2. Background

Spiking Neural Network (SNN) differs from the traditional Artificial Neural Network in a way that it simulates more closely how neural systems work inside our brains. Instead of using continues values between neurons, SNN uses binary spikes. Each neuron has a property called potential with corresponding excitation threshold, which once crossed, will make the neuron fire a spike to consecutive neuron through a synapse connecting both neurons. When a spike arrives the consecutive neuron, its potential will rise by the weight of the incoming synapse. The SNN incorporates time concept such that everything happens in time instead of one static moment. Therefore, it will take certain amount of time for a neuron to receive enough spikes to reach its threshold and fire a new spike to the consecutive neuron. For each sample, the network will run a certain amount of time to generate enough spikes for learning purpose.

Model architecture

…

…

…

Input Layer

Excitatory Layer

Inhibitory Layer

(100)

Encoded Spike Train

MNIST

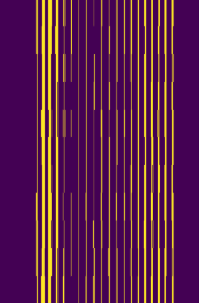


Fig. 1. Architecture of our Spiking Neural Network. The input layer and excitatory layer are fully connected. The synapses that connect these two layers are the core learning part of the network. Each excitatory neuron only connects forward to one inhibitory neuron, and each inhibitory neuron connects backward to every excitatory neuron except the one that connects forward to it.

Our SNN consists of 3 layers: input layer, excitatory layer, and inhibitory layer. The input layer consists of 784 neurons corresponding to 784 pixels of each image sample from MNIST dataset. The excitatory layer consists of 100 neurons responsible for classifying samples. The inhibitory layer also consists of 100 neurons responsible for lateral inhibition. The original MNIST data consists of statics images, we encoded those static images into Poisson spike train with fire rate based on pixel value and make input neurons fire at corresponding rates to represent an image sample. An input neuron representing a bright pixel will fire at high frequency and vice versa. The fire rate is calculated as follow

where is the fire rate, is the pixel value, is the maximum firing rate the network is designed to handle, is the maximum value a pixel can have, which is 255 for most images.

For neurons, we are using leaky integrate-and-fire model to simulate potential behavior of biological neurons. The potential of each neuron will exponentially decay in time to its reset value unless a spike is received. Each spike will increase the potential of a neuron by a set value. Once enough spikes are received causing the potential to cross threshold, the neuron will fire a spike to its consecutive neuron. Each neuron has a refractory period, , that prevents potential update for a short amount of time when it fires a spike. The potential dynamics is described by the following equations

where (1) describes potential update when a spike is received, is the new potential, is the potential before spike arrives, is a scaler that reduces weight ranging from 0 to 1 to a small enough value as threshold is usually around , and is determined based on the number of input neuron because more input neurons require lower average synaptic weights to achieve similar potential increase of excitatory neurons. is an efficacy term that determines how effective the neuron will update its potential, is the synaptic weight, (2) describes exponential decay of the potential when there is no spike received, is the decay time constant. It is notable that the potential will eventually decay to zero is no spike is ever received.

We also incorporate homeostasis model for neurons to prevent some neurons from being too active and thus suppress other neurons’ learning ability. Homeostasis will raise a neuron’s threshold whenever it reaches threshold and fires a spike. This makes it harder for that neuron to fire again in short amount of time and gives other neurons opportunity to fire. The threshold dynamics is described as follow

where (1) describes how threshold is increased upon firing, is the threshold after firing, is the threshold before firing, is the amount the threshold is increased by, (2) describes how threshold decays if there is no spike being fired, is the rest threshold, is the decay constant that determines how fast the threshold should decay to its rest value.

For synapses connecting neurons, we are using spike-timing-dependent plasticity (STDP) model to simulate learning behavior of biological synapses, and thus train our network. When a post-synaptic neuron fires a spike, the synaptic weight will update based on the synapse activity before the neuron reaches its threshold. If the synapse is very active and sends a lot of spikes to the post-synaptic neuron, the synapse is highly relevant to the neuron firing and therefore, its weight should increase. The closer those spikes are to the neuron firing in time, the greater the weight increase should be. On the other hand, if the synapse is relatively inactive and does not send many spikes, its weight should decrease. The activity is increased whenever a spike is sent to post-synaptic neuron through a synapse. Only synapses that connect input layer to excitatory neuron will have STDP-based weight update. Synapses that connect excitatory layer and inhibitory layer, both forward and backward, do not have weight update mechanics. The synapse dynamics are described by the following equations

where (5) describes activity increase of a synapse when a spike is sent to post-synaptic neuron through that synapse, is the activity after the increase, is the activity before the increase, is the activity increment, is the smoothing term. (6) describes how the activity of a synapse decays exponentially in time if there is no spike sent through that synapse, (7) describes how weight update of a synapse is calculated upon a post-synaptic neuron firing event, is the weight update, is the learning rate, is the activity of that synapse at the moment of the firing event, is an offset that makes synapses with low activity to have negative weight update, is a smoothing term that makes weight updates less when the weight is higher.

Because some classes have higher effective pixel counts than others, these classes will always activate excitatory neurons more frequently and become dominant, which prevents the system from learning other classes. Homeostasis alone does not prevent all neurons from responding to one single pattern, and thus cannot solve such problem. Inhibition is needed to allow the best neuron with the most suitable synaptic weight for a certain pattern to fire and then suppress other neurons. For each sample, once an excitatory neuron fires the first spike, its consecutive inhibitory neuron will fire suppressing spikes to every other neuron to reset their potential. In addition to resetting the potential, every neuron’s efficacy will decrease due to inhibition. The neuron that fires spike and activates inhibition will also have decreased efficacy. Decreased efficacy means every neuron will gain less potential update from received spikes, and the system will not train too much on a single sample. The dynamics of efficacy is described by the following equations

where is the efficacy after the update, is the efficacy before the update, is the ratio that determines how much the efficacy should decrease upon inhibition or firing event.

We train the network by starting with randomized synaptic weight and feed each sample images to the network for a certain training period . At the start of each sample training period, every variable that changes during learning will be reset to its rest value, including , except for threshold, , as it will need to stay at high value and slowly decay through multiple samples to prevent a neuron from learning multiple samples as described in homeostasis section.

3. Optimization Framework

Once the network is trained, target pruning ratio, , is set. The pruning threshold, , then can be found at percentile of all synaptic weights. Any synapses with weight lower than will be removed from the network. Weight quantization is also realized when accessing each synaptic weight based on different quantization levels set for the network. Synaptic weight will be rounded to nearest quantized value. After pruning, any neuron without any synapses connected will then be removed from the network as well. The algorithm for pruning and quantization is described by the pseudo code below:

4. Experimental Results

We trained our network on samples from MNIST dataset using parameters listed in the following table

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
|  |  | Maximum firing rate for encoded spike train |
|  |  | Initial potential of each neuron |
|  |  | Initial efficacy of each neuron |
|  |  | Potential decay time constant |
|  |  | Initial threshold |
|  |  | Scaler |
|  | 0.01 | Amount of threshold increased upon firing event |
|  |  | Threshold decay time constant |
|  | 0 | Initial activity |
|  |  | Activity increment |
|  |  | Activity decay time constant |
|  |  | Learning rate |
|  |  | Offset of weigh update |
|  |  | Efficacy update ratio upon inhibition |

Tab. 1. Parameters used in our model.

Currently the overall accuracy of our un-pruned network is .

## Related Works on SNN-Based Machine Learning

Peter et al.[1] implemented a network with similar learning rule as used in our model, with different excitatory layer size, including 100, 400, 1600, 6400 neurons, and has achieved accuracies of , , , respectively on MNIST dataset, without any compression method. Nitin et al.[2] implemented a batch-by-batch pruning-during-training compression method. and reached accuracy of for non-pruned model and for compressed model. They have also conducted energy analysis on their network model with different pruning ratios. Yuhan et al. [3] implemented a network utilizing different mechanics that uses global information across the network. They used probabilistic model to determine output neuron firing event, and the weight update of each single synapse needs to know the number of other synapses that also need updates, which is different from our model where every update is determined by local information. Their excitatory consists of 500 excitatory neurons, and they reached accuracy of without pruning and with pruning rate.

[1] Diehl, Peter U., and Matthew Cook. "Unsupervised learning of digit recognition using spike-timing-dependent plasticity." *Frontiers in computational neuroscience* 9 (2015): 99.

[2] Rathi, Nitin, Priyadarshini Panda, and Kaushik Roy. "STDP-based pruning of connections and weight quantization in spiking neural networks for energy-efficient recognition." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38.4 (2018): 668-677.

[3] Shi, Yuhan, et al. "A Soft-Pruning Method Applied During Training of Spiking Neural Networks for In-memory Computing Applications." *Frontiers in neuroscience* 13 (2019): 405.