Optimizing neural networks for next-gen AI

Andrew Deutsch, SULI Intern, Department of Physics, Stevens Institute of Technology, Hoboken, NJ 07030
Yihui “Ray” Ren (Mentor), Sandeep Mittal (Co-Mentor), Computational Science Initiative, Brookhaven National Lab, Upton, NY 11973

Abstract

Most scientific data challenges involve real-time decision making on high-volume data flows. **Deep Neural Networks (DNNs)** and **Convolutional Neural Networks (CNNs)** have been promising solutions for many challenging problems but are limited to training and evaluation in an offline manner on a Graphics Processing Unit (GPU). In this work, we aim to transform pre-trained **DNNs** and **CNNs** into **Spiking Neural Networks (SNNs)**, a better, biologically inspired form that can be implemented and deployed on novel hardware. Unlike discrete matrix multiplication based **DNNs**, an **SNN** acts on continuous trains of signals. First, we investigate the sparsity nature of **DNNs**, pruning techniques, and their connections to **SNNs**. We then investigate **DNN** to **SNN** conversion using NengoDL, an existing framework based on TensorFlow. Specifically, we optimize the accuracy of converted models by properly scaling the firing rates of neurons and applying a lowpass filter over the spike trains. We systematically run these experiments using the popular benchmark image datasets MNIST and CIFAR-10. For further optimization, we limit scaling of firing rates to only one layer and find evidence that scaling late layers in a converted **DNN** and early layers in a converted **CNN** leads to higher accuracy. Further, we investigate the cause of this discrepancy by extracting firing rate data from both networks. Our findings provide insight into the co-design of novel energy-efficient neuromorphic chips for **SNNs**. As a result of this summer, I have gained a concrete understanding of neural network theory and the implementation of these networks using open-source libraries. Being able to draw connections between mathematical equations and their computational implementation is an invaluable experience. I have also developed skills in data analysis, data visualization and scientific communication that will benefit me in my future career.

Introduction

Network Pruning

We first explore pruning techniques as a method of reducing the size of a network while maintaining a baseline level of accuracy. Neuron weights that, when decreased, produce a negligible change in accuracy are sent to zero during training. We test the limits of pruning and determine an optimal network size for a specific **DNN**.

The Spiking Neural Network

SNNs (Figure 1) are fundamentally time-dependent, as they operate on trains of discrete spikes. Instead of representing information as a vector of real numbers, the **SNN** encodes these values in the frequency domain, with higher spiking frequencies corresponding to higher values and vice versa. The Rectified Linear Unit (ReLU) activation functions at each neuron are replaced with Leaky Integrate and Fire (LIF) neurons (Figure 2), which compute membrane potential as a weighted sum of input spikes followed by exponential decay, and fire when this potential exceeds a threshold. In this work we investigate the sparsity of spikes as they propagate through the **SNN**.

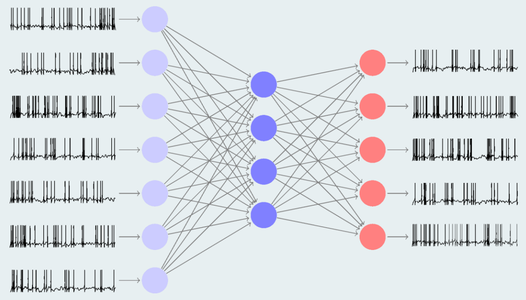


Figure 1: An illustration of a simple 3-layer Spiking Neural Network, complete with input and output spike trains. **Source:** https://fzenke.net/index.php/2017/02/19/learning-in-multi-layer-spiking-neural-networks/

Motivation

The sparse nature **SNNs** means they provide the same functionality as **DNNs**, but with much less communication. Although these tests are performed on a digital computer, **SNNs** are capable of implementation on energy-efficient novel analog neuromorphic chips.

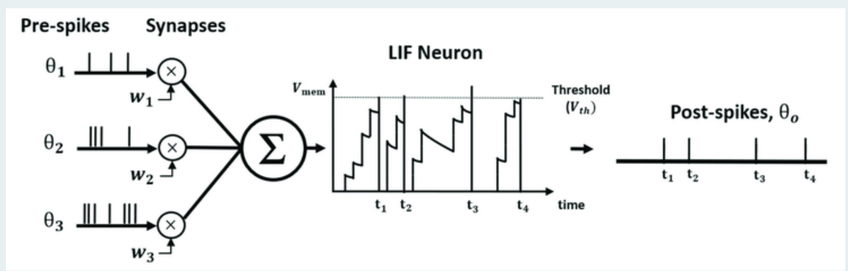


Figure 2: The Leaky Integrate and Fire (LIF) neuron, transforming three presynaptic spike trains into a single postsynaptic spike train. **Source:** Lee, Chankyu & Sarwar, Syed & Panda, Priyadarshini & Srinivasan, Gopalakrishnan & Roy, Kaushik. (2020). Enabling Spike-Based Backpropagation for Training Deep Neural Network Architectures. Frontiers in Neuroscience. 14.119.10.3389/fnins.2020.00119.

Methods

NengoDL is used to convert pre-trained **DNNs** and **CNNs** to **SNNs**. This framework comes equipped with two key parameters which we set out to optimize: **synapse** and **scale_firing_rates**. Respectively, these allow us to control the time constant of a low-pass filter to be applied over all the spike trains and a direct upscaling of the firing rates by layer. First, we compute the test accuracy of a converted **DNN** consisting of 5 dense (fully connected) layers and a converted **CNN** consisting of 7 layers (3 pairs of convolutional and pooling layers, plus one dense layer) for a wide range of these two parameters. Once these optimal values are found, we scale the firing rates of each layer individually and compare test accuracy. Finally, we probe the neurons directly and find extract firing rate statistics for each layer. The CIFAR-10 image dataset (Figure 4) is normalized for all tests.

Results

SNN Optimization

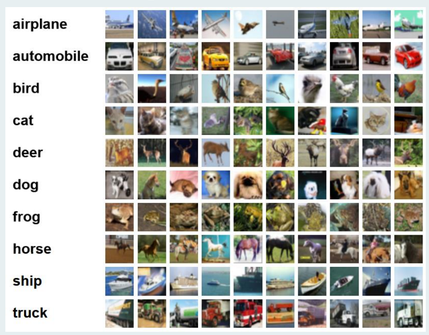
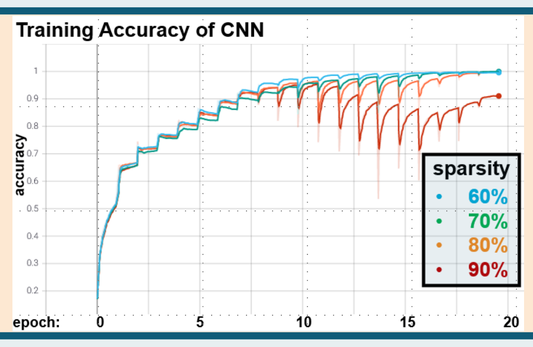


Figure 4: The CIFAR-10 image dataset used to train and test our networks consists of 60,000 images belonging to 10 classes. **Source:** https://www.cs.toronto.edu/~kriz/cifar.html

Network Pruning

We find that a simple **CNN** trained on the MNIST dataset (Figure 3) can be pruned by up to 80% (only 20% of the neurons remain) before the training accuracy fails to recover. Here, I made use of a polynomial pruning technique which begins pruning at epoch 5, preventing the network from removing critical neurons early on.



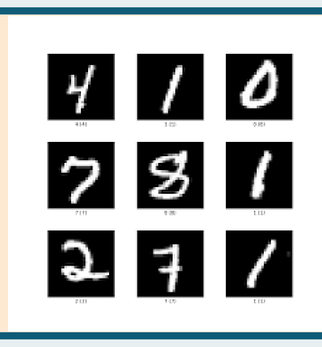


Figure 3: The MNIST image dataset of handwritten digits used to train this pruned network. networks consists of 70,000 images belonging to 10 classes. **Source:** https://deepai.org/dataset/mnist

2D Optimization Problem

A heat map of test accuracy for various combinations of the low-pass filter time constant and the scaling of firing rates. We see a peak value for **synapse** and a constantly increasing trend for **scale_firing_rates**. Accuracy of the SNN approaches that of the non-spiking network as **scale_firing_rates** goes to infinity.

Scaling Individual Layers

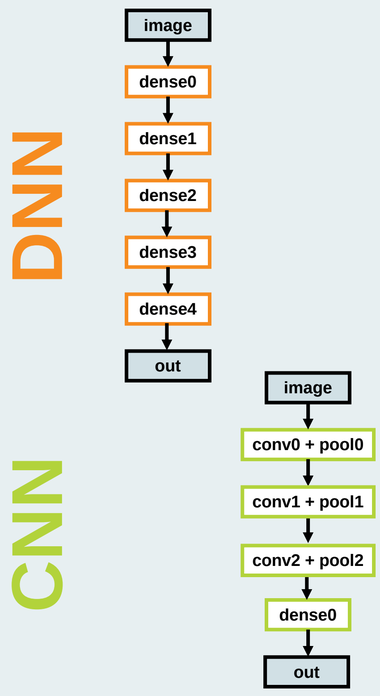
To scale individual layers, we provide a base scaling for all layers to preserve accuracy and boost the scaling in only one layer at a time. This is illustrated by the arrow on the accuracy heat map. We find that scaling the last dense layer in the **DNN** results in the highest accuracy. Conversely, scaling the first convolutional layer in the **CNN** results in the highest accuracy. Scaling of pooling layers has no effect.

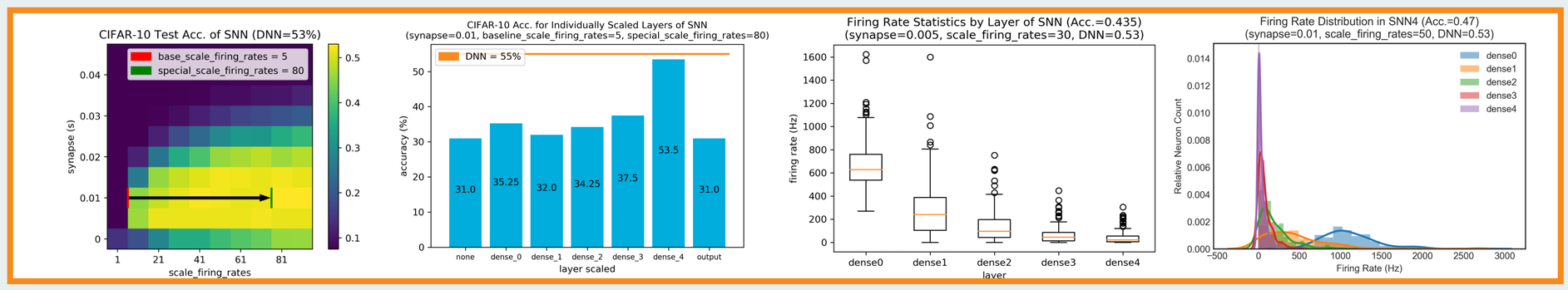
Firing Rate Statistics

We give each layer of the models an equal number of neurons and extract firing rate statistics as a box-and-whisker plot for each layer. In the **DNN**, we observe that firing rates die off significantly as we go deeper into the network. In the **CNN**, we see that firing rates increase across convolutional layers. Firing rates in the **DNN** are initially much higher.

Firing Rate Distribution

To better visualize this data, we plot a histogram showing the relative distribution of firing rates for each layer. A best fit line is included. We notice in the **DNN** that the distribution is initially normal with a wide deviation and becomes more concentrated at zero and heavily skewed right as we go deeper. For the **CNN**, the opposite appears to be true.





Discussion

This research is aimed at optimizing the conversion of **DNNs** and **CNNs** to **SNNs** to allow for computation on novel neuromorphic hardware. Such hardware will come with constraints, making it preferable to keep the firing rates as low as possible. Lowest firing rates and highest accuracy occur in the convolutional network, specifically when only the initial convolutional layers are scaled. Future work involves the expansion of these networks into deeper residual spiking networks, which are capable of obtaining higher accuracy than the networks presented.

Acknowledgements

- I would like to give special thanks to my mentor, **Ray**, for all his guidance and support through the SULI program.
- I would also like to thank **Sandeep Mittal** for his thoughtful help and generosity.
- In addition, I wish to thank the **Office of Educational Programs** (OEP) and my OEP team leader **Amy Engel** for fostering a virtual environment conducive of advanced research and science communication.
- This project was supported in part by the **U.S. Department of Energy**, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI).

U.S. DEPARTMENT OF ENERGY

STEVENS INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

Brookhaven™ National Laboratory