Paramater Sensitivity of Spiking Neural Network Using MNIST Dataset

Zi-Yi Tai
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan
b06901007@ntu.edu.tw

***Abstract*—Spiking Neural Network are able to perform digit recognition with unsupervised learning successfully [1]. However, in the original network there were a lot of fixed parameters during the simulation while there were not any obvious reason to set them that way. In this work, we looked into the original network and analyzed how different parameters may affect the results. We also ran the network on another dataset that involves hand-written letters recognition. The results show that the original network is highly sensitive to its parameters and the initial weights of the network may have a large influence on the performance.**

Keywords—spiking neural network, parameter sensitivity

I. Introduction

Spiking neural networks (SNN) are artificial network that simulates natural neural networks. Different from the usual ANN in machine learning which passes 32-bit or 64-bit values between units and units, SNN encodes the values in one bit by sending a spike from one neuron to another. It is shown that SNN are able to perform digit recognition with unsupervised learning successfully [1]. To simulate SNN, we use Python and the BRIAN simulator [3]. However, in the original network there were a lot of fixed parameters during the simulation while there were not any obvious reason to set them that way. In this work, we looked into the original network and analyzed how different parameters may affect the results. We also ran the network on another dataset that involves hand-written letters recognition.

In this remaining section, a brief introduction on SNN would be presented. The network architecture is in section 2. Section 3 and 4 shows the results of adjusting different parameters of the network. The model performance on the letter recognition dataset is in Section 5.

A. Spiking Neural Network: Neurons and Synapes

Unlike the usual ANN in machine learning which passes 32-bit or 64-bit values between units and units, in SNN, information are transmitted between neurons through synapses with different weight and delay. Synapses are connections between neurons. Figure 1 shows the membrane potential of a post-synaptic neuron. When a spike is received in a post-synaptic neuron, the membrane potential increase. If the membrane potential reach the neuron threshold voltage, the post-synaptic neuron would fire a spike and set its voltage back to its resting potential. A spike can occur anytime, traveling along the synapse. Since the timing of spikes also encodes information, every neuron would fire spike at different time. Hence, the network can be seen as an asynchronous network.

B. Learning Rule: Spike Timing Dependent Plasticity

Spike Timing Dependent Plasticity is a way to update the weights of synapses. The mechanism can be stated as follows:

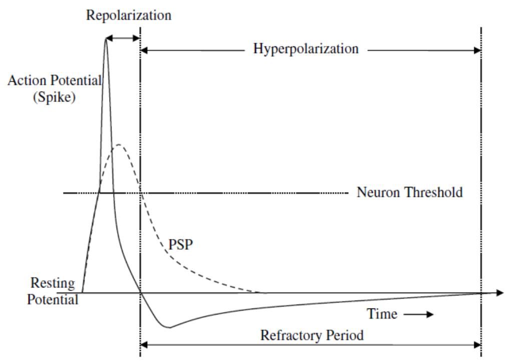


Figure 1. Membrane potential of a post-synaptic neuron [4].

- If a pre-synaptic spike causes a post-synaptic spike immediately, the synapse tends to become stronger.
- On the contrary, if a post-synaptic spike fires right before a pre-synaptic spike, the synapse should become weaker.

The change of the weight depends on the time difference between the spikes. In our work, this dependence can be shown in Figure 2, where Δt stands for the time difference between the post-synaptic and pre-synaptic spike and W stands for the value added to the weights of the synapse. We can tell that the synapse is strengthened the most when the post-synaptic spike fires right after the presynaptic spike.

II. Network Architecture For Digit Recognition

A. Network Architecture

The network consists of three layers (Figure 3). The input layer has 28 x 28 neurons (one neuron per image pixel). The second and third layer are the excitatory and inhibitory layer. We transformed each image into a Poisson spike-train, where the rate of each neuron is proportional to the intensity to the corresponding pixel. Since the input layer is fully connected to the excitatory layer, some of the excitatory neurons would spike when each image is shown to the network.

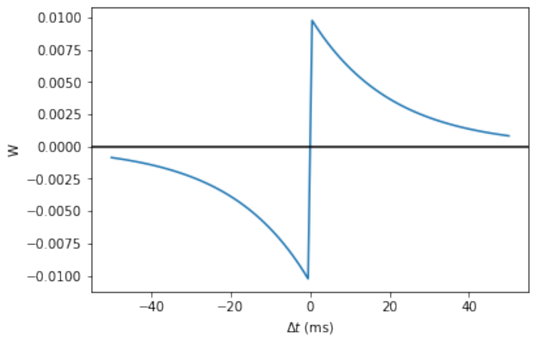


Figure 2. The mechanism of STDP rule.

There are the same amount of neurons in the excitatory and inhibition layer, where the neurons are connected in a one-to-one fashion. This means each spike in an excitatory neuron would trigger a spike in its corresponding neuron. Each inhibition neuron is connected to every other excitatory neuron except the one it corresponds to. When an excitatory neuron spikes, it would inhibit all the other excitatory neurons from spiking. This provides a lateral inhibition and leads to competition among excitatory neurons.

B. *Running Simluations*

The network is trained on 60000 samples and tested on 10000 samples from the MNIST dataset. Both training data and testing data are provided with their labels. There are three phases of a simulation, which are listed as follows.

- *Training Phase.* The training dataset (60000 samples) are shown to the network. Weights between the input and excitatory neurons are updated through the Spiking Time Dependent Plasticity (STDP) rule so that every excitatory neuron is trained to learn a specific type of sample. The labels are not used in this phase. Hence, training phase is simulated by unsupervised learning.
- *Inference Phase.* Since each excitatory neuron is trained to learn a specific type of sample, we would like to assign each of them to a label. This is accomplished by showing the training dataset to the network again without updating the weights. For each excitatory neuron, we calculated average number of spikes caused by each class label, and assigned the neuron to the label with the highest value.
- *Testing Phase.* The testing dataset (10000 samples) are shown to the network. For each sample, we find the excitatory neuron that spikes the most and assign its label to the sample.

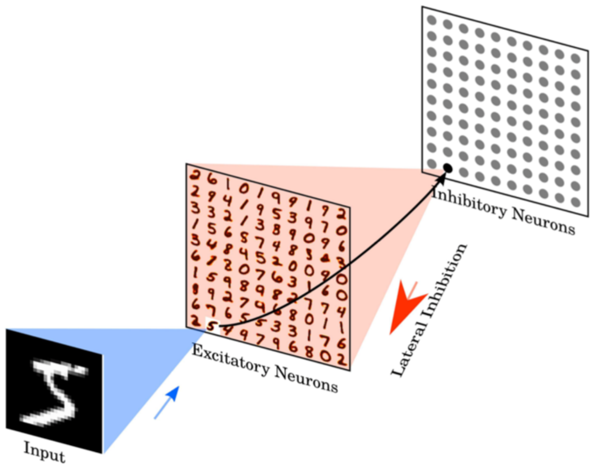


Figure 3. Network Architecture [1].

III. Tuning Parameters

We would like to know how each parameter in the network affects the performance. Hence, in this following section, we focus on several parameters listed as follows and how a slight difference would contribute to the network performance. For comparison, we first trained a model on 60000 training samples and 10000 testing samples with 400 neurons. This model achieves a 88.64% accuracy on the testing set. The weights between the input layer and the excitatory layer can be rearranged from 784 into 28 x 28 and shown in Figure 4. Each neuron is trained to learn a type of sample. It takes 12 hours to run through the whole simulation.

A. *Number of Training Samples*

Although the size of the MNIST dataset is fixed, we can expect the performance of our model to improve if the training dataset is presented more than once. The results are shown in Figure 5. The accuracy on testing data improves from 83.36% to 89.61% as the number of training samples increased from 10000 to 18000. However, the accuracy does not increase unlimitedly and stops around 89%. Hence, we would have to adjust some other parameters to reach a better accuracy.

B. *Voltage of the Neuron Membrane*

Figure 6 shows the membrane potential of a post-synaptic neuron. When spikes are received from a pre-synaptic neuron, the membrane potential increases. If the potential bypass a threshold voltage, it would cause a spike in the post-synaptic neuron and the membrane potential falls to the resting voltage. In the original code, the threshold voltage and the resting voltage of the neurons are set to a constant, which are shown in Figure 6 as well. The value of the voltages are set so as to simulate the actual voltage in human neurons. However, we would like to know if the value of these voltage actually affects the performance of the network. To do so, we added a voltage offset to both voltages in order to keep the voltage difference the same.

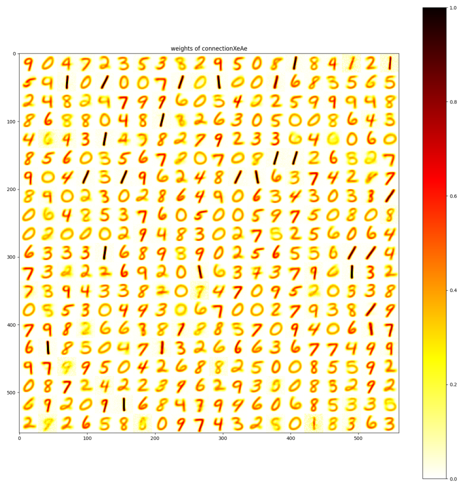


Figure 4. Weights of connections from input to excitatory neurons.

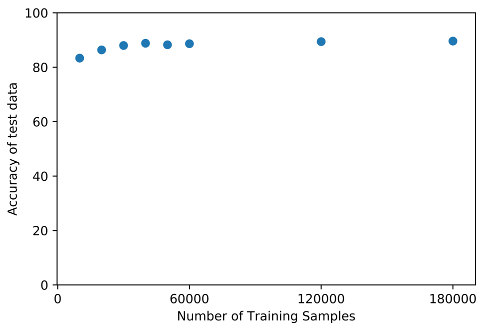


Figure 5. Network Performance on different number of training samples.

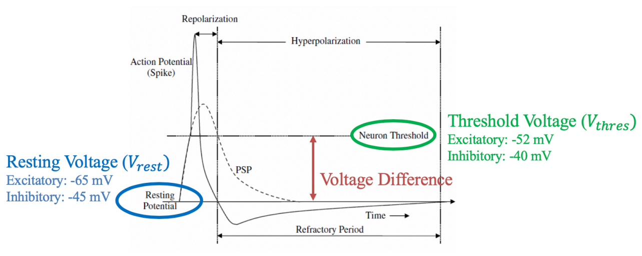


Figure 6. Membrane Potential of a post-synaptic neuron [1].

The voltage offset does not affects the network performance much when the voltage offset is negative. However, the accuracy drops a lot when both voltage becomes larger. Furthermore, the training cannot be completed (no spikes caused by each sample) if the voltage offset is larger than 20 mV. This can be explained by the equation of the membrane potential of a post-synaptic neuron, as in (1).

$$\tau \frac{dV}{dt} = (E_{rest} - V) + g_e(E_{exc} - V) + g_i(E_{inh} - V) \quad (1)$$

If both voltages increase, the absolute values of voltages become smaller and closer to zero. This may cause problems since the membrane potential is governed by the differential equation in (1). When the value of voltage is closer to zero, the voltage changes at a faster rate. Thus, when a post-synaptic neuron receives spikes at different time, the effects from these spikes might not be able to accumulate, leading to the results that no spikes would occur in the excitatory neurons.

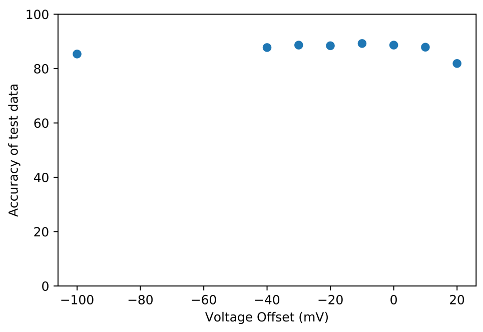


Figure 7. Network Performance on different voltage offset.

C. Adaptive Membrane Threshold

To ensure that every neuron has an equal firing rate, an *adaptive membrane threshold* is applied. When an excitatory neuron is excited and released a spike, the threshold voltage is increased by θ. It would be harder for this neuron to spike for a while, as θ would decay exponentially. This provide a better chance for other neurons to spike and prevent single neurons from dominating the response. Let this time constant be τ. Originally in the code, τ has a value of 10^7 (s), which may seem like a too large time constant that does not decay at all. Hence, we would like to change τ and see how it affects the performance.

We can tell from Figure 8 that when τ becomes smaller, the performance is affected tremendously. The accuracy drops to 56.06% and Figure 9 shows the weight connections between the input layer and excitatory layer when τ is set to 10^5 (s). Only some of the neurons were trained and learned a pattern while the others never participated in the training phase. The accuracy slightly increases to 90.66% when τ is raised to 10^8. However, the excitatory neurons stopped spiking after 30000 training samples are shown since the voltage threshold eventually becomes too high and no spikes are able to occur anymore. Hence, setting τ to 10^7 (s) is a rather accurate choice.

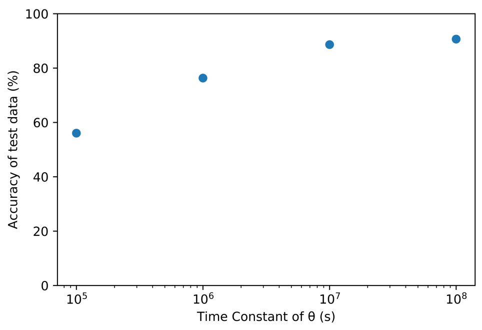


Figure 8. Network Performance on different time constant of θ.

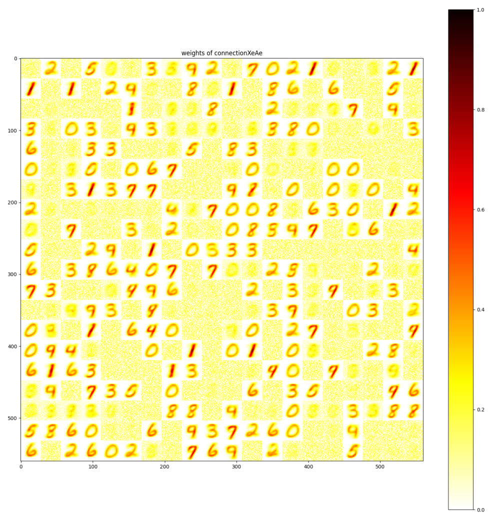


Figure 9. Weights of connections from input to excitatory neurons with $\tau = 10^5$.

D. Number of Neurons

If there are more neurons in the excitatory and inhibition layer, the network is expected to perform better since there are more neurons to learn the different patterns. However, it turns out that the accuracy falls dramatically when the number of neurons is not 400, as shown in Figure 10. This may result from the fact that a specific set of initial random weights is used for 400 neurons. This set of initial weights is from the original code. For the other number of neurons, we generated the initial weights from a uniform distribution between 0 and 0.33, since 0.33 is the maximum value in that specific set of initial weights.

Intuitively, this shall not be a problem since a slight difference in the initial random weights should not affect the results. If we look into the weights after training 225 neurons, we can see that most neuron learned a pattern well. However, during the inference phase, only a small portion of the neurons spike, which would mislead the label assignments of the neurons. Table I shows the comparison between 225 and 400 neurons. All neurons have equal spike rates for 400 neurons. Hence, most neurons are used in the inference and testing phase. However, the spikes are not distributed evenly throughout 225 neurons, which implies that only a small portion of neurons are used. These neurons are also the ones with smaller θ value and thus are easier to spike. We concluded that the network performance is very sensitive to the parameters, and that the number of neurons must be adjusted with other parameters such as θ and τ.

We can tell that as the noise increases, the model would have lower accuracy despite the same amount of training data. This is consistent with the assumption that this specific initial weight is important to the model performance. The reason to this is still unclear for now.

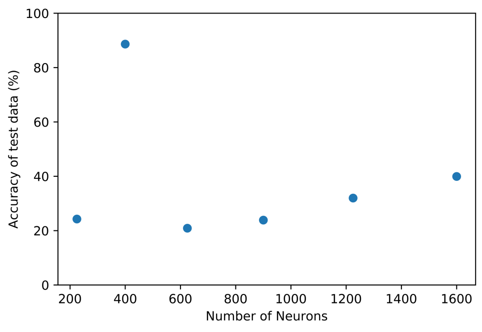


Figure 10. Network Performance on different number of neurons.

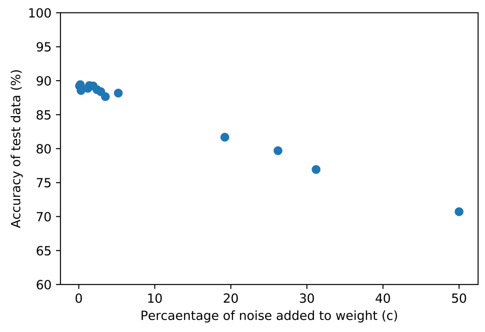
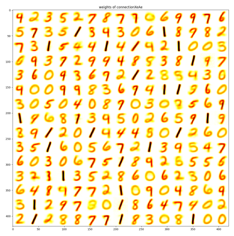
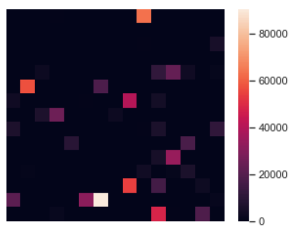
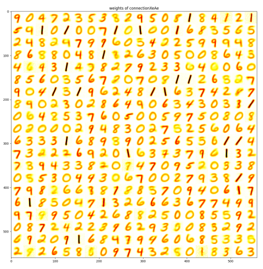
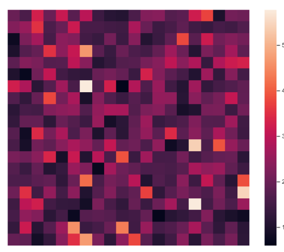


Figure 11. Network Performance on different percentage of weight noise.

TABLE I. COMPARISON BETWEEN 225 AND 400 NEURONS

225 Neurons	
Weights between input and excitatory neurons	***Number of spikes for each neuron in inference phase***
	
400 Neurons	
Weights between input and excitatory neurons	***Number of spikes for each neuron in inference phase***
	

IV. ADD NOISE TO INITIAL RANDOM WEIGHT

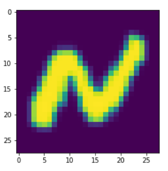
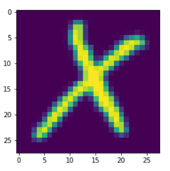
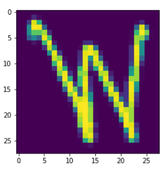
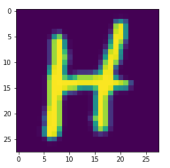
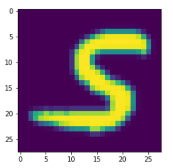
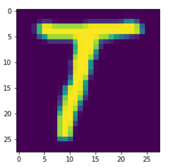
We ran the simulation on 400 neurons with our own generated initial weights and the accuracy falls to 20.1% from 88.64%. This indicated that the network achieves better performance with 400 neurons because of the specific set of initial random weights provided in the original code. We would like to verify this hypothesis by adding some noise to the specific set of weights.

We added a random value sampled from a uniform distribution to the set of weights. Let the average of the original weight be w_{ave}. The noise can be represented as c × r × w_{ave}, where c is a constant and r is a sample from a uniform distribution between 0 and 1. In other words, c is the percentage of noise added to the weight.

V. RUNNING THE SIMULATION ON ANOTHER DATASET

The network is also trained on another dataset, the EMNIST dataset. This dataset is a variant of the original MNIST dataset, following the same conversion paradigm used to create MNIST dataset [2]. It involves more challenging classification tasks involving not only hand-written digits but also letters as well. Originally there are 62 classes, considering all digits and letters. However, since some letters have very similar upper and lower cases, some classes can be merged and rearranged into 47 classes in total. For 400 neurons, it may be difficult to distinguish this large number of classes. Hence, we only consider the upper case letters, which is 26 classes in total, in our simulation.

TABLE II. SOME EXAMPLES IN THE EMNIST DATASET

124800 of training samples and 20800 testing samples was used, which contains 4800 training and 800 testing samples for each letter. With the large number of dataset size and number of classes, the training of the model takes about 36 hours for one simulation. There are 400 neurons in the network and the specific initial random weight mentioned above was applied. However, the results were barely satisfactory. The accuracy on testing data was only 23.97%.

There are a lot of possible reasons for the poor results on EMNIST dataset. The initial random weights may have to be adjusted due to the new dataset since we now know that these weights have a large influence on the performance of the model. To do so, we shall understand the reason for the importance of that set of weights and how it affects the network training. Another reason might be the parameter sensitivity of SNN. Since the number of classes increased from 10 to 26, it is reasonable that more neurons are needed in order to learn the different patterns. However, we cannot yet conduct this experiment until we solve the problem of the initial weights.

VI. CONSLUSIONS

The Spiking Neural Network proposed in [1] that performs digit recognition is very sensitive to its parameters and may be likely to fail when one parameter is slightly adjusted. In this work, we discussed the effects on the number of training samples, voltage offset, adaptive membrane threshold, and the number of neurons. We found out that the initial random weights provided in the original code has a significant influence on the results, where the reason to this remain unclear. The network is also run on another dataset that involves letter recognition.

REFERENCES

[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] Cohen, Gregory, et al. "EMNIST: Extending MNIST to handwritten letters." *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017.

[3] Stimberg, M, Brette, R, Goodman, DFM. “Brian 2, an Intuitive and Efficient Neural Simulator.” eLife 8 (2019): e47314. doi: 10.7554/eLife.47314.

[4] Ghosh-Dastidar, Samanwoy, and Hojjat Adeli. "Spiking neural networks." *International journal of neural systems* 19.04 (2009): 295-308.