**PROJECT REPORT**
**ON**

# Image Classification Using Spiking Convolutional Neural Networks

**BY**


Name of the Student ID Number

**Yash Dabhade 2020A7PS0974G**


*Submitted in partial fulfillment of the requirements of*
*CS F266 Study Project*

Supervised by

**Dr. Basabdatta Sen-Bhattacharya**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**
**K.K. Birla Goa Campus**


**April 2023**

# Abstract

Spiking neural networks (SNNs) have a significant advantage over traditional artificial neural networks (ANNs) in that they can leverage spike timing for coding and efficient processing. In this study, we examine whether neuromorphic datasets recorded
SNNs' capacity to employ spike timings in their computations may be evaluated using static pictures. We examined N-MNIST and N-Caltech101 along these lines, although our concentration is on N-MNIST. SNNs can be simulated on regular computers or can run much more efficiently on neuromorphic computers. neuromorphic computers, are incredibly energy-efficient, latent, and they possess massive parallelism, hence we can combine the accuracy and power of deep learning with the efficiency and latency of SNNs and neuromorphic computers.

[Code]

# Acknowledgments

I would like to express my heartful gratitude towards Dr. Basabdatta Sen-Bhattacharya for giving me this opportunity and for her continuous support and motivation. Her guidance helped me in all aspects of this study project.
Through this project, I was exposed to a considerable amount of new information regarding the most rapidly developing technologies based on Image processing and Spiking Neural Networks that was in uncharted waters for me prior, the exploration of which then caused my interests to bloom.

# TABLE OF CONTENTS

# 1. Introduction

Scientists have built systems that imitate the brain's extraordinary performance and efficiency in order to research biological function and improve technical systems. The neurons in early neural networks, those of the first and second generations, do not spike. These networks, dubbed artificial neural networks (ANNs), have real-valued outputs and can be thought of as time averaged firing rates of neurons. Spiking neural networks (SNN) of the third generation clearly use spikes as their mechanism for computation.

In this work, we employed two datasets, the MNIST and the Caltech datasets. The MNIST dataset contains images with ten digits. As a result, the dense layer of our proposed SCNN, which corresponds to the MNIST database, includes ten spiking neurons. For the vehicle recognition test, two sets of photos from the Caltech dataset are employed; the thick layer of our SCNN corresponding to this database comprises two spiking neurons. The outputs of the neurons in the final layer are expressed in millivolts (mV) and are created using the Nengo library's probe function. These outputs depict increasingly increasing membrane potentials, and the neuron with the greatest voltage throughout a 30 ms simulation time period represents the class that was chosen in a given epoch. The SCNN's loss function is then computed using a Cross-Entropy function.

# 2. Spiking Neural Networks

Spiking neural networks (SNNs) are artificial neural networks that closely resemble natural neural networks. SNNs incorporate time into their working model in addition to neuronal and synaptic state. The idea is that neurons in the SNN only transmit information when a membrane potential - a neuron's intrinsic quality related to its membrane electrical charge - reaches a certain value, known as the threshold.

When the membrane potential reaches the threshold, the neuron fires, sending a signal to neighboring neurons, which increase or decrease their potentials in response to the signal. A spiking neuron model is a neuron model that fires when a threshold is crossed.

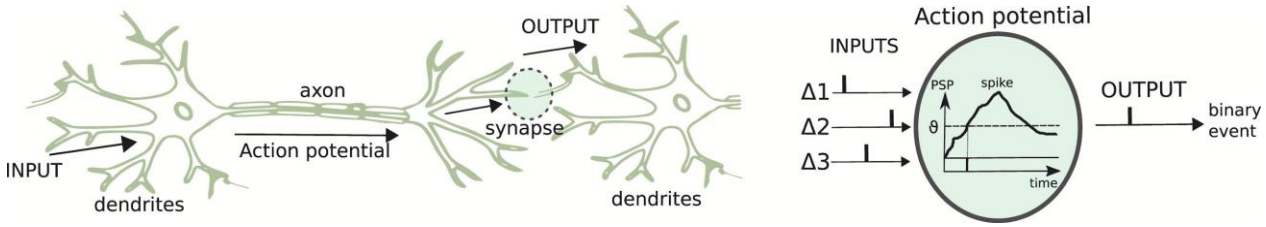


Fig. 1: This figure shows how spiked are propagated and generated in a neuron

A spiking neuron's firing rate can be defined in numerous ways:

(1) The time averaged firing rate is the number of spikes fired by a neuron over a given period;

(2) The instantaneous population firing rate is the number of spikes elicited by a population of neurons in a short period of time;

(3) The trial averaged firing rate of a neuron is the average number of spikes across trials.

To quantify big values, spike time coding does not require a huge number of spikes or many neurons, but can do so simply altering the spike timing of a few neurons. As a result, spike time codes enable faster processing. If an SNN only uses time-averaged or instantaneous population rate codes, it is less efficient than ANNs since it must run for lengthy periods of time or employ numerous neurons to compute correct spike rate averages. The fundamental advantage of SNNs over the previous two generations of neural networks is that they can use spike time coding to increase efficiency. We provide a novel method for converting deep ANNs into spiking networks for implementation on neuromorphic hardware in this research.

# 3. Methodology

## 1) Leaky Integrate and Fire Model:

Leaky integrate and fire (LIF) model represents neuron as a parallel combination of a "leaky" resistor (conductance, $g_L$) and a capacitor ($C$). A current source $I(t)$ is used as synaptic current input to charge up the capacitor to produce a potential $V(t)$.

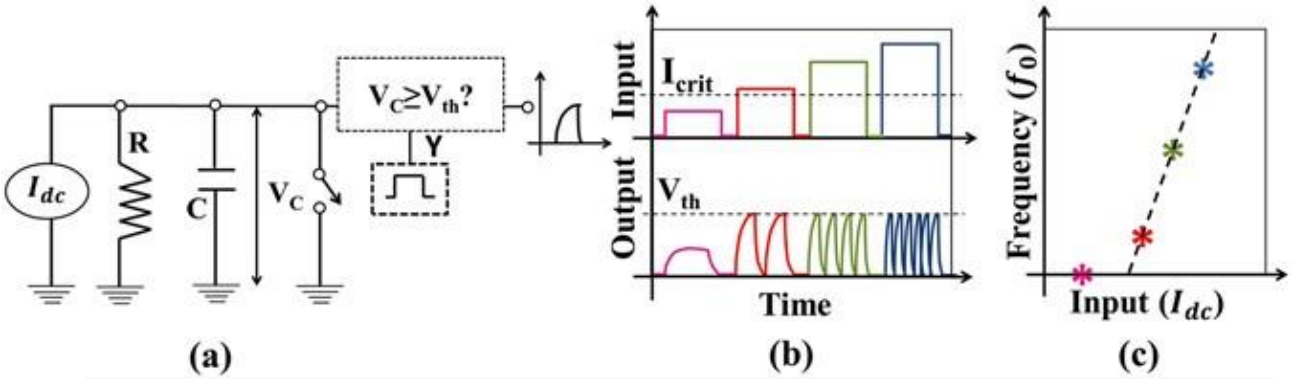


Fig 2: A neuron is represented by an RC circuit with a threshold. Each input pulse (e.g. caused by a spike from a different neuron) causes a short current pulse. Voltage decays exponentially. If the threshold is reached an output spike is generated and the voltage is reset.

$$C_{\mathrm{m}} \frac{dV_{\mathrm{m}}(t)}{dt} = I(t) - \frac{V_{\mathrm{m}}(t)}{R_{\mathrm{m}}}$$

where $V_{\mathrm{m}}$ is the voltage across the cell membrane and $R_{\mathrm{m}}$ is the membrane resistance. (The non-leaky integrate-and-fire model is retrieved in the limit $R_{\mathrm{m}}$ to infinity, i.e., if the membrane is a perfect insulator). The model equation is valid for arbitrary time-dependent input until a threshold $V_{\mathrm{th}}$ is reached; thereafter the membrane potential is reset.

For constant input, the minimum input to reach the threshold is $I_{\mathrm{th}} = V_{\mathrm{th}} / R_{\mathrm{m}}$. Assuming a reset to zero, the firing frequency thus looks like

$$f(I) = \begin{cases} 0, & I \leq I_{\mathrm{th}} \\ \left[ t_{\mathrm{ref}} - R_{\mathrm{m}} C_{\mathrm{m}} \log\left(1 - \frac{V_{\mathrm{th}}}{I R_{\mathrm{m}}}\right) \right]^{-1}, & I > I_{\mathrm{th}} \end{cases}$$

which converges for large input currents to the previous leak-free model with refractory period. The model can also be used for inhibitory neurons

The Spiking neurons in this work are Leaky Integrate and Fire (LIF) neurons, as defined above. To transform the rank-ordered image characteristics to spikes, a layer of LIF neurons of the same size M x N as the input image is used. Each rank-ordered coefficient is associated with a current input value that drives the spatially matching LIF. We now have a neural layer S with M x N spiking.

2) Nengo & NengoDL

We can construct a convolutional spiking network after preprocessing the input and flattening the images. This is done with Nengo, a Python library that combines SNN architecture and Python in a simple and intuitive manner. We're using NengoDeepLearning, which combines TensorFlow and Nengo to build a convolutional network (traditionally a deep learning network) with spiking neurons.

3) Training Data

SNNs are time-based networks, hence the data must have a time dimension. We accomplish this by introducing time-steps into the data. We just repeat the dataset for the number of time-steps we require. However, this is only done for the test dataset because it is the set that the actual convolutional SNN uses. Because the rate-based convolutional SNN approximation is used, the train dataset only needs one time step. For the digit recognition task, we perform the training for a set of 55,000 input images and for the vehicle recognition task, this procedure is carried out for a corpus of 1400 training images

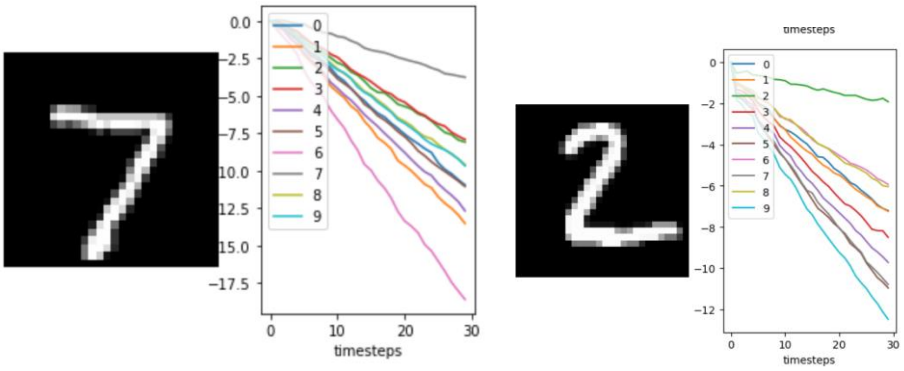


Fig 3: Training images with timestamps

4) Generating predictions:

During the testing phase, we assess our network's performance on 10,000 photos from the testing datasets for the digit identification job and 200 images from the testing datasets for the vehicle recognition task. We propagate each input image through the trained network for 30 timesteps for the MNIST dataset and 60 timesteps for the Caltech dataset, and the output of our network's final layer is a Kx1 vector, where K is the number of classes. As a result, the projected label for the input image is allocated as the index of the neuron with the highest spike count among all K values. This label is then compared to the ground truth label of the input image to assess our network's classification accuracy.

# 4. Results

The best results of 98.7% are obtained over 10 epochs, which is the perfect amount of time for the network to converge.

The MNIST testing set achieves this level of precision. The network tends to overfit as it runs longer, learning extremely particular features related to the training data and hence performing poorly on the testing data.

```
sim.compile(loss={out_p_filt: classification_accuracy})
print("accuracy after training:",
      sim.evaluate(test_images, {out_p_filt: test_labels}, verbose=0)["loss"])
```

```
accuracy after training: 0.9876999855041504
```

Fig 4: The Accuracy of the SCNN based on MNIST

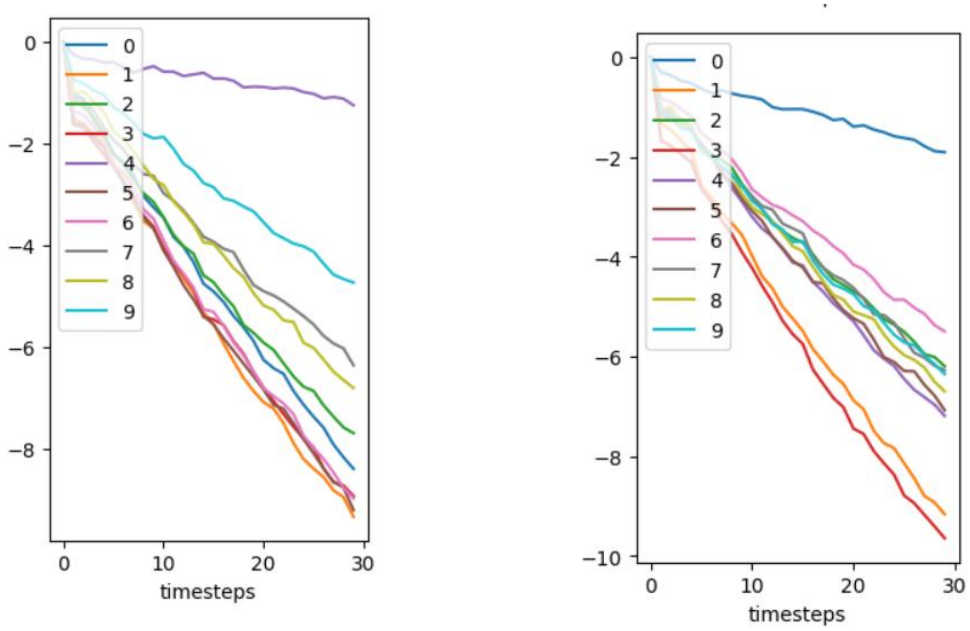


Fig 5 : Outputs of the final layer neurons for (a) digit 4 and (b) digit 0. The timestep of simulation is 1 ms and total time of simulation is 30ms

For the Caltech dataset the network is trained such that the neuron 0 will represent the motorbike category, and the neuron 1 will represent the airplane category. In the plots of Fig. 6, we can see the outputs from the final layer neurons during the testing phase, obtained using the probe function of the Nengo library. For an input image of an airplane, the output from the neuron 1 of the output layer is the highest and hence this image is classified correctly by the network. For the motorbike image, the highest output is for the neuron 0, once again classifying correctly the input image

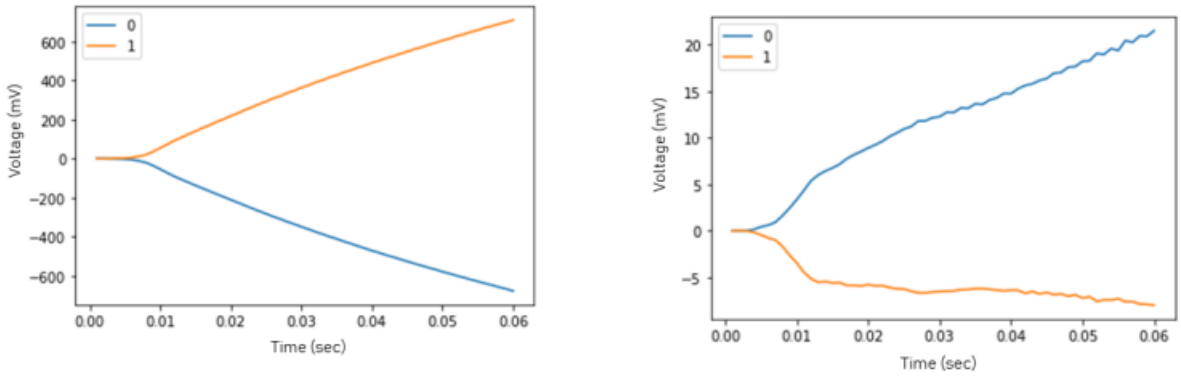


Fig 6: Final layer neuron outputs for the (a) aeroplane and (b) motorcycle images. The 0 in the image legend represents neuron 0's output, which corresponds to the motorcycle category; the 1 in the figure legend represents neuron 1's output, which belongs to the aeroplane category. The simulation timestep is 1 ms, and the overall simulation time is 60ms.

# 5. Conclusion

In this paper, we present a Spiking Convolutional Neural Network (SCNN) to enable the classification of images in both a noise-free and a noisy environment. We investigated the qualitative effect of the filtering on the input image in order to discover the best combination of DoG filters for each of the datasets utilised in our experiments. We employed a supervised training technique based on the leaky integrate and fire model to train our network, which is a modified version of the classic backpropagation approach used in artificial deep learning networks. We tested our model's performance using two types of datasets: one for digit identification and another for vehicle recognition. Our model performs well on both datasets, with an accuracy of up to 99.00% on the MNIST dataset, comparable to existing SCNN models.

# 6. References

1. Implementing a foveal-pit inspired filter in a Spiking Convolutional Neural Network: a preliminary study - **Shriya T.P. Gupta** and Basabdatta S. Bhattacharya *International Joint Conference on Neural Networks (IJCNN). Glasgow, UK. IEEE, 2020*

2. P. U. Diehl and M. Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity,” Frontiers in computational neuroscience, vol. 9, p. 99, 2015.

3. S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, “Stdp-based spiking deep convolutional neural networks for object recognition,” Neural Networks, vol. 99, pp. 56–67, 2018.

4. S. Ghosh-Dastidar and H. Adeli, “Third generation neural networks: Spiking neural networks,” in Advances in Computational Intelligence. Springer, 2009, pp. 167–178

5. “Is Neuromorphic MNIST Neuromorphic? Analyzing the Discriminative Power of Neuromorphic Datasets in the Time Domain” - Laxmi R. Iyer, Yansong Chua1 and Haizhou Li

6. “Training Spiking Deep Networks for Neuromorphic Hardware” - Eric Hunsberger, Chris Elias