applications: A review," *Brain Sciences*, vol. 12, no. 7, p. 863, 2022.

[56] A. Yanguas-Gil, "Coarse scale representation of spiking neural networks: Backpropagation through spikes and application to neuromorphic hardware," in *International Conference on Neuromorphic Systems 2020*, 2020, pp. 1–7.

[57] D. Zambrano, R. Nusselder, H. S. Scholte, and S. M. Bohté, "Sparse computation in adaptive spiking neural networks," *Frontiers in neuroscience*, vol. 12, p. 987, 2019.

[58] F. Zenke, S. M. Bohté, C. Clopath, *et al.*, "Visualizing a joint future of neuroscience and neuromorphic engineering," *Neuron*, vol. 109, no. 4, pp. 571–575, 2021.

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

[60] F. Zenke and M. Halvagal. "Spytorch - a tutorial on surrogate gradient learning in spiking neural networks." (2019), [Online]. Available: `https://github.com/fzenke/spytorch/`.

[61] F. Zenke and T. P. Vogels, "The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks," *Neural Computation*, vol. 33, no. 4, pp. 899–925, 2021.

Appendix A

A.1 Supplementary results of MNIST and SHD experiments

The following sections cover the results that were excluded from the main body of the thesis report for the sake of brevity. The excluded parts contain the figures and detailed descriptions of the MNIST and SHD Optuna hyperparameter optimization studies (Sec. 3.1.1 and Sec. 3.2.1 respectively).

A.1.1 MNIST

Hyperparameter optimization

The results described here exclude the description of the LIF optimization study which has already been explained in detail in Sec. 3.1.1 as an example description.

Izhikevich The Optuna study that used RS-Izhikevich SNNs (Fig. A.1) reached peak network performances of $acc_{train} \approx 98.01\%$ and $acc_{test} \approx 96.68\%$. The mean absolute difference between the train and test accuracies of those trials for which $acc_{test} \geq 20\%$ was around 1.63%. We made the following observations about the hyperparameters: overall, the Izhikevich SNNs were not as robust to hyperparameter changes as the LIF networks. Nevertheless, the Izhikevich networks did reach high accuracy values along the full range of the HL size, batch size, L2 penalty and the

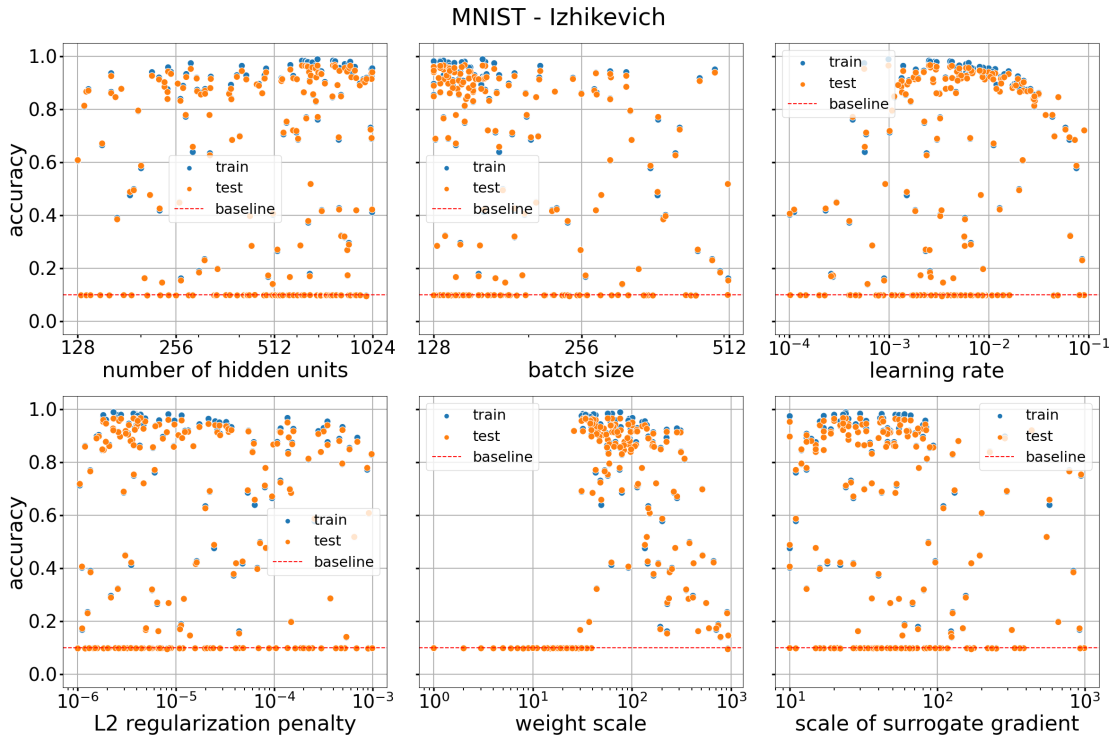

Figure A.1: Optuna study similar to Fig. 3.2, except that here **Izhikevich**-SNNs were trained on the **MNIST** dataset. Peak accuracies reached $acc_{train} \approx 98.01\%$ and $acc_{test} \approx 96.68\%$ for the train and test set respectively.

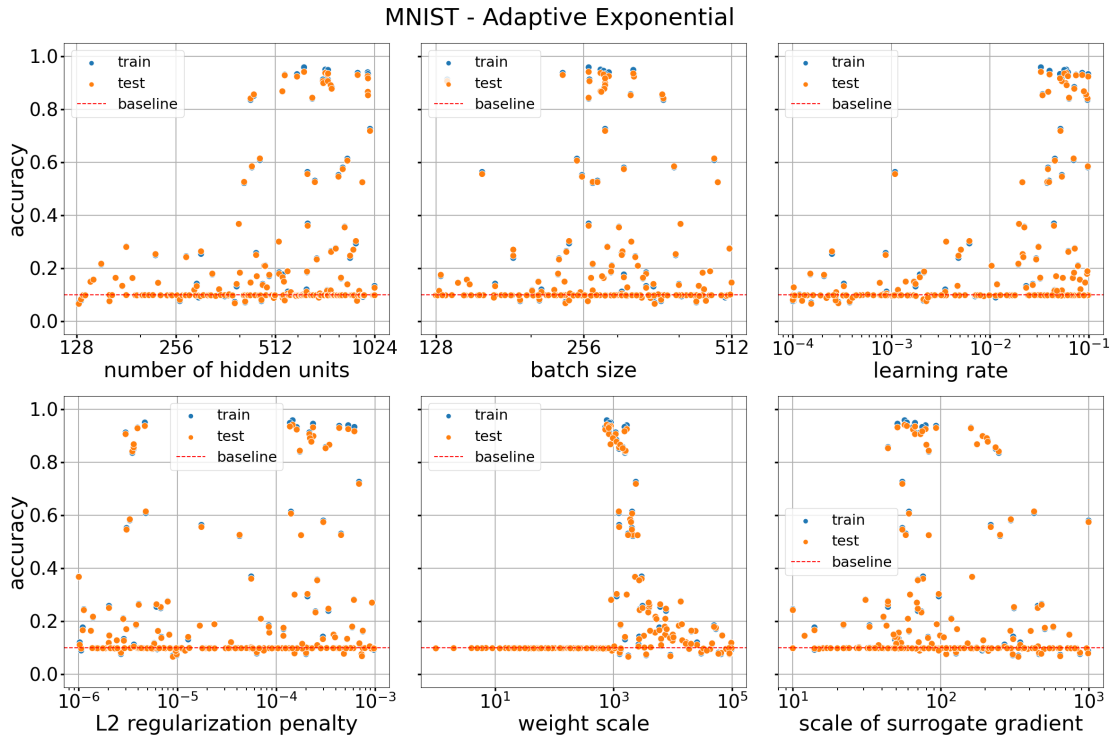

Figure A.2: Optuna study similar to figure 3.2, except that here **AdEx**-SNNs were trained on the **MNIST** dataset. Peak accuracies reached $acc_{train} \approx 95.87\%$ and $acc_{test} \approx 94.13\%$ for the train and test set respectively.

SG scale. However, Optuna favored large values for the HL size and smaller values for the batch size, L2 penalty and the SG scale. Again, if we look at the learning rate sub-plot, values of $10^{-3} \leq lr \geq 10^{-2}$ did result in peak accuracy networks whereas networks outside that range did visibly decrease in network performance. For the Izhikevich SNNs the weight scale was the most influential as $W_s \leq 25$ did lead to chance-level accuracies in all trials. Weight scales of $W_s > 300$ did result in a decrease in accuracies as well. The weight scale sub-plot indicates that almost all chance-level accuracies in all the sub-plots were mainly caused by a too low initial weight scale. Networks that resulted in accuracy values between chance-level and peak accuracies can be attributed to either a sub-optimal learning rate or the pruning and early stopping of trials.

AdEx The Optuna study that used TO-AdEx SNNs (Fig. A.2) reached peak network performances of $acc_{train} \approx 95.87\%$ and $acc_{test} \approx 94.13\%$. The mean absolute difference between the train and test accuracies of those trials for which $acc_{test} \geq 20\%$ was around 0.59%. We made the following observations about the hyperparameters: due to the fact that the majority of the trials only reached accuracies below 40%, there was but one hyperparameter that guaranteed robust and good training of a network. Only the L2 penalty sub-plot shows trials that reached peak accuracies with low and high values. All other hyperparameters indicate clear ranges in which AdEx SNNs learned better than for other values. Again, when inspecting the data and the figure, the weight scale played the most important role for learning. Here, only weight scales in the rough range of $[750, 1500]$ led to network models with peak

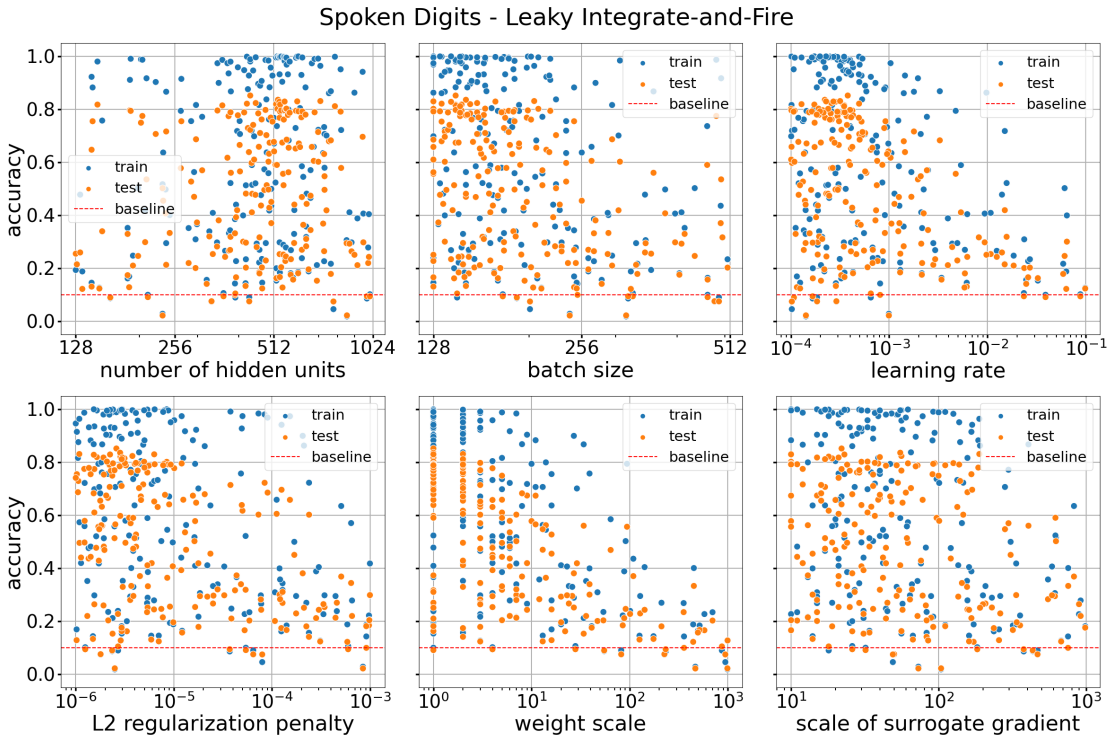

Figure A.3: Optuna study similar to Fig. 3.2, except that here **LIF**-SNNs were trained on the **Spoken Digits** dataset. Peak accuracies reached $acc_{train} \approx 98.89\%$ and $acc_{test} \approx 85.21\%$ for the train and test set respectively.

accuracies. Other weight scale values resulted in chance-level networks or them being close to it. Generally, large values for the HL size, learning rate and values in the medium range for the batch size and SG scale are shown by Optuna to result in peak accuracy networks.

A.1.2 Heidelberg Spoken Digits

Hyperparameter optimization

The results reported here supplement the summarized results in Sec. 3.2.1.

LIF The Optuna study that used LIF SNNs (Fig. A.3) reached peak network performances of $acc_{train} \approx 98.89\%$ and $acc_{test} \approx 85.21\%$. The mean absolute difference between the train and test accuracies of those trials for which $acc_{test} \geq 20\%$ was around 14.22%. We made the following observations about the hyperparameters: the results of LIF SNNs trained on the SHD data do indicate that the networks learned well with small and large values for the HL size and the SG scale. These result are similar to the LIF Optuna study from the MNIST dataset (Fig. 3.2). However, the results also indicate that the LIF SNNs only learned well with a low batch size, learning rate, L2 penalty and weight scale. Furthermore, the results show that training on the SHD data was more difficult because there were more trials whose accuracies ranged between the peak and chance-level ones. Finally, the differences between the train and test accuracies were quite higher, compared to the results of the MNIST-LIF Optuna study (Fig. 3.2).

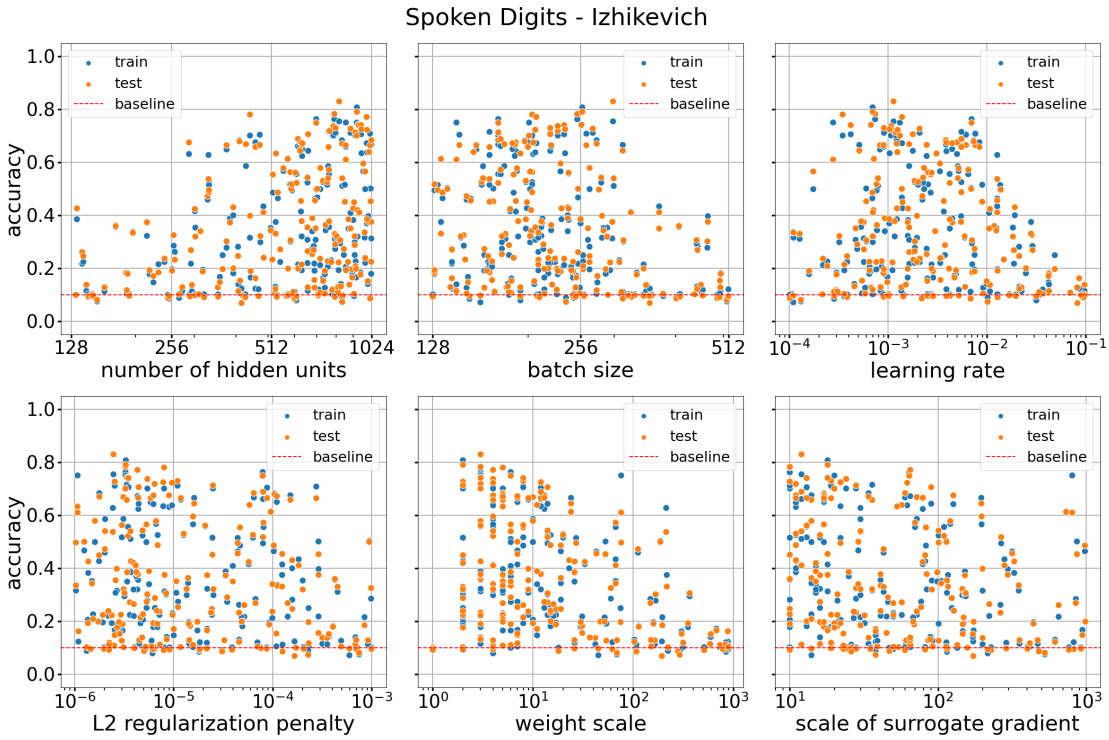

Figure A.4: Optuna study similar to Fig. 3.2, except that here **Izhikevich**-SNNs were trained on the **Spoken Digits** dataset. Peak accuracies reached $acc_{train} \approx 75.45\%$ and $acc_{test} \approx 83.00\%$ for the train and test set respectively.

Izhikevich The Optuna study that used RS-Izhikevich SNNs (Fig. A.4) reached peak network performances of $acc_{train} \approx 75.45\%$ and $acc_{test} \approx 83.0\%$. The mean absolute difference between the train and test accuracies of those trials for which $acc_{test} \geq 20\%$ was around 03.14%. We made the following observations about the hyperparameters: the results do not indicate that the Izhikevich SNNs were invariant to any of the search space ranges of the hyperparameters. All of the hyperparameters did influence the network's learning capabilities either weakly or strongly. Here, larger values for the HL size, medium ranged values for the batch size and learning rate, as well as lower values for the L2 penalty, weight scale (except 1) and the SG scale seemed to have caused peak performances. Surprisingly, some of the trials returned higher classification accuracy on the test set than on the train set. That is uncommon because the test set should be more difficult and novel to classify compared to the train set.

AdEx The Optuna study that used TO-AdEx SNNs (Fig. A.5) reached peak network performances of $acc_{train} \approx 62.08\%$ and $acc_{test} \approx 65.52\%$. The mean absolute difference between the train and test accuracies of those trials for which $acc_{test} \geq 20\%$ was around 3.37%. We made the following observations about the hyperparameters: the results show that the training of those networks was more sensitive compared to the MNIST-AdEx SNNs or the SHD-Izhikevich SNNs. The AdEx results further indicate that the model was slightly invariant to changes of the L2 penalty. Concluding, the results indicate that AdEx SNNs required large HL sizes, a medium valued learning rates, small batch sizes and SG scales, as well as a very narrow range for

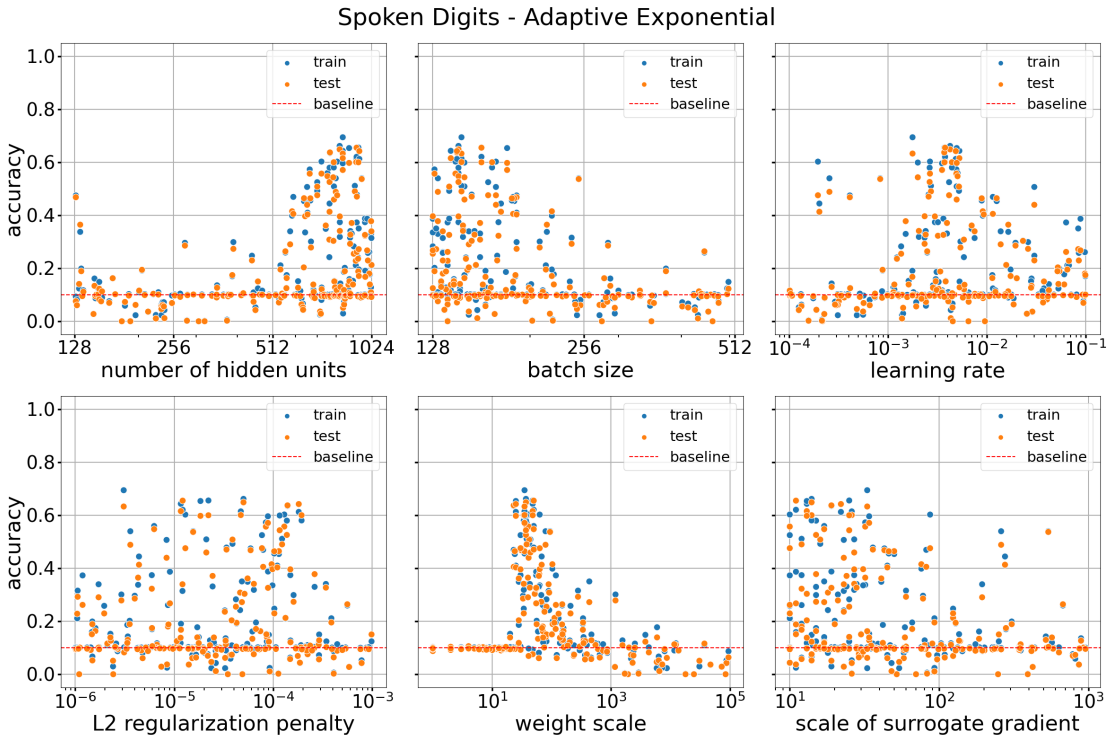

Figure A.5: Optuna study similar to Fig. 3.2, except that here **AdEx**-SNNs were trained on the **Spoken Digits** dataset. Peak accuracies reached $acc_{train} \approx 62.08\%$ and $acc_{test} \approx 65.52\%$ for the train and test set respectively.

the initial weight scales. The networks were also very sensitive to the weight scale which should be set around 35. Similar to the results from the Izhikevich SNNs above (Fig. A.4), the classification accuracies of AdEx SNNs for the train and test set were very close to one another, considerably closer than the accuracy differences from the LIF SNNs (Fig. A.3).

A.2 Random Manifolds Dataset

As a third way of comparing the Izhikevich and AdEx SNNs to the LIF SNNs, next to the MNIST and SHD experiments (Sec. 3.1 and Sec. 3.2 respectively), we wanted to train Izhikevich and AdEx network models on synthetic data consisting of smooth Random Manifolds (RandMan) [61].

Zenke *et al.* [61] created this "synthetic classification data set" as another benchmark to test the classification abilities of LIF SNNs using SGL [61]. Also, see https://github.com/fzenke/randman for their code that we adapted to generate the data.

The goal for this third dataset was to have a more difficult dataset than the MNIST and SHD datasets. A requirement here was, that the data is easy to overfit on and thus difficult to generalize on. We explored various configurations to simulate data that suited our needs.

We excluded the results of the experiments on this dataset from the main corpus of the thesis because the results did not return any additional valuable insights. Still, the results of the regime comparisons are reported here for completeness. We skipped the reporting of detailed Optuna results and neuron statistics because the outcome of the first two experiments can also be derived by studying the results from the regime comparisons (Sec.A.2.3).

Configuration of the RandMan dataset

The simulation of the RandMan data was configured as follows: the amount of input neurons was set to 20 neurons and the amount of classes was set to 10 to match the number of classes of the MNIST and SHD experiments. We used a manifold dimension value of 1 with $\alpha = 1$ and $fr = 5Hz$. With the comparatively small input size (to MNIST and SHD), the size of the train and test set were also chosen to be smaller with 800 and 100 samples respectively. In hindsight, more samples could have let to improved generalization capabilities of the Izhikevich and AdEx models, but would have likely let to more overfitting of at least the LIF as well. For more information on how to generate the data, we refer to the GitHub link above from Zenke *et al.* [61].

A.2.1 Hyperparameter Tuning

In summary, the hyperparameter search space allowed the LIF-SNNs to reach high training accuracies, whereas the Izhikevich and AdEx did fail to reach train accuracies of $acc_{train} \geq 60\%$. However, the LIF networks did seem to overfit because the test accuracies were on average 28.6% lower than the train accuracy, only reaching a test accuracy of $acc_{test} \approx 59.8\%$ in the best trial. Except for the weight scale and learning rate, LIF SNNs were less dependent on specific hyperparameter values to reach their respective peak performances. The Izhikevich and AdEx networks did require a very narrow range for the weight scale to be able to learn any representations which led to Optuna loosing a considerable number of trials to chance-level performances. In turn, that lead to less informative results for the other hyperparameters.

Dataset	Model	Hyperparameter					
		learning rate	weight scale	SG scale	batch size	HL size	L2 penalty
RandMan	LIF	0.01	3	30	150	800	10^{-6}
	Izh.	0.017	125	150			
	AdEx	0.01	1700	70			

Table A.1: Chosen hyperparameter for the RandMan and neuron model pairs. The abbreviations *scale of SG* and *HL size* refer to the scale of the surrogate gradient during the backpropagation step and the size of or number of neurons in the hidden layer respectively. The weight scale refers to the scaling of the network weights and applies to all weight matrices.

The chosen hyperparameter values to train the SNNs on in the subsequent experiments are reported in Tab. A.1.

A.2.2 Firing Regime Drift

All networks trained for 50 epochs. An inspection of the classification losses shows (Fig. A.6) that the training of the LIF and Izhikevich networks converged because their train loss did not further improve. However, for both networks the test loss did not improve much after the first couple of epochs. The 50 epochs of training were too short for the AdEx network to converge, their train loss was still improving (decreasing) slightly. Interestingly, the AdEx's test losses followed the train losses very closely until the last epoch. Thus, if the AdEx SNN would train for longer, higher train and test accuracies than shown below in Sec. A.2.3 would be likely.

Fig. A.7 shows the neuronal metrics of the LIF SNN. The figures for the Izhikevich and the AdEx were omitted because there was no visible change in any of their distributions. From the figure we can see that the LIF stays in a sparse spiking regime as was similarly observed in Sec. 3.1.2 and Sec. 3.2.2.

A.2.3 Firing Regime Comparisons

Fig. A.8 shows that the LIF SNNs did overfit on the train set and had problems generalizing to the unseen test data. In general, the Izhikevich and the AdEx networks reached bad performances, with the best Izhikevich networks (RS and IB regimes) reaching train accuracies close to the LIF's test accuracies.

In Fig. A.9 the previously observed trend is again visible that generalization capabilities are worse for the LIF and better for the Izhikevich and AdEx. However, that observation is somewhat flawed because the performances of the Izhikevich and AdEx were not very good, leaving a smaller margin between train and test accuracies. It makes sense that the closer the network are overall to chance-level performance the smaller the difference between train and test accuracy.

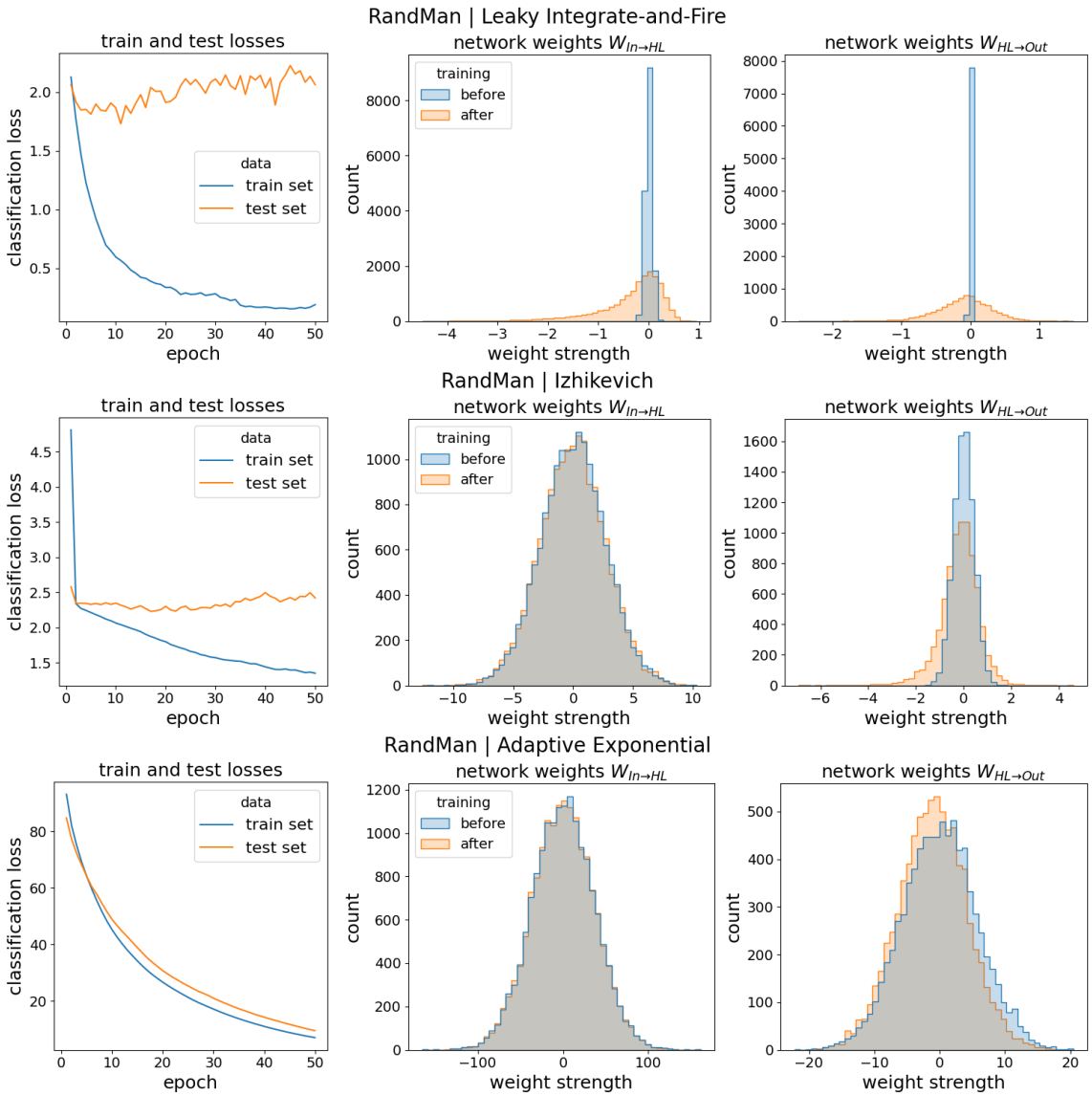

Figure A.6: RandMan classification losses and weight distribution of LIF, Izhikevich and AdEx SNNs (top to bottom). For a detailed description of the figure panels, see Fig. 3.3.

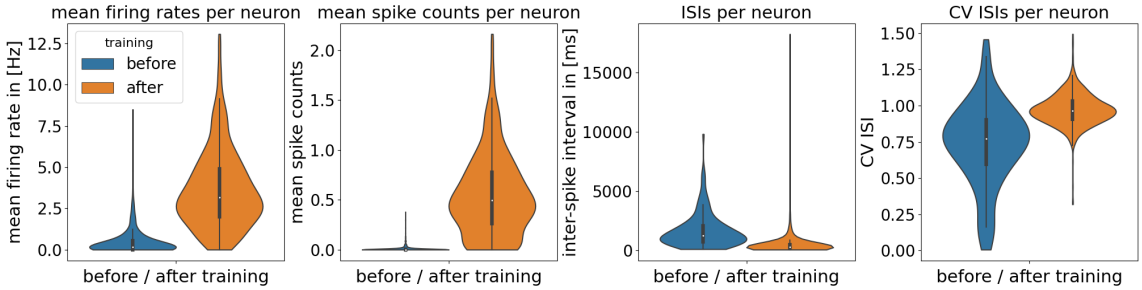

Figure A.7: Network and neuronal statistics of an Izhikevich network trained on RandMan data. The trial reached a train and test accuracy of 96.79% and 56.90% respectively. Samples without spikes before and after training: 2% and 0% respectively. For a detailed description of the figure panels, see Fig. 3.3.

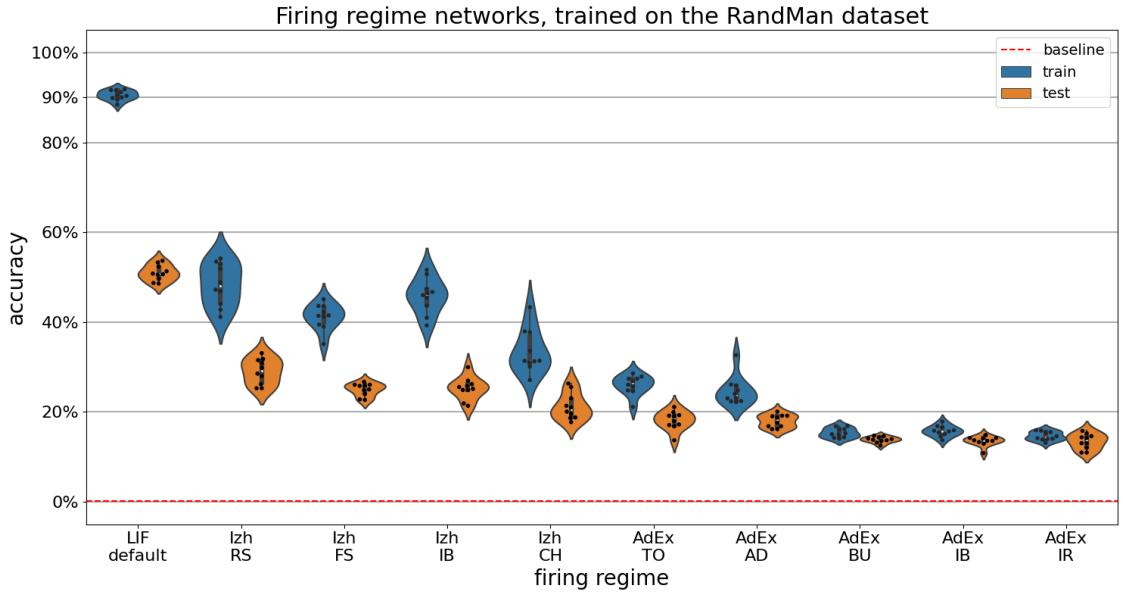

Figure A.8: Test accuracies of various firing regimes, trained on RandMan data. Each regime was trained 10 times.

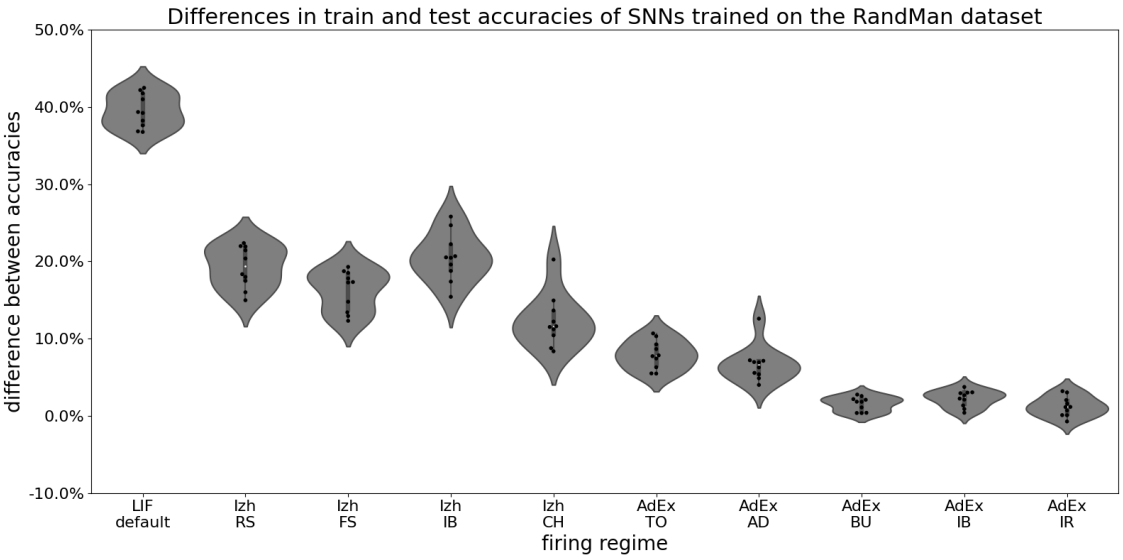

Figure A.9: Test accuracies of various firing regimes, trained on RandMan data. Each regime was trained 10 times.