University of Nottingham
UK | CHINA | MALAYSIA

# Adversarial Robustness of Rate - Encoded Spiking Neural Networks

Thesis submitted to the University of Nottingham for the degree of Master of Science in Statistics, **2023/24**

**Timileyin Folaranmi**

Supervised by

**Dr. Stephen Coombes**

I have read and understood the School and University guidelines on plagiarism. I confirm that this work is my own, apart from the acknowledged references.

# Abstract

Constructing neural networks robust to deception has recently been a topic of growing interest as the applications of neural networks have become more mission-critical. Spiking Neural Networks (SNNs) are a promising family of neural network architectures which offer bio-realism as well as increasingly accurate models capable of various applications, including image classification. Recent work explores the adversarial robustness of SNNs through comparing encoding schemes [33], or looking at the effect of structural parameters on adversarial robustness [61]. This work aims to build on existing work by studying how the type of neurons used to build the network affects robustness to adversarial examples. We compare how SNNs constructed exclusively using three different neuron types: the Leaky-Integrate-and-Fire (LIF), Quadratic-Integrate-and-Fire (QIF), and Izhikevich neuron, fare against each other when presented with adversarial examples crafted from the MNIST [11] dataset. To do so, two white-box attacks: Projected Gradient Descent (PGD), Fast Gradient Sign Method (FGSM) and one black-box attack, Pixle, is used. Our analysis finds that in the white-box setting, models built from LIF neurons are generally more robust when presented with adversarial examples, relative to QIF and Izhikevich-based models, whether or not the attack is targeted. In the black-box setting, QIF and Izhikevich-based models are relatively more robust, across targeted and non-targeted attacks.

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

Inspired by the brain, Artificial Neural Networks (ANNs) are designed to imitate and enhance human task performance across a plethora of domains. Such tasks include pattern recognition (see for example [7]). The area of artificial *spiking* neural networks brings to life the neuro-biological components of artificial neuron computation. Communication of real biological neurons occurs via electrical pulses, or *spikes*. Throughout the 1990s, more studies demonstrated that real neurons could produce precisely timed spikes (an example is [9]). This was a key influence in Maass' writing of his seminal paper [44] on 'the third generation of neural networks', setting the foundations for the workings of spiking neurons.

Neural networks have undergone significant evolution since their inception. Despite the various advances made, traditional ANNs are limited primarily by their ability to process temporal information efficiently. Due to the characteristics of SNNs communicating via binary signals (i.e. spikes), computationally demanding matrix multiplication operations are not needed,

leading to gains in energy efficiency.

The applications of SNNs are being considered for autonomous cars, drones, and robotics. Having such a mission-critical nature, there is an important need to improve the robustness of these networks against adversarial attacks. Adversarial attacks [20, 51] are generated through the injection of carefully - crafted perturbations to a *clean* input, to deceive the model into producing incorrect outputs with a high probability. It is important to note that the perturbation is typically small enough to be imperceptible to the human eye.

Numerous studies have explored the extent to which traditional ANNs are susceptible to adversarial attacks [68]. This vulnerability concern extends to SNNs, as effective adversarial attacks are architecture agnostic [61]. There has recently been growing interest in achieving adversarial robustness with SNNs, and this remains a pertinent open problem [14, 36].

## 1.2 Objectives

This thesis is primarily concerned with the exploration of the robustness of Spiking Neural Networks under adversarial attack. An analysis is done by comparing SNNs built exclusively using different neuron types. These neurons are the Leaky Integrate and Fire (LIF), Quadratic Integrate and Fire (QIF) and Izhikevich neurons. The *adversarial accuracy*, which is the accuracy of the model on a set of images once perturbations have been introduced, is compared across the models when varying the intensity of an attack. Three main attack methods are deployed: Projected Gradient Descent (PGD) [45], the Pixle method [55], and Fast Gradient Sign Method (FGSM) [20]. Using these attack schemes, the effect of time steps will also be compared across networks constructed from the three neurons. To this

end, the key contribution of this thesis is the exploration of the robustness to adversarial examples of networks constructed on QIF and Izhikevich neurons, compared with LIF neuron-based SNNs.

## 1.3 Thesis Structure

The structure of the thesis can be summarised as follows. In Chapter 2, an introduction is provided to biological neurons and artificial networks. Artificial Neural Networks are bio-inspired, and thus networks of neurons can be created. By providing background on its workings, we then introduce Spiking Neural Networks as a step further in bio-inspired network construction.

In Section 2.3, we introduce the key neuron models this thesis will be based on: the LIF, QIF, and Izhikevich neurons, inspired by the Hodgkin-Huxley model. After establishing the various neuron types, an in-depth exploration of *neural encoding* is explored in Section 2.4. Various encoding schemes are explored. Following this section, the practical process of training SNNs is provided, giving background and context to SNN-based methods of learning. The important problem of the *dead-neuron* problem is introduced and its solution, the use of a surrogate gradient, is explained.

Following an introduction to basic Spiking Neural Network architecture, the idea of adversarial robustness is then introduced in Chapter 3. This chapter begins with a background on a key application of SNNs: computer vision. Following this, the mathematical foundations and concepts underpinning adversarial attacks are then outlined, followed by background provided on the main attacks used in this analysis: PGD, Pixle method, and FGSM. Section 3.5 details the experiments conducted in the thesis, beginning with the motivation and experiment set-up. We find in the first experiment that

networks built from LIF neurons generally tend to be more robust in white-box settings, relative to QIF- and Izhkieviech-based networks, whether or not the attack is targeted. However, the LIF neuron-based network is less strong in response to the FGSM attack. This sentiment reverses when attacks are black-box and LIF-based networks yield greater accuracy degradation relative to QIF- and Izhikevich-based networks.

Chapter 4 follows on by providing a discussion on the results obtained, and possible explanations for the results. The current landscape of the state of SNN research on adversarial robustness is then explored in Section 4.2.

Finally, the final chapter of this thesis presents the conclusions of the study. In particular, it explores the limitations of the approach utilised in the study, including computational constraints and experiment design. Suggestions for directions for further work are also presented, including providing rigorous explanations for why some networks are more or less robust in black-box/white-box and targeted/non-targeted attack scenarios.

# Chapter 2

# Background

## 2.1 Networks of Neurons

### 2.1.1 Biological Neurons

In the brain, neurons propagate electrochemical signals through events called action potentials. To understand action potentials, the idea of *membrane potentials* must first be understood. The membrane potential is the result of the distribution of the ions between the inside and outside of a cell, across a membrane. Action potentials are fast, temporary changes in this electrical membrane potential of a cell, and these are crucial events concerning the communication of neurons. Neurons consist of four main parts: synapses, dendrites, somas, and axons [54]. The role of synapses is to form connections between neurons. Synapses lie on dendrites, which handle the direct input to neurons. Dendrites are tasked with translating chemical signals into electrical signals. The soma is the cell body, where the membrane potentials propagated from synaptic inputs are integrated. Here, it will be determined whether the post-synaptic cell fires action potentials

before being transmitted to the axon, which propagates action potential toward other nerve cells. A diagram of a neuron's structure is presented in Figure 2.1. In the following section, we outline the translation of biological neuron models to artificial models.

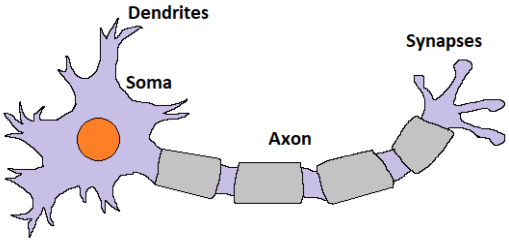


Figure 2.1: Biological neuron's structure. [54]

### 2.1.2 Artificial Neural Networks

Rate-based neurons model the activity of a neuron only by its firing rate, $\mathbf{r}$, irrespective of any changes in its membrane potential or times when the neuron spikes. The first such rate-coded artificial neuron, known as the formal neuron, or threshold logic unit, was first proposed by [47]. This work then inspired the perceptron, introduced by [57]. Rosenblatt utilised the Heaviside step function given by Equation 2.1, as the activation function.

$$H(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{2} & x = 0 \\ 1 & x > 0 \end{cases} \tag{2.1}$$

First-generation neurons constructed in this way would fire binary signals at the point when the sum of incoming signals reaches a threshold of the neuron. Continuous activation functions, such as the sigmoid [24], or hyperbolic tangent function, are inspired by this. Such activation functions could work with analog inputs and outputs, leading to the training of the

neural network through a prominent backpropagation algorithm based on gradient descent. Generally, the discrete-time firing rate model is formulated as Equation 2.2 [54]:

$$\mathbf{r} = f(\mathbf{W}\mathbf{u} + \mathbf{b}) \tag{2.2}$$

where $\mathbf{u} \in \mathbb{R}^{N_{pre}}$ is the firing rate of pre-synaptic neurons, $\mathbf{r} \in \mathbb{R}^{N_{\text{post}}}$ is the firing rate of post-synaptic neurons, $\mathbf{W} \in \mathbb{R}^{N_{\text{post}} \times N_{\text{pre}}}$ is the weight matrix which represents the synaptic strength between pre- and post-synaptic neurons, $\mathbf{b} \in \mathbb{R}^{N_{\text{post}}}$ is the bias, and $f(\cdot)$ is the non-linear activation function.

At present, the Rectified Linear Unit (ReLU) [49], and similar variations are utilised as activation functions due to their strong convergence performance, compared with the sigmoidal activation function [35]. Grouping such rate-based neurons is commonly referred to as a fully connected layer. Modern neural network architectures typically stack variants of this layer, building deep networks of neurons, commonly referred to as deep neural networks (DNNs). A neural network is deep when it has at least two hidden layers performing non-linear transformations of the input data.

In the construction of a DNN, an important, commonly used building block is a convolutional layer. A convolutional layer is a type of fully connected layer. The convolutional layer utilises the sharing of weights applied to the input data, best used for processing data with a grid-like topology, such as images. Due to this property, convolutional neural networks (CNNs) [42, 41] can understand spatial relationships in the input data more easily. The representational properties of these layers in CNNs are similar to the response properties of the neurons in the primary visual cortex (V1).

As such, CNNs have two key properties demonstrating their strong suitability for vision-based tasks: (i) spatially shared weights, and (ii) spatial

pooling. Spatially shared weights allow for the property of shift-invariance to be present in the network, allowing it to learn shift-invariant features. Spatial pooling allows for the increased robustness of the output to slight input shifts and perturbations. In 2012 and onwards, one of the most prominent results in deep learning has been the use of CNNs in achieving an improvement in the ImageNet classification challenge [10, 34]. This technological breakthrough paved the way to various proposals for improvements in network architectures in vision-based models [64, 26, 69]. We now demonstrate how developments in ANNs have laid the foundation for spiking neural network models.

## 2.2 Artificial to Spiking Neural Networks

We introduce this by setting the foundations for neural code. The neural code is concerned with how information is represented in the brain [15]. Numerous theories exist as to what the code is, and the code is yet to be cracked. Across the theories, we now explore three key traits underlying them.

### 2.2.1 Spikes

This characteristic is grounded in the observation that biological neurons interact via spikes, or action potentials. These are electrical impulses of approximately 100 mV in amplitude. Most computational models of neuron behaviour simplify the representation of a spike to a discrete event. Multiplying high-precision activations with high-precision weights requires conversion to integers, and decomposition of multiplication into numerous additions, which can be a computationally expensive endeavour. Mean-

while, utilising a spike-based approach only requires the weight to be multiplied by a spike, or a 1.

### 2.2.2 Sparsity

Most of the time, biological neurons spend their time at rest. Therefore, most of the activations would be zero, at any given time. Having sparse tensors eases the computational burden, as these are far easier to store. The space that a basic data structure requires to store a matrix increases with the number of entries to store. On the other hand, a data structure that stores a sparse matrix would only consume memory with the number of nonzero elements. Take the following list as an example:

$$[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1].$$

Here, as most of the entries are 0, writing out only the non-zero elements leads to run-length encoding (indexing from zero), a far more efficient representation.

### 2.2.3 Static suppression

Also known as event-driven processing, this idea is rooted in the idea that the sensory system is more sensitive to changes in the input than to static input. Suitably, there are numerous features of the sensory periphery which promote neuron excitability when exposed to changing stimuli, simultaneously suppressing its response to unchanging, static information.
The receptive field of a neuron refers to a section of the visual field, or input, where a stimulus would trigger the firing of that particular neuron. In

the visual system, neurons in the primary visual cortex (V1) have receptive fields, which respond to orientations of edges and movement. Analogously, mechanisms in early auditory processing include spectrotemporal receptive fields, where neurons are more sensitive to changing frequencies in the input sound than static frequencies [2].

### 2.2.4 Spiking Neurons

Similar to the artificial neuron model, spiking neurons also operate on a weighted sum of inputs. However, instead of passing the result through a sigmoid or ReLU non-linearity, the weighted sum is integrated into, or contributes to, the membrane potential $u(t)$ of the neuron. If there is sufficient excitement in the neuron by this weighted sum, so that the membrane potential reaches a threshold $\theta$, the neuron emits a spike that propagates to its following connections. However, most of the time, neuronal inputs are spikes of short bursts of electrical activity [15]. Therefore, the likelihood of all input spikes arriving at the neuron body in unison is small, implying the presence of temporal dynamics which sustain the membrane potential over periods of time.

Lapicque quantified these dynamics in 1907 [38] through the stimulation of the nerve fibre of a frog leg. Using a current source, Lapicque observed how long it took for the frog leg to twitch based on the amplitude and duration of the driving current $I_{in}$. [8]. He found that a spiking neuron crudely resembles a low-pass filter circuit which has a resistor $R$ and a capacitor $C$. Such a filter allows signals of a lower frequency than a cut-off to pass, whilst attenuating higher signals with frequencies higher than the cut-off. This was later dubbed the *leaky integrate-and-fire (LIF)* neuron. We explore this neuron in depth in the following section. The capacitance

comes from the insulating lipid bilayer which forms the membrane of the neuron. The resistance comes from the gated ion channels that open and close across the membrane [27]. We can model the dynamics of the passive membrane using an RC circuit, represented as

$$\tau_m \frac{du(t)}{dt} = -u(t) + I_{\text{in}}(t)R \tag{2.3}$$

where $\tau_m = RC$ is the time constant of the circuit. Typically, $\tau_m$ lies in the 1-100 ms range. The solution to Equation 2.3 for a constant current input is

$$u(t) = I_{\text{in}}\,(t)R + [u(0) - I_{\text{in}}\,(t)R]\, e^{-\frac{t}{\tau_m}} \tag{2.4}$$

demonstrating how exponential relaxation of $u(t)$ to a steady - state value takes place, where $u(0)$ is the initial membrane potential at $t = 0$. To allow this time-varying solution to be compatible with a sequence-based neural network, the forward Euler method is used, in the simplest case, to obtain an approximate solution to Equation 2.3, consider

$$u(t) = \beta u(t-1) + (1-\beta) I_{\text{in}}\,(t) \tag{2.5}$$

where we have that time is explicitly discretised and $\beta = e^{-1/\tau_m}$ is the decay rate of $u(t)$. Following from derivations in [15], the weighting factor of an input is usually a learnable parameter. Therefore, the coefficient of the input current in 2.5, $(1-\beta)$, is subsumed into a learnable weight $W$, and the simplification $I_{\text{in}}\,(t) = WX(t)$ is made, decomposing the effect of $\beta$ on the input $X(t)$. We have that $X(t)$ is treated as a single input.

Typically, a full-scale neural network would vectorise $X(t)$, and $W$ would be a matrix. However, here, we treat $X(t)$ as a single input to a single neuron for simplicity. When we account for spiking and membrane potential reset, we obtain

$$u(t) = \underbrace{\beta u(t-1)}_{\text{decay}} + \underbrace{WX(t)}_{\text{input}} - \underbrace{S_{\text{out}}\,(t-1)\theta}_{\text{reset}}. \tag{2.6}$$

Here, $S_{\text{out}}\,(t) \in \{0, 1\}$ is the output spike the neuron generates. If the neuron is activated ($S_{\text{out}} = 1$), the reset term subtracts the threshold $\theta$ from the membrane potential. If the neuron is not activated ($S_{\text{out}} = 0$), the reset term does not have any effect.

Spike generation in a neuron occurs if the membrane potential exceeds the threshold

$$S_{\text{out}}\,(t) = \begin{cases} 1 & \text{if } u(t) > \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

In Figure 2.2, a graphical representation of the LIF neuron is provided. A recurrent neuron is unrolled across time steps, where the reset mechanism is given by $S_{\text{out}} - \theta$.

A wide spectrum of neuron models exist, ranging from models that are biophysically accurate, to purely simple artificial neurons which currently proliferate modern deep-learning architectures.

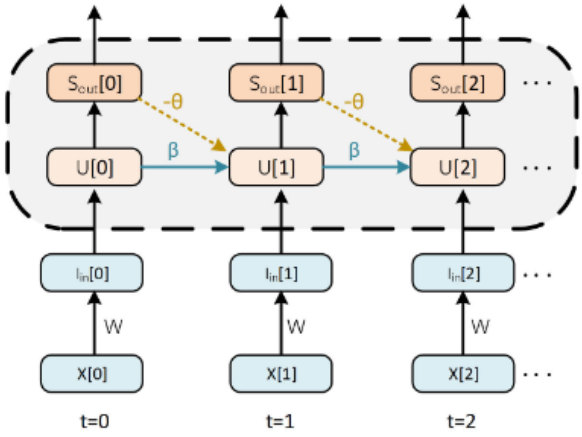


Figure 2.2: Recurrent neuron from [15]. An unrolled computational graph of the neuron, where time flows from left to right. Each column represents the propagation of weighted input current $I_{in}[t] = WX[t]$ at time $t$. Input current is integrated into membrane potential $U[t]$ and spikes if $U[t] > \theta$, where $\theta$ is the threshold.

## 2.3 Neuron Models

### 2.3.1 The variety of neuron models

Variants of spiking neuron models has been proposed, with the major trade-off being between biophysical accuracy and computational feasibility. The intuition behind spiking neurons can be found through exploring the question of *what determines whether a neuron spikes in the first place?* The past century has seen many experiments to this end, many of which have generally concluded that provided a neuron experiences sufficient stimulus, then it may become excited and emit a voltage spike. In this section, numerous models of spiking neurons are presented.

#### Hodgkin-Huxley (HH)

Hodgkin and Huxley experimented on the giant axon of a squid and came to the conclusion that two types of ion channels $K^+$ channel and $Na^+$ channel,

are integral in the formation of action potentials [27]. Even though the Hodgkin-Huxley model bears biological accuracy, various limitations have been pointed out [48, 67], including its intensive demand on computational resources and lack of scalability for large-scale networks.

### Artificial Neuron

A more computationally feasible model is the artificial neuron model, where inputs to neurons are multiplied by their corresponding weights and passed through an activation function. Incredibly, these simple models have led to large-scale accomplishments across the fields of computer vision, natural language processing, and numerous machine learning tasks.

### Leaky Integrate and Fire (LIF)

This variant of the neuron model lies in the middle of the spectrum of realism and practicality. Similar to its artificial counterpart, it is still passed a weighted sum of inputs. The difference is that instead of passing the weighted sum to an activation function, an LIF neuron will integrate the input current over time, with leakage. At a certain point, the integrated current may exceed a threshold, in which case the LIF neuron emits a voltage spike. A simulation of this phenomenon is shown in Figure 2.3.

Concerning modelling, the LIF neuron treats the output spike as a discrete event, not considering the shape and profile of the output spike. As such, information obtained from a neuron is stored not in the spike but in the frequency of spikes. When a pre-synaptic neuron fires, the voltage spike makes its way through the axon of the neuron. This releases neurotransmitters into the synaptic cleft, which is the small gap between the presynaptic

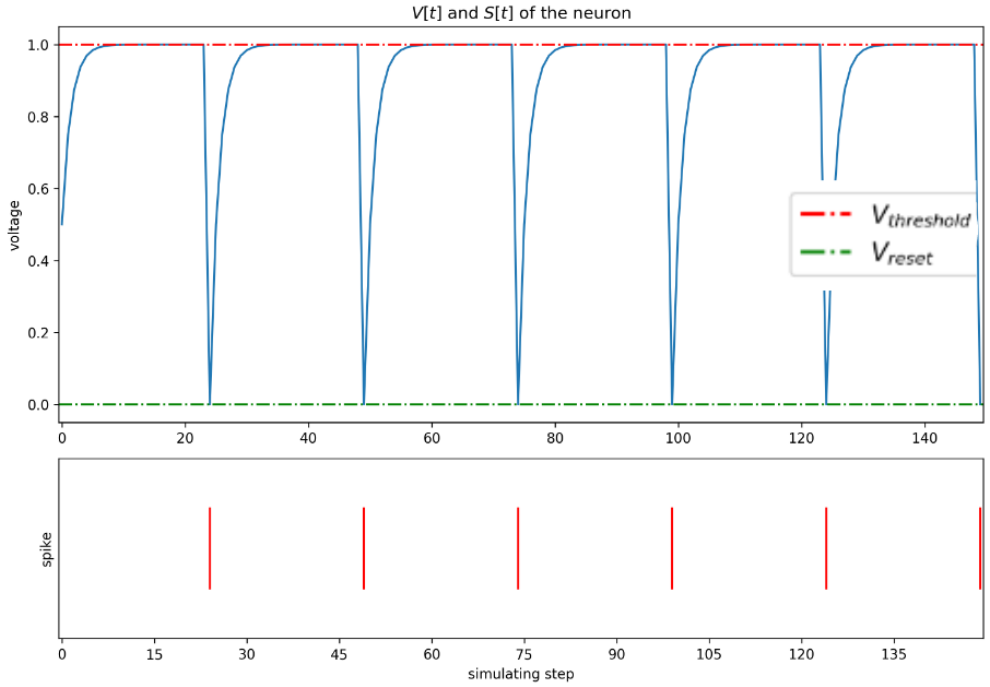


Figure 2.3: Simulation of LIF neuron following injection of input current equal to 1. 150 time steps are simulated. Code for the simulations can be found at: github.com/timif2/snnthesis.

and postsynaptic neurons.

A choice that naturally arises when designing a LIF network is choosing between a current-based (first order) or a conductance-based (second order) model. We can then explore the *synaptic conductance-based* LIF neuron. Up to now, we have assumed that an initial input voltage spike results in an instantaneous jump in synaptic current, which is then integrated into the membrane potential of the neuron. This is the first-order synaptic neuron model. However, in reality, the input spike would lead to a *gradual* release of neurotransmitters from the pre-synaptic neuron which arrive at the post-synaptic neuron. Second-order conductance-based LIF models account for this by incorporating the gradual temporal dynamic nature of the input current. In practice, second-order neurons may be more appropriate if the input data exhibits temporal relationships over long durations, or if the input spike pattern is infrequent. Therefore, first-order neuron models may

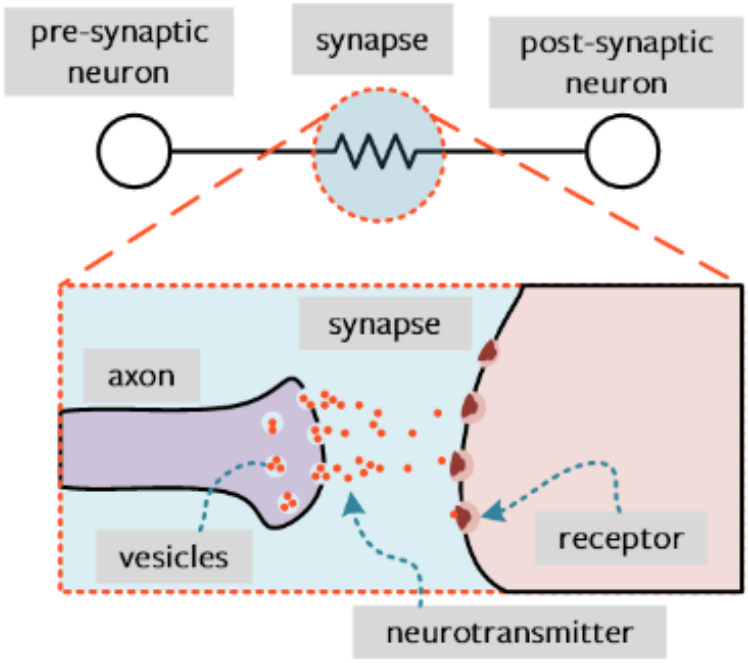


Figure 2.4: Firing of neuron, with the evolution of voltage spike [15]. The input current travels from the pre-synaptic neuron, through the synapse, to the post-synaptic neuron. Within the synapse is a synaptic cleft where information transports through the axon, to the receptor.

be more suitable in settings outside of these, as well as less computational strain due to the backpropagation process being made more simple.

### Quadratic Integrate - and - Fire (QIF)

In general, for a non-linear integrate-and-fire model, we would replace Equation 2.3 by

$$\tau_m \frac{\mathrm{d}}{\mathrm{d}t} u(t) = F(u) + G(u) I_{\mathrm{in}}(t); \tag{2.8}$$

cf. [1]. In the same fashion as before, the dynamics stop once the membrane potential $u$ reaches the threshold $\theta$, and gets reinitialised at $u = u_{\mathrm{rest}}$. We can interpret this general non-linear model by comparing it with Equation 2.3. We see that $G(u)$ may be interpreted as a voltage-dependent input

resistance, and $F(u)/\left(u - u_{\text{rest}}\right)$ corresponds to a voltage-dependent decay constant. In particular, an instance of a non-linear integrate-and-fire model is the *quadratic* model [18, 25, 39].

$$\tau \frac{\mathrm{d}}{\mathrm{d}t} u = a_0 \left(u - u_{\text{rest}}\right)\left(u - u_{\text{c}}\right) + RI_{in}(t). \tag{2.9}$$

.

Here, $a_0 > 0$ and $u_c > u_{\text{rest}}$. When $I_{in} = 0$ (no external current), and $u < u_c$, the voltage (membrane potential) decays to the resting potential $u_{\text{rest}}$. If $u > u_c$, the membrane potential increases so that an action potential occurs. Therefore, $u_c$ can be interpreted as a critical voltage for a spike to occur from an initial short current injection. We can see a simulation of this in Figure 2.5.

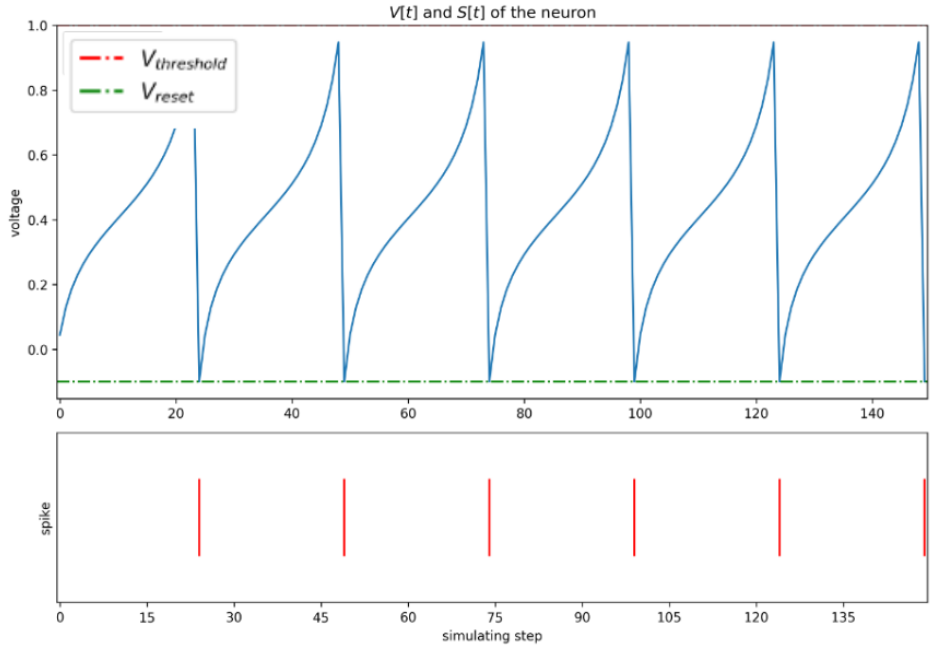


Figure 2.5: Simulation of Quadratic-Integrate-and-Fire neuron following a current injection of 0.2, with a spike train of length 150 timesteps. Code for simulations can be found at: github.com/timif2/snnthesis.

### Izhikevich Neuron

The Izhikevich neuron model was introduced by Eugene Izhikevich in 2003 [30], standing out for its ability to balance biological realism with computational efficiency. Whilst the Hodgkin-Huxley (HH) model previously mentioned is highly accurate, it also carries a high computational burden. This is due to the solution of various non-linear differential equations having to occur, and this level of complexity makes it difficult to simulate large networks of neurons based on the Hodgkin-Huxley model. On the other hand, the LIF model previously mentioned provides a more computationally efficient method of neuron simulation but lacks in a bio-realistic sense, and fails to adequately capture the range of neuronal behaviours that occur in real biological systems. To bridge this gap, the Izhikevich model is introduced, offering a computationally efficient model, as well as being capable of capturing a wide diversity of neuronal dynamics.

Mathematically, the Izhikevich neuron model is defined by a pair of coupled different equations that describe the membrane potential at time $t$ $u(t)$ and a recovery variable $w$:

$$\frac{du(t)}{dt} = 0.04u(t)^2 + 5u(t) + 140 - w + I_{\text{in}}(t) \tag{2.10}$$

$$\frac{dw}{dt} = a(bu(t) - w) \tag{2.11}$$

In these equations, $u(t)$ represents the membrane potential of the neuron, $w$ is a recovery variable accounting for the activation of potassium ($K^+$) and inactivation of sodium ($Na^+$) ion channels, $I_{\text{in}}(t)$ representing the input current to the neuron at time $t$, and $a, b, c, d$ are parameters controlling the

dynamics of the neuron. In addition, the model has a reset condition which applies whenever the membrane potential $u$ exceeds a certain threshold, which is usually 30mV:

$$\text{If } u \geq 30\text{mV, then } \begin{cases} u \leftarrow c \\ w \leftarrow w + d \end{cases}$$

The parameters $c$ and $d$ are used to reset the membrane potential and adjust the recovery variable, respectively. We can see what happens to neurons following a simulated input injection in Figure 2.6.

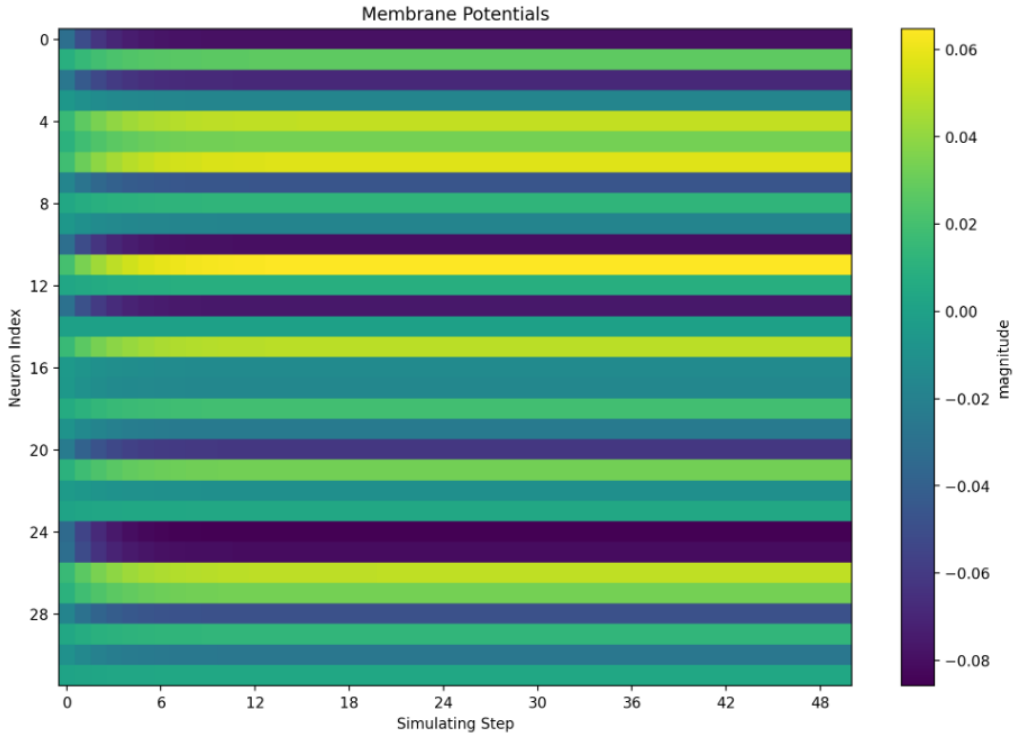


Figure 2.6: Heatmap simulation of membrane potential across 150 time-steps for Izhikevich neurons.

## 2.4 Neural encoding/decoding schemes

In Section 2.1.1, we briefly mention how neural code is rooted in how information is represented in the brain. In this section, we unpack this notion and explore in more depth various neural encoding schemes that are commonly used. Whilst numerous encoding schemes are explored, however, the focus of this thesis is primarily on rate-encoded networks.

When the retina converts photons into spikes, we see light. Odours are experienced when the nose intricately processes volatilised molecules into spikes. Generally, the brain understands the world through spikes. We now explore the question of *how do spikes carry meaning if they are identical?* We probe this question through the lens of two components: (i) input encoding, which is the process of converting input data into spikes, and subsequently passing it onto a neural network and, (ii) output decoding, which is training the output of a network to spike in a way that we can perform meaningful inference.

### 2.4.1 Input Encoding

When providing an SNN with an input, this data need not be converted into spikes. Static data such as images, may be treated as a direct current (dc) input, which has the same features passed to the input layer at every time step. A drawback of this approach is that it does not leverage the benefits of SNNs concerning performing meaningful inference on temporal data. There are three main encoding mechanisms we will now explore.

### 2.4.2 Rate-coded inputs

Rate coding converts the intensity of the input into a spike count or firing rate. This is calculated by finding the average number of spikes, or action potentials, in a specified time interval. However, this naive approach is flawed in the sense that neuronal dynamics typically vary across time. In a more sophisticated fashion, which is also more biologically plausible, spike events can be modelled as Poisson processes with constant average firing rates [12, 74]. Using this approach implies that we care only about the average firing rates, to perform inference.

A basic example is the processing of an image: a bright pixel is encoded into a high-frequency firing rate, as opposed to a dark pixel which would have a low-frequency firing rate. Figure 2.7 demonstrates the rate coding mechanism.

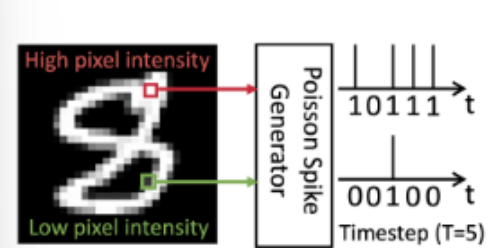


Figure 2.7: Figure of rate coding mechanism from [33]. For each timestep, whether or not the neuron spikes is determined by the pixel intensity. The pixel intensity directly contributes to a Poisson spike generation process.

Hubel and Wiesel presented Nobel prize-winning research on visual processing, with the key findings demonstrating that a brighter input or a favourable orientation of light directly corresponds to a greater firing rate [28]. This thesis project will explore the adversarial robustness of these rate-encoded networks in more depth. To achieve this rate-encoding, the spiking intensities of the input data are converted into spike trains using a Poisson Distribution. The time window $T$ is divided into bins, with a probability $p$ of spiking occurring in each bin. For every bin, there is a 50%

chance of a spike event happening. The number of spikes in $T$ follows a Poisson distribution.

### 2.4.3 Latency-coded inputs

This form of encoding primarily leverages the temporal component of the SNN, but not the multiplicity. Information from a stimulus is encoded as the latency from a particular event until the first spike occurs. The motivation for this type of encoding is inspired by the observation that important sensory stimuli would elicit spikes earlier in upstream neurons. Here, *when* the spike happens carries the information. For instance, a time-to-first-spike mechanism encodes a bright pixel as an early spike, but a dark pixel would spike last, or perhaps never spike. In practice, this encoding method has been used in unsupervised learning [50], and supervised learning methods such as Chronotron [19].

### 2.4.4 Delta-modulated inputs

In principle, delta modulation is rooted in the observation that neurons are stimulated by change. The silicon retina camera highlights this notion: the camera only generates an input provided there has been a sufficient change in input intensity over time. In the case of small amounts of change in your field of view, the photoreceptor cells are far less likely to fire. From an implementation perspective, a network encoded in this way would receive a time-series input, feeding a matrix difference subjected to a threshold into a network. Once this is done, a common condition implemented for a spike to be generated is for the difference to be both *positive* and *greater* than a predefined threshold.

### 2.4.5 Output decoding

Importantly, we now consider decoding the spikes to interpret the firing behaviour of the output neurons. A rate-encoded scheme would choose the output neuron with the highest spike count, or firing rate, as the predicted class. Latency encoding, on the other hand, would select the output neuron that fires first as the predicted class. In general, if we consider a multi-class classification problem, where $N_C$ is the number of classes, a traditional ANN would select the neuron with the largest output activation as the predicted class for the input. However, a rate-encoded SNN chooses the neuron that fires with the highest frequency as the predicted class.

When choosing between rate and latency encoding schemes for an SNN, it is important to consider both advantages and disadvantages. For rate codes, a key strength is error robustness. Suppose a neuron fails to fire. Then, there are many more spikes to ease the burden of this error. Furthermore, the additional spikes of rate-coded schemes provide a stronger gradient signal for learning through error backpropagation. For latency-coded networks, an important advantage is lower power consumption. As this encoded scheme by design communicates fewer spikes, less dynamic power and memory access is used, due to sparsity. In biology, this benefit is also observed, due to the notion that nature optimises for efficiency. In Olshausen and Field's [52] work, they observe that rate-coding can only explain, at most, the activity of 15% of neurons in the primary visual cortex (V1).

### 2.4.6 Objective Functions for Rate Encoding

When evaluating the performance of the network, either the cross-entropy loss or Mean Square Error (MSE) is applied to the spike count or the membrane potential of the output layer of neurons. Given that enough time

has been utilised, passing the spike count through the objective function is more commonly used in practice, as it operates directly on spikes. Using the actual number of neuron spikes directly in calculations yields greater efficiency as spikes are directly observable events. However, in the use of the neuron's membrane potential to estimate or influence the spike count, complications arise as the membrane potential is not directly measurable. In this context, cross-entropy loss aims to discourage neurons from incorrect classes to fire. The sum of the spikes for each neuron in the output layer is accumulated over time to form a spike count $\vec{c} \in \mathbb{N}^{N_C}$, across $N_C$ classes. The softmax function is then applied to the spike counts, which are treated as logits in the final layer. The membrane potential can also be used to form the network predictions. In this case, logits are obtained by taking the maximum value of the membrane potential across all time steps, these are then applied to a softmax cross entropy function.

For the MSE or Mean Square Spike Rate, spike counts for both the correct and incorrect classes are set as targets. Mean square errors between the observed spike counts and these targets are calculated for all output classes and are then summed. In practice, the target spike count is often represented as a proportion of the total number of time steps the neuron fires [63]. In using the membrane potential, each neuron in the output layer is set a target membrane potential for each time step, with losses summed across both time and outputs.

## 2.5 Training Spiking Neural Networks

Using the loss functions explored in the previous section, we now study the utilisation of these loss functions in the learning and training process of the neural network.

### 2.5.1 Learning Rules

After training data has been passed through a network, the output is then used to calculate a loss using a loss function. From here, this loss must be used to update the network parameters (weights and biases) to improve the network at the trained task. The weights take some blame for their contribution to the total loss, and this is known as *credit assignment.* In particular, credit assignment can be decomposed into two components: *spatial* and *temporal* credit assignment problems.

The objective of spatial credit assignment is to find the spatial location in the network, of the weight contributing to the error. Temporal credit assignment, on the other hand, is concerned with finding the time at which the weight contributes to the error. In the following section we explore the backpropagation algorithm which solves the spatial credit assignment problem through the application of a distinct backward pass after a forward pass during the learning process [21]. This can be visualised in Figure 2.8.

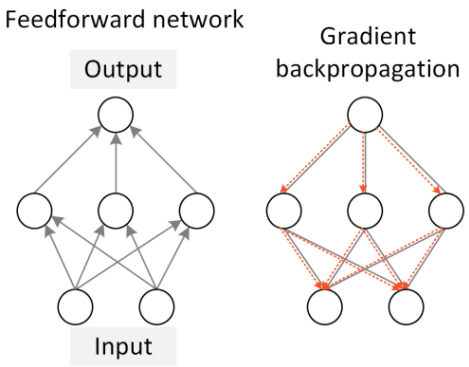


Figure 2.8: Feedforward neural network and gradient backpropagation [15]. Feedforward network calculates outputs from inputs by passing data through layers of neurons in one direction. Gradient backpropagation network calculates gradients of the loss function with respect to the network's weights by propagating errors backwards through the network.

### 2.5.2 Spike timing-dependent plasticity (STDP)

Spike timing-dependent plasticity (STDP) is an unsupervised learning algorithm that is bio-physically accurate [65]. Crucially, it is based on a Hebbian rule which may be summarised as follows: suppose that two connected neurons fire at the same time. Then, the weight of the synapse between the two neurons should be strengthened [46]. However, STDP is based on the principle that if a presynaptic neuron spikes before a postsynaptic neuron, the weight of the synapse is either strengthened or weakened. Similarly, if a postsynaptic spike occurs before a presynaptic spike, the weight of the synapse is either weakened or strengthened. The first scenario is the principle of *Hebbian* STDP. The second, reverse scenario, is termed *anti-Hebbian* STDP.

### 2.5.3 Backpropagation Through Spikes

Primarily, the objective of the backpropagation algorithm is loss minimisation. This is done through the computation of the gradient of the loss, concerning each learnable parameter via the chain rule from the final layer back to each weight [43, 59]. The gradient provides the direction for updating the weights of the network in a way that the error should fall. A gradient of 0 yields no weight update. Therefore, a pertinent limitation of training SNNs via error backpropagation is the non-differentiable nature of spikes, leading to the phenomenon known as the *dead neuron* problem.
The objective of the backpropagation-based approach is to learn the complex spatio-temporal relationships between the spikes. An issue arises in this approach, however, in that networks conduct forward passes frequently, and thus utilising this approach would lead to long training times. A more common approach in recent years has been to use the generalised backprop-

agation algorithm on an expanded computational graph, a phenomenon known as backpropagation through time (BPTT). In essence, this involves tracing the gradient of the final output of the network back to all of its earlier states. This process bears similarity to computing the gradient in recurrent neural networks (RNN), through the repeated application of the chain rule. The final loss is calculated as the sum of the instantaneous losses at each time step. In essence, the backpropagation algorithm can not be applied directly on SNNs, due to the non-differentiable nature of the spikes. As such, using methods such as *surrogate gradients* have been widely used in practice.

### 2.5.4 ANN-SNN Conversion

Originally presented in [13], and then improved in [60] for applications in deep networks, this algorithm provides a strong framework for training SNNs with IF neurons. The mechanism works via a threshold balancing technique that, in an iterative manner, refines the threshold voltage of each layer. Provided a trained ANN is available, the key step is the conversion of the input to a Poisson spike train over the training set for a sufficiently large enough time window such that the timed average adequately describes the input.

In terms of computational efficiency, for ANN-SNN conversion-based networks, the complexity is similar to that of regular ANNs. This may be attributable to the fact that the cost of calculating the output SNNs parameters is relatively small. Such calculations only occur one time for every SNN network, after training the source ANN.

### 2.5.5 Surrogate Gradients

To leverage the advantages of the widely-used backpropagation-based optimisation procedures, [71, 75, 5] proposed the surrogate gradient technique. In this technique, the gradient of the step function of the dynamics of an LIF (or IF) neuron is approximated by a pseudo-derivative, such as a linear or exponential function. A novel procedure was presented in [56]: an approximation function for these gradients found by leveraging spike time information in the derivative. We can explore this further:
Suppose the membrane potential of a neuron is $U$, the threshold $\theta$, and recall the spike indicator function $S$. Consider the following cases:

1. Membrane potential is below threshold: $U < \theta$,
2. Membrane potential is above threshold $U > \theta$,
3. Membrane potential is exactly threshold $U = \theta$.

In the first case, since the membrane potential is not sufficient enough to excite the neuron to spike, there is no spike, and the derivative would be $\partial S/\partial U_{U<\theta} = 0$. In the second case, a spike would fire but the derivative would remain the same: $\partial S/\partial U_{U>\theta} = 0$. This is because traditional spike indicator functions do not distinguish between membrane potential values above threshold; they only indicate whether or not a spike has occurred. Case 3 implies that $\partial S/\partial U_{U=\theta} = \infty$, an improbable boundary case. In such cases, we can approximate the gradient $\partial \tilde{S}/\partial U$ as a solution to this problem. Reference [15] provides a useful derivation. We might consider replacing the non-differentiable term with a threshold-shifted sigmoid activation function, but only during the backward pass. Consider Equation 2.12.

$$\sigma(\cdot) = \frac{1}{1 + e^{\theta - U}} \tag{2.12}$$

and we have that

$$\frac{\partial S}{\partial U} \leftarrow \frac{\partial \tilde{S}}{\partial U} = \sigma'(\cdot) = \frac{e^{\theta - U}}{(e^{\theta - U} + 1)^2}, \tag{2.13}$$

implying that learning only occurs provided spikes have occurred. This phenomenon is presented in Figure 2.9.

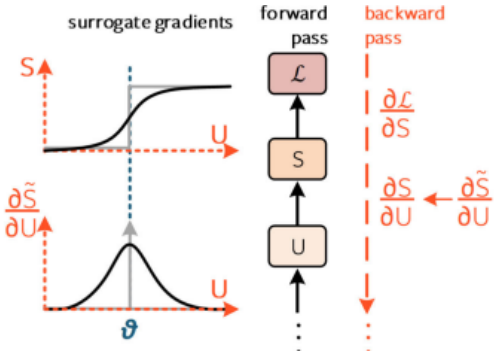


Figure 2.9: Reproduced from [15]. Spike generation function is approximated as a continuous function during the backward pass. Surrogate function used to compute backward propagation step.

Suppose that a synaptic weight $W_{\text{in}}$ is coupled with the input of a spiking neuron, and another weight $W_{\text{out}}$ is attached to the output of that same neuron. Suppose that the following sequence of events happens:

1. Input spike, $S_{\text{in}}$ is scaled by $W_{\text{in}}$,
2. Weighted spike is added as input current injection to spiking neuron,
3. Neuron may or may not spike, $S_{\text{out}}$,
4. Output spike is weighted by output weight $W_{\text{out}}$,
5. Weighted output spike changes loss function $\mathcal{L}$.

The (arbitrary) loss function chosen will be the Manhattan distance between the ground truth label $y$ and the weighted spike:

$$\mathcal{L} = |W_{\text{out}}\, S_{\text{out}}\ - y|$$

and updating $W_{\text{out}}$ means that

$$\frac{\partial \mathcal{L}}{\partial W_{\text{out}}} = S_{\text{out.}}$$

Therefore, in general, we find that for a weight to be updated, a spike must occur. If we consider the case for updating $W_{\text{in}}$, the following derivative must be found:

$$\frac{\partial \mathcal{L}}{\partial W_{\text{in}}} = \underbrace{\frac{\partial \mathcal{L}}{\partial S_{\text{out}}}}_{A} \underbrace{\frac{\partial S_{\text{out}}}{\partial U}}_{B} \underbrace{\frac{\partial U}{\partial W_{\text{in}}}}_{C}.$$

Term A is simply $W_{\text{out}}$, based on the equation we gave for $\mathcal{L}$ earlier. Term B is 0 most of the time, unless it is replaced with a surrogate gradient. The third term C is $S_{\text{in}}$.

As an example of a surrogate gradient, in practice, one might consider the arctan surrogate gradient, first utilised in [17]:

$$\frac{\partial \tilde{S}}{\partial U} = \frac{1}{\pi} \frac{1}{1 + (\pi U)^2}.$$

Surrogate gradients allow the training process of SNNs to be computationally feasible and have therefore paved the way for large-scale implementation of SNNs in practice. A prominent area where such an implementation

might occur is in the field of computer vision. In particular, image classification tasks. In the next chapter, we explore the adversarial robustness of SNNs by measuring changes in accuracy on image classification tasks.

# Chapter 3

# Adversarial Robustness of Spiking Neural Networks

In the previous chapter, we provide background on the construction and implementation of Spiking Neural Networks. This chapter will give an overview of a key application of SNNs: computer vision. We then move onto a discussion of general adversarial robustness in machine learning systems. Following this discussion, the adversarial robustness of SNNs is explored, through the comparative analysis of networks built using Leaky Integrate and Fire (LIF), Quadratic Integrate and Fire (QIF) and Izhikevich neurons. As mentioned in Section 1.2, the metric of adversarial accuracy, which measures the accuracy of the network under adversarial attack, is used to compare the networks built from these neurons. The main experimental methodologies we use to conduct this analysis will be outlined in the chapter.

## 3.1 SNNs in Computer Vision

Recently, SNNs have been a driving force in the development of numerous modern computer vision and signal processing techniques. Gradually, the applications of SNNs are being considered for computer vision, where data consisting of temporal information is utilised, or where the objective is to ease computational burden [74].

Despite various studies demonstrating the proficiency of SNNs to be used for image classification on large datasets such as ImageNet [58, 29], most applications of SNNs are still restricted to less complex datasets such as MNIST [40]. One potential reason for this is the non-differentiability of the spiking neurons which complicates the backpropagation process. Some notable studies have applied SNNs for object detection tasks [76, 31]. These studies have shown comparative results with deep neural networks, whilst also showing SNNs as being more energy efficient.

## 3.2 Adversarial Attacks

In today's mission-critical applications in deep neural networks, such as in transportation or medical applications, adversarial attacks remain one of the biggest challenges to their success [73, 16, 37]. As briefly mentioned in Chapter 1, the fundamental concept behind adversarial attacks is the purposeful modulation of an input to a neural network such that the change is subtle enough to remain undetectable to human eyes, but also fools the network into making inaccurate decisions. In Szegedy *et. al.* [68], this deceptive behaviour was first introduced in computer vision. Within malware detection, Biggio *et. al.* [6] first introduced adversarial examples. Following this research, various defensive mechanisms have been proposed

to address the issue of adversarial robustness. One class of defence consists of fine-tuning the network parameters, *adversarial training* [20, 45], or network distillation [53]. Another class of defences focus specifically on pre-processing the input before it is given to the network. Such methods include input quantization [72], or compression [22]. Despite these advances being made, many counterattack techniques have been proven effective in fooling neural networks with adversarial examples. Numerous defences based on gradient-masking were shown to be futile when exposed to an attack demonstrated in [4]. In addition, adversarial training tends to overfit to training samples and remains exposed to transfer attacks [70]. The threat of adversarial attacks is still a pertinent issue.

Suppose we have a clean image $x$, belonging to class $i$, and a trained neural network $M$. An adversarial example $x_{adv}$ must satisfy two criteria:

1. $x_{adv}$ looks similar to $x$. That is, $|x - x_{adv}| = \epsilon$, where $\epsilon$ is small, and
2. The neural network misclassifies $x_{adv}$ such that $M(x_{adv}) \neq i$.

Choosing the distance metric $|\cdot|$ is dependent on the method used to generate the adversarial image $x_{adv}$, and $\epsilon$ is a hyperparameter. In practice, the $L_2$ or $L_\infty$ norm is used to measure similarity. The normalised pixel intensity is transformed to $x \in [0, 1]$, where the original $x \in [0, 255]$.

Adversarial attacks can also be categorised according to the attacker's knowledge of the model under attack. White-box attacks are where the attacker is assumed to have some information about the target model, including its architecture, weights, or training procedure. This attack category is also known as a *gradient-based* attack. On the other hand, in a *score-based* attack setting, the attacker is assumed only to have access to the output (predicted class probabilities or logits) of the model under attack. These are also known as black-box attacks.

We can also subdivide adversarial attacks into *targeted* or *non-targeted* attacks. Targeted attacks occur when an adversarial example is generated to guarantee a predetermined label or classification, whereas a non-targeted attack has to objective of ensuring an incorrect classification, irrespective of the class.

## 3.3 White-Box Adversarial Attacks

### 3.3.1 Fast Gradient Sign Method (FGSM)

The Fast Gradient Sign Method attack is one of the most basic methods for the construction of adversarial attacks, introduced in [20]. FGSM has the main objective of seeking a perturbation $\delta$ such that the cost function for the perturbed input of the network $J\left(x+\delta, y_{\text{true}}\right)$ is maximised, where $x$ is the original, clean input image and $y_{true}$ is the ground truth label. Note that this maximisation is subject to the constraint that $|\delta|_\infty < \epsilon$. The closed-form representation of the attack is:

$$x_{adv} = x + \epsilon \times \text{sign}\left(\nabla_x J\left(x, y_{\text{true}}\right)\right) \tag{3.1}$$

where $\epsilon$ denotes the strength of the perturbation to the clean image.

### 3.3.2 Projected Gradient Descent (PGD)

Initially proposed in [45], this method is a $k$-step variation of FGSM, and is formally computed in the following way:

$$x_{adv}^{(k+1)} = \Pi_{x+\epsilon}\left\{\left(x_{adv}^{(k)} + \alpha \times \text{sign}\left(\nabla_x\left(J\left(x_{adv}^{(k)}, y_{\text{true}}\right)\right)\right)\right)\right\} \tag{3.2}$$

Here, $x_{adv}^{(0)} = x$ and $\alpha(\leq \epsilon)$ is the quantity of disturbance applied per iteration or step, $k$ denotes the total number of iterations used in the attack. $\Pi_{x+\epsilon}\{\cdot\}$ projects its operand on an $\epsilon$-ball around $x$ such that the operand is clipped between $x + \epsilon$ and $x - \epsilon$.

## 3.4 Black-Box Adversarial Attacks

### 3.4.1 Pixle Method

Introduced in [55], the Pixle method offers a strong adversarial attack designed to generate adversarial examples by strategically rearranging pixels in an image. Unlike gradient-based attacks that require knowledge of a victim model's internal parameters, Pixle only requires access to the model's output (such as class probabilities) and does not rely on gradients, making it a black-box, decision-based attack.

We begin with defining an input image as $x$ and its true label $y$. Initialise a best adversarial example $\bar{x}$ as the original image $x$, and set the best loss $l$ to the loss of $x$ on the model with respect to the label $y$, denoted $f_y(x)$.

The algorithm for the Pixle method is the following:

**Algorithm 1** Pixle: Restart-Iterative algorithm

**Require:** Input image $x$ with its associated label $y$. Maximum and minimum dimensions for source patch. The number of restarts $R$ and the iterations to perform for each restart step $T$. The mapping function $m$.

**Ensure:** $y = x^n$

$\bar{x} \leftarrow x$
$l \leftarrow f_y(x)$
**for** $r = 0$ to $R$ **do**
    $x^r \leftarrow \bar{x}$
    **for** $t = 0$ to $T$ **do**
        Sample $p = (o_x, o_y, w_p, h_p)$
        Calculate the set $P$
        $x^t \leftarrow \bar{x}$
        **for** each $(i, j) \in P$ **do**
            $(z, k) \leftarrow m(i, j)$
            $x^t_{z,k} \leftarrow x_{i,j}$
        **end for**
        **if** $f_y(x^t) < l$ **then**
            $l \leftarrow f_y(x^t)$
            $x^r \leftarrow x^t$
        **end if**
    **end for**
    $\bar{x} \leftarrow x^r$
**end for**
    **return** $\bar{x}$

The process is to, for each iteration $t$ (up to a maximum number $T$), randomly select a rectangular patch within the image defined by its top-left

corner $(o_x, o_y)$ and dimensions $(w_p, h_p)$. Apply a mapping function $m$ to rearrange the pixels within the selected patch. The mapping function can be random, similarity-based or distance-based. An adversarial candidate image $x^t$ is then formed by applying the mapping to the patch in the original image $\bar{x}$. Query the model with $x^t$ and update the loss $f_y(x^t)$. If the loss $f_y(x^t)$ is lower than the current best loss $l$, we update the best adversarial example $\bar{x}$ and the best loss $l$. After completing the iterations, the algorithm returns the best adversarial example $\bar{x}$ found.

## 3.5 Experiments

Comparison will now be done across the three SNN models: LIF neuron-based, QIF neuron-based, and Izhikevich neuron-based. Experiments are broadly divided into two categories: white-box and black-box adversarial attacks. PGD and FGSM are white-box attacks, with the adversary having complete knowledge of the model and its parameters, whilst Pixle is a black-box, decision-based attack, where the attacker does not have full access to the model in terms of its architecture and weights. The Pixle approach is used in a representative fashion for black-box attacks due to its strong performance compared to other black-box methods [55].

Within these classes of attacks, each model will be attacked in a targeted and non-targeted way. That is, white and black-box attacks will either be targeted or non-targeted.

### 3.5.1 Motivation

In this section, we explore the adversarial robustness of SNNs constructed from QIF, LIF and Izhikevich neurons. We aim to provide a comparative analysis where accuracy degradation across neuronal schemes can be compared, to provide insight into the extent to which neuronal choice has a part to play in adversarial robustness. In this portion of the thesis, we specifically pay attention to the following adversarial aspects:

1. How does adversarial robustness change across different neuron choices?
2. The extent to which the number of time steps affects adversarial robustness.

To the best of our knowledge, there has not been rigorous, comparative analysis done on the effect of choice of neuron on the adversarial robustness of SNNs. In particular, for rate-encoded spiking networks, looking at how QIF, LIF and Izhikevich-based models compare when such models are queried with adversarial examples. This is what the analysis will focus on in this thesis.

### 3.5.2 Experimental Settings

#### Network Architecture

In our experiments, we study three types of SNNs built from different neurons: i) LIF-based SNNs; ii) QIF-based SNNs and iii) Izhikevich-based SNNs. Let $O_1$ denote the LIF-based SNN; $O_2$ denote the QIF-based SNN, and $O_3$ denote the Izhikevich-based SNN. The SNNs of the second and third types are built using the SpikingJelly framework [66], whilst SNNs of

the first type are built using the snnTorch framework [15] in Python. As both frameworks are based on PyTorch, conventional learning techniques for the training of SNNs are utilised. Torchattacks [32] is used to generate all adversarial examples. It contains an interface similar to PyTorch and allows for the implementation of adversarial attacks. Further details about the attacks may be found in the Torchattacks documentation.

All experiments are carried out on an Intel Core i5. Section 3.2 outlines the attack methods deployed. We pay particular attention to rate-encoded SNNs in our experiments, with input data encoded using the Poisson encoder introduced in Section 2.4.2. The dataset is divided into a training set of 60,000 images and a test set of 10,000 images. Each image in the dataset corresponds to a digit from 0 to 9, leading to a ten-class classification problem.

The SNNs used in the experiments each have a flatten layer, fully connected layer, followed by a neuron layer (LIF, QIF, or Izhikevich) which outputs spikes. The architectures of the models constructed are found in Table 3.1. We use the image classification dataset MNIST [11]. It contains 70,000 grayscale images of handwritten digits, with each image being $28 \times 28$ pixels in size. The pixel values range from 0 to 255. We set the total number of epochs to 1, due to computational constraints. During the training process, we use a learning rate of $1e-2$, with an Adam optimiser. For the QIF and Izhikevich neurons, we set the time constant $\tau_m$ to 2, and the firing threshold $\theta$ to 1. For the LIF neuron, we set $\beta$ to 0.6. As the original pixels vary from 0 to 255, we apply a normalisation of the dataset to have zero mean and that the absolute values of the input pixel intensity for a clean image range from 0 to 1. Therefore, we also normalise the noise intensity for the attacks, using $\epsilon = 8/255, 16/255, 32/255$ and $64/255$. All architectures use a sigmoid surrogate gradient function with a slope of 25.

As for gradient optimisation, with an image passed into the network, neurons accumulate pre-synaptic spikes and generate output spikes. The accumulated spike count is then used to calculate the MSE loss, which is the loss function used over all architectures in the study. Based on this calculated loss, backpropagation occurs via the surrogate gradient. The number of time steps used for training is 15 for all experiments.

Table 3.1: Architecture of SNNs used in experiments

| Layer | Type | Output Shape | Connected to |
|---|---|---|---|
| flatten | Flatten | 784 | Dense |
| fc | Dense | 10 | {LIF, QIF, Izhikevich} |
| {LIF, QIF, Izhikevich} Node | Neuron/Activtion | 10 | |

The code for the experiments can be found at: github.com/timif2/snnthesis.

### 3.5.3 Experiment 1: White-box attacks

#### Attack: Projected Gradient Descent

The adversarial accuracy of a model is the accuracy a model achieves on a sample of images on which perturbations have been added. We now study, for each attack, how the adversarial robustness of networks changes depending on the neuron type used. No hybrid schemes are used in the experiments so that only one type of neuron is used in each network (cf. Table 3.1). Attack intensity is varied across each network type, and attack scheme, using $\epsilon = 8/255, 16/255, 32/255$ and $64/255$. For the targeted schemes, the labels are shifted by 1, so that label 2 becomes a 3, and an original label 3 becomes a 4, for instance.

Analysis of the performance of the networks when subject to a non-targeted PGD attack is demonstrated in Figure 3.1, in Panel (a). Each network is

given 16 samples from the MNIST dataset. The clean accuracy is the accuracy of the networks on these samples without perturbations to the input image. The clean accuracies are 100%, 81.25%, and 75% for the $O_1$, $O_2$ and $O_3$ networks, respectively. After perturbations are added to the input image, $O_1$ appears to be the most robust across the attack intensities compared to the other schemes. Meanwhile, the $O_3$ model suffers higher levels of accuracy degradation than the other two models at lower attack intensities. The $O_2$ model is in the middle, in terms of relative performance after adversarial examples have been introduced.

Similarly, presented in Panel (b) in Figure 3.1, we see that $O_1$ demonstrates the highest levels of robustness compared to the other network types. The $O_3$ network appears to have higher levels of robustness in a targeted attack, compared with a non-targeted attack at the same levels of attack intensity. However, for $O_2$, the targeted attack seems slightly more effective in reducing accuracy than the non-targeted attack. In terms of overall accuracy degradation, $O_2$ and $O_3$ appear to react similarly to the targeted PGD attack.

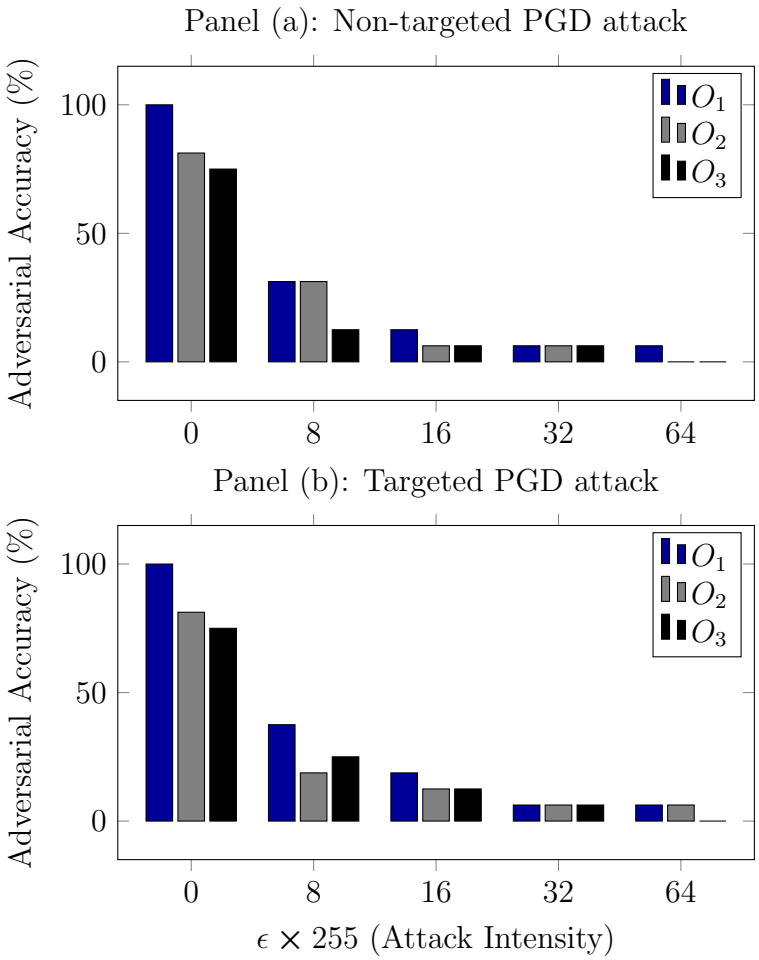


Figure 3.1: Comparison of adversarial accuracy after targeted and non-targeted PGD attack across SNNs. Clean accuracy is when $\epsilon = 0$, and attack intensities used are $\epsilon = 8/255, 16/255, 32/255, 64/255$. $O_1$ network appears to be most resistant to adversarial perturbation, whilst $O_3$ is least resistant, on average, across targeted and non-targeted attacks.

### Attack: Fast Gradient Sign Method

As for how the models perform in the face of a non-targeted FGSM attack, as demonstrated in Figure 3.2, the LIF neuron-based model still appears to show superiority concerning robustness overall. Between the $O_2$ and $O_3$ models, the $O_2$ model appears to have greater robustness across the levels of intensity than the $O_3$ model. When the models are queried in a targeted fashion, under a FGSM attack, all three networks studied are

more adversarially robust, across all attack intensities (apart from $O_1$ when $\epsilon = 8/255$), compared to their performance in the PGD attack.

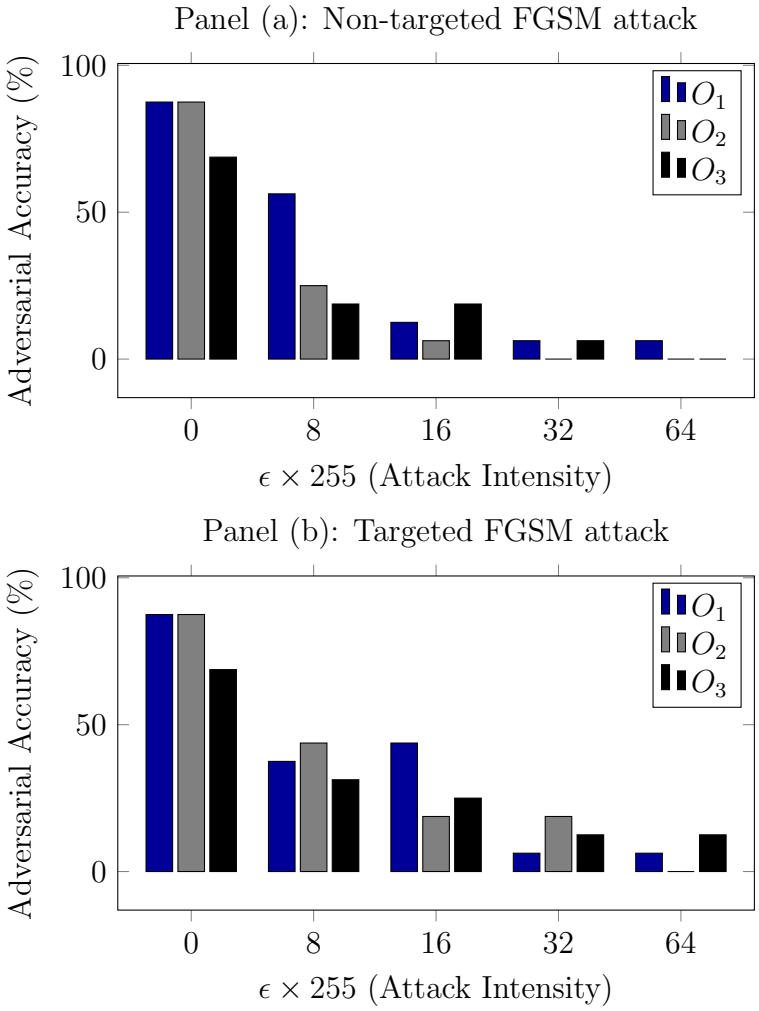


Figure 3.2: Comparison of adversarial accuracy after targeted and non-targeted FGSM attack across SNNs. Clean accuracy is when $\epsilon = 0$, and attack intensities used are $\epsilon = 8/255, 16/255, 32/255, 64/255$. $O_2$ network shows stronger levels of robustness relative to PGD attack, and similar for $O_3$ network. $O_1$ network is less resistant, compared to the PGD attack.

### 3.5.4 Experiment 2: Black-box attacks

#### Attack: Pixle

The Pixle attack, as explained in Section 3.4.1, works by randomly selecting a rectangular patch and applying a mapping function to rearrange the pixels

within the selected patch. Interestingly, in the black-box attack setting, the $O_1$ network is the least robust compared to the other network types, under both a targeted and non-targeted attack. This is in contrast to the white-box setting performance in the previous experiment. For the non-targeted Pixle attack, $O_3$ demonstrates the highest levels of adversarial robustness compared to the other networks, followed by $O_2$. This is shown in Figure 3.3. $O_1$, across all patch sizes studied, identifies 1 image correctly of 16, leading to the adversarial accuracy of 6.25%. $O_2$ appears to be stronger in the black-box non-targeted setting compared to the white-box non-targeted setting.

We now pay attention to the targeted black-box attack, which is performed similarly to the white-box targeted attack regarding how the labels are changed. Figure 3.3 demonstrates this. $O_2$ leads in this setting being the most adversarially robust. $O_3$ does not correctly identify any images once the patch size is beyond 60% of the image. In comparison to all other attack settings so far, the targeted black-box is the most effective across neuron schemes overall. The $O_1$ network responds the same to all attack intensities, identifying only 1 image correctly of the batch of 16 images.

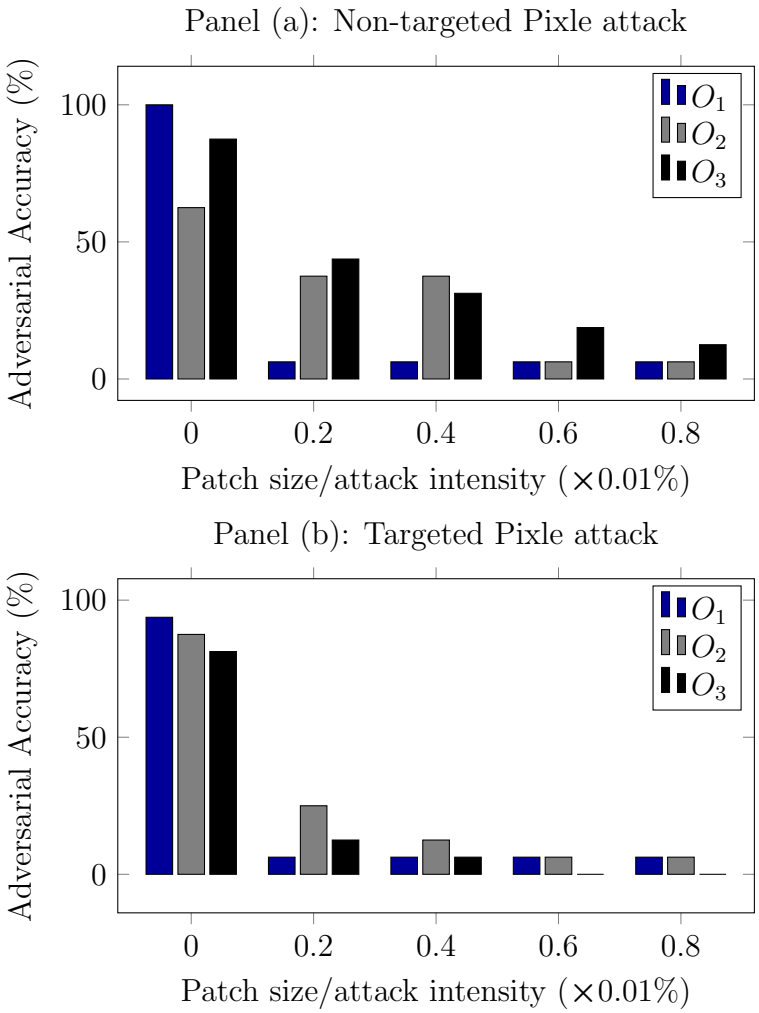


Figure 3.3: Comparison of adversarial accuracy after targeted and non-targeted Pixle attack across SNNs. Clean accuracy is when the patch size is 0, and (square) patch sizes used are 20%, 40%, 60% and 80% of the input image from the top-left corner. $O_1$ is least adversarially robust, whilst $O_2$ demonstrates highest levels of adversarial robustness.

The following set of experiments will study the interaction between time steps and the choice of network and how the adversarial robustness is affected.

### 3.5.5 Experiment 3: Effect of time steps

We now study the effect of time steps on the adversarial robustness of SNNs, looking at how this compares across the neuron schemes looked at thus far.

To do this, we perform analysis in a white-box, non-targeted setting, using the PGD and FGSM attacks. The intensity for these attacks will be fixed at $\epsilon = 8/255$, and the sample size of the dataset on which the networks will be tested will remain at 16 images. The networks will be trained using 4, 6, 8, and 10 time steps, providing insight into how robustness varies across these time scales.

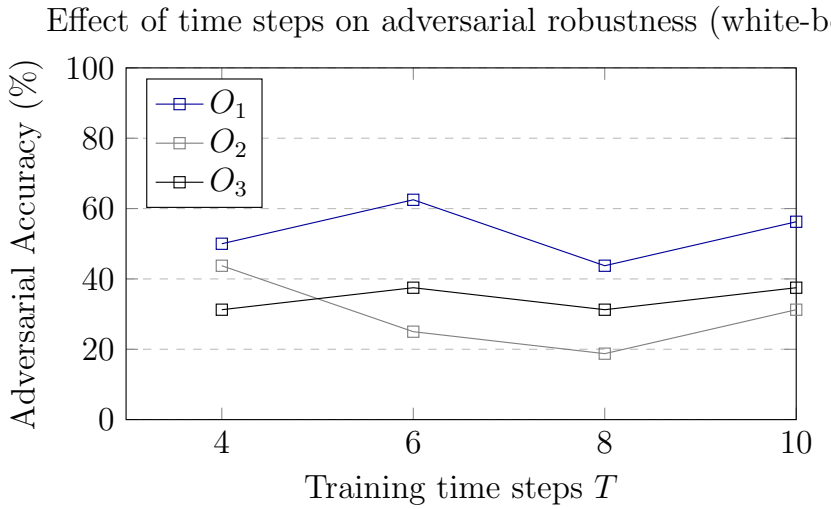


Figure 3.4: Effect of network training time steps on adversarial accuracy. Timesteps $T = 4, 6, 8, 10$. Non-targeted PGD attack with attack intensity $\epsilon = 8/255$ for all training timesteps. Model $O_1$ has the lowest levels of accuracy degradation, whilst $O_2$ has the highest levels of accuracy degradation.

Paying attention to Figure 3.4, In the white-box, non-targeted setting, $O_1$ demonstrates the highest levels of adversarial robustness when trained over all time steps compared to the other models. Model $O_2$ appears to have the lowest levels of adversarial robustness compared to the other models. $O_3$ makes less intense fluctuations over all time steps than the other models. Now considering the effect of training time steps when there is a black-box attack. In Figure 3.5, we can see that the adversarial accuracy does not change for $O_1$, then decreases once the time steps are equal to 10 when the patch size is 40% for the Pixle attack. The $O_2$ model stays in between 40% and 60%. $O_3$ across all time steps maintains a relatively lower level of

adversarial robustness when trained across all time steps.

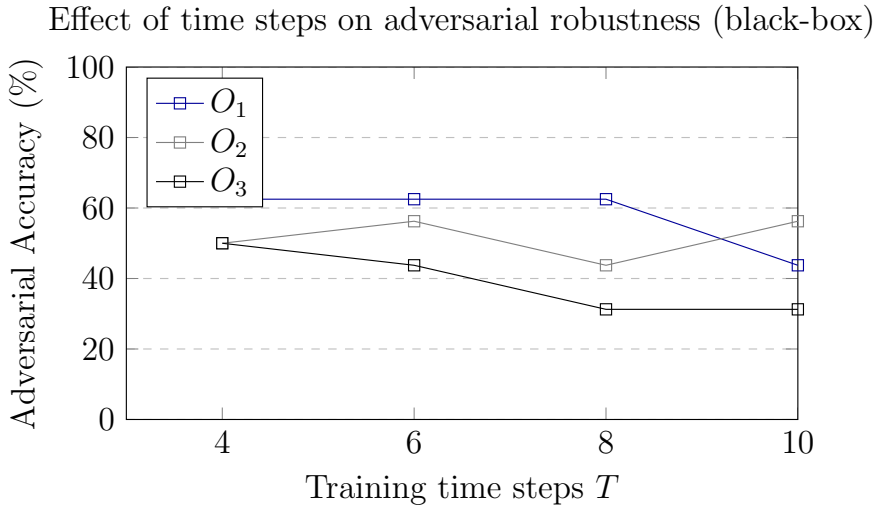


Figure 3.5: Effect of network training time steps on adversarial accuracy. Time Steps $T = 4, 6, 8, 10$. Non-targeted Pixle attack with patch size equal to 40% for all training time steps. $O_1$ is the most robust, whilst $O_2$ is the least robust.

The following section will provide an in-depth discussion of the results found in these experiments. In particular, it will look at possible explanations for the observed phenomena in all three experiments. In addition, we will look at related work on the topic of security and vulnerability of SNNs in general.

# Chapter 4

# Discussion

Section 3.5 started with an explanation of the motivation behind conducting the experiments, largely on the lack of literature exploring this aspect of SNNs. Next, we continue this section by outlining the architectures and frameworks utilised in the experiments and explaining the method for conducting them. Following this, we study 3 experiments. The first 2 experiments focus on the relationship between attack intensity and neuron choice, and the third focuses on the relationship between time steps, neuron choice and adversarial accuracy.

This part of the thesis will explore potential reasons why and offer explanations for, the results obtained in the previous section. In addition, we also provide an outlook on related literature on the topic of adversarial robustness in SNNs.

## 4.1 Neuron choice and robustness

Whilst LIF neurons offer easier computational implementation, at the expense of complexity and depth, the findings of the experiments in this

thesis demonstrate networks constructed from them to be more robust to white-box adversarial attacks when compared with QIF and Izhikevich-neuron-based networks. In the black-box setting, however, this notion reverses. The LIF-neuron-based networks are the least robust to adversarial examples in the targeted and non-targeted black-box setting, relative to the other networks studied. When studying the relationship between the time steps the models are trained on, adversarial robustness and the choice of neuron, the LIF-based network is most robust in both the white and black-box scenario when a common adversary attacks all network types.

The differences observed in the adversarial performance of the models may be attributable to the key parameters governing the neuronal dynamics. For the LIF model, [62] demonstrates that the leak rate of the membrane potential has a direct relationship with the adversarial robustness of the network. In the QIF-based model, differences in accuracy degradation may be caused by the quadratic non-linearity coefficient. Potentially, this term might cause neuron behaviour near the threshold to be more sensitive to small perturbations. This sensitivity is then exploited by adversarial inputs, and propagates throughout the network, providing a potential reason for this model's performance in the white box non-targeted setting. The Izhikevich model has various parameters that control the recovery dynamics and reset behaviour of the model. This makes it difficult to isolate and understand the causal effect between individual parameters of the neuron and the adversarial robustness of networks built from Izhikevich neurons and would require further analysis. However, this increased complexity may explain reduced performance when presented with adversarial examples as this neuron type is more sensitive to small changes in input, leading to a propagation effect from adversarial examples.

All three networks studied have qualities of inherent adversarial robustness,

compared to traditional SNNs. Sharmin *et. al.* [62] finds that due to SNNs being binary spike-based networks, the encoding of any analog valued input signal to a binary spike train leads to the property of inherent robustness in SNNs. In particular, they propose that it is this discretisation process that makes SNNs more robust than their ANN counterparts. The amount of discretisation introduced by using the Poisson encoder varies with the number of time steps used. Therefore, the adversarial accuracy of a network can be controlled directly by changing the number of time steps as long as the clean accuracy of the network remains reasonable.

## 4.2 Related Work

There have been some studies to analyse the security and vulnerability of SNNs in general. Sharmin *et. al.* [62] reviews the state of SNN adversarial robustness research. They present a simple and effective framework to construct adversarial attacks. Their work demonstrates that input discretisation via Poisson encoding and non-linear activations of LIF (or IF) neurons are inherent reasons why SNNs may be more robust than their ANN counterparts. The robustness of SNNs against attacks with varying neuron firing voltage thresholds and time window boundary values is explored by Rida *et. al.* [3]. Here, they query robustness under numerous combinations $(V, T)$, where $V$ is the voltage threshold of the LIF neuron, and $T$ refers to the training time step. Work done by Kundu *et. al.* [36] discovers that LIF neurons reduce the input perturbation, and the extent to which this occurs depends on the effect of combined parameters such as weight, leakage, threshold and time step. In Guo *et. al.* [23], four neural coding schemes are compared to find the best coding scheme. However, the analysis is constrained to only 2 layer fully connected networks and

STDP training. Scalability is a concern in such a case, and the key metric of adversarial robustness is not considered. Kim *et. al.* [33] conducts an extensive comparison on the robustness, energy efficiency and accuracy of rate and direct encoding schemes, but does not compare to other encoding methods.

# Chapter 5

# Conclusion

In this project, we study Spiking Neural Networks constructed solely from three different types of spiking neurons: the LIF, QIF and Izhikevich neurons. We compare the robustness to adversarial examples for the three networks when subject to white-box attacks and black-box attacks, across varying attack intensities. This is supplemented by analysing the effect of deploying targeted and non-targeted attacks on adversarial robustness. Relationships between the number of timesteps the model is trained on and the adversarial robustness of the network are studied. This chapter draws the report to a close, putting the work achieved in a wider context.

## 5.1 Summary of results

Section 3.5 demonstrates through the first two experiments that when the models are subject to white-box attacks, a model constructed solely from LIF-neurons generally seems to be the most robust to adversarial perturbations, in both the targeted and non-targeted attack settings. Primarily, we determine one reason for this may be due to simpler dynamics mean-

ing lower levels of sensitivity to attacks as there are fewer parameters to manipulate. However, we find in the black-box setting that this robustness does not extend to LIF-based networks, and the explanation for this is an open question. A similar phenomenon is seen in QIF-based networks, where in the white-box attack setting, across targeted and non-targeted attacks, they are relatively less robust, but in the black-box setting appear to be relatively stronger. As for Izhikevich-based models, the difference in adversarial accuracy across attack intensities does not appear to be significant when comparing the white-box and black-box attack settings. This is the key contribution of the project: analysis of how the neuron choice of a SNN affects its robustness to adversarial attacks.

Experiment three of the thesis studies the effect of training time steps on adversarial accuracy across the network types. In both the white-box and black-box settings, using the non-targeted PGD and Pixle attack as representative white-box and black-box attacks, respectively, the LIF-based network offers the highest levels of adversarial robustness when trained with $T = 4, 6, 8, 10$ time steps. Interestingly, using work from [62], we expect as the training time steps increase, adversarial robustness generally falls across networks, or that the most robust networks are those trained with the smallest time steps provided the clean accuracy of the models remains in sensible ranges. This appears to be the case on average for the networks until the networks are trained with 10 time-steps, then the robustness increases. A plausible explanation for this phenomenon is the experiment design: the networks were not trained with enough epochs, and the training time steps could have been increased further to study the long-term effect of training time steps. These limitations in experiment design are explored further in the following section.

## 5.2 Limitations of current approach

Limitations of the thesis project are largely due to experiment design and lack of computational resources. Due to computational constraints in our experiments, training epochs are restricted to 1. This has the drawback of the models studied not yielding strong robustness in general when presented with adversarial examples. Similarly, the analysis of time steps in the third experiment of Section 3.5 would yield more consistent, robust results provided the networks are trained with much larger time steps, such as $T = 50, 100, 150$ and so on. Models trained at such large timestep scales provide a clearer idea of robustness in general for a SNN.

Whilst the FGSM and PGD attacks are strong representative white-box adversarial attacks, and the Pixle attack is a strong black-box adversarial attack, more diverse attack methods can give a more holistic view of model performance in white-box and black-box attack settings. Practically, implementing a score-like system to quantify relative performance across a large, diverse range of white-box and black-box attacks and comparing models in this way could achieve this.

Another limitation of the approaches used in this project is restricting experimental analysis to shallow networks. Each network constructed in the experiments only has a flatten, fully connected and neuron layer. Extending analysis to deeper networks, incorporating convolution and pooling layers, for instance, can give a more holistic view of adversarial robustness on architectures of varying complexity. Similarly, whilst performing analysis on rate-encoded networks is beneficial for gaining insight into this area of interest, extending the study to SNNs which use other encoding schemes such as latency, phase or direct, for instance, would provide a more comprehensive outlook on the subject matter. Such limitations form a direction

for future work suggestions, which are explored in the next section.

## 5.3 Further Work

Crucially, offering concrete, rigorous explanations for why there are significant differences in adversarial accuracy across network types is needed. In particular, it is of interest to look at why targeted attacks disproportionately affect some networks more than others, compared to non-targeted attacks. Rigorous explanations for why some networks constructed from certain neurons are more sensitive to black-box, or white-box attacks than others are also needed. To this end, doing a detailed analysis into the effect of various parameters within, say, QIF neurons, on adversarial robustness is a helpful initiative. Doing so will contribute to a greater understanding of how specific neuron parameters play a role in SNN robustness. As an example, [62] demonstrates this with the analysis of the effect of *leak rate* $\lambda$ on the adversarial robustness of rate-encoded LIF-based SNNs.

Giving insight into interaction effects, a study on hybrid effects may also be useful. This might involve looking at how networks constructed from both LIF and QIF neurons fare in the face of adversarial attacks, compared to LIF and Izhikevich neurons. Importantly, when doing such analysis, it must be taken into consideration that the differences in computational load and the trade-offs for different use cases for these neurons affect whether a researcher builds a network based on a particular neuron type for their experiments. Extending this analysis to other neuron types offers interesting further work, such as networks built from Hodgkin-Huxley neurons, for instance.

# Bibliography


[1] L. F. Abbott and Carl van Vreeswijk. "Asynchronous states in networks of pulse-coupled oscillators". In: *Physical Review E* 48.2 (Aug. 1993). Publisher: American Physical Society, pp. 1483–1490. DOI: 10.1103/PhysRevE.48.1483. URL: https://link.aps.org/doi/10.1103/PhysRevE.48.1483 (visited on 08/12/2024).

[2] A. M. H. J. Aertsen and P. I. M. Johannesma. "The Spectro-Temporal Receptive Field". en. In: *Biological Cybernetics* 42.2 (Nov. 1981), pp. 133–143. ISSN: 1432-0770. DOI: 10.1007/BF00336731. URL: https://doi.org/10.1007/BF00336731 (visited on 07/16/2024).

[3] Rida El-Allami et al. "Securing Deep Spiking Neural Networks against Adversarial Attacks through Inherent Structural Parameters: 2021 Design, Automation and Test in Europe Conference and Exhibition, DATE 2021". In: *Proceedings of the 2021 Design, Automation and Test in Europe, DATE 2021*. Proceedings -Design, Automation and Test in Europe, DATE (Feb. 2021). Publisher: Institute of Electrical and Electronics Engineers Inc., pp. 774–779. DOI: 10.23919/DATE51398.2021.9473981. URL: http://www.scopus.com/inward/record.url?scp=85111074375&partnerID=8YFLogxK (visited on 07/29/2024).

[4] Anish Athalye, Nicholas Carlini, and David Wagner. *Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples*. arXiv:1802.00420 [cs]. July 2018. DOI: 10.48550/arXiv.1802.00420. URL: http://arxiv.org/abs/1802.00420 (visited on 07/26/2024).

[5] Guillaume Bellec et al. *Long short-term memory and learning-to-learn in networks of spiking neurons*. arXiv:1803.09574 [cs, q-bio]. Dec. 2018. DOI: 10.48550/arXiv.1803.09574. URL: http://arxiv.org/abs/1803.09574 (visited on 07/26/2024).

[6] Battista Biggio et al. "Evasion Attacks against Machine Learning at Test Time". In: vol. 7908. arXiv:1708.06131 [cs]. 2013, pp. 387–402. DOI: 10.1007/978-3-642-40994-3_25. URL: http://arxiv.org/abs/1708.06131 (visited on 07/26/2024).

[7] Christopher M Bishop. "Neural Networks for Pattern Recognition". en. In: ().

[8] Nicolas Brunel and Mark van Rossum. "Lapicque's 1907 paper: From frogs to integrate-and-fire". In: *Biological cybernetics* 97 (Jan. 2008), pp. 337–9. DOI: 10.1007/s00422-007-0190-0.

[9] Ce Carr and M Konishi. "A circuit for detection of interaural time differences in the brain stem of the barn owl". en. In: *The Journal of Neuroscience* 10.10 (Oct. 1990), pp. 3227–3246. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.10-10-03227.1990. URL: https://www.jneurosci.org/lookup/doi/10.1523/JNEUROSCI.10-10-03227.1990 (visited on 07/05/2024).

[10] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. ISSN: 1063-6919. June 2009, pp. 248–255. DOI: 10.1109/CVPR.

2009.5206848. URL: https://ieeexplore.ieee.org/abstract/document/5206848 (visited on 07/09/2024).

[11] Li Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012). Conference Name: IEEE Signal Processing Magazine, pp. 141–142. ISSN: 1558-0792. DOI: 10.1109/MSP.2012.2211477. URL: https://ieeexplore.ieee.org/document/6296535 (visited on 08/12/2024).

[12] Peter U. Diehl and Matthew Cook. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity". English. In: *Frontiers in Computational Neuroscience* 9 (Aug. 2015). Publisher: Frontiers. ISSN: 1662-5188. DOI: 10.3389/fncom.2015.00099. URL: https://www.frontiersin.org/journals/computational-neuroscience/articles/10.3389/fncom.2015.00099/full (visited on 07/19/2024).

[13] Peter U. Diehl et al. "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing". In: *2015 International Joint Conference on Neural Networks (IJCNN)*. ISSN: 2161-4407. July 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280696. URL: https://ieeexplore.ieee.org/document/7280696 (visited on 08/15/2024).

[14] Jianhao Ding et al. "SNN-RAT: Robustness-enhanced Spiking Neural Network through Regularized Adversarial Training". en. In: ().

[15] Jason K. Eshraghian et al. "Training Spiking Neural Networks Using Lessons From Deep Learning". In: *Proceedings of the IEEE* 111.9 (Sept. 2023). Conference Name: Proceedings of the IEEE, pp. 1016–1054. ISSN: 1558-2256. DOI: 10.1109/JPROC.2023.3308088. URL: https://ieeexplore.ieee.org/document/10242251/?arnumber=10242251 (visited on 07/16/2024).

[16] Kevin Eykholt et al. *Robust Physical-World Attacks on Deep Learning Models*. arXiv:1707.08945 [cs]. Apr. 2018. DOI: 10.48550/arXiv.1707.08945. URL: http://arxiv.org/abs/1707.08945 (visited on 07/26/2024).

[17] Wei Fang et al. *Deep Residual Learning in Spiking Neural Networks*. arXiv:2102.04159 [cs]. Jan. 2022. DOI: 10.48550/arXiv.2102.04159. URL: http://arxiv.org/abs/2102.04159 (visited on 07/23/2024).

[18] J. Feng. "Is the integrate-and-fire model good enough?–a review". eng. In: *Neural Networks: The Official Journal of the International Neural Network Society* 14.6-7 (2001), pp. 955–975. ISSN: 0893-6080. DOI: 10.1016/s0893-6080(01)00074-0.

[19] Răzvan V. Florian. "The chronotron: a neuron that learns to fire temporally precise spike patterns". eng. In: *PloS One* 7.8 (2012), e40233. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0040233.

[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. arXiv:1412.6572 [cs, stat]. Mar. 2015. DOI: 10.48550/arXiv.1412.6572. URL: http://arxiv.org/abs/1412.6572 (visited on 07/05/2024).

[21] Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. "Towards deep learning with segregated dendrites". In: *eLife* 6 (Dec. 2017). Ed. by Peter Latham. Publisher: eLife Sciences Publications, Ltd, e22901. ISSN: 2050-084X. DOI: 10.7554/eLife.22901. URL: https://doi.org/10.7554/eLife.22901 (visited on 07/22/2024).

[22] Chuan Guo et al. *Countering Adversarial Images using Input Transformations*. arXiv:1711.00117 [cs]. Jan. 2018. DOI: 10.48550/arXiv.1711.00117. URL: http://arxiv.org/abs/1711.00117 (visited on 07/26/2024).

[23] Wenzhe Guo et al. “Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems”. In: *Frontiers in Neuroscience* 15 (Mar. 2021), p. 638474. ISSN: 1662-4548. DOI: 10.3389/fnins.2021.638474. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7970006/ (visited on 07/29/2024).

[24] Jun Han and Claudio Moraga. “The influence of the sigmoid function parameters on the speed of backpropagation learning”. en. In: *From Natural to Artificial Neural Computation*. Ed. by José Mira and Francisco Sandoval. Berlin, Heidelberg: Springer, 1995, pp. 195–201. ISBN: 978-3-540-49288-7. DOI: 10.1007/3-540-59497-3_175.

[25] D. Hansel, G. Mato, and C. Meunier. “Synchrony in excitatory neural networks”. eng. In: *Neural Computation* 7.2 (Mar. 1995), pp. 307–337. ISSN: 0899-7667. DOI: 10.1162/neco.1995.7.2.307.

[26] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: 2016, pp. 770–778. URL: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html (visited on 07/09/2024).

[27] A. L. Hodgkin and A. F. Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of Physiology* 117.4 (Aug. 1952), pp. 500–544. ISSN: 0022-3751. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1392413/ (visited on 07/16/2024).

[28] D. H. Hubel and T. N. Wiesel. “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex”. In: *The Journal of Physiology* 160.1 (Jan. 1962), pp. 106–154.2. ISSN: 0022-3751. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1359523/ (visited on 07/20/2024).

[29] Eric Hunsberger and Chris Eliasmith. "Training Spiking Deep Networks for Neuromorphic Hardware". In: (2016). arXiv:1611.05141 [cs]. DOI: 10.13140/RG.2.2.10967.06566. URL: http://arxiv.org/abs/1611.05141 (visited on 07/25/2024).

[30] E.M. Izhikevich. "Simple model of spiking neurons". en. In: *IEEE Transactions on Neural Networks* 14.6 (Nov. 2003), pp. 1569–1572. ISSN: 1045-9227. DOI: 10.1109/TNN.2003.820440. URL: http://ieeexplore.ieee.org/document/1257420/ (visited on 08/14/2024).

[31] Saeed Reza Kheradpisheh et al. "STDP-based spiking deep convolutional neural networks for object recognition". In: *Neural Networks* 99 (Mar. 2018), pp. 56–67. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2017.12.005. URL: https://www.sciencedirect.com/science/article/pii/S0893608017302903 (visited on 07/25/2024).

[32] Hoki Kim. *Torchattacks: A PyTorch Repository for Adversarial Attacks*. en. Sept. 2020. URL: https://arxiv.org/abs/2010.01950v3 (visited on 08/15/2024).

[33] Youngeun Kim et al. *Rate Coding or Direct Coding: Which One is Better for Accurate, Robust, and Energy-efficient Spiking Neural Networks?* en. arXiv:2202.03133 [cs]. Apr. 2022. URL: http://arxiv.org/abs/2202.03133 (visited on 07/04/2024).

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (visited on 07/09/2024).

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". en. In: *Com-*

*munications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386. URL: https://dl.acm.org/doi/10.1145/3065386 (visited on 07/09/2024).

[36] Souvik Kundu, Massoud Pedram, and Peter A. Beerel. *HIRE-SNN: Harnessing the Inherent Robustness of Energy-Efficient Deep Spiking Neural Networks by Training with Crafted Input Noise*. arXiv:2110.11417 [cs]. Oct. 2021. DOI: 10.48550/arXiv.2110.11417. URL: http://arxiv.org/abs/2110.11417 (visited on 07/05/2024).

[37] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. *Adversarial examples in the physical world*. arXiv:1607.02533 [cs, stat]. Feb. 2017. DOI: 10.48550/arXiv.1607.02533. URL: http://arxiv.org/abs/1607.02533 (visited on 07/26/2024).

[38] Lapique L. "Researches quantatives sur l'excitation electrique des nerfs traitee comme une polarization". In: *Journal of Physiology, Pathology and Genetics* 9 (1907), pp. 620–635. URL: https://cir.nii.ac.jp/crid/1570291225205713280 (visited on 07/16/2024).

[39] P. E. Latham et al. "Intrinsic dynamics in neuronal networks. I. Theory". eng. In: *Journal of Neurophysiology* 83.2 (Feb. 2000), pp. 808–827. ISSN: 0022-3077. DOI: 10.1152/jn.2000.83.2.808.

[40] Y. Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (Nov. 1998). Conference Name: Proceedings of the IEEE, pp. 2278–2324. ISSN: 1558-2256. DOI: 10.1109/5.726791. URL: https://ieeexplore.ieee.org/document/726791 (visited on 07/25/2024).

[41] Yann Lecun. "A Theoretical Framework for Back-Propagation". In: (Aug. 2001).

[42] *LeCun: M uLLER, K.-R. Efficient backprop - Google Scholar*. URL: https://scholar.google.com/scholar_lookup?title=Efficient+backprop&author=LeCun,+Y.&author=Bottou,+L.&author=Orr,+G.B.&author=M%C3%BCller,+K.R.&publication_year=1998&pages=9%E2%80%9350 (visited on 07/09/2024).

[43] Seppo Linnainmaa. "Taylor expansion of the accumulated rounding error". en. In: *BIT Numerical Mathematics* 16.2 (June 1976), pp. 146–160. ISSN: 1572-9125. DOI: 10.1007/BF01931367. URL: https://doi.org/10.1007/BF01931367 (visited on 07/22/2024).

[44] Wolfgang Maass. "Networks of spiking neurons: The third generation of neural network models". In: *Neural Networks* 10.9 (Dec. 1997), pp. 1659–1671. ISSN: 0893-6080. DOI: 10.1016/S0893-6080(97)00011-7. URL: https://www.sciencedirect.com/science/article/pii/S0893608097000117 (visited on 07/05/2024).

[45] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. arXiv:1706.06083 [cs, stat]. Sept. 2019. DOI: 10.48550/arXiv.1706.06083. URL: http://arxiv.org/abs/1706.06083 (visited on 07/26/2024).

[46] Henry Markram, Wulfram Gerstner, and Per Jesper Sjöström. "Spike-Timing-Dependent Plasticity: A Comprehensive Overview". English. In: *Frontiers in Synaptic Neuroscience* 4 (July 2012). Publisher: Frontiers. ISSN: 1663-3563. DOI: 10.3389/fnsyn.2012.00002. URL: https://www.frontiersin.org/journals/synaptic-neuroscience/articles/10.3389/fnsyn.2012.00002/full (visited on 07/26/2024).

[47] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". en. In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602.

DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259 (visited on 07/08/2024).

[48] Claude Meunier and Idan Segev. "Playing the Devil's advocate: is the Hodgkin–Huxley model useful?" English. In: *Trends in Neurosciences* 25.11 (Nov. 2002). Publisher: Elsevier, pp. 558–563. ISSN: 0166-2236, 1878-108X. DOI: 10.1016/S0166-2236(02)02278-6. URL: https://www.cell.com/trends/neurosciences/abstract/S0166-2236(02)02278-6 (visited on 07/24/2024).

[49] Vinod Nair and Geoffrey E Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". en. In: ().

[50] T. Natschläger and B. Ruf. "Spatial and temporal pattern analysis via spiking neurons". eng. In: *Network (Bristol, England)* 9.3 (Aug. 1998), pp. 319–332. ISSN: 0954-898X.

[51] Martin C. Nwadiugwu. *Neural Networks, Artificial Intelligence and the Computational Brain*. arXiv:2101.08635 [cs, q-bio]. Dec. 2020. DOI: 10.48550/arXiv.2101.08635. URL: http://arxiv.org/abs/2101.08635 (visited on 07/05/2024).

[52] Bruno A Olshausen. "What is the other 85% of V1 doing?" en. In: ().

[53] Nicolas Papernot et al. "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks". In: *2016 IEEE Symposium on Security and Privacy (SP)*. ISSN: 2375-1207. May 2016, pp. 582–597. DOI: 10.1109/SP.2016.41. URL: https://ieeexplore.ieee.org/document/7546524 (visited on 07/26/2024).

[54] Paweł Pietrzak et al. "Overview of Spiking Neural Network Learning Approaches and Their Computational Complexities". en. In: *Sensors* 23.6 (Jan. 2023). Number: 6 Publisher: Multidisciplinary Digital Pub-

lishing Institute, p. 3037. ISSN: 1424-8220. DOI: 10.3390/s23063037. URL: https://www.mdpi.com/1424-8220/23/6/3037 (visited on 07/08/2024).

[55] Jary Pomponi, Simone Scardapane, and Aurelio Uncini. "Pixle: a fast and effective black-box attack based on rearranging pixels". In: *2022 International Joint Conference on Neural Networks (IJCNN)*. arXiv:2202.02236 [cs, stat]. July 2022, pp. 1–7. DOI: 10.1109/IJCNN55064.2022.9892966. URL: http://arxiv.org/abs/2202.02236 (visited on 08/19/2024).

[56] Nitin Rathi et al. *Enabling Deep Spiking Neural Networks with Hybrid Conversion and Spike Timing Dependent Backpropagation*. arXiv:2005.01807 [cs, stat]. May 2020. DOI: 10.48550/arXiv.2005.01807. URL: http://arxiv.org/abs/2005.01807 (visited on 07/26/2024).

[57] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65.6 (1958). Place: US Publisher: American Psychological Association, pp. 386–408. ISSN: 1939-1471. DOI: 10.1037/h0042519.

[58] Bodo Rueckauer et al. "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification". English. In: *Frontiers in Neuroscience* 11 (Dec. 2017). Publisher: Frontiers. ISSN: 1662-453X. DOI: 10.3389/fnins.2017.00682. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2017.00682/full (visited on 07/25/2024).

[59] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". en. In: *Nature* 323.6088 (Oct. 1986). Publisher: Nature Publishing Group, pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0 (visited on 07/22/2024).

[60] Abhronil Sengupta et al. “Going Deeper in Spiking Neural Networks: VGG and Residual Architectures”. English. In: *Frontiers in Neuroscience* 13 (Mar. 2019). Publisher: Frontiers. ISSN: 1662-453X. DOI: 10.3389/fnins.2019.00095. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2019.00095/full (visited on 08/15/2024).

[61] Saima Sharmin et al. *A Comprehensive Analysis on Adversarial Robustness of Spiking Neural Networks*. en. arXiv:1905.02704 [cs, eess]. May 2019. URL: http://arxiv.org/abs/1905.02704 (visited on 07/04/2024).

[62] Saima Sharmin et al. *Inherent Adversarial Robustness of Deep Spiking Neural Networks: Effects of Discrete Input Encoding and Non-Linear Activations*. en. arXiv:2003.10399 [cs]. July 2020. URL: http://arxiv.org/abs/2003.10399 (visited on 07/04/2024).

[63] Sumit Bam Shrestha and Garrick Orchard. “SLAYER: Spike Layer Error Reassignment in Time”. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/hash/82f2b308c3b01637c607ce05f52a2fed-Abstract.html (visited on 07/22/2024).

[64] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv:1409.1556 [cs]. Apr. 2015. DOI: 10.48550/arXiv.1409.1556. URL: http://arxiv.org/abs/1409.1556 (visited on 07/09/2024).

[65] Sen Song, Kenneth D. Miller, and L. F. Abbott. “Competitive Hebbian learning through spike-timing-dependent synaptic plasticity”. en. In: *Nature Neuroscience* 3.9 (Sept. 2000). Publisher: Nature Publishing Group, pp. 919–926. ISSN: 1546-1726. DOI: 10.1038/78829.

URL: https://www.nature.com/articles/nn0900_919 (visited on 07/26/2024).

[66] *spikingjelly/docs/source/index.rst at master · fangwei123456/spikingjelly*. en. URL: https://github.com/fangwei123456/spikingjelly/blob/master/docs/source/index.rst (visited on 08/12/2024).

[67] Adam F. Strassberg and Louis J. DeFelice. "Limitations of the Hodgkin-Huxley Formalism: Effects of Single Channel Kinetics on Transmembrane Voltage Dynamics". In: *Neural Computation* 5.6 (Nov. 1993), pp. 843–855. ISSN: 0899-7667. DOI: 10.1162/neco.1993.5.6.843. URL: https://doi.org/10.1162/neco.1993.5.6.843 (visited on 07/24/2024).

[68] Christian Szegedy et al. *Intriguing properties of neural networks*. arXiv:1312.6199 [cs]. Feb. 2014. DOI: 10.48550/arXiv.1312.6199. URL: http://arxiv.org/abs/1312.6199 (visited on 07/05/2024).

[69] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: 2016, pp. 2818–2826. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.html (visited on 07/09/2024).

[70] Florian Tramèr et al. *Ensemble Adversarial Training: Attacks and Defenses*. arXiv:1705.07204 [cs, stat]. Apr. 2020. DOI: 10.48550/arXiv.1705.07204. URL: http://arxiv.org/abs/1705.07204 (visited on 07/26/2024).

[71] Yujie Wu et al. "Spatio-Temporal Backpropagation for Training High-Performance Spiking Neural Networks". English. In: *Frontiers in Neuroscience* 12 (May 2018). Publisher: Frontiers. ISSN: 1662-453X. DOI: 10.3389/fnins.2018.00331. URL: https://www.frontiersin.

org/journals/neuroscience/articles/10.3389/fnins.2018.00331/full (visited on 07/26/2024).

[72] Weilin Xu, David Evans, and Yanjun Qi. "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks". In: *Proceedings 2018 Network and Distributed System Security Symposium*. arXiv:1704.01155 [cs]. 2018. DOI: 10.14722/ndss.2018.23198. URL: http://arxiv.org/abs/1704.01155 (visited on 07/26/2024).

[73] Weilin Xu, Yanjun Qi, and David Evans. "Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers". In: Jan. 2016. DOI: 10.14722/ndss.2016.23115.

[74] Kashu Yamazaki et al. "Spiking Neural Networks and Their Applications: A Review". en. In: *Brain Sciences* 12.7 (July 2022). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 863. ISSN: 2076-3425. DOI: 10.3390/brainsci12070863. URL: https://www.mdpi.com/2076-3425/12/7/863 (visited on 07/09/2024).

[75] Friedemann Zenke and Surya Ganguli. "SuperSpike: Supervised learning in multi-layer spiking neural networks". In: *Neural Computation* 30.6 (June 2018). arXiv:1705.11146 [cs, q-bio, stat], pp. 1514–1541. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco_a_01086. URL: http://arxiv.org/abs/1705.11146 (visited on 07/26/2024).

[76] Shibo Zhou et al. "Deep SCNN-Based Real-Time Object Detection for Self-Driving Vehicles Using LiDAR Temporal Data". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 76903–76912. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2990416. URL: https://ieeexplore.ieee.org/document/9078792 (visited on 07/25/2024).