Spiking Neural Networks for Image Classification

AHMADHOSSEIN YAZDANI

OSAZE SHEARS

Outline

- Problem Definition
- Motivation
- Related Work
- Approach
- Results
- Discussion

Problem Definition

Optimizing Hardware for Neural Networks

- Von Neumann Architecture (CPU + Memory)
  - Memory Bottleneck
  - Limited Parallelism

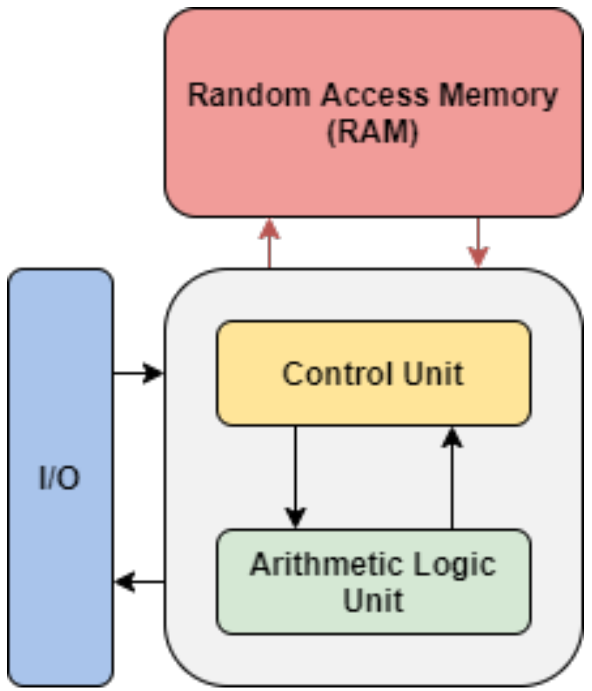


Figure 1. von Neumann Architecture

Problem Definition

Optimizing Hardware for Neural Networks

- Von Neumann Architecture (CPU + Memory)
  - Memory Bottleneck
  - Limited Parallelism
- Graphics Processing Unit (GPU)
  - Power Consumption
  - Different Architecture

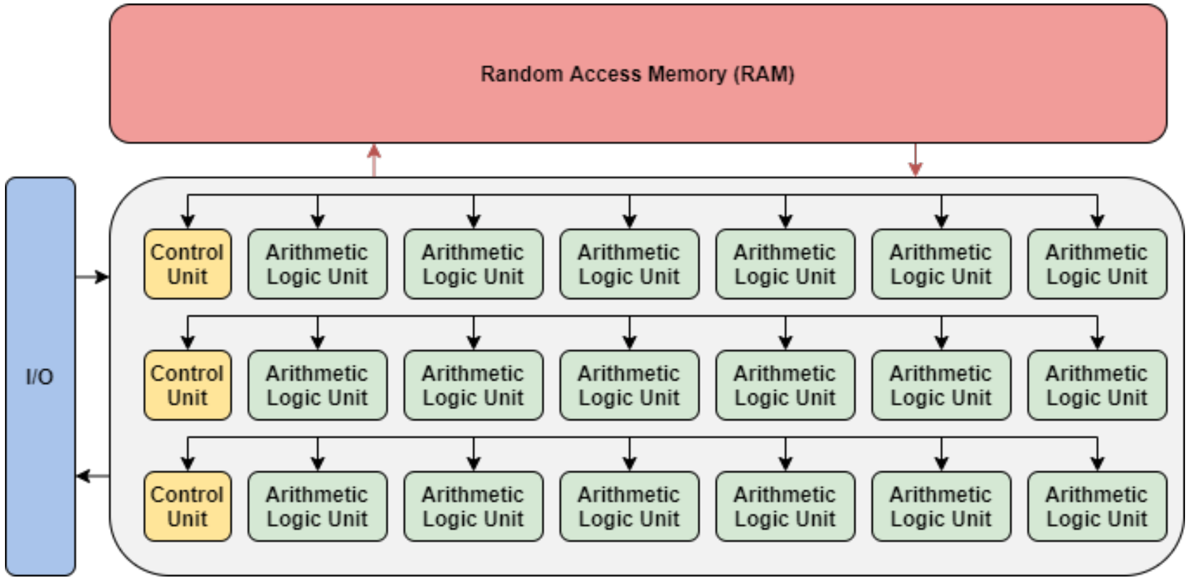


Figure 2. GPU Architecture

Problem Definition
Optimizing Hardware for Neural Networks

- Von Neumann Architecture (CPU + Memory)
  - Memory Bottleneck
  - Limited Parallelism
- Graphics Processing Unit (GPU)
  - Power Consumption
  - Different Architecture
- Neuromorphic Hardware
  - Neurons Individually Represented
  - More Scalable for Power and Area

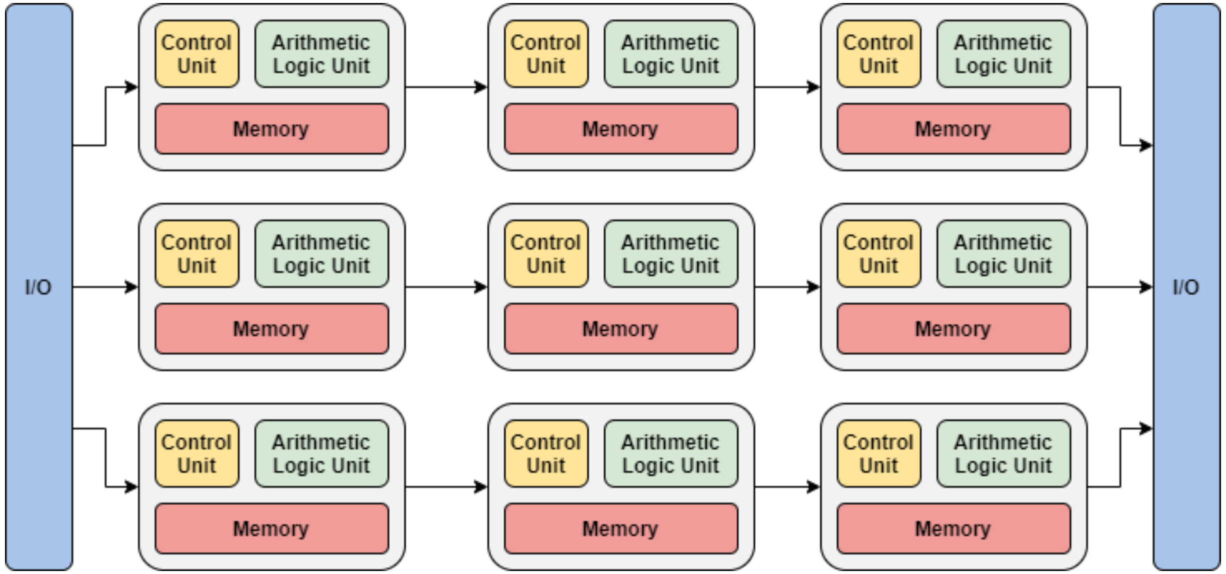


Figure 3. Neuromorphic Architecture

Problem Definition
ANN Hardware Performance Issues

- Computation Speed
- Area Efficiency
- Energy Efficiency

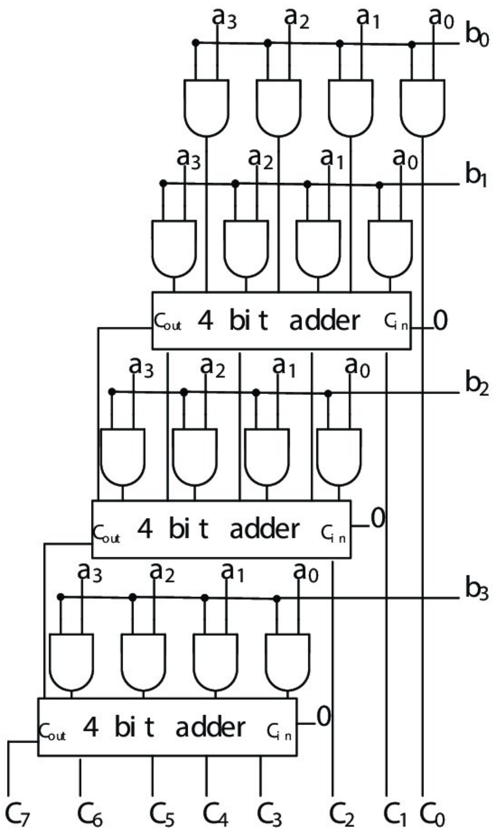


Figure 4. Multiplier Circuit. Adapted from “Traditional 4 bit array multiplier.” by Junzhou Qian and Junchao Wang, 2014, retrieved from researchgate.net

Motivation
Spiking Neural Networks

- Spiking Neural Networks (SNNs) are ANNs that more closely mimic natural neural networks
- "Third generation of neural networks"
- Neurons only transmit data when their "membrane potential" reaches a threshold
- Transmitted spikes will either increase or decrease the membrane potentials of other neurons
- SNNs utilize the concept of time in their execution

Motivation
Traditional ANNs vs SNNs

Network	Execution Time	Neuron Inputs/Outputs	Output Mechanism
Traditional ANN	Instantaneous	Raw Numerical Value	Activation Function (e.g., ReLU, Sigmoid)
SNN	Duration of Time	Binary Spike Value	Threshold Value

Motivation
SNN Advantages

- Energy Efficiency
- Area Efficiency
- Efficient On-Chip Learning
- Fault Tolerance
- Temporal Properties
- Biological Plausibility

Motivation
SNN Components

- Encoding Scheme
- Neural Model
- Learning Technique

Motivation
Encoding Data for SNNs

- Rate Coding
  - The neurons corresponding to inputs with the highest intensities fire more frequently
- Temporal Coding
  - The neurons corresponding to inputs with the highest intensities fire first
- Population Coding
  - The spike times of several input neurons are used to represent the input data

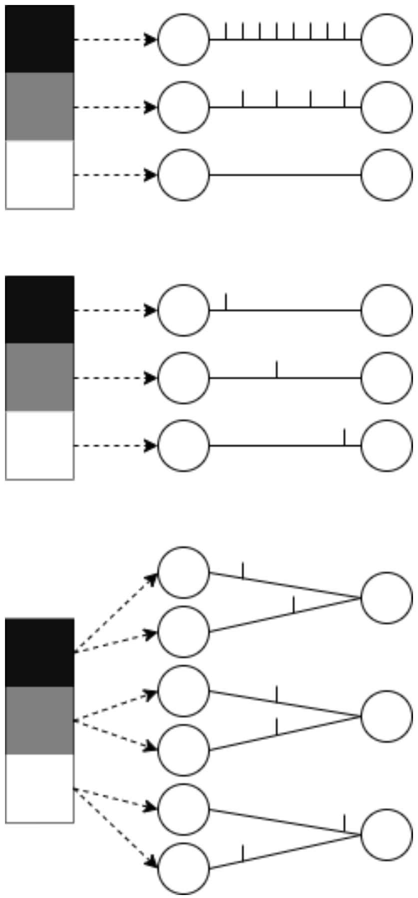


Figure 5. Encoding Schemes

Motivation
Neural Models for SNNs

- Integrate-and-Fire (IF) Model
  - The membrane potential increments until it reaches a specified threshold
- Leaky Integrate-and-Fire (LIF) Model
  - Like the IF model except the membrane potential decrements towards a resting value
- Adaptive LIF Model
  - Like the LIF model except the threshold value increments each time the neuron fires

Motivation
Neural Models for SNNs

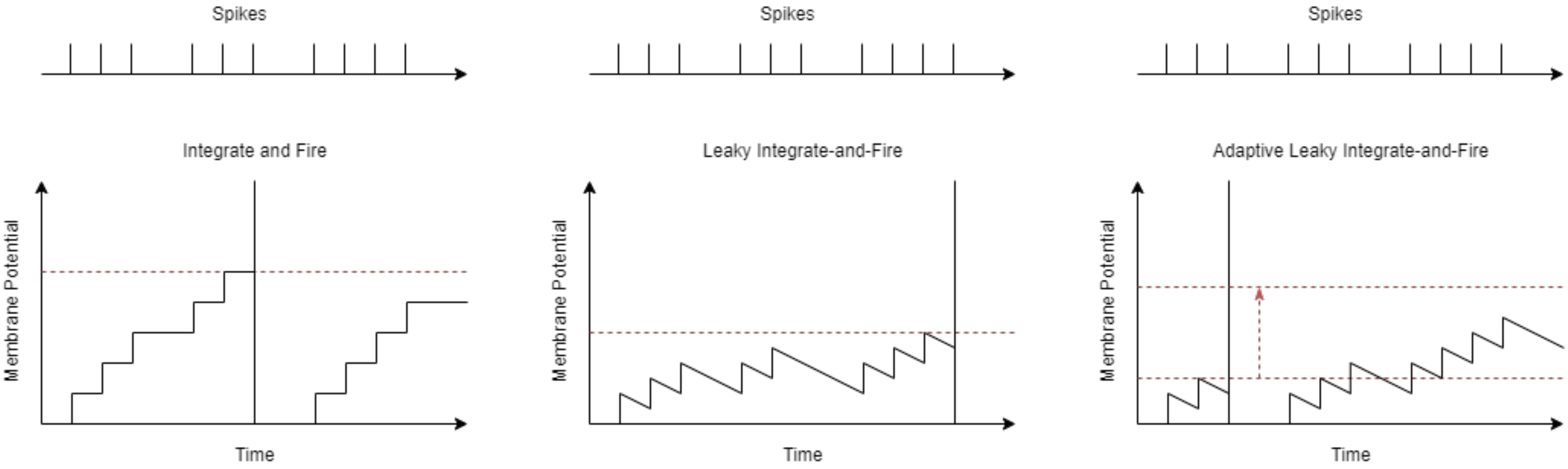


Figure 6. Neural Models

Motivation
Learning Techniques for SNNs

- Spike-Timing-Dependent-Plasticity (STDP)
  - Weight is increased if pre-synaptic neuron fires just before post-synaptic neuron
    - Also known as long-term potentiation (LTP)
  - Weight is decreased if post-synaptic neuron fires just before pre-synaptic neuron
    - Also known as long-term depression (LTD)

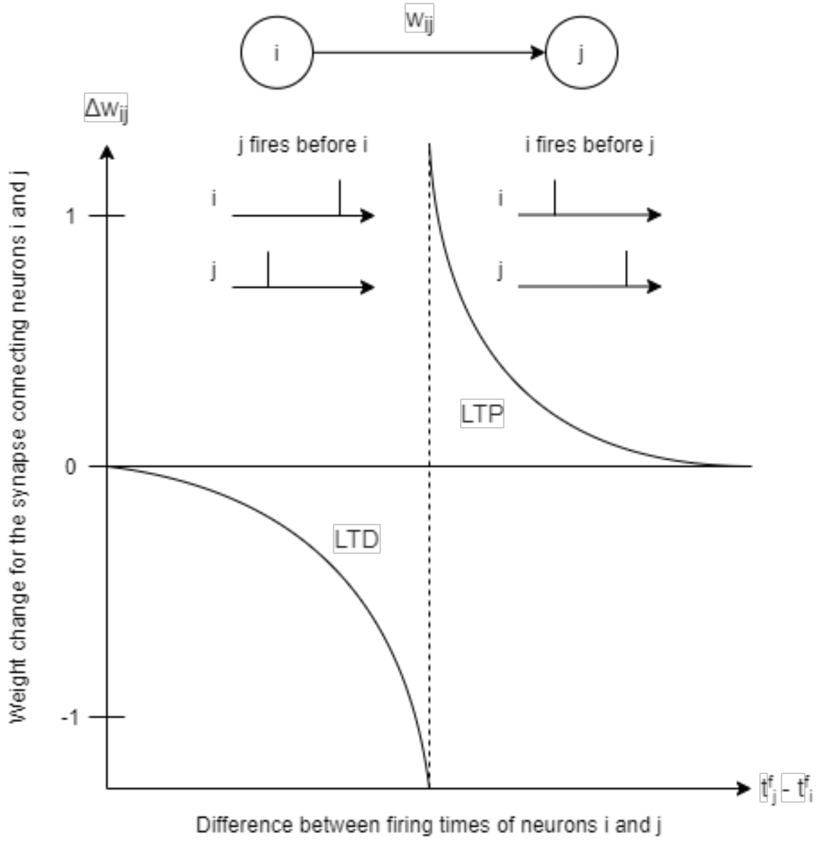


Figure 7. STDP Learning

Motivation
Learning Techniques for SNNs

- Supervised STDP via a Teacher Signal
  - Output neurons are forced to spike for their corresponding labels
- Weighted STDP
  - The negative and positive updates are weighted
- Hebbian learning
  - The update is only positive regardless of the order
- Biologically Inspired Backpropagation

Motivation
Challenges with Implementing SNNs

- Training is difficult
- Accuracy does not match that of traditional ANNs
- Need better metrics to benchmark SNN performance relative to ANNs
- Programming frameworks are still in their infancy
- **Additional research need to determine ideal encoding schemes, neural models and learning techniques**

Related Work
Diehl and Cook (2015)

- Poisson Rate Encoding Scheme
- Adaptive Threshold LIF Neurons + standard LIF Neurons
- 3 Layer Network (Input, Excitatory, Inhibitory)
- 4 Variants of STDP
- Up to 95% unsupervised classification accuracy

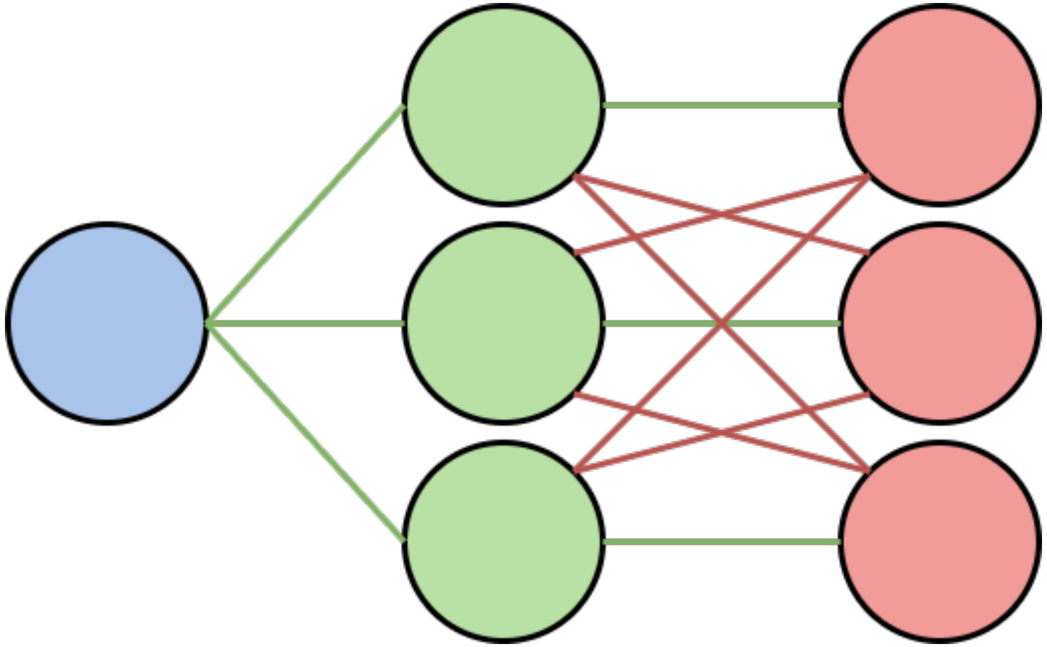


Figure 8. Diehl and Cook Network Architecture

Related Work
Deng et al. (2020)

- Poisson or Bernoulli Rate Encoding Scheme
- LIF Neurons
- 3 Layer Network (Input, Hidden, Output)
- Backpropagation inspired training
- Best SNN achieved:
  - 99.31% accuracy on MNIST
  - 99.08% accuracy on NMNIST

Approach
Objectives

- Expand on the research of Diehl and Cook (2015) and Deng et al. (2020)
  - Report the effect of different encoding schemes on the classification accuracy
  - Report the effect of different neural models on the classification accuracy
  - Report the effect of different learning techniques on the classification accuracy

Approach
Learning Components

- Task: Labelling images of handwritten digits in the MNIST dataset.
- Performance: The accuracy of each model variation when performing classification.
- Experience: The labelled images in the dataset.

Approach
BindsNET Framework

- Software framework published by Hazan et al. (2018) for prototyping SNNs
- Built on-top of PyTorch to support runtime optimizations (e.g. CUDA)
- Provide support for several encoding schemes, neural models and learning rules

```
# initialize network
network = Network()

# initialize input and LIF layers
input_layer = Input(n=input_neurons)
lif_layer = LIFNodes(n=lif_neurons)

# add layers to network
network.add_layer( layer=input_layer, name="Input Layer" )
network.add_layer( layer=lif_layer, name="LIF Layer" )

# connection between the input layer and the LIF layer
connection = Connection( source=input_layer, target=lif_layer)

# add connection to network
network.add_connection( connection=connection, source="Input Layer", target="LIF Layer" )

# simulate network on input data
network.run(inputs=inputs, time=time)
```

Figure 10. Creating a SNN in BindsNET

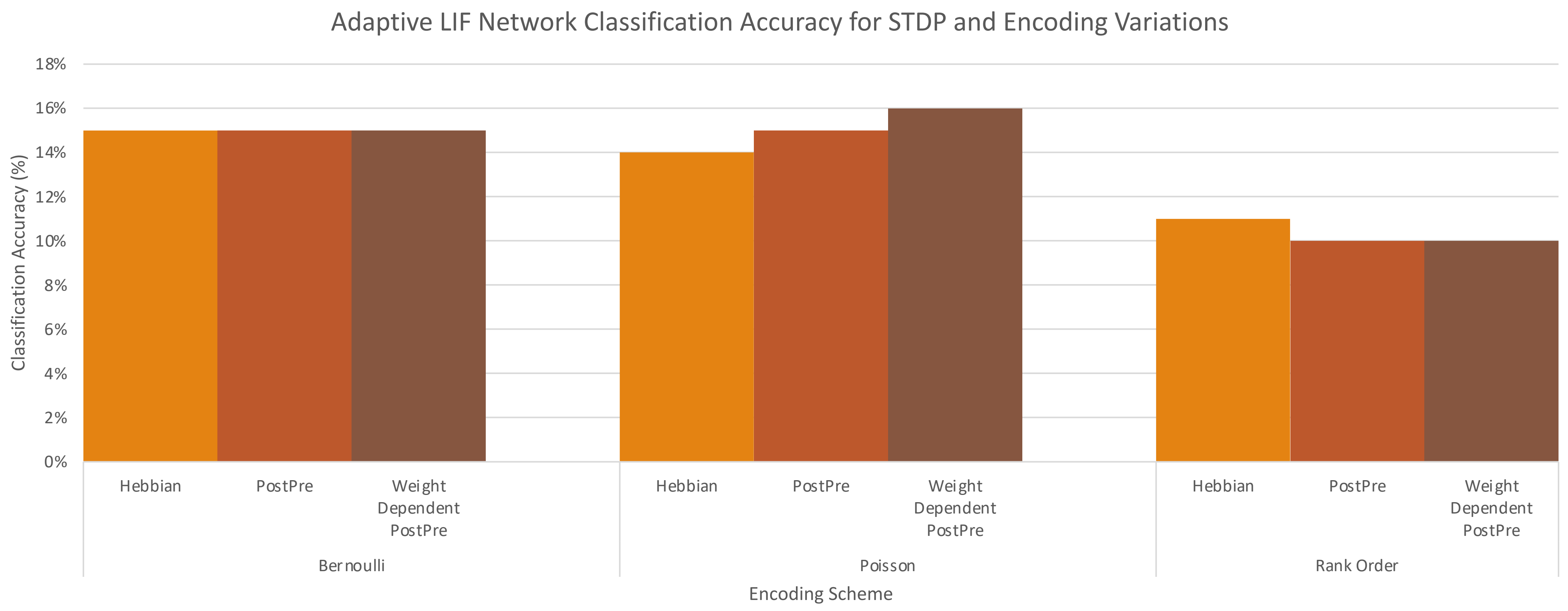
Approach
Experimental Variables

- Encoding Schemes
  - Poisson Rate Encoding (Rate)
  - Bernoulli Rate Encoding (Rate)
  - Rank Order Encoding (Temporal)
- Neural Models
  - IF Model
  - LIF Model
  - Diehl and Cook Model (Adaptive Threshold)
- Learning Techniques
  - STDP
  - Weight Dependent STDP
  - Hebbian

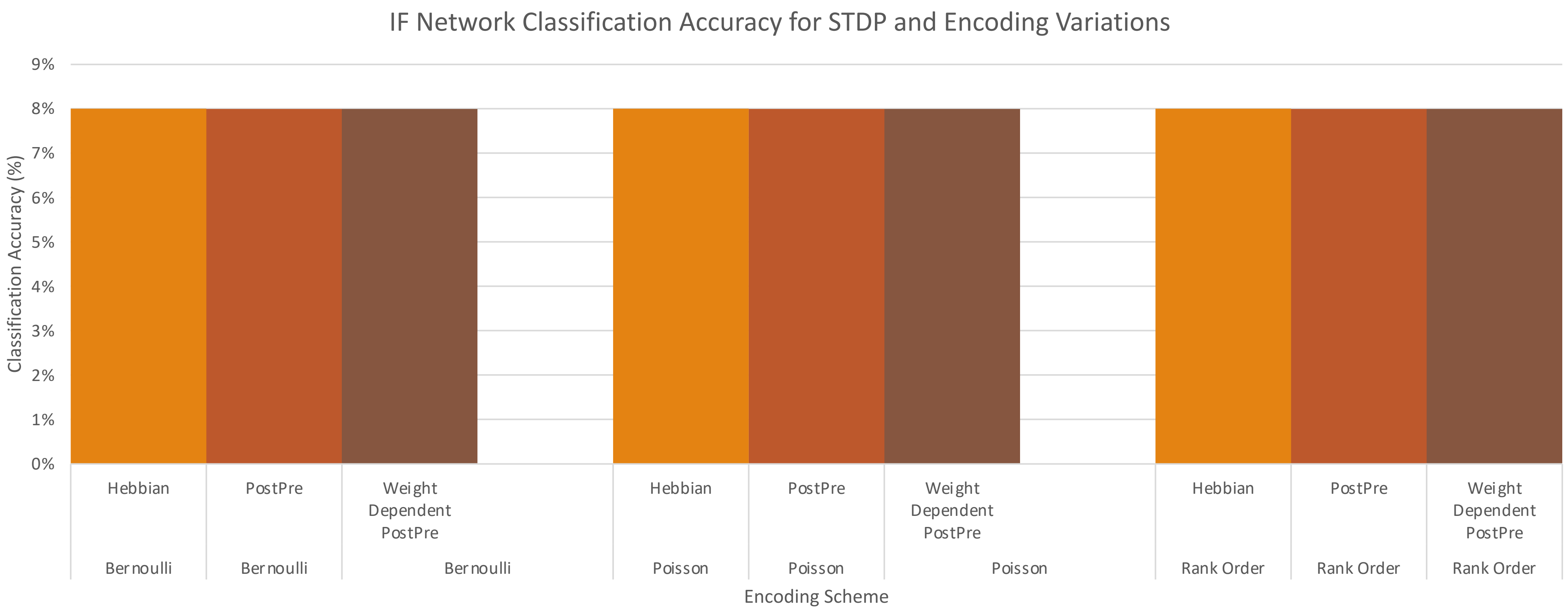
Results
Summary

- Observed 79.75% accuracy for the network using adaptive LIF neurons with Poisson rate coding and STDP learning
- Increased minibatch size to 64 to improve speed of testing different combinations
- Resulting classification accuracy was poor for all combinations
  - Best accuracy was 16% using adaptive LIF neurons with Poisson rate coding and Weight Dependent STDP

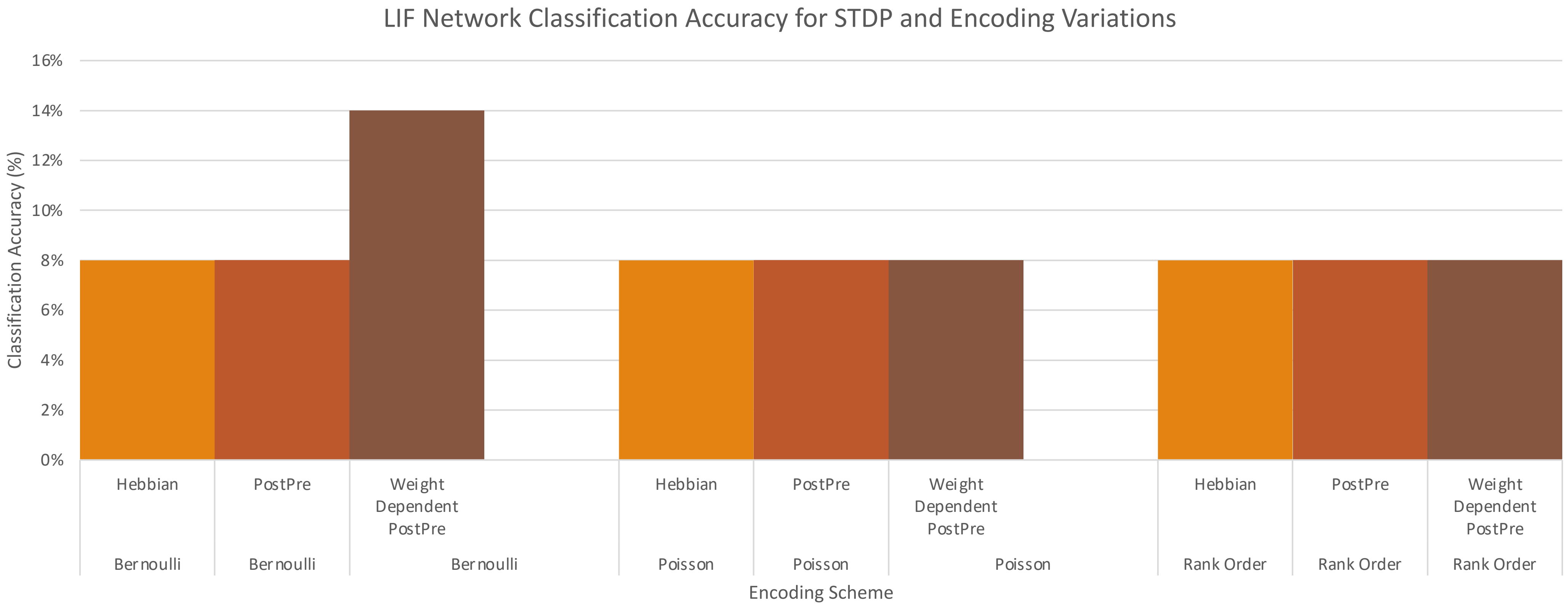
Results
Encoding Scheme Variations



Results
Encoding Scheme Variations



Results
Encoding Scheme Variations



Discussion
Potential Areas of Improvement

- Modify the learning rate based on the batch size
- Use a smaller batch size to improve training accuracy
- Increase the neuron count in the excitatory layer
- Increase the number of simulation timesteps to improve accuracy
- Increase the number of epochs (training samples) to improve accuracy
- Adjust the hyperparameters to accommodate the variations in encoding schemes, neural models and learning techniques

References

[1] Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks*, *10*(9), 1659-1671.

[2] Bouvier, M., Valentian, A., Mesquida, T., Rummens, F., Reyboz, M., Vianello, E., & Beigne, E. (2019). Spiking neural networks hardware implementations and challenges: A survey. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, *15*(2), 1-35.

[3] Li, G., Deng, L., Chua, Y., Li, P., Neftci, E. O., & Li, H. (2020). Spiking Neural Network Learning, Benchmarking, Programming and Executing. *Frontiers in Neuroscience*, *14*.

[4] Diehl, P. U., & Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, *9*, 99.

[5] Deng, L., Wu, Y., Hu, X., Liang, L., Ding, Y., Li, G., ... & Xie, Y. (2020). Rethinking the performance comparison between SNNS and ANNS. *Neural Networks*, *121*, 294-307.

[6] LeCun, Y. (1998). The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*.

[7] Hazan, H., Saunders, D. J., Khan, H., Patel, D., Sanghavi, D. T., Siegelmann, H. T., & Kozma, R. (2018). Bindsnet: A machine learning-oriented spiking neural networks library in python. *Frontiers in neuroinformatics*, *12*, 89.

Image References

Figure 4. Multiplier Circuit. Adapted from “Traditional 4 bit array multiplier.” by Junzhou Qian and Junchao Wang, 2014, retrieved from researchgate.net

Figure 9. Example digits from the MNIST dataset. Adapted from “MNIST Examples” by Steppan, 2017, retrieved from commons.wikimedia.org