UNIVERSITY OF CALIFORNIA

SANTA CRUZ

DESIGN AND ANALYSIS OF LEAKY INTEGRATE-AND-FIRE AND MEMRISTIVE INTEGRATE-AND-FIRE SPIKING NEURAL NETWORKS

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Tin Thurein

June 2022

The Thesis of Tin Thurein
is approved:

Professor Sung-Mo "Steve" Kang, Chair

Professor Nobuhiko Kobayashi

Professor Yu Zhang

Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Design and Analysis of Leaky Integrate-and-Fire
and Memristive Integrate-and-Fire Spiking
Neural Networks

by

Tin Thurein

A set of new neuron model and neural network architectures are introduced for the exploration of spiking neural networks to classify handwritten digits and images. Brain-like neural network powered artificial intelligence has become a driving force behind everyday applications from autonomous vehicles, facial recognition applications, to business analytics and energy market prediction. Biological brain-inspired Deep Neural Networks (DNN) have been around for a while. However, spiking Neural Networks (SNNs) have come to the spotlight due to higher energy efficiency and brain-like computing power. In this thesis, various models of spiking neurons such as Leaky Integrate-and-Fire (LIF), and Memristive Integrate-and-Fire (MIF) are examined and explored for applications of spiking neural networks. It has become apparent that large-scale neural network models will not only be in data-centers but also in edge computing devices and embedded systems, thus more energy-efficient and faster neural network types are needed.

Acknowledgments

I would like to thank Professor Sung Mo Kang for providing guidance throughout this research study. I appreciate all the feedback that I have gotten back from Professor Kang. I would also like to express my gratitude to Professor Yu Zhang and Professor Nobby Kobayashi for serving on the thesis reading committee and providing insights and valuable suggestions. Donguk Choi has provided me with many sample simulations done with Cadence tools. He has explained many important concepts behind the neural networks and memristor-based neural networks.

Lastly, I would like to thank my parents, sister, and friends for the support and love, especially during the COVID time. They were always there when I needed them the most. Thanks to my beloved wife for her boundless patience and for having faith in me.

LIST OF ABBREVIATIONS

ASIC Application Specific Integrated Circuit

ANN Artificial Neural Network

CNN Convolutional Neural Network

DNN Deep Neural Networks

GPU Graphics Processing Units

HH Hodgkin-Huxley Neuron Model

HH-T Hodgkin-Huxley Neuron Model using look-up tables

IZH Izhikevich Neuron Model

LIF Leaky Integrate-and-Fire Neuron Model

MIF Memrisitive Integrate-and-Fire Neuron Model

MIF2 Memrisitive Integrate-and-Fire 2 Neuron Model

MSE Mean-Squared Error

NEF Neural Engineering Framework

NVN Non-von Neumann

ReLu Rectified Linear Unit

RNN Recurrent Neural Network

SNN Spiking Neural Network

TPU Tensor Processing Unit

VLSI Very Large Scale Integration

Chapter 1

Introduction

With the advancement of Application Specific Integrated Circuit (ASIC) hardware and software libraries, together with the availability of large datasets and the development of high-performance computing systems, neural networks have pushed the boundaries to enable many new AI-powered applications. The advanced artificial neural networks, inspired by biological neuron models, have revolutionized the entire landscape by solving difficult and diverse machine learning problems of today. While data-centers have ample computing power for training large datasets, edge devices such as laptops and mobile phones have limited resources in terms of battery power and computation for inference and operating large-scale neural networks. Though the semiconductor industry has followed Moore's law to increase power efficiency and computation power, a completely new paradigm shift is now required. Therefore, many researchers have ventured into various forms of neural networks, computing devices, and Non-von Neumann (NVN) architecture. Spiking Neural Network (SNN) is a third revolution in the AI space where neurons exchange the information in

encoded spike patterns in the spatio-temporal domain. This type of neural network is of much more resemblance to a biological brain network.

1.1 Motivation

The goal of this thesis is to compare various network architectures (shallow vs deep), compare the training times and model accuracies. The use of GPU training time allows us to get a sense of required computation power and time. We will use the classical standard datasets in this thesis to explore, implement and compare various non-spiking activation such as the Rectified Linear Unit (ReLU) and spiking neuron models such as Leaky-Integrate and-Fire (LIF), Izhikevich, and Memristive Integrate-and-Fire (MIF). The standard dataset consists of *MNIST* [23] (28x28 pixel of hand-written digits), and a more complex image dataset such as *CIFAR10* [22](32 x 32 x 3 color channels images of 10 image classes, such as truck, automobile, and so on).

The majority of the AI-powered applications are based on some type of "classical" or non-spiking neural networks. Research efforts have been put toward designing AI accelerator for training and inference applications for non-spiking neural networks. In this thesis, we will explore single spiking neuron behaviors, various architectures of spiking neural networks and spiking neuron types. Much inspiration came from Xiaoyang Jia's thesis "Design and Training of Memristor-Based Neural Networks" [17] which explored implementing memristor as synapses and demonstrated MNIST image classification and Yinghao Shao's thesis on the "Application of Memristive Device Arrays for Pattern Recognition" [30].

1.2 The Need for More Efficient Computing Devices

Huge volumes of data are being generated each day while data centers and data warehouses are unable to keep up with the increased volume and algorithmic data analysis. With Von Neuman's architectural bottleneck in communication between the computing and the memory units, the new research focus more on ASIC and exploring different computing architecture especially for brain-inspired-computation to reduce power consumption. Once a network has been trained, inference can be done with low-power devices to compute the results.

1.3 Artificial Intelligence Neural Network (ANN)

ANN or feed-forward network is a very generic neural network architecture in which each neuron of a layer is fully connected to every other neuron in the next layer. In ANN is also capable of learning any nonlinear function thus it can be used as a universal function approximator. While ANN is good for certain applications, it lacks in learning spatial features such as image pixels. It has been known in the machine learning community that the optimality of neural network architectures can vary significantly from one application to another. For instance, CNN architecture is deployed widely across the computer vision space while Recurrent Neural Networks (RNN) are deployed in applications such as time series analysis, temporal, or sequential information such as voice recognition.

1.4 Convolutional Neural Network (CNN)

One of the drawbacks of using ANN for applications such as image recognition is that the sheer number of required neurons would explode exponentially with the increase in the input image pixel counts thus making the scaling and real-time image recognition much harder. CNN is one of deep learning algorithm that used mainly in computer vision and image recognition. It has some special hidden layers that have multiple convolution layers, pooling layers, batch normalization layers, drop-out layers, and fully connected layers. Convolution layers are feature extraction layers where a portion of a 2-D image is scanned across to define edges and extract smaller details.

1.4.1 Convolution layer

The heart of the convolution layer is *Kernel* and output filter. The *Kernels* are a window that scans across the 2-D or 3-D (with RGB color channels) input image or previous output of the convolution layer to further extract the features [34].

1.4.2 Filters and Kernel

The network models are built using *Keras* high level API for 2D convolution layer. Some key terminologies are define below:

filters = the number of the output filters

kernel size = 2D convolution window, usually 3x3, 5x5 or 7x7

strides = stride of the convolution window

epoch = one iteration to go through the entire dataset

batch = the training dataset is split into chunks of data called "batch"

1.4.3 Gradient descent and loss function

A key component in the supervised learning model is the gradient descent and the loss function. Gradient descent describes a mathematical equation that is used to update the model's weight and biases. The loss function can be thought of as an error between what the output neuron predicts and the ground truth. A popular loss function is a Mean Squared Error (MSE) which can be defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{1.1}$$

where Y_i is the ground truth and $\hat{Y}_i$ is the predicted value.

1.4.4 Defining accuracy

As a method of defining accuracy, we use built-in function called sparse categorical cross entropy [36] by using the correct output label and our network prediction which computes the error for the gradient descent function.

1.4.5 Putting it all together

During the training process, the data is split and grouped into smaller chunks called batches. One epoch is where the entire training data set is passed through the neural network once. One of the important hyperparameters for any gradient descent network is called "learning rate", which defines how much error in the network prediction can affect the rate of which the network weights are being updated. By default, the learning rate can be varied from 10^{-6} to 10^{-3} or 0.0001% to 0.1%[6]. The network passes through multiple

epochs so that each time it passes through, it can extract features and learn more each time. The process for the training has 4 individual parts and repeats for each piece of data: **1)** Forward pass where the input data is fed to the network; **2)** Calculating loss function such that the network minimizes the error between the ground truth and the calculated output; **3)** Backward pass to determine the weights to be updated; **4)** Updating the weights accordingly with the learning rate.

During the test phase, the network uses a new set of data that it has never seen before. This is to ensure that the network design is robust and can be used for any generalized-use cases.

1.5 Spiking Neuron Types and Spiking Neural Networks

One important distinction between Deep Neural Network (DNN) and Spiking Neural Network (SNN) is how the input data is moved and how the output is computed. In DNN, the input data is fed forward all through the layers. However, the SNN carries event-based spikes information thorough time. For the output, DNN computes all at once with certain probability. However, SNN's outputs are gradually achieved over time. Spiked-based computing can be more energy-efficient due to sparse computation and use of hardware that is specifically made for spike-based computation [24]. Multiple papers have pointed to the reduced network performance due to shallower network architecture (typically less than 5 or so hidden layers) [29]. We can verify this with some of the more complex classification dataset such as CIFAR10.

1.5.1 Leaky Integrate-and-Fire (LIF) Neuron

Although neural networks are modeled after the biological brain, a challenge arises with the way neuron-to-neuron communications are done with the current implementation of neural networks. Many neuron models have been developed to mimic the biological neurons that can be computationally simple and reproduce properties of real neurons. Integrate-and-Fire is a neuron model that was first proposed by Louis Lapicque in 1907, long before the advancement of electronic hardware technology [3]. This neuron model is meant to mimic biological neurons, consists of a capacitor and a resistor in parallel. However, the Leaky Integrate-and-Fire (LIF) model has been further developed to include a current source, capacitor, and a leaky resistor as shown in the figure below.

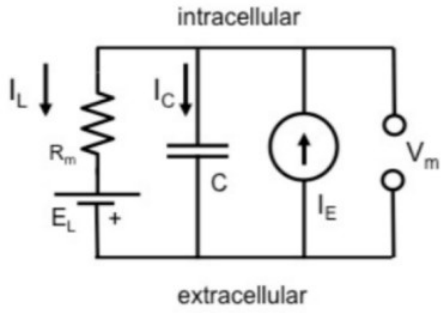


Figure 1.1: Equivalent RC model of LIF neuron [26]

As the model is made up of electrical components, a differential equation can be written as:

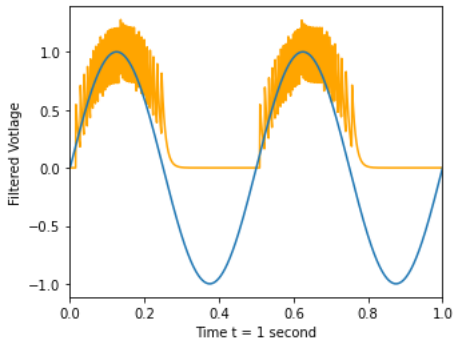
$$I_E(t) = C\frac{dV_{\mathrm{m}}(t)}{dt} + \frac{V_m(t) - E_{\mathrm{L}}(t)}{R_{\mathrm{m}}} \tag{1.2}$$

Solving for $V_m(t)$, we get:

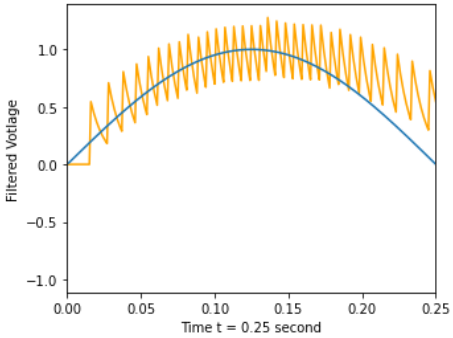
$$V_{\mathrm{m}}(t) = E_L + (V_0 - E_L)\, e^{-t/\tau_m} \tag{1.3}$$

The equation describes a mathematical model where $V_m(t)$ is the membrane potential, R_{m} the membrane resistance, C_{m} the membrane capacitance, τ_m the membrane time con-

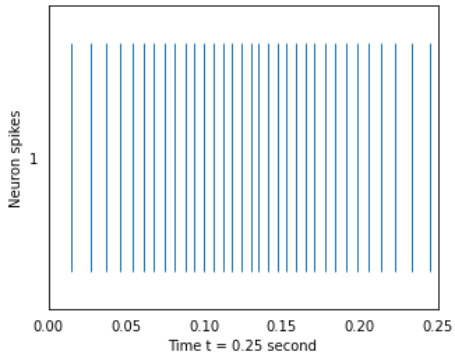
stant and V_0 is the voltage of $V_m(t)$ at time t = 0. In the absence of input current I_E, the model relaxes exponentially toward E_L. When $V_m(t)$ reaches the threshold voltage, a spike is registered and the voltage gets reset to E_L.



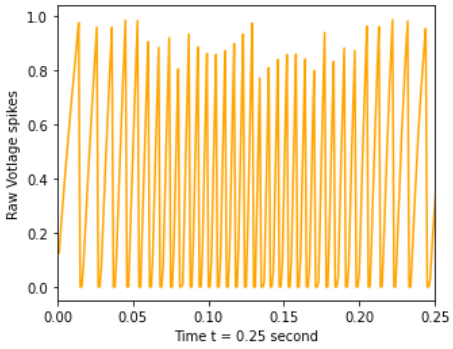
(a) Sine wave input and single neuron output for 1 second duration



(b) Filtered output spikes for t = 0.25 second duration



(c) Spike events for t = 0.25 second duration



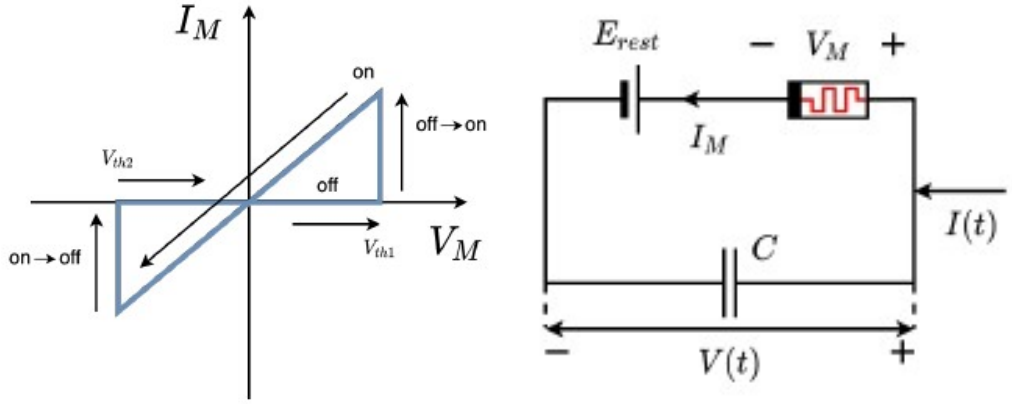
(d) Raw voltage spikes for t = 0.25 second duration

Figure 1.2: A single LIF neuron responding to sine wave input

A simulated single LIF neuron with voltage spikes responding to an input sine wave can be seen above. For the easier distinction between various neuron types, the color palette of orange is chosen for the LIF neuron type in the following chapters.

1.5.2 Memristive Integrate-and-Fire (MIF) Neuron

Memristor was first discovered by Leon Chua as the fourth fundamental circuit component [8] and further generalized by Chua and Kang for memristive systems [9]. Memristor has a hysteresis property whose resistance depends on the magnitude and direction of the applied voltage. Internally it has two states: R_{off} and R_{on} depend on two threshold voltages, V_{th1} and V_{th2} respectively. V-I characteristics of memristor can be seen below:



(a) I-V curve of a memristor [18] (b) Equivalent RC model of a MIF neuron [18]

Figure 1.3: Memristor I-V curve and MIF neuron RC model

Memristive Integrate-and-Fire is a new type of spiking neuron model that is presented in the paper by Kang, et al. [18]. This new model replaces the leaky resistor in LIF with a memristor device.

We can derive the equation as such:

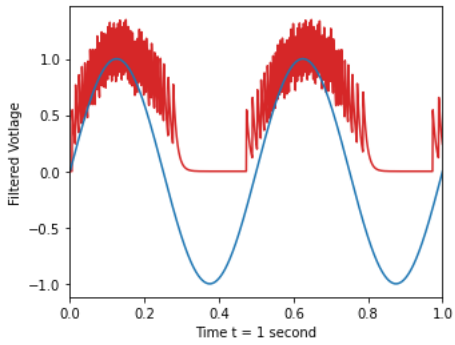
$$I(t) = C\frac{V(t)}{dt} + \frac{V(t) - E_{\text{rest}}}{R_{\text{M}}(x, V(t))} \tag{1.4}$$

where $R_{\text{M}}(x, V(t))$ is a voltage controlled memristance. For simple solution of the differential equation 1.4, we can assume that the state of the memristor does not change and

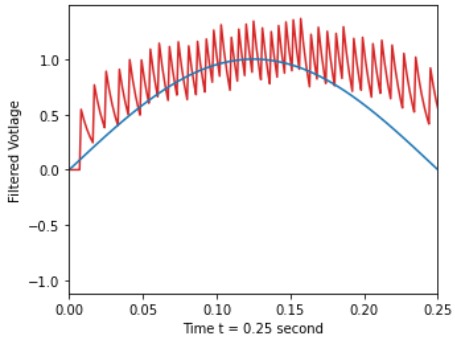
I(t) is a constant I_0, which yields:

$$V(t) = V_{\text{rest}}\, e^{-\frac{t}{\tau}} + (E_{\text{rest}} + R_{\text{off}}\, I_0)\left(1 - e^{-\frac{t}{\tau}}\right) \tag{1.5}$$

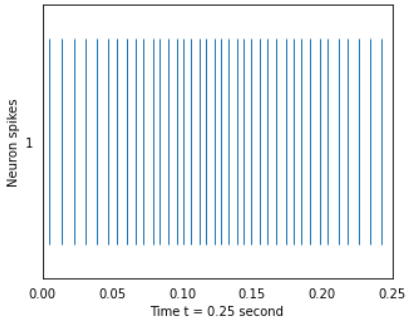
with $\tau = R_{off}C$. A simulated single MIF neuron with voltage spikes responding to an input sine wave can be seen below.



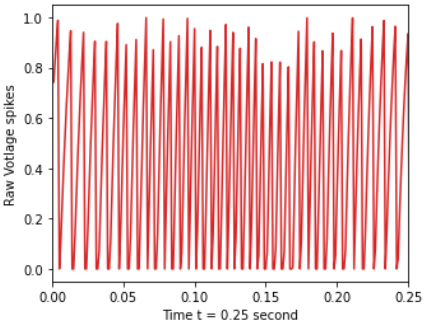
(a) Sine wave input and single neuron output for 1 second duration



(b) Filtered output spikes for t = 0.25 second duration



(c) Spike events for t = 0.25 second duration



(d) Raw voltage spikes for t = 0.25 second duration

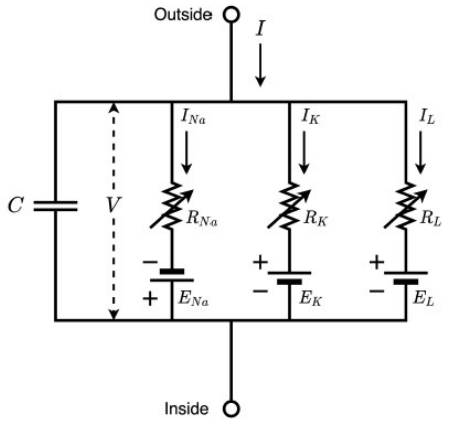
Figure 1.4: A single MIF neuron responding to sine wave input

The color palette of red is chosen for the MIF neuron type in the following chapters.

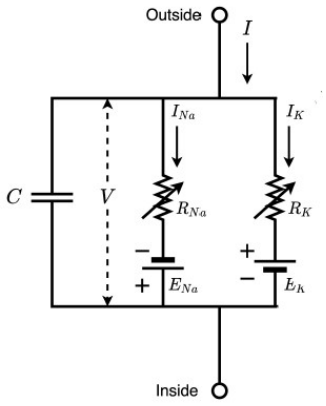
1.5.3 Memristive Integrate-and-Fire 2 (MIF2) Neuron

The Hodgkin-Huxley (HH) neuron model is a mathematical model that depicts how action potentials are generated in neurons with a series of equations [13]. In HH mode, the

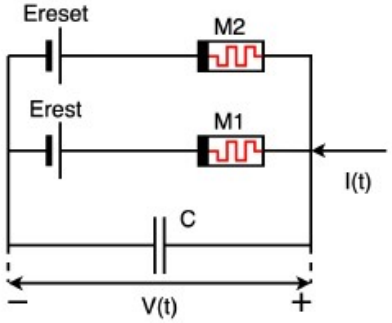
three conductance terms or ion channels are voltage gated. Later, HH model was reduced to two ion channels by setting $R_L = 0$, which provides the best fit for human cardiac action potentials [18].



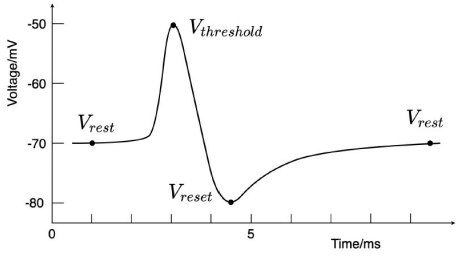
(a) Hodgkin-Huxley (HH) neuron model [18]



(b) Reduced HH model [18]



(c) Equivalent RC model of a MIF2 neuron [18]

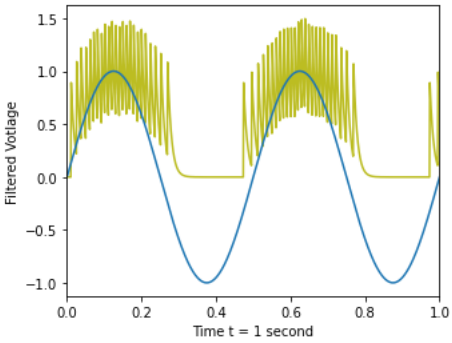


(d) Action potential of a neuron [18]

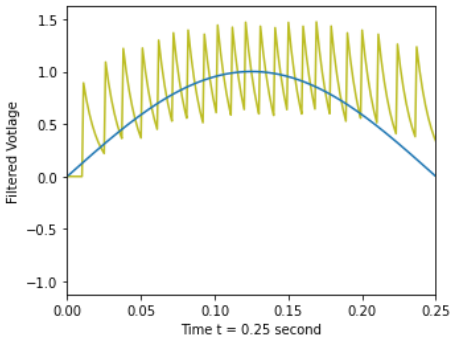
Figure 1.5: HH model and equivalent MIF2 neuron model

MIF2 is an extension to MIF neuron model with additional voltage source E_{reset} and memristor M_2. The additional voltage E_{reset} raises the threshold voltage with V_{reset} during refractory time, which accurately describes a biological neuron. MIF2 model state equation can be written as:

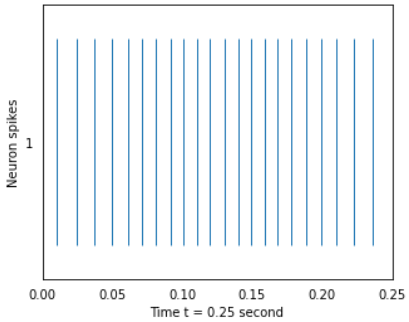
$$I(t) = C\frac{V(t)}{dt} + \frac{V(t) - E_{\text{rest}}}{R_{M1}\left(x_1, V(t)\right)} + \frac{V(t) - E_{\text{reset}}}{R_{M2}\left(x_2, V(t)\right)} \tag{1.6}$$



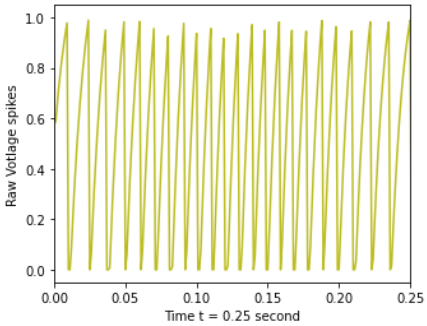
(a) Sine wave input and single neuron output for 1 second duration



(b) Filtered output spikes for t = 0.25 second duration



(c) Spike events for t = 0.25 second duration



(d) Raw voltage spikes for t = 0.25 second duration

Figure 1.6: A single MIF2 neuron responding to sine wave input

Chapter 2

Simulation Frameworks and Test Datasets

The key objective of this thesis work is to explore neuromorphic spiking network architectures using two datasets for a specific task - image classification. The thesis work is presented to examine as follows: Chapter 2 presents simulation frameworks, python libraries and datasets; Chapter 3 presents MNIST classification results using various types of neurons, associated experiments with various network architectures and ANN-SNN conversion techniques; Chapter 4 follows with much more difficult CIFAR10 classification using shallow and deep neural network architectures; Chapter 5 concludes the thesis work along with future work in Chapter 6, and lastly appendix with simulation results for various learning rates and neuron types.

2.1 Neuromorphic Computing Platforms

The Neural Engineering Framework (NEF) is a method of building large-scale biological neural models [32]. Although the majority of the neural network applications are powered by GPUs and ASICs, none of the commercially available hardware is optimized for the brain-like computing. The Tensor Processing Unit (TPU), for example, is a highly optimized machine learning ASIC, made for running *TensorFlow* code much more efficiently than GPU counterparts on watt-to-watt comparison for training large datasets and large batches, but GPUs are more versatile and can achieve higher throughput on smaller batches [38]. On the other hand, *Loihi* is Intel's research chip for neuromorphic computing platforms and is capable of mimicking biological neural computation based on event-based Spiking neural networks [10]. Although we can simulate the Spiking behaviors of the neural networks and construct spiking neural network architectures with the von Neumann architecture-based GPU, a highly efficient neuromorphic computing platform such as Loihi can dramatically speed up the parametric learning, and more research needs to be done for implementing efficient computation on the edge devices.

2.2 Software Libraries and Dependencies

Current implementation of the neural network most relies on matrix multiplication and thus requires raw parallel computational power for a reasonable convergence time. This thesis will rely on using hardware-accelerated libraries such as TensorFlow and Keras. Since computation done on GPU is more efficient for training, Google's Colab with Jupyter notebook-style is utilized as it runs entirely on the Google servers. A spiking neuron model

from the Keras library is also examined. Keras has a spiking activation function that mimics the spiking neuron behavior where neurons only mimic an equivalent spiking activation function when forwarding to the next layer, and a non-spiking activation function when calculating the gradient descent. This will become important as we will see in the future chapters for network implementations. A few other python libraries are also imported for numerical calculations, image augmentation, error metrics, and plotting graphs.

For training the neural networks, we will use the Tesla K-80 GPU accelerator across all spiking and non-spiking neural networks so that we can get a sense of training time for each network[2]. TensorFlow is the underlying library that has both high-level and low-level APIs to run the models and Keras is built on top of TensorFlow.

2.3 Nengo

Nengo is a python framework package that is built specifically to experiment with a brain-inspired neuromorphic computing with pre-defined neuron types such as LIF, and Izhikevich neuron models [1][5]. *Nengo* can also be extended by adding various custom neuron types to simulate the relationship between an input voltage and output spikes. While we can simulate various types of neurons and network models on GPUs, the same network can also be deployed on *Loihi* hardware [10] and the hardware developed by the SpinNAKER project aiming to simulate a large spiking network in real-time [11].

One of the challenges with the spiking neuron model is that it has a temporal component. A neuron outputs a spike when a certain threshold is reached at a certain time. Therefore, we need to incorporate a time component in the dataset for us to visualize any spiking behavior.

During training, the network's weights get updated by utilizing back-propagation. However, for a spiking neural network, the output has spiking noise at different time steps which led to an implementation of ANN-SNN conversion technique during training where the output neurons behave much like *Rectified Linear Unit (ReLU)* and switch to spiking neurons at the testing phase with the trained weights. While this technique has been proposed previously in the papers presented by Cao et al. [7] and Querlioz et al. [27], we face challenges with deeper network architecture using this technique.

While *Keras* and *Nengo* has built-in conversion technique for ANN-SNN conversion in an almost loss less manner, there are still some limitations such as converting max-pooling layers (to reduce dimensionality), and batch-normalization layers in the model. The auto conversion technique from Keras and Nengo needs manually adjusting of post-synaptic spikes by filters or firing rates.

2.4 MNIST Training Dataset

As mentioned above, we will use various standard datasets to train and test our network performance from an accuracy and training time standpoint. MNIST dataset includes 70,000 images, where 60,000 images are used toward training the network, while test images consist of 10,000 hand-written digits. MNIST consists of a single color channel of hand-written digits, ranging from 0 to 9, where it is relatively easier for the neural network to learn the shapes and features with just a few hidden layers. The image pixel contains 8-bit values range from 0 to 255 and to pre-process the image data, we need to scale the pixel values by 255 to get values ranging from 0 to 1. These values are then processed by the input layers where the activation neurons respond to the values by registering a spike once the threshold value is reached.

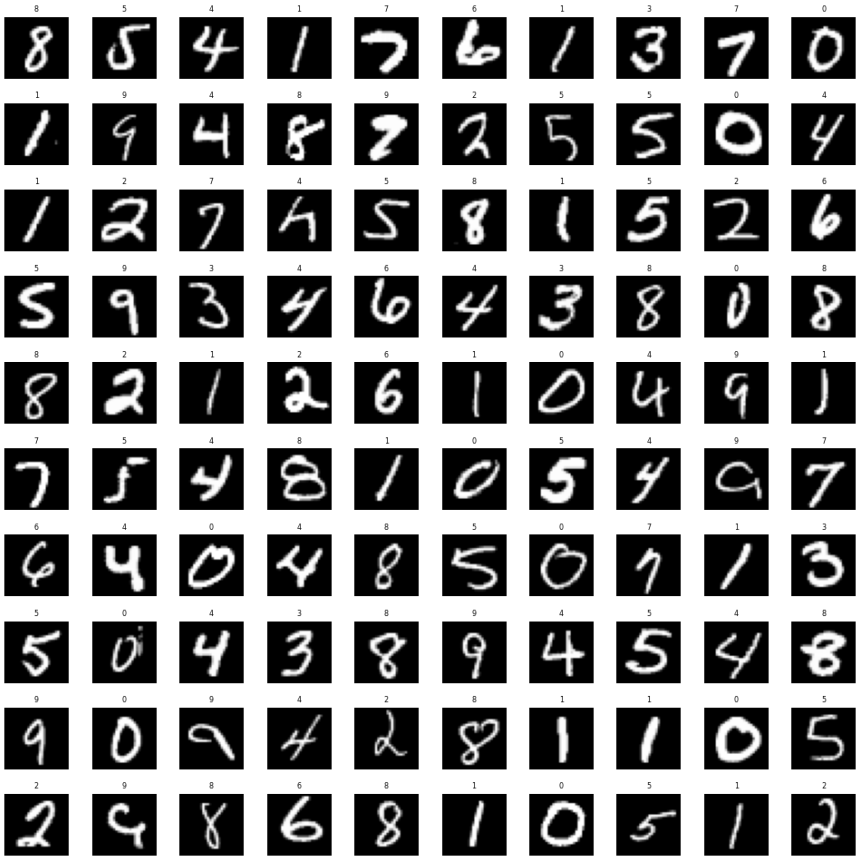


Figure 2.1: Sample Images of MNIST dataset with labels

MNIST handwritten digits for distribution of each digit can be seen below and the distribution seems fairly balanced.

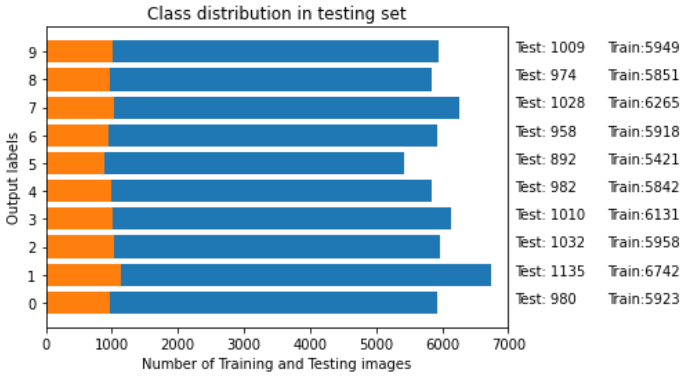


Figure 2.2: Distribution of MNIST training and test dataset

2.5 CIFAR10 Training Dataset

CIFAR10 dataset with sample images is shown below. As we can see, the dataset has 10 different classes of images: 'airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck'. Similar to the other standardized dataset, CIFAR10 consists of 60,000 images with 50,000 images allocated for training purposes and 10,000 for testing. CIFAR10 image classes are made up of 32x 32x 3 (color channels) where the images sometimes include backgrounds and other objects such as grass, water, windows, shadows, etc. This makes a much harder problem for the neural network to classify which usually requires multiple feature extraction layers, and a large array of training image sets.

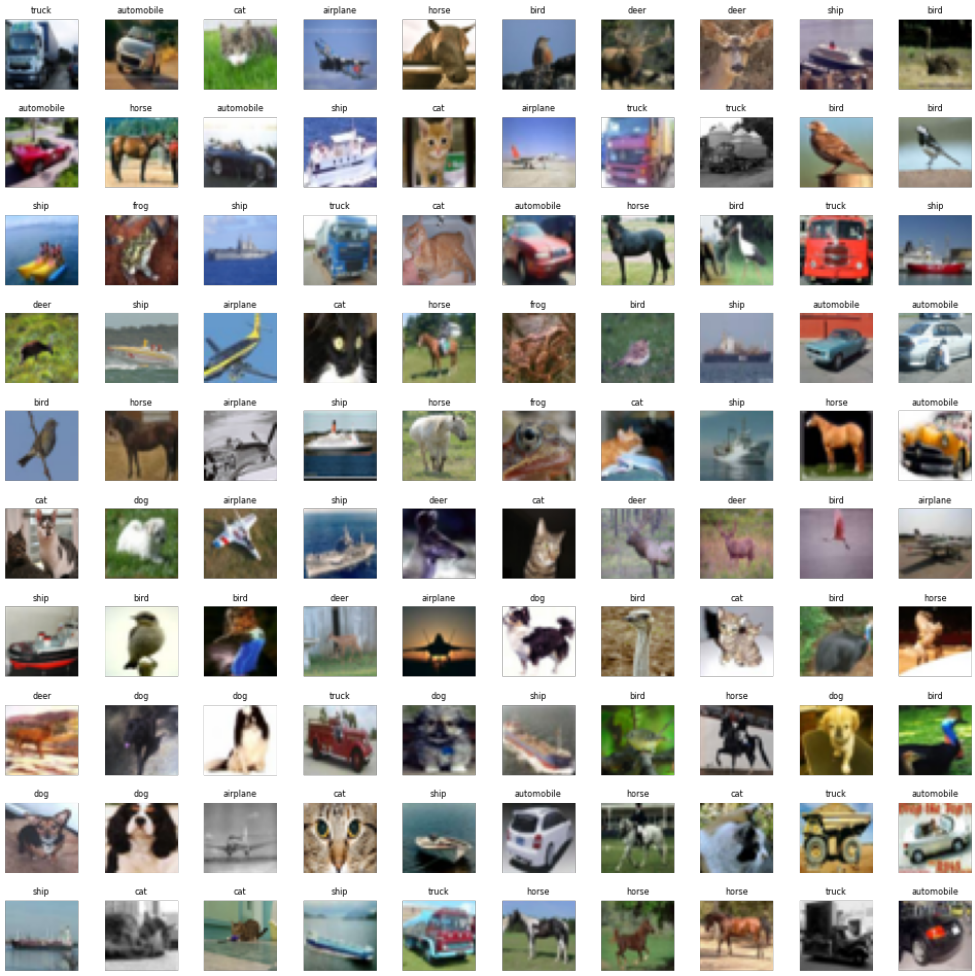


Figure 2.3: CIFAR10 dataset with image labels

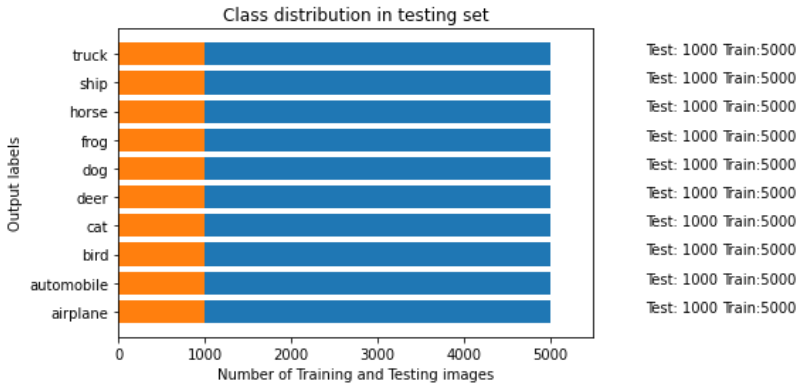


Figure 2.4: CIFAR10 distribution

Chapter 3

MNIST Classification Results

3.1 MNIST Classification Network Architecture

In this chapter, we explored three paths:

1. Building Keras model to transform non-spiking model to spiking model and compare the results. The color palette of brown is chosen for non-spiking relu neuron and green for spiking relu neuron type. (chapter: 3.2)

2. Using Nengo's built-in LIF neuron type to compare model's performance (chapter: 3.3). The color palette of orange is chosen for LIF neuron type network performance results.

3. Using MIF neurons in the neural network (chapter: 3.4) The color palette of red is chosen for MIF neuron type network performance results.

For all 3 paths, the basic network is built using 3 convolution layers with fixed 3x3 Kernel window size: first convolution layer consists of 28x28 neurons, a second convolution layer

with 26x26 neurons, and the last convolution layer with 12x12 neurons and a fully connected output layer. The entire network is trained through 30 epochs of the training dataset and 30 time steps to compute output. The network architecture has the following parameters as default:

Table 3.1: SNN Model Parameters

	Value
Learning Rate	0.1%
Max Neuron Spiking rate	100Hz
Input Layer	28x28 neurons
1st conv2D Layer	320 parameters
2nd conv2D Layer	18496 parameters
3rd conv2D Layer	73856 parameters
Output Layer	10 fully-connected neurons, 32010 parameters

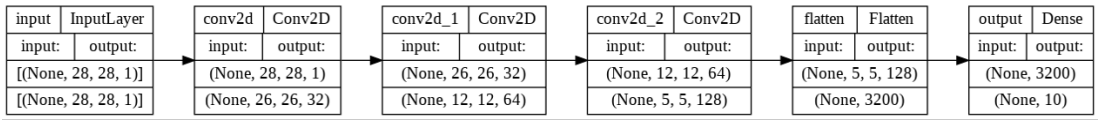


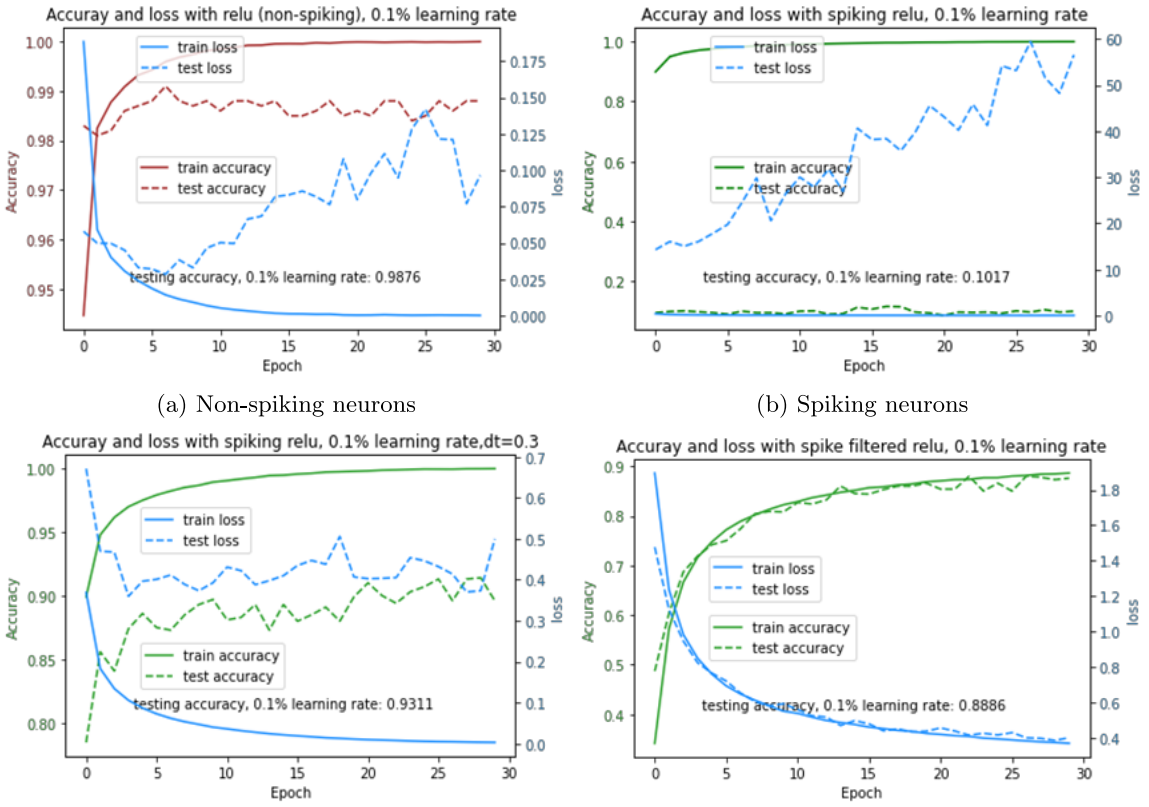
Figure 3.1: Basic Network Architecture for MNIST classification

3.2 Classification Results with Keras Spiking and Non-spiking Neurons

In this section, we explore how the spiking neural network will perform and the technique involved with the spiking network. During training, the network uses appropriate activation

functions such as *relu* to calculate the weights and biases. To convert from non-spiking neuron activation to spiking neuron, we first need to train the model using built-in activation functions such as *relu*. Once the network has been trained with the desired accuracy, we can call Keras API to swap the activation function to spiking neuron. While the weights and biases do not get influenced by the change of activation function, the output results can change dramatically. While the MNIST dataset is relatively straightforward for a non-spiking neuron, we find the drop in accuracy once we swap to the spiking version, largely due to the spiking nature of the output train. We can further refine the network performance by increasing the spike rate and adding post-synaptic low-pass filtering. Also note that training epochs are limited to 30, though some network architecture could potentially achieve higher accuracy by training longer with more epochs.

Non-spiking and spiking neuron architecture are shown in Figure 3.2. We can observe from the graphs that: **i)** swapping to spiking neuron can drop in network accuracy dramatically **ii)** though the network performs well with the training dataset, test accuracy can be quite low (largely due to slower neuron spiking thus losing information and over-fitting) **iii)** higher neuron spike rate can be achieved by turning dt which is the time step of a spiking neuron, effectively binning the neuron activities longer and **iv)** filtering the output can help with test accuracy during training and validation.



(a) Non-spiking neurons

(b) Spiking neurons

(c) Spiking neurons with spike rates adjusted

(d) Spiking neurons with post-synaptic filters

Figure 3.2: Non-spiking and spiking neurons with Keras model

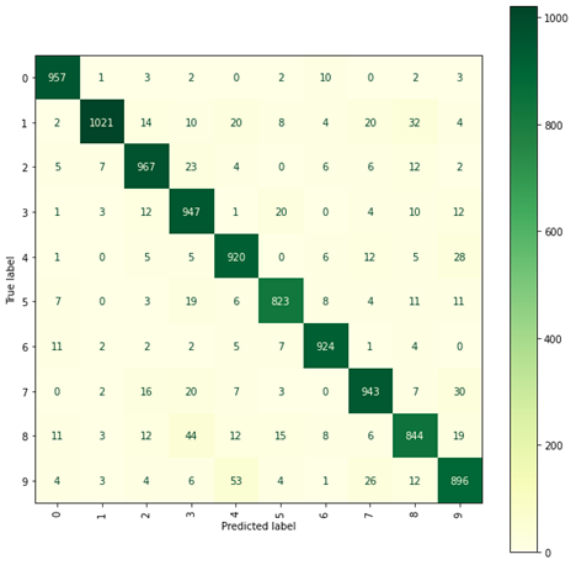
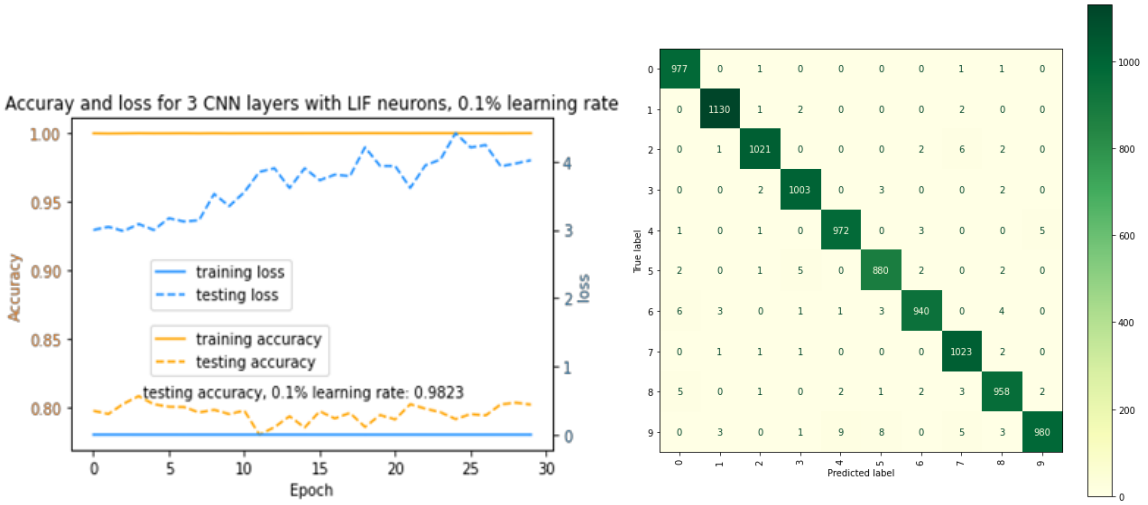


Figure 3.3: Matrix of correctly and incorrectly identified MNIST image with post-synpatic filtered Keras Spiking neuron

An observation can be made that though spiking neural network doesn't perform as well as the non-spiking counterpart initially, we can achieve 90+ % accuracy by fine-tuning some of the spiking behaviors and post-processing.

3.3 Classification Results with *Nengo* LIF Neurons

Unlike Keras spiking model, we do not have to swap between spiking and non-spiking activation functions on the neural network because Nengo does this automatically for us. Also, we can add an additional output layer with the synaptic filter to further refine the network performance which can be seen at Figure 3.5 and Figure 3.6. Sample classification of MNIST hand-written test dataset is shown below in the appendix section A.2. We can probe the output neurons at each time step and observe that the neurons make a correct prediction for the majority of the dataset with an overall accuracy of 98%. To save time during each training and validation epoch, the model gets validated using a smaller subset of the test dataset of 1000 images.



(a) Accuracy and loss for LIF neuron

(b) Matrix of correctly and incorrectly identified image counts for LIF neurons

Figure 3.4: MNIST results for LIF neurons

One of the most important hyperparameters for any neural network is the learning rate [6], which states how the model weights are computed based on the output gradient and error. We can observe in appendix section A.2 that spiking neural networks can be resilient to a higher learning rates ($> 0.01\%$) which could translate to lower training time and a wider range of learning rates.

From the confusion matrix above, we observe that it is less likely for a digit 0 to be incorrectly classified (only 3 incorrect outputs) whereas a digit 9 can be 10 times more likely to be incorrectly labeled. Overall, the network strikes a good balance between accuracy and over-fitting which is common in neural network training.

3.3.1 Incorrect classifications with LIF spiking nuerons

Although the network seems to be robust and has good overall accuracy, at times it has difficulties identifying certain digits like shown below. We can observe that the filtered

output neurons have more than one distinct spikes whereas the second spike is likely the correct digit. If the network performance is based on Top-2 identification, the error rate can be less than 1%.

The second graph also shows that the unfiltered output neurons can be quite noisy, indication that low-pass filtering is needed to smooth out the signals. Once the output signals are filtered, the predictions can be made by taking the highest output signal.

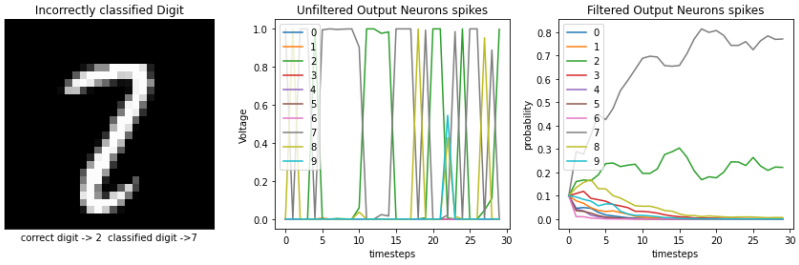


Figure 3.5: Incorrectly classified handwritten digit as 7

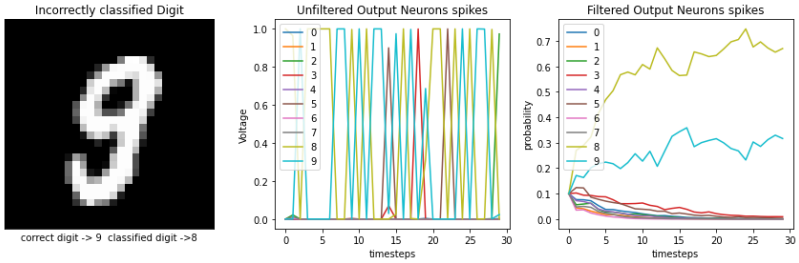


Figure 3.6: Incorrectly classified handwritten digit as 8

3.3.2 Training time with LIF Neuron

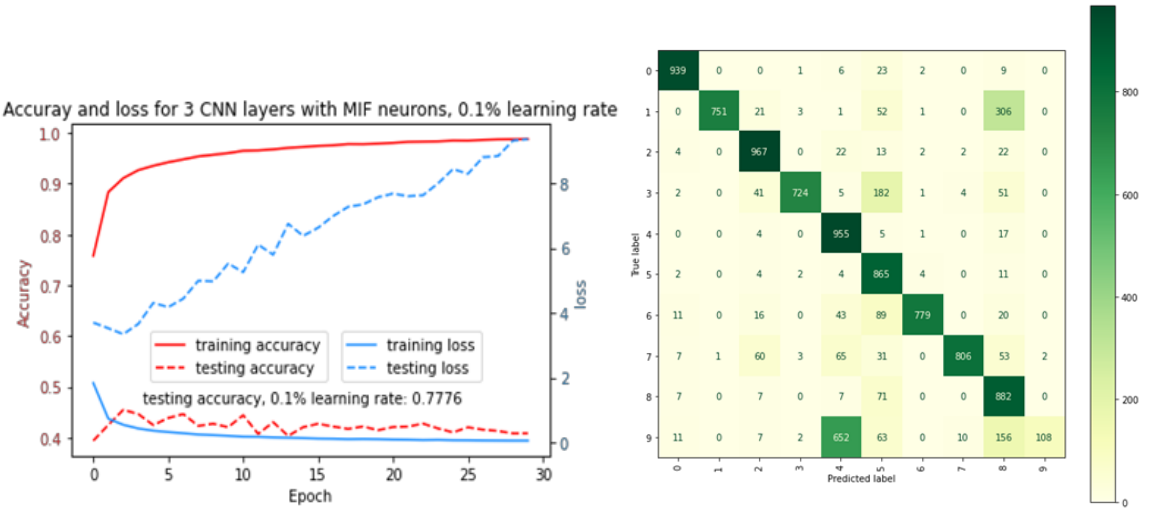
In terms of training time, it took the Google Cloud Tesla K-80 GPU roughly 200 seconds (3.3 minutes) to train the network with 30 epochs to achieve an accuracy of 98%. The state-of-the-art MNIST classification using CNN can achieve up to 99.97% by the researchers at Sogang University [4]. The state of the art CNN classification network is much more complex compared to our network with 3 convolution layers and a lot more training time as well

as training parameters thus we can conclude that spiking neural network can be a good alternative for smaller image classification problems.

3.4 MNIST Classification with MIF Spiking Neuron Network

The MIF neuron type described in Section 1.5.2 is implemented in replacement of LIF neuron to the 3 layers CNN with the same parameters. The value of $R_{off} = 10M\Omega$ and $C = 0.2nF$ is used for voltage characteristic Equation 1.5.

While the training dataset accuracy is quite good, the testing accuracy lags behind. A quick observation can be made that digit 9 is the most likely number to get incorrectly identified by the network.



(a) Accuracy and loss for MIF neuron

(b) Matrix of correct and incorrect classification for MIF neuron

Figure 3.7: MNIST results for MIF neurons

In terms of training time for MIF neurons, it took the Google Cloud Tesla K-80 GPU

roughly 60 minutes to train the network with 30 epochs to achieve an accuracy of 78%. The MIF neuron type is more computationally intensive compared to the traditional LIF neuron. The decrease in accuracy could be contributed to the higher neuronal activity of the MIF neuron thus a lot more spikes as we can see in Figure: 1.4. Although in the previous section where we swap non-spiking neurons to spiking neurons, adjusting to a higher neuron spike rate can help the network performance by extending the output neurons' time-step longer, our MIF neurons are excited all throughout the network activation layers. Therefore, the neurons spiked more frequently and the overall network is nosier.

An implementation of MIF2 neuron described in the paper by Kang et al. [18] could be an alternative solution to our MIF neuron where it has additional reset voltage and sub-reset voltage level. Having a secondary reset voltage helps to drive the neuron membrane voltage below sub-rest potential, effectively raising the threshold voltage to fire another spike during the refractory period thus lowering the firing rates and can potentially be used as rate adaptive neuron.

3.5 Summary of MNIST Classification Results

A summary table of various techniques and network performance accuracy is provided below. We can take note that while a non-spiking neural network can achieve the highest accuracy among all, the SNN with LIF neuron can achieve similar performance with lower training time. ANN-SNN conversion technique could work well with the fine-tuning of post-synaptic filtering or spiking behaviors. Since probing at the output neurons directly is much noisier at each time step, all output layers of SNN will require some form of filtering, which

in our case is low-pass filtering. As we will see in chapter 4, high firing rates with MIF neurons can sometimes defeat the SNN purpose of encoding information in spike trains.

Table 3.2: Summary of MNIST model accuracy

	Neuron Type	Layers	Training Time [minute]	Accuracy	Remarks
non-spiking	relu	3	5.2	98.76%	
ANN-SNN conversion	relu-spiking relu	3	8.2	88.86%	Filtered output
ANN-SNN conversion	relu-spiking relu	3	8.2	93.11%	Spike rates adjusted
SNN	LIF	3	3.3	98.23%	
SNN	MIF	3	60	77.76%	

Chapter 4

CIFAR10 Classification Results

4.1 CIFAR10 Classification with LIF Spiking Neuron Network

The CIFAR10 dataset proves to be much more difficult for the neural network to correctly identify each class. Our baseline non-spiking deep neural network has 21 hidden layers and the computation was quite intensive. While SNN are more attractive for energy efficiency, the network architecture has been limited to shallow networks and more straightforward classification problems[29]. The work done by Tavanaei et al. [33] has summarized the recorded spiking neural network performance with various techniques involving versions of back-propagation techniques on MNIST and CIFAR-10 dataset. Some of the SNN performances on MNIST are equally impressive as the non-spiking DNN counterpart. However, when it comes to more complex CIFAR-10 dataset, most neural networks struggle to get comparable results to some of the state-of-the-art DNN results [33].

A few techniques were used to improve the network performance and robustness of the SNN which include:

Data Augmentation: both training and testing data have been transformed using *Keras's* built-in *ImageDataGenerator* [35]. The images are randomly rotated up to ± 15 degrees, randomly shifted by 3 pixels in x and y direction and randomly flip image horizontally. This is to help improve network resiliency from over-fitting and increase in generalization which we can see by comparing Figure 4.3g to Figure 4.3c where overall test accuracy increased slightly.

Pooling technique: A technique used to reduce the input dimensions of the convolution layer by ways of taking an average over a fixed window size or maximum value [19]. In our case, we use the pooling layer to "bin" the neuron spikes over the time steps. This is to help with slower spiking neurons.

Batch-normalization technique: A technique involving normalization of the previous layers so that inter-dependency between each layer is reduced, thus more robust network [20][15].

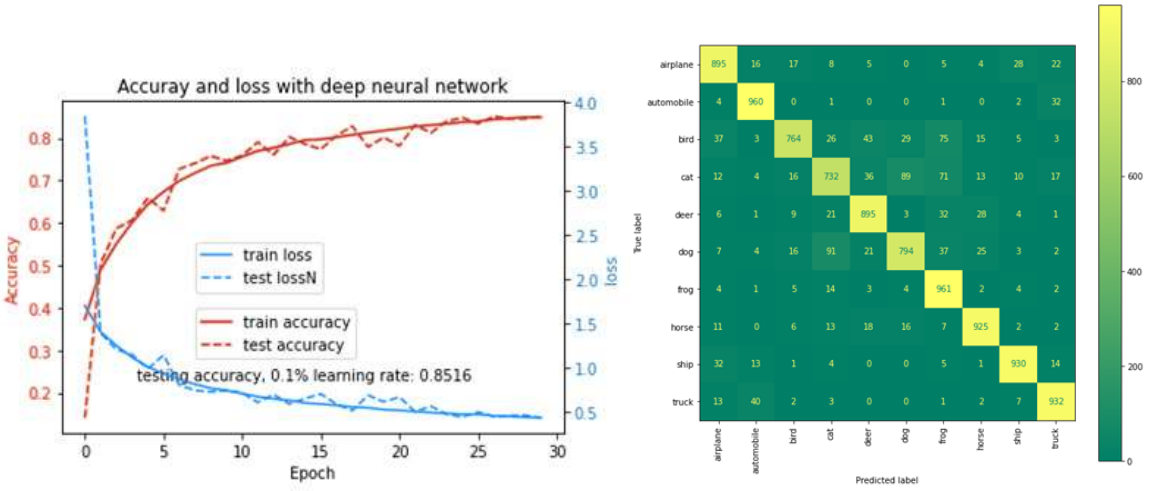
Dropout technique: A technique to regularize networks by randomly select neurons to be dropped out during training. This prevents model from over-fitting. One way we can observe the model is over-fitted by looking at the gap between accuracy for train dataset and test dataset at each epoch. An over-fitted model has a high accuracy while testing on test dataset is rather low [21][31].

Deeper convolution neural network: By combining various Kernel window sizes

of the convolution layer, we can achieve relatively good performance with deep neural network such as Figure. 4.3a, which serves as a baseline benchmark for our various spiking neural network architecture.

Gray-scale images: Gray-scaling is a technique where RGB channels are converted to a single channel of black using $Y = 0.2125R + 0.7154G + 0.0721B$ [28]. The original CIFAR10 images are converted to gray-scaled images, however we find that network performs very poorly. Figure. A.7a shows that we lose quite a bit of image data when converted to gray scale thus decrease in network ability to learn the features.

Baseline non-spiking DNN performance can be seen below. This will serve as a benchmark for the Spiking networks.



(a) Accuracy and loss for DNN

(b) Matrix of correct and incorrect classification for DNN

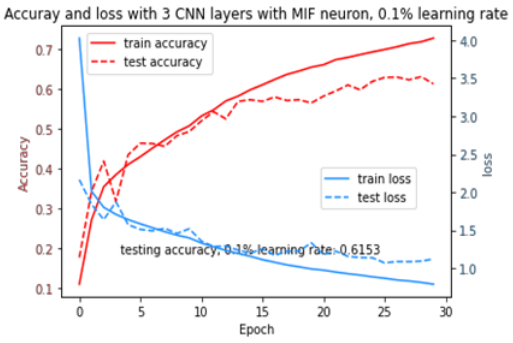
Figure 4.1: Baseline DNN classification performance

The fundamental issue with the Spiking Neural network is the non-differentiable nature of the spiking neurons. Once a neuron crosses a threshold, the neuron fires an action potential or spike in time step and rapidly decays thus the function becomes discontinuous.

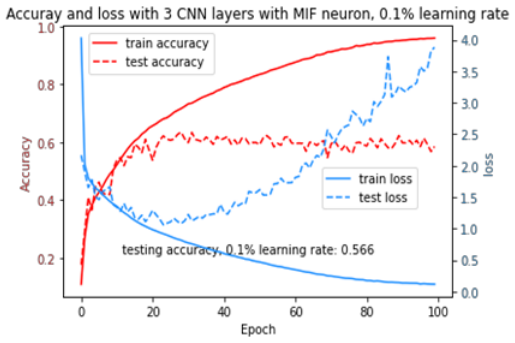
To improve the network performance, several network architectures have been deployed which are summarized in Table 4.1. As a baseline, we have a non-spiking deep neural network whose accuracy is roughly at 85 %. As we can see from the table summary, the highest neural network performance with spiking neuron is MIF network at roughly 74% with 3 convolution layers, dropout and batch-normalization layers. Adding more convolution layers, pooling layers, and batch-normalization layers to form a deeper neural network architecture didn't benefit to improving network performance.

4.2 CIFAR10 Classification with MIF Spiking Neuron Network

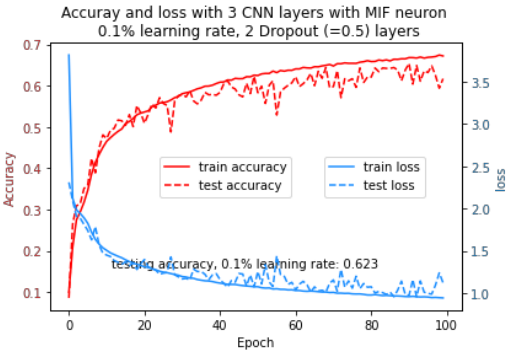
MIF neuron was implemented to the 3 CNN layers neural network architecture since 3 layers seem to be optimal. Overall the MIF network takes roughly 3 hours to train on the GPU and the overall performance is slightly better than the best LIF network accuracy. Future work can be done to improve the MIF network performance by deploying on neuromorphic hardware such as Loihi to reduce the training time, as well as implementing and tweaking MIF2 parameters to help reduce spike noises during the refractory period.



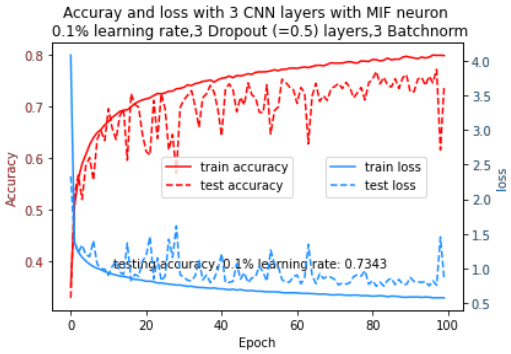
(a) Accuracy and loss for MIF



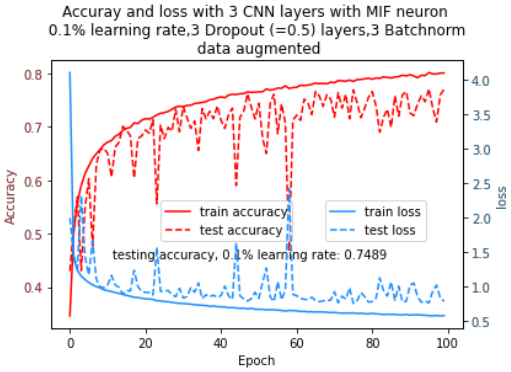
(b) 3 CNN layers with MIF neuron, 100 epochs



(c) 3 CNN layers, MIF neuron with 2 dropout layers (50%)



(d) 3 CNN layers, MIF neuron with 3 dropout layers (50%) and 3 Batch-normalization layers



(e) 3 CNN layers, MIF neuron with 3 dropout layers (50%) and 3 Batch-normalization layers data augmented

Figure 4.2: SNN with MIF neuron performance

Figure 4.2 b) suggests the network is over-fitted as we note the decline in performance at 100 epochs versus 30 epochs, as well as the difference between train dataset and test dataset accuracy. The network was learning other non-parametric in the images instead of generalized shapes of each object, such as background, and noise in the train image dataset,

such that when model is validated against the test dataset, the model performs poorly. Adding dropout layers in between the CNN layers improved the network performance as we noted by the training and testing accuracy closely follows one another. Additional batch normalization layers helped speed up the training time, thought not by much, but further improved network performance.

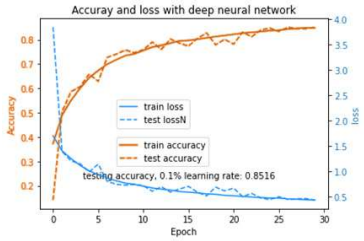
4.3 Summary of CIFAR10 Classification Results

Testing on the sub-dataset of 1000 images, the network test accuracy is quite low although the training accuracy s relatively high during training epochs. While data augmentation is a great tool to generate more data points, our network benefited only a few percentages. Deep SNN doesn't scale as well as non-spiking neural networks where going deeper usually benefits the overall robustness and performance [33]. Adding the dropout layer has significantly improved by about 10% to 12% for LIF neuron and MIF neuron types. Adding batch-normalization layer together with the dropout layer improved the model by another 10%, therefore, closing the gap to the baseline DNN performance of 85% accuracy.

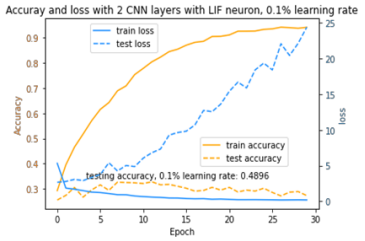
However, a simple model such as LIF can be trained within 10-30 minutes, a more complex and equivalent Hodgkin-Huxley circuits such as MIF and MIF2 can be computationally intensive [18][37], which could take a few hours to close to ten hours. The paper by Valadez-Godínez et al. compares the computation time for variations of spiking neurons (LIF,IZH, HH-T, and HH) and in table 8, it describes CPU time takes the shortest to simulate LIF neuron, whereas anywhere from 5x to 10x longer to simulate HH. [37].

Table 4.1: Summary of CIFAR10 model accuracy

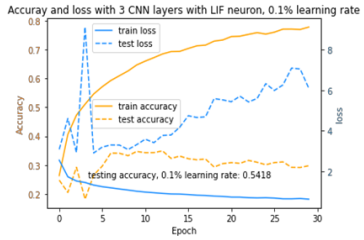
	Neuron Type	Network Layers	Accuracy	Training Parameters	Training Time [min]	Reference
DNN	relu	21	85.16%	0.55 million	230	Fig. 4.3a
SNN	LIF	2	48.96%	6.44 million	5	Fig. 4.3b
SNN	LIF	3	54.18%	2.86 million	7.8	Fig. 4.3c
SNN	LIF	4	40.15%	0.06 million		Fig. 4.3d
SNN	LIF	5	10.02%	0.8 million	16.6	Fig. 4.3e
SNN	LIF	9	10.01%	0.76 million	21.5	Fig. 4.3f
SNN	LIF	3	50.47%	2.86 million	4.4	Fig. 4.3g
SNN	LIF	8	17.11%	0.1 million	9.1	Fig. 4.3h
SNN	LIF	3	10.00%	2.86 million	4.2	Fig. A.7a
SNN	LIF	3	13.19%	2.86 million	7	Fig. A.7b
SNN	LIF	3	62.82%	2.86 million	26	Fig. 4.3i
SNN	LIF	3	66.26%	2.86 million	27	Fig. 4.3j
SNN	MIF	3	61.53%	2.86 million	175	Fig. 4.2a
SNN	MIF	3	56.60%	2.86 million	616	Fig. 4.2b
SNN	MIF	3	62.30%	2.86 million	589	Fig. 4.2c
SNN	MIF	3	73.43%	2.86 million	550	Fig. 4.2d
SNN	MIF	3	74.89%	2.86 million	550	Fig. 4.2e



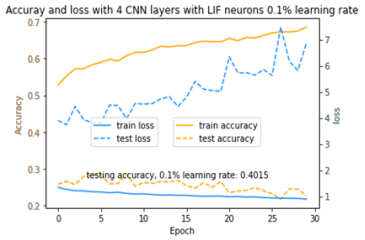
(a) Deep Neural Network



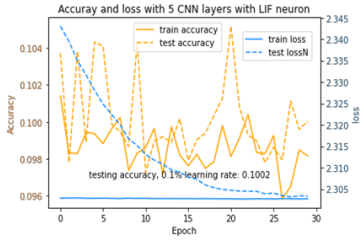
(b) 2 CNN layers with LIF neuron



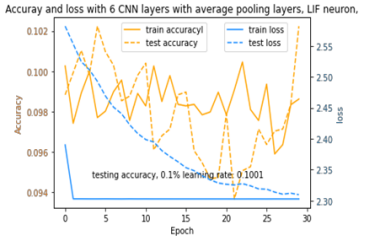
(c) 3 CNN layers with LIF neuron



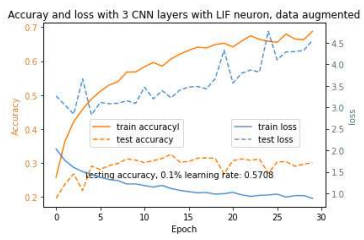
(d) 4 CNN layers with LIF neuron



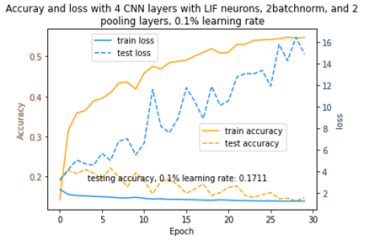
(e) 5 CNN layers with LIF neuron



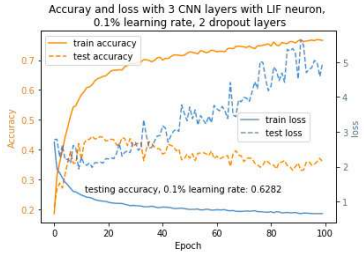
(f) 6 CNN layers with LIF neuron



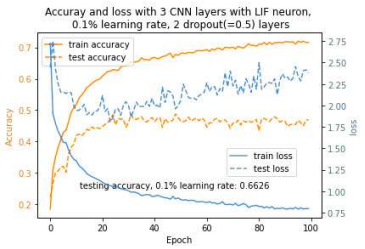
(g) 3 CNN layers with LIF neuron, data augmented



(h) 4 CNN layers with LIF neuron, batch normalization and pooling layers added



(i) 3 CNN layers with LIF neuron, dropout = 25%



(j) 3 CNN layers with LIF neuron, dropout = 50%

Figure 4.3: Network performance at various CNN layer depth, architecture and neuron types

Chapter 5

Conclusion

The potential of spiking neural networks with implementing various neuron types such as LIF and MIF was demonstrated in this paper. The implementation is done by replacing spiking neuron types at the activation layer. The purpose of this paper is to explore the possibility of deep neural network architecture using spiking neurons. We can conclude that while spiking neurons can be resilient to a high learning rate (Appendix: A.1) but they can be poor in performance with a complex temporal dataset. The SNN can also be a good alternative to non-spiking neural networks in smaller and simpler classification problems for quicker training time and energy efficiency.

We have demonstrated the use of Memristive Integrate-and-Fire (MIF) in the neural network and verified the performance with two different datasets. Although a MIF network can be computationally intensive to train on current von Neumann's architecture based GPU, future work can be expanded by implementing in VLSI architecture [18], much like Intel's Loihi chip for energy efficiency [10] and on-chip inference. The results obtained were

somewhat counter-intuitive in some cases where higher neuron activity (i.e. firing rates) improved the network accuracy dramatically using Keras's ANN-SNN conversion technique in MNIST dataset, whereas Nengo LIF outperforms Keras's spiking network result by 5% to 10%.

With the CIFAR10 dataset, it can be observed that training longer (more epochs) with MIF network can lead to lower overall network performance as seen in Figure 4.2 due to network over-fitting, however adding dropout layers and batch-normalization layers can boost dramatically in performance . We can also make a note from the various experiments conducted in this thesis using LIF neurons, it can be observed that there is a diminishing return up until 3 to 4 layers deep as the SNN hidden layers go from 2 to 9.

Chapter 6

Future Work

Future work can be done in an effort to further develop the MIF2 neuristor to add to the Nengo back-end for simulation and comparison. MIF2 is described in the paper by Kang et al., where the second memristor has V_{reset} in addition to the first memristor with V_{rest} [18], which can help to suppress action potential spikes during the refractory period by raising the threshold voltage with V_{reset}. Due to time constraints to train and simulate MIF2 based SNN, I was unable to continue to further study the performance and comparison of MIF and MIF2 neuron types. More work can be done to implement a robust deep neural network with spike-aware neurons thus the correct amount of spikes can be propagated through the network without losing encoded information.

This work reveals the research opportunities where more work can be done to refine the network architectures presented in this thesis and deploy the networks on *Loihi* hardware to implement MIF and MIF2 neuron types. We can also use the technique described in the paper by Hunsberger [14] to fine-tune the MIF2 model parameters such that the function is

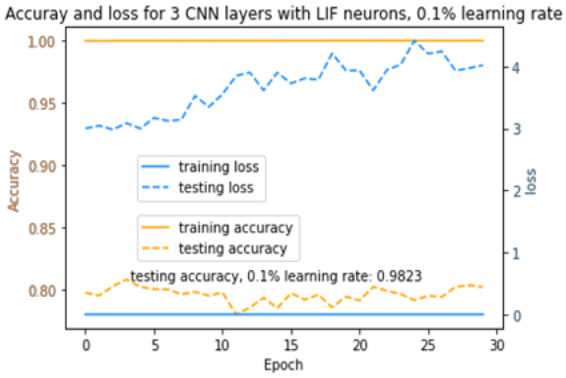
smooth when the membrane potential reaches the threshold voltage, thus back-propagation technique can be used, and another technique to use a constant value is assigned to the derivative of the neuron activation function in the paper by Gerum et al. [12]. Together with classical techniques such as pooling layers, drop out layers and batch-normalization layers, the SNN have potential to be more efficient and have high accuracy.

Appendix A

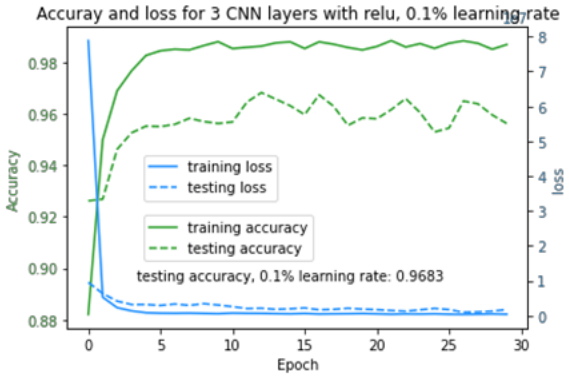
Simulation Results

A.1 Various learning rate with LIF neurons, on MNIST dataset

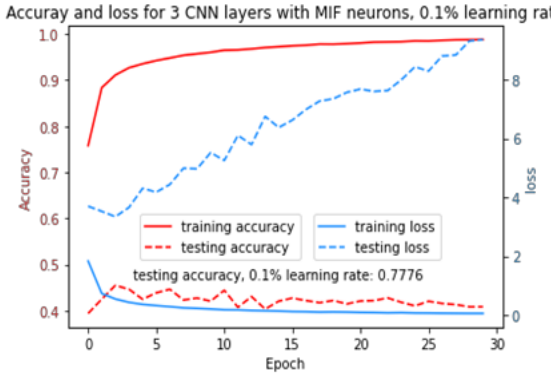
The runs in this appendix are performed with the same network architecture as in Chapter 3, but with various learning rates and neuron types. We can conclude that spiking neurons can be more resilient to higher learning rate (thus less epochs): SNN can retain higher accuracy through higher learning rate for both LIF and MIF type networks, however quicker training time for simpler neuron such as LIF.



(a) 0.1 % learning rate LIF

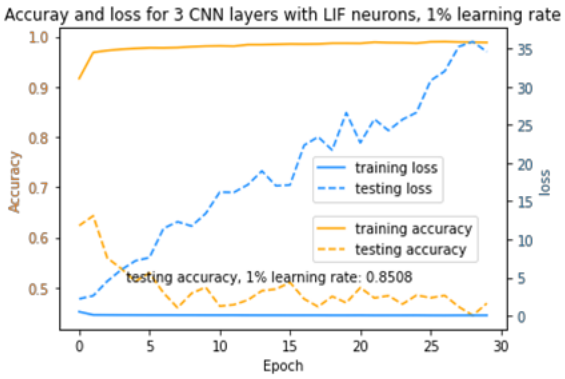


(b) 0.1 % learning rate relu

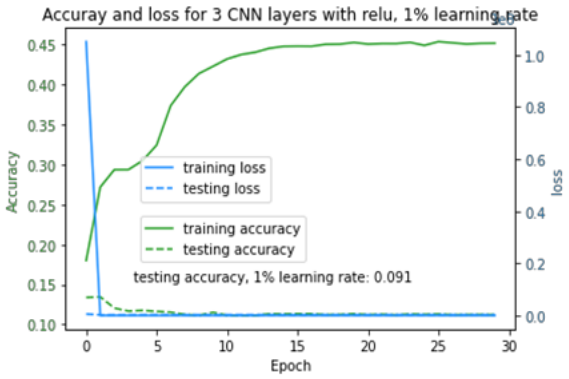


(c) 0.1 % learning rate MIF

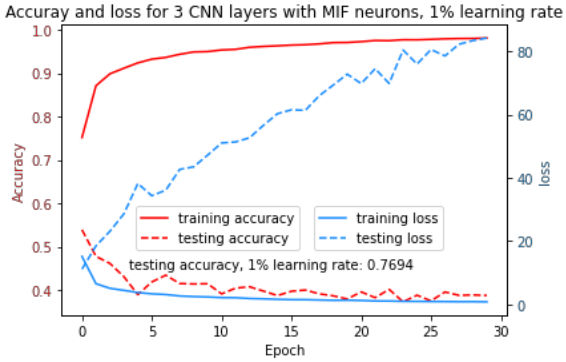
Figure A.1: Network performance with various neuron types, 0.1% learning rate



(a) 1 % learning rate LIF

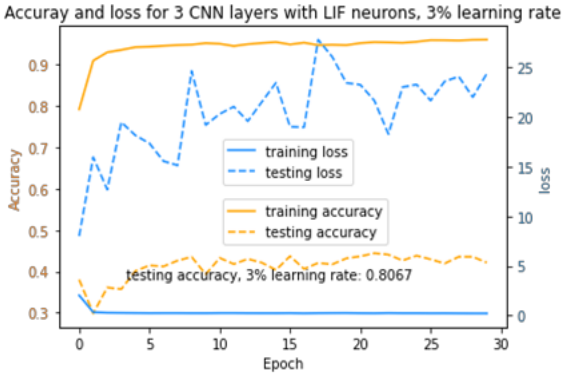


(b) 1 % learning rate relu

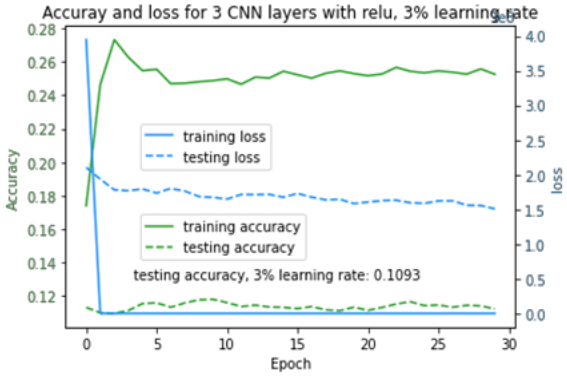


(c) 1 % learning rate MIF

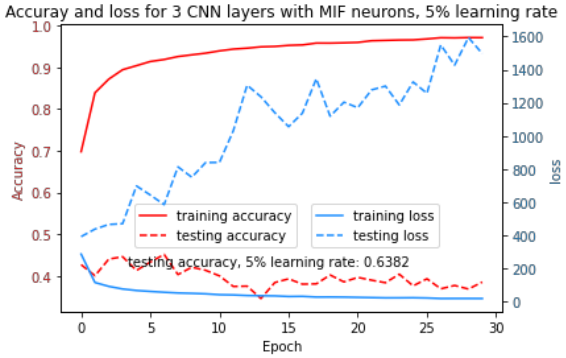
Figure A.2: Network performance with various neuron types, 1% learning rate



(a) 3 % learning rate LIF

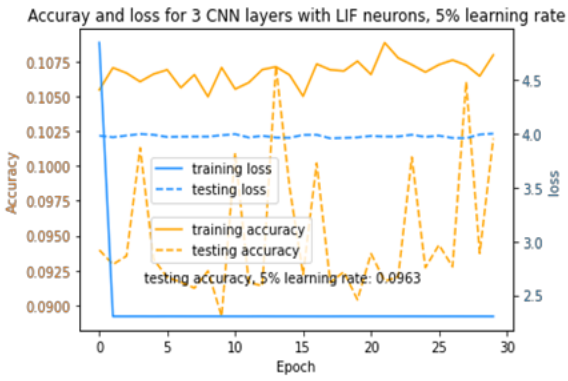


(b) 3 % learning rate relu

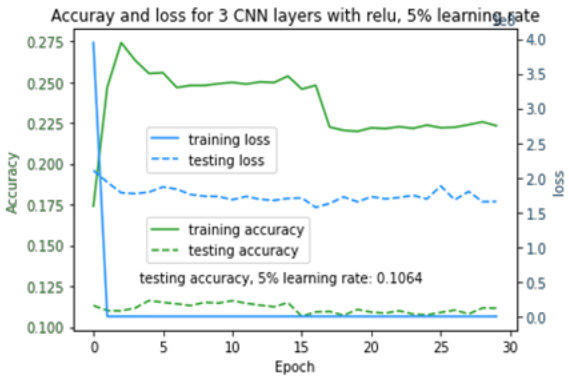


(c) 3 % learning rate MIF

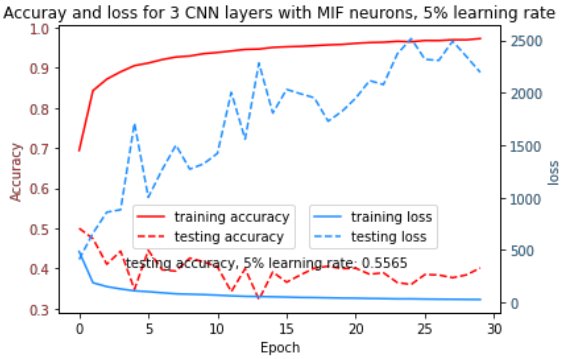
Figure A.3: Network performance with various neuron types, 3% learning rate



(a) 5 % learning rate LIF



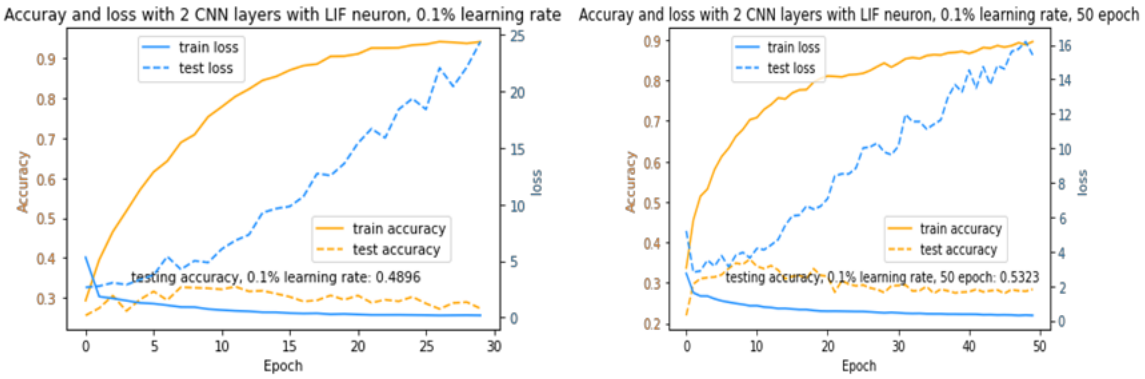
(b) 5 % learning rate relu



(c) 5 % learning rate MIF

Figure A.4: Network performance with various neuron types, 5% learning rate

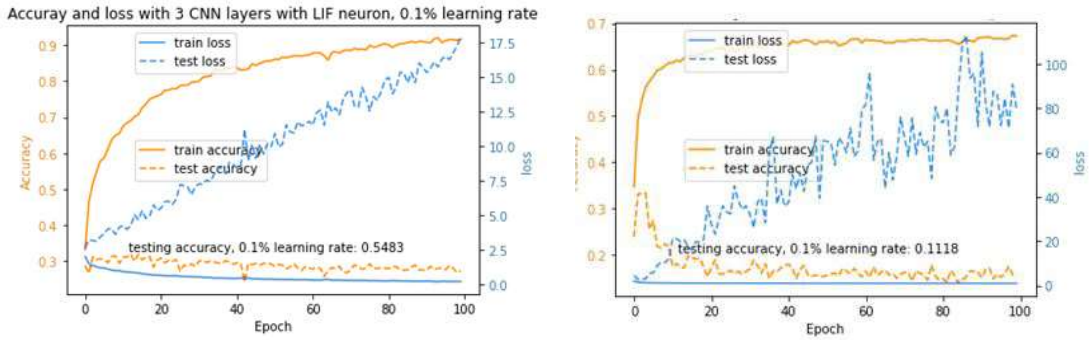
A.2 Various learning rate with LIF neurons, on CIFAR10 dataset



(a) 2 CNN layers with LIF neurons, 30 epochs (b) 2 CNN layers with LIF neurons, 50 epochs

Figure A.5: 2 layer CNN performance at various epoch

Higher the accuracy can be reached with more epochs, however, it seems there is an upper limit asymptotically.

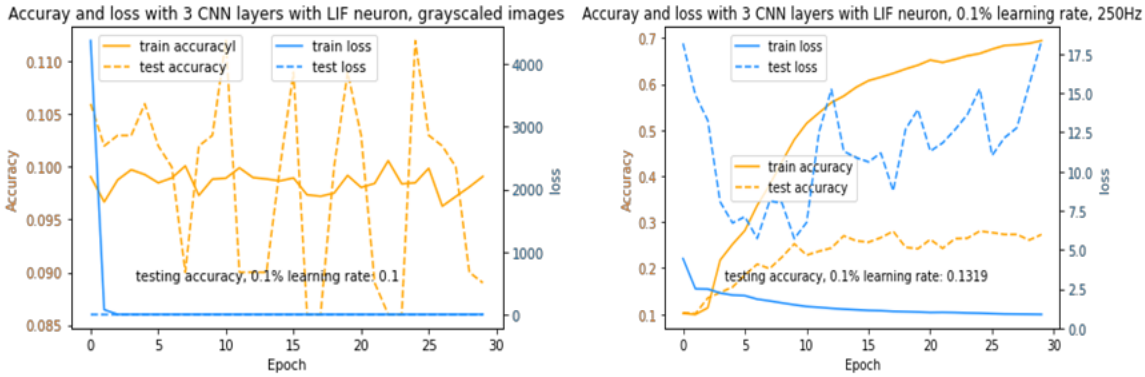


(a) 3 CNN layers with LIF neurons, 100 epochs, batch-size = 200 (b) 3 CNN layers with LIF neurons, 100 epochs, batch-size = 32

Figure A.6: 3 layers CNN performance at various batchsize

In the paper by Masters [25], a smaller batchsize of 32 to 64 has the best results overall with the CIFAR10 dataset, however we can observe that the SNN with batchsize of 32 is on contrary to the non-spiking DNN. Larger batchsize of 200 outperforms batchsize of 32.

Although the accuracy is quite good during training, the test accuracy is only at 11% or so.



(a) 3 CNN layers with LIF neuron, gray-scaled (b) 3 CNN layers with LIF neuron, 250Hz

Figure A.7: 3 CNN layers performance of gray-scaled image dataset and higher firing rate neurons

Gray-scaled images of CIFAR10 dataset seem to lose important details in the images, thus the accuracy for the training data hovers around 10% at best. While higher firing rates of neurons seem to be somewhat effective during training, the overall accuracy is not that great.

A.3 Sample MNIST classification with LIF neuron type

Exemplary images of unfiltered and filtered neuron spiking can be seen below for SNN with LIF neuron.

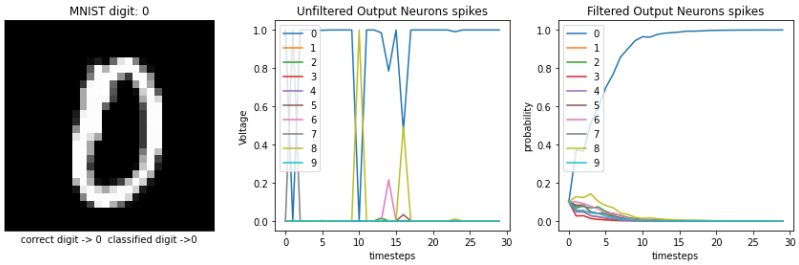


Figure A.8: Classified Handwritten digit 0, LIF SNN

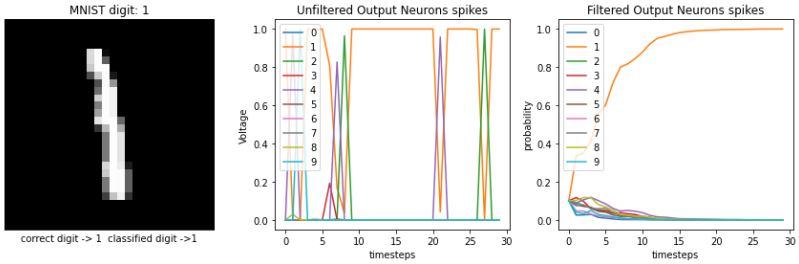


Figure A.9: Classified Handwritten digit 1, LIF SNN

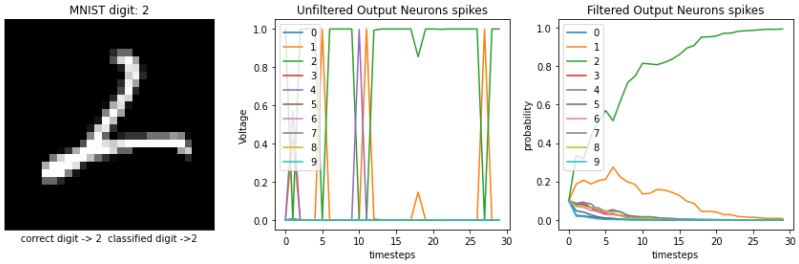


Figure A.10: Classified Handwritten digit 2, LIF SNN

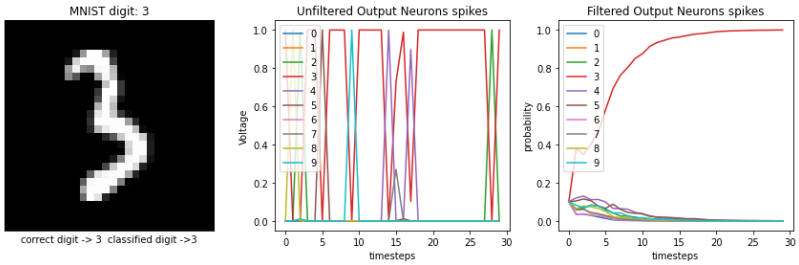


Figure A.11: Classified Handwritten digit 3, LIF SNN

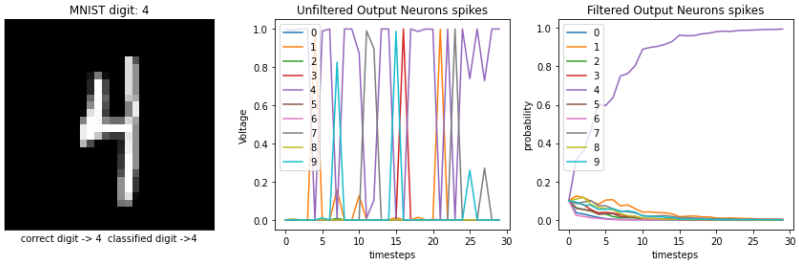


Figure A.12: Classified Handwritten digit 4, LIF SNN

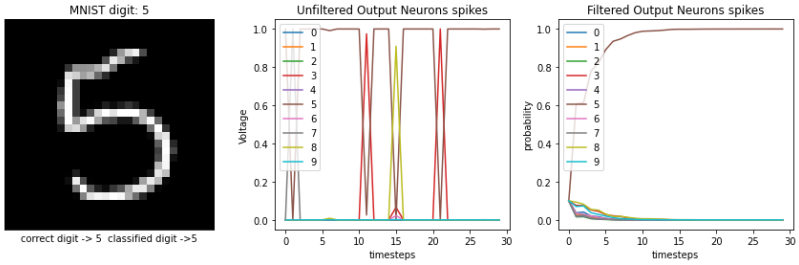


Figure A.13: Classified Handwritten digit 5, LIF SNN

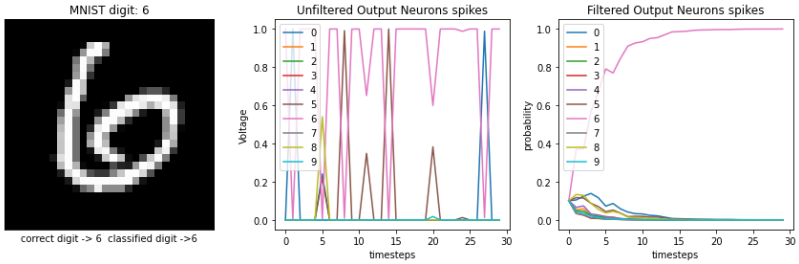


Figure A.14: Classified Handwritten digit 6, LIF SNN

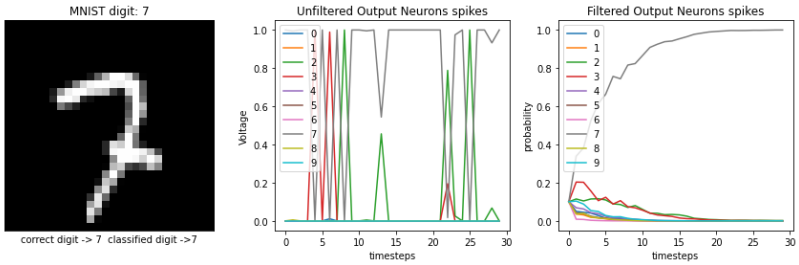


Figure A.15: Classified Handwritten digit 7, LIF SNN

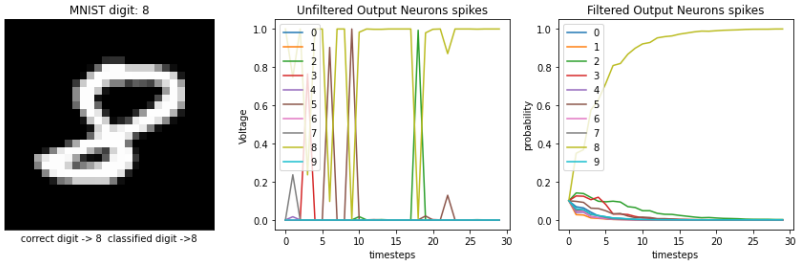


Figure A.16: Classified Handwritten digit 8, LIF SNN

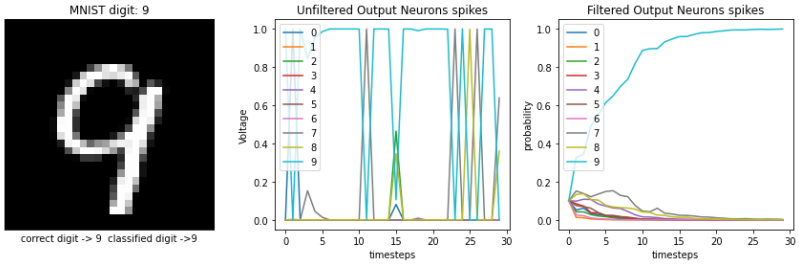


Figure A.17: Classified Handwritten digit 9, LIF SNN

A.4 Sample MNIST classification with MIF neuron type

Exemplary images of unfiltered and filtered neuron spiking can be seen below for SNN with MIF neuron. We can see that the output neurons are more active compared to the LIF neuron.

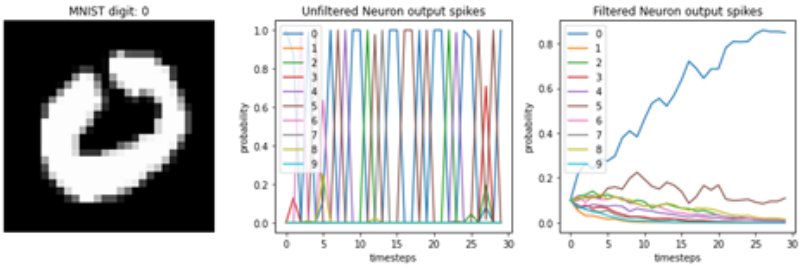


Figure A.18: Classified Handwritten digit 0, MIF SNN

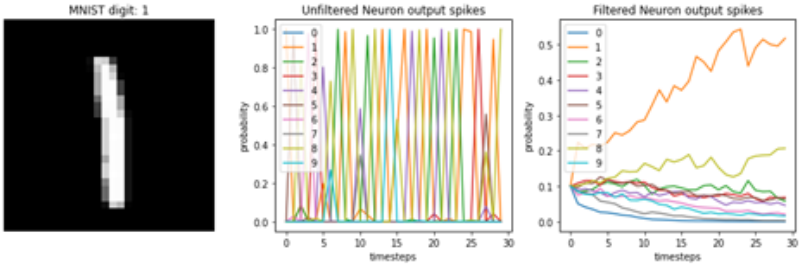


Figure A.19: Classified Handwritten digit 1, MIF SNN

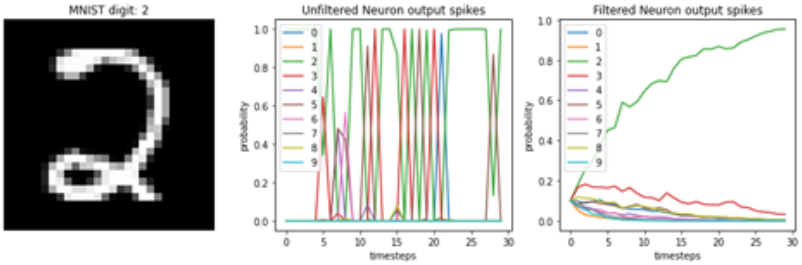


Figure A.20: Classified Handwritten digit 2, MIF SNN

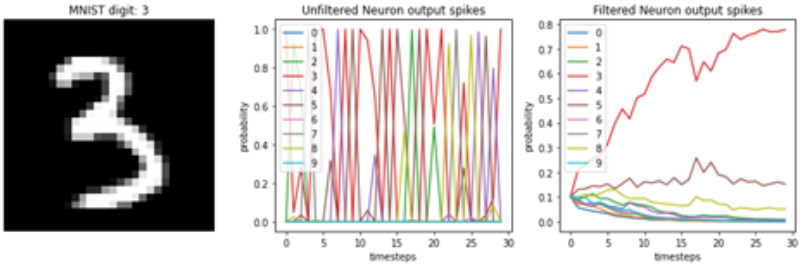


Figure A.21: Classified Handwritten digit 3, MIF SNN

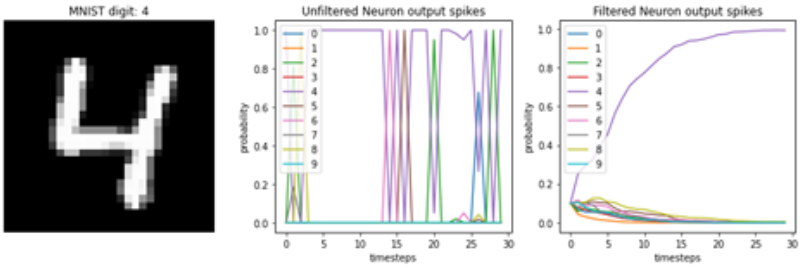


Figure A.22: Classified Handwritten digit 4, MIF SNN

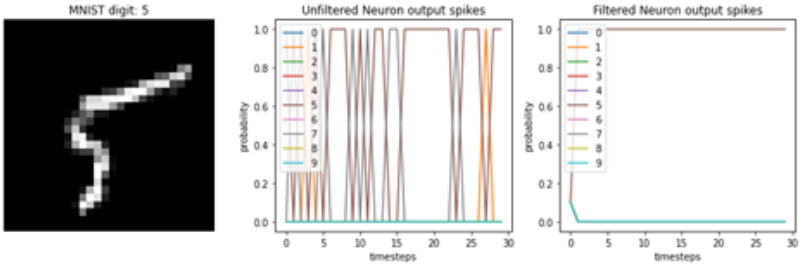


Figure A.23: Classified Handwritten digit 5, MIF SNN

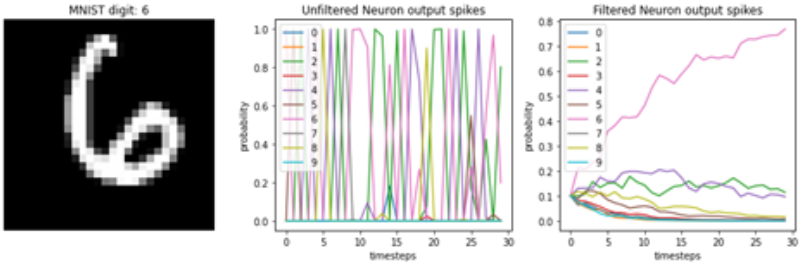


Figure A.24: Classified Handwritten digit 6, MIF SNN

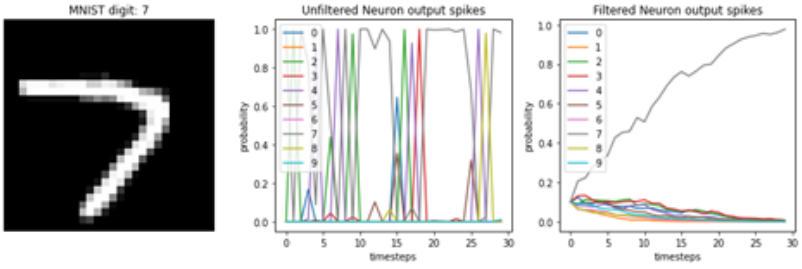


Figure A.25: Classified Handwritten digit 7, MIF SNN

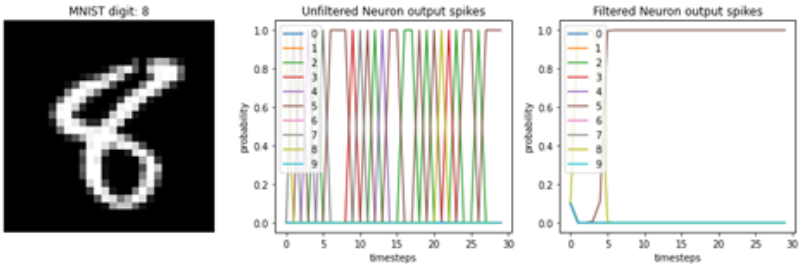


Figure A.26: Classified Handwritten digit 8, MIF SNN

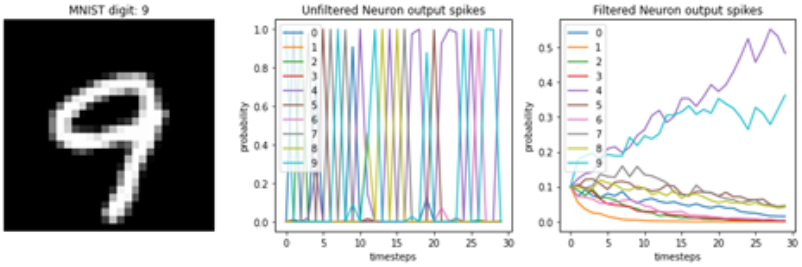


Figure A.27: Classified Handwritten digit 9, MIF SNN

Bibliography

[1] Nengo. `https://www.nengo.ai/`. Accessed: 2021-12-3.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[3] L. F. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *columbia.edu*, May 1999.

[4] S. An, M. Lee, S. Park, H. Yang, and J. So. An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition. *ArXiv e-prints*, October 2020.

[5] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T.C. Stewart, D. Rasmussen, X. Choo, A.R. Voelker, and C. Eliasmith. Nengo: a python tool for building large-scale functional brain models. In *Frontiers in Neuroinformatics*, volume 7, 2014.

[6] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*. Springer, 2013.

[7] Y. Cao, Y. Chen, and D. Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *Int J Comput Vis*, pages 54–66, 2015.

[8] L. O. Chua. Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, 1971.

[9] L. O. Chua and S. M. Kang. Memristive devices and systems. *Proceedings of the IEEE*, 64(2):209–223, 1976.

[10] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S.H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.H. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. In *IEEE Micro*, volume 38, pages 82–99, 2018.

[11] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[12] R.C. Gerum and A. Schilling. Spiking machine intelligence: What we can learn from biology and how spiking neural networks can help to improve machine learning. *CoRR*, abs/2004.13532, 2020.

[13] A. L. Hodgkin and A. F. Huxley. Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *The Journal of Physiology*, 116(4):449–472, 1952.

[14] E. Hunsberger and C. Eliasmith. Spiking deep networks with LIF neurons. *CoRR*, abs/1510.08829, 2015.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[16] E. M. Izhikevich. Simple model of spiking neurons. In *IEEE Transactions on Neural Netwworks*, volume 14, pages 1569–15720, November 2003.

[17] X. Jia. Design and Training of Memristor-Based Neural Networks. *escholarship.org*, June 2020.

[18] S. M. Kang, D. Choi, J. K. Eshraghian, P. Zhou, J. Kim, B. Kong, X. Zhu, S. D. Ahmet, A. Ascoli, R. Tetzlaff, W. D. Lu, and L. O. Chua. How to build a memristive integrate-and-fire model for spiking neuronal signal generation. In *IEEE Transactions on Circuits and Systems I: Regular Papers*, volume 68, pages 4837–4850, November 2021.

[19] Keras. Averagepooling2d layer. `https://keras.io/api/layers/pooling_layers/average_pooling2d/`. Accessed: 2022-1-3.

[20] Keras. Batchnormalization layer. `https://keras.io/api/layers/normalization_layers/batch_normalization/`. Accessed: 2022-1-3.

[21] Keras. Dropout layer. `https://keras.io/api/layers/regularization_layers/dropout/`. Accessed: 2022-1-3.

[22] A. Krizhevsky, V. Nair, and Hinton G. The cifar-10 dataset. `https://www.cs.toronto.edu/~kriz/cifar.html`. Accessed: 2021-12-28.

[23] Y. LeCun, C. Cortes, and Burges C.J.C. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`. Accessed: 2021-12-27.

[24] R. Massa, A. Marchisio, M. Martina, and M. Shafique. An efficient spiking neural network for recognizing gestures with a DVS camera on the loihi neuromorphic processor. *CoRR*, abs/2006.09985, 2020.

[25] D. Masters and C. Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018.

[26] T. Poggio and G. Kreiman. Mitopencourseware: Brains, minds and machines summer course. `https://ocw.mit.edu/resources/res-9-003-brains-minds-and-machines-summer-course-summer-2015/tutorials/tutorial-2.-matlab-programming/MITRES_9_003SUM15_fire.pdf`. Accessed: 2021-12-1.

[27] D. Querlioz, O. Bichler, P. Dollfus, and C. Gamrat. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Transactions on Nanotechnology*, 12(3):288–295, 2013.

[28] Scikit-Image. rgbtogray. `https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_rgb_to_gray.html`. Accessed: 2022-1-4.

[29] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy. Going deeper in spiking neural networks: VGG and residual architectures. *CoRR*, abs/1802.02627, 2018.

[30] Y. Shao. Application Of Memristive Device Arrays For Pattern Recognition. *escholarship.org*, June 2020.

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[32] T. Stewart. A Technical Overview of the Neural Engineering Framework. *Centre for Theoretical Neuroscience technical report*, October 2012.

[33] A. Tavanaei, M. Ghodrati, S.R. Kheradpisheh, T. Masquelier, and A. S. Maida. Deep learning in spiking neural networks. *CoRR*, abs/1804.08150, 2018.

[34] TensorFlow. Convolutional neural network. `https://developers.google.com/machine-learning/glossary/#convolutional_neural_network`. Accessed: 2022-1-27.

[35] TensorFlow. Imagedatagenerator. `https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator`. Accessed: 2022-1-3.

[36] TensorFlow. Sparse categorical cross entropy. `https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy`. Accessed: 2021-12-27.

[37] S. V. Valadez-Godinez, Humberto J. Azuela, S., and R. Santiago-Montero. How the accuracy and computational cost of spiking neuron simulation are affected by the time span and firing rate. *Computación y Sistemas*, 21(4), 2018.

[38] Y. Wang, G.Y. Wei, and D. Brooks. Benchmarking tpu, gpu, and CPU platforms for deep learning. *CoRR*, abs/1907.10701, 2019.