Information Encoding Methods and Spike-Timing Dependent Plasticity

Amir Faridi - 610300087

Department of Computer Science, University of Tehran

Computational Neuroscience

1. Introduction

In this project I begin by dicussing and discovering different methods of information encodings and their efficiencies and accuracy after trying to decode them. Then I will move on to Spike-Timing Dependent Plasticity (STDP) and Reward-Modulaated STDP (RSTDP) learning methods for spiking neurons.

2. Encoding Methods

2.1. Time to First Spike

In this method, I used grayscale images as inputs and take one neuron for each pixel in the image, and then map the intensity of the pixel to the latency of the first spike of the corresponding neuron.

Throughout this report, I will use the following images as inputs, and refer to them as human, circles, and circle.

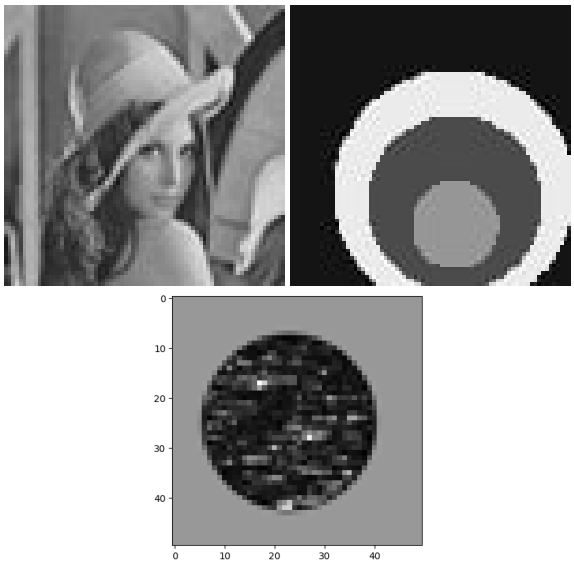

Figure 1. *The images will be resized to 50x50 or 25x25 pixels in the experiments to avoid heavy computations.*

Time to first spike of each neuron is calculated by the following formula:

$$ttfs_i = T * pixel_i/255 \quad (1)$$

where T is the time that the input is presented to the network. This gives a one-to-one mapping between the pixel intensity and the time to first spike of the corresponding neuron, which can be reversed to decode the image.

The followings are the results of the encoding and decoding of the images using the Intensity to Latency method.

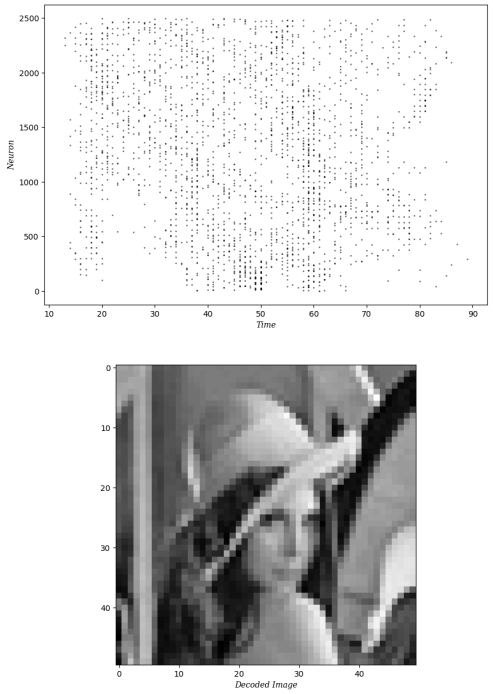

Figure 2. *In this figure, the input is the human image and T is 100 units.*

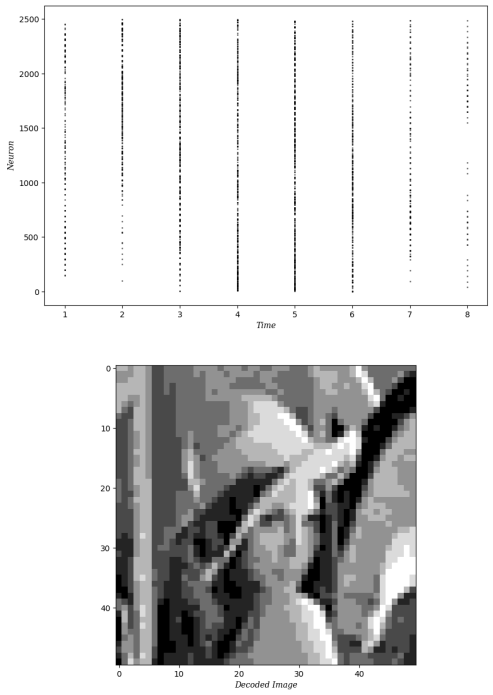

Figure 3. *In this figure, the input is the human image and T is set to 10 units.*

As it is clear from the figures, the coding is almost perfect when the parameter T is set to 100 (or higher), but when it is

set to 10, the image is not decoded correctly and the quality is very low. This can be interpreted as the fact that the image does not encode correctly when the time is not enough for the network to encode the input. From the perspective of the encoding formula, the pixels are mapped into some numbers between 0 and T, and when T is small, the difference between the latencies of the neurons is not enough to distinguish the pixels from each other.

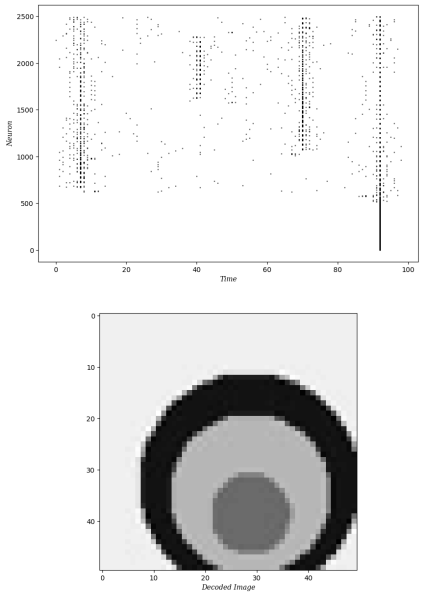

Figure 4. *In this figure, the input is the human image and T is 100 units.*

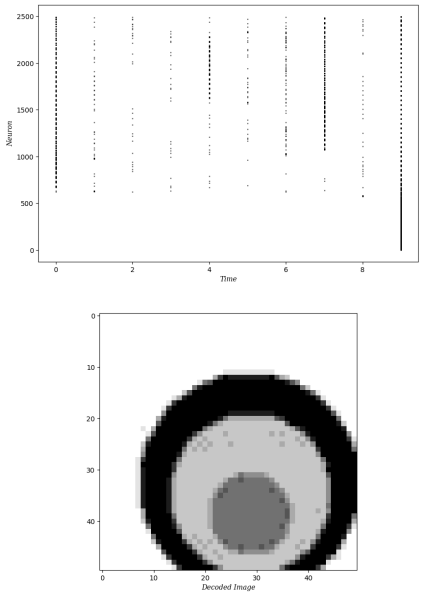

Figure 5. *In this figure, the input is the human image and T is set to 10 units.*

This is the results for the circles image as the input. Each pixel in this image is inversed; meaning that the new value of each pixel, $pixel_i$, is $255 - pixel_i$. This is because that the raster plots of the activity of the neurons are more readable now.

Here as before, the coding is perfect for $T = 100$, but it lacks quality for $T = 10$. The same interpretation can be made here. In the raster plots, there are a time steps when the spikes of the neurons are more dense, and those are the parts of the image that are more intense (darker).

2.2. Representing Numbers

In this coding method, neurons with gaussian distribution is used to divide the input space into some regions, and the intensity of the pixel is mapped to the probability of the neurons in the corresponding region to spike. Hence, each number is represented by a vector of the required time of the neurons to spike. Depending on the number of the neurons and the parameters of the gaussian distribution as well as T, the quality of the coding can be changed. This method can be more computationally expensive than the previous one depending on the number of neurons we are using.

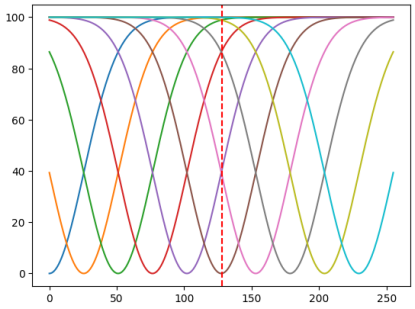

Figure 6. *representing $x = 128$ using 10 neurons*

As mentioned before, each number in interval $[0, 255]$ is represented by a unique vector of the required time of the neurons to spike.

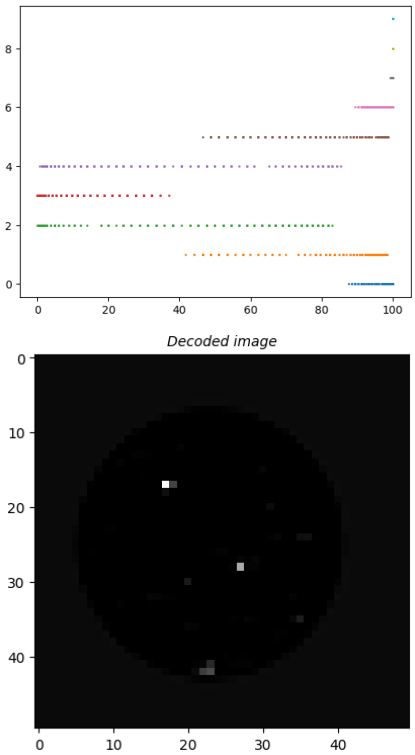

Figure 7. $T = 100, N = 10$

This is the 50×50 circle image encoded and then decoded using the gaussian neurons. The number of the neurons and

the time T are set to 10 and 100 respectively. The image is decoded using a simple summation over the neurons' activities. The quality of the decoded image is really bad due to the probabilistic nature of this method. This can be improved by **decreasing** the number of the neurons as shown in the next figure.

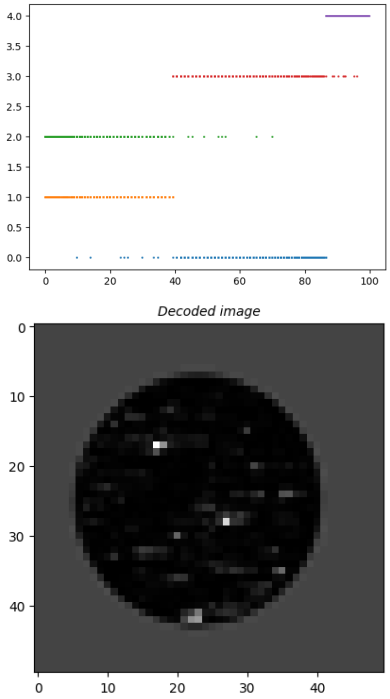

Figure 8. $T = 100, N = 5$

Same thing goes for the human image. The quality of the decoded image is really bad when the number of the neurons is high. But when I decrease N, the quality gets better.

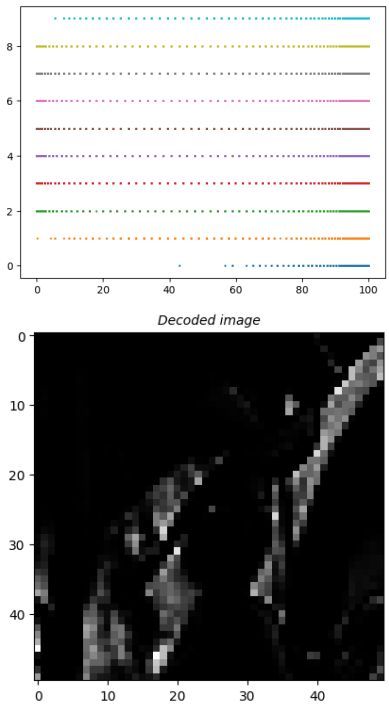

Figure 9. $T = 100, N = 10$

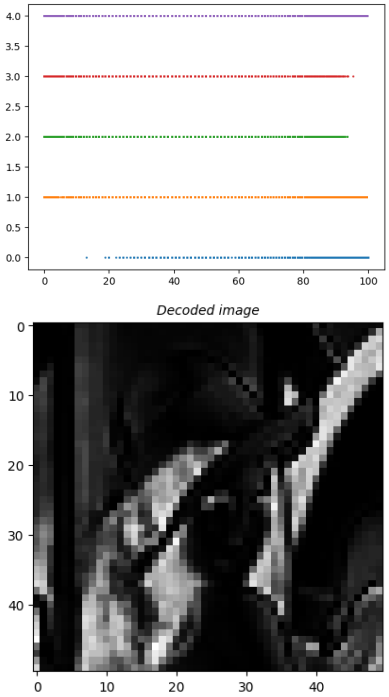

Figure 10. $T = 100, N = 5$

2.3. Poisson Encoding

In this method, the intensity of the pixel is mapped to the probability of the corresponding neuron to spike in a time step, and then translated into spikes. In this method we also have information loss like the previous one due to the probabilistic nature of the method. But if we set the parameters of the poisson distribution and the time T correctly, somehow the edges in the picture can be detected really well. So this method is suitable for encoding the images that have sharp edges. The decoding is done by a simple summation over the spikes of the neurons.

The results of the encoding and decoding of the images using the Poisson encoding method are as follows:

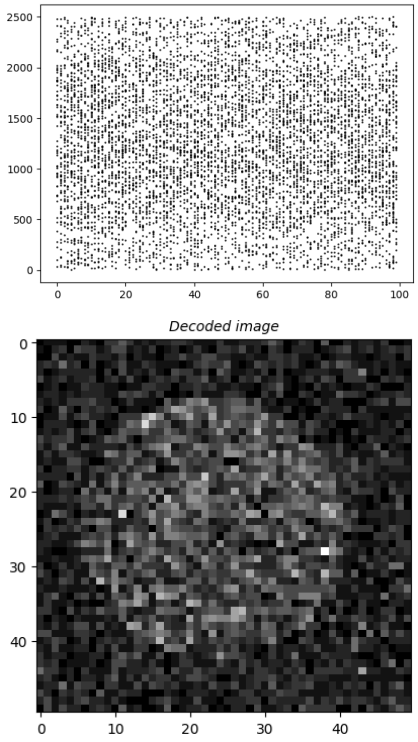

Figure 11. $T = 100, k = 2$

Here, the time T is set to 100, and the maximum number of spikes that a neuron can have in a time step is set to 2. The quality can be improved by increasing the T. I leave the k as it is in the next figure.

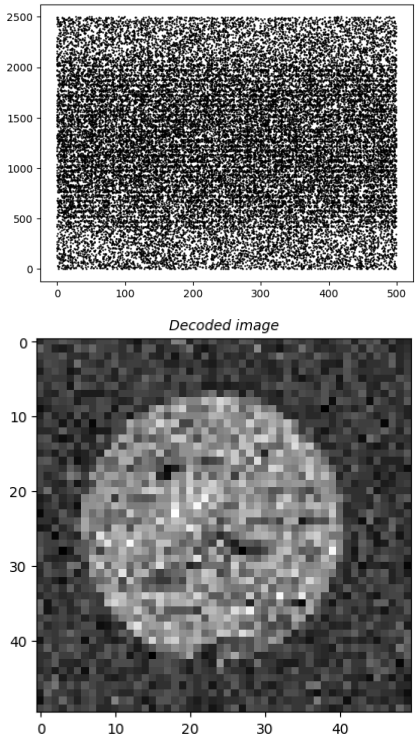

Figure 12. $T = 500, k = 2$

Presenting the input for a longer time improves the quality of encoding and the edges are perfectly distinguishable.

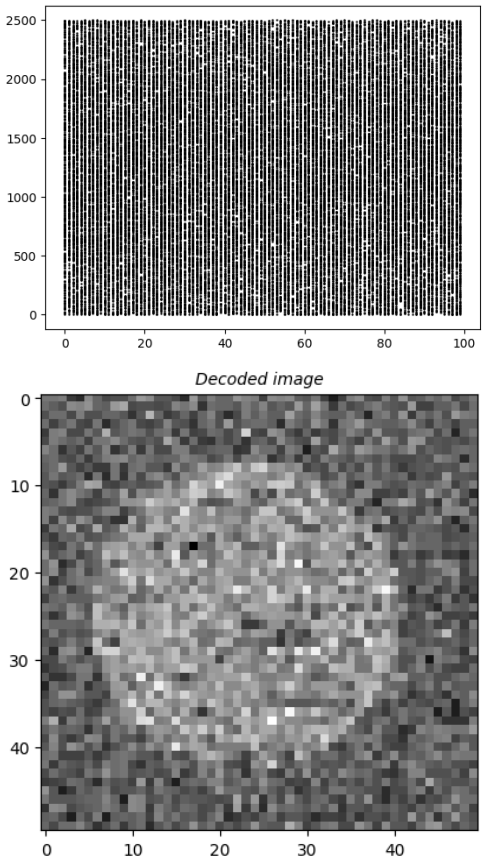

Figure 13. $T = 100, k = 1$

Decreasing the parameter k can also improve the quality of the encoding. But the edges are not as sharp as the previous

one. But increasing the k more than a certain number (Perhaps it depends on the size of the image, time T, and the image itself. But here the best k I could find was 2) decreases the quality drammatically. See the next figure where $k = 3$.

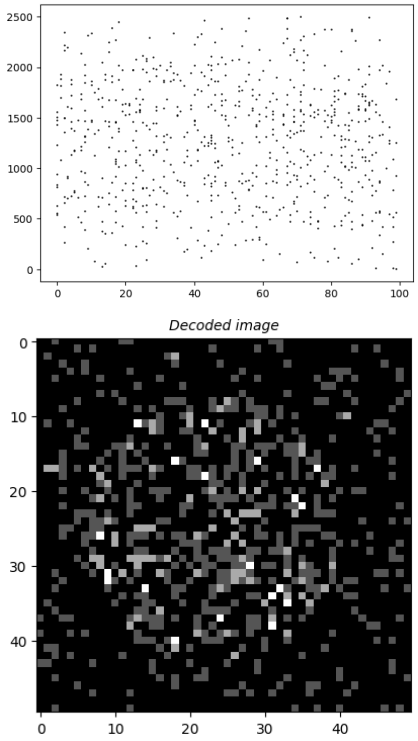

Figure 14. *steps = 100 and k = 3*

Same goes for the human image.

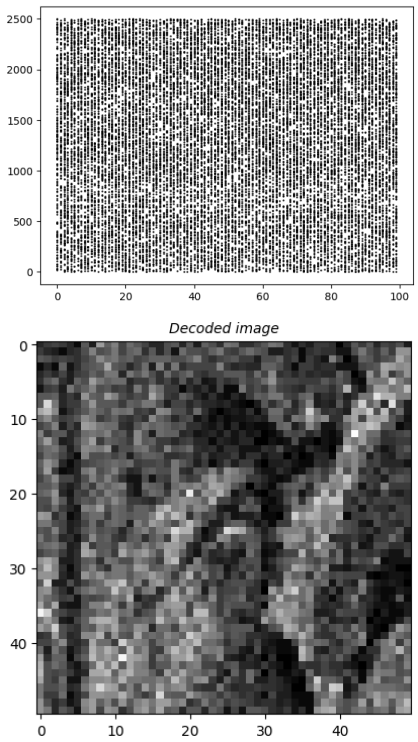

Figure 15. *steps = 100 and k = 2*

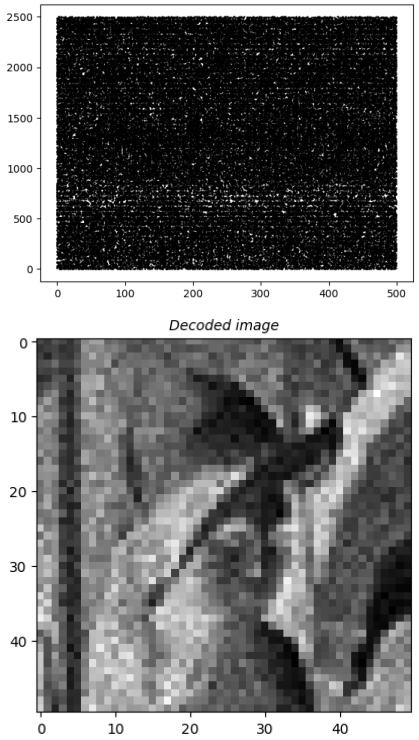

Figure 16. *steps = 500 and k = 2*

Increasing parameter k decreases the activity of the neurons as shown in the raster plots and when the neurons don't have enough activity, the quality of the decoded image decreases.

3. Learning Methods

In this section, I will analyze the performance and the behavior of the STDP and RSTDP learning methods for spiking neurons. Throughout this section, the base parameters for the LIF neurons are given the following table. We use these fixed parameters throughout the report. If needed, we will mention the changes in the parameters. There is also a small background noise in the input of the neurons which does not affect the results significantly. The randomness of some spikes will be added later in the experiments.

Parameter	Value	Description
u_{rest}	-80	resting potential
u_{reset}	-90	reset potential
u_{init}	N(-75, 1)	initial potential
$threshold$	-60	threshold potential
R	1	membrane resistance
τ	10	membrane time constant
$Time\ Resolution$	0.1	dt in euler's method
$Iterations$	1000	number of iterations

Table 1. *Base parameters for the LIF model. $N(\mu, \sigma)$ denotes a normal distribution with mean μ and standard deviation σ.*

Furthermore, the following parameters are used for the STDP and RSTDP learning methods.

Parameter	Value
A_+	2
A_-	2
τ^+_{trace}	40
τ^-_{trace}	40
W_{max}	10
W_{min}	0.1

Table 2. *Parameters for the STDP and RSTDP learning methods.*

3.1. Spike-Timing Dependent Plasticity

In this method, the weights of the synapses are updated based on the relative timing of the pre-synaptic and post-synaptic spikes. But since there is no supervision in the learning process, the weights of the synapses can be updated in a way that the network does not learn anything. This can be understood from the cosine similarity of the output neurons' activities. If the cosine similarity is close to 1, the network has not learned anything. The weights also are updated randomly and the network didn't learn the patterns.

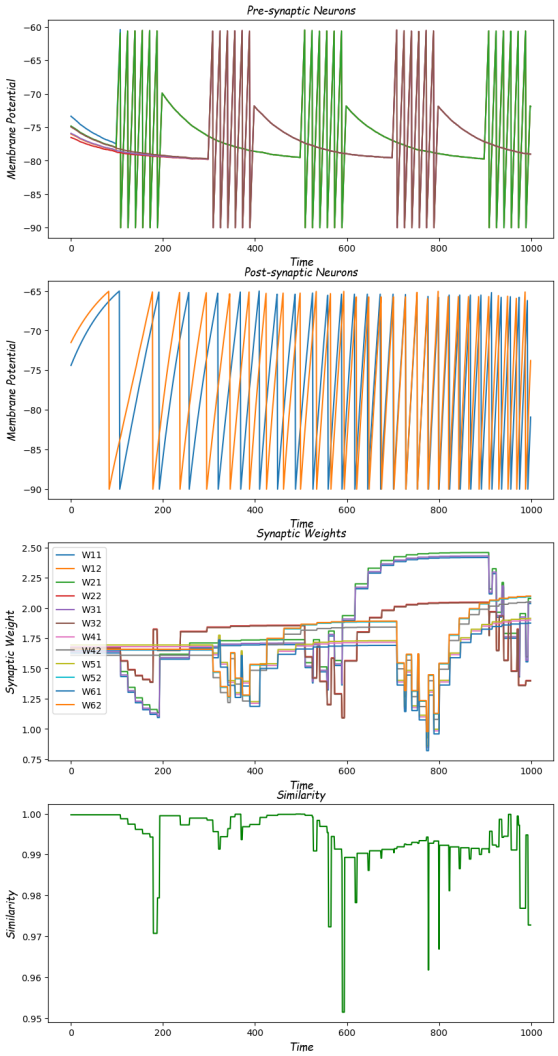

Figure 17.
$pattern_1 = [200, 200, 200, 0, 0, 0]$ *and* $pattern_2 = [0, 0, 0, 200, 200, 200]$

Decreasing the input current in the system does not change the learning results of the network and obviously by the previous figure, it does not lter the results either by increasing the input current. Same thing goes when the patterns do intersect. The network does not learn anything as shown in the figure 19 and 20.

Even adding noise and random spikes on both input and output layers (Figures 21 and 22 with the mentioned probability in the caption) or adding a dummy neuron to both layers (Figure 23) does not change the results.

This method does not capture the reality of the learning process in the brain. This is why we need a reward-modulated learning method to make the network learn the patterns.

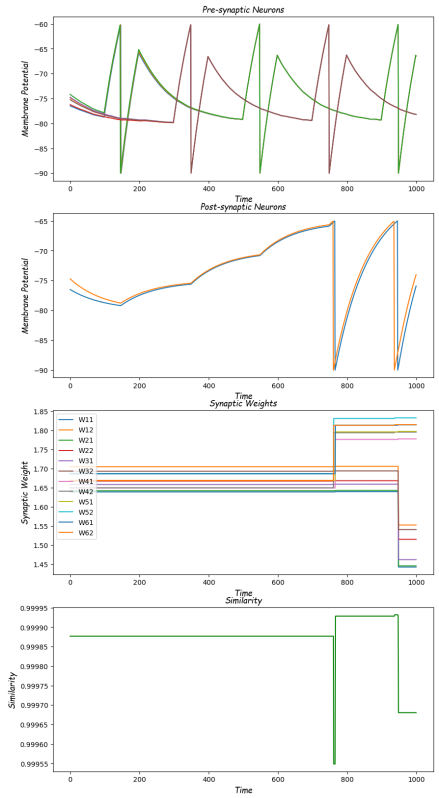

Figure 18. $pattern_1 = [50, 50, 50, 0, 0, 0]$ *and* $pattern_2 = [0, 0, 0, 50, 50, 50]$

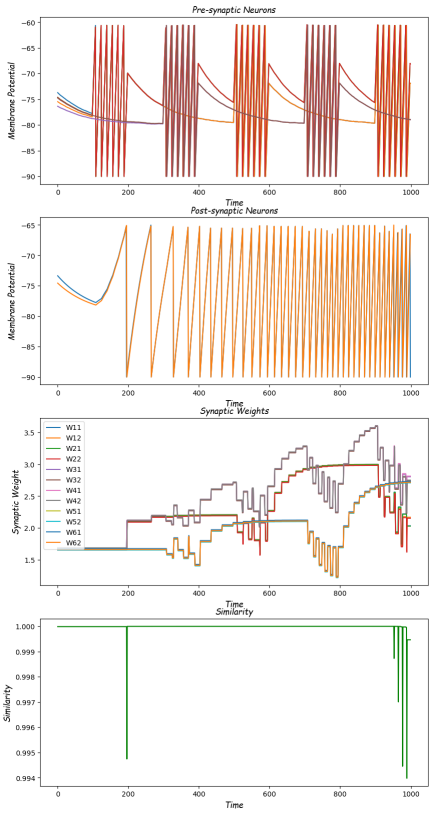

Figure 20.
$pattern_1 = [200, 200, 200, 200, 0, 0]$ *and* $pattern_2 = [0, 0, 200, 200, 200, 200]$

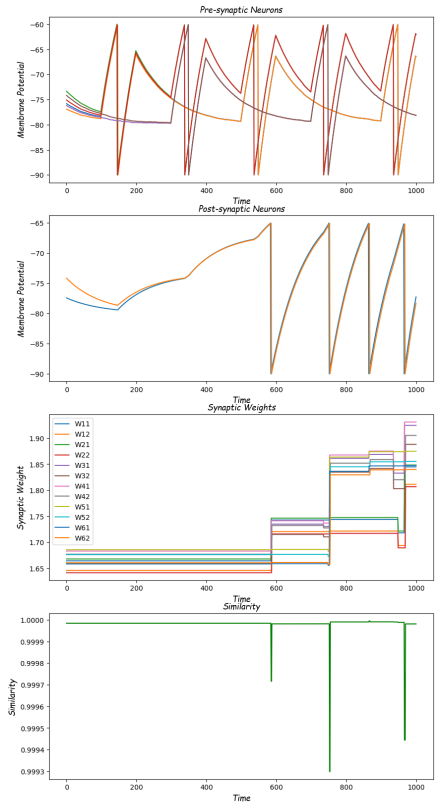

Figure 19. $pattern_1 = [50, 50, 50, 50, 0, 0]$ *and* $pattern_2[0, 0, 50, 50, 50, 50]$.

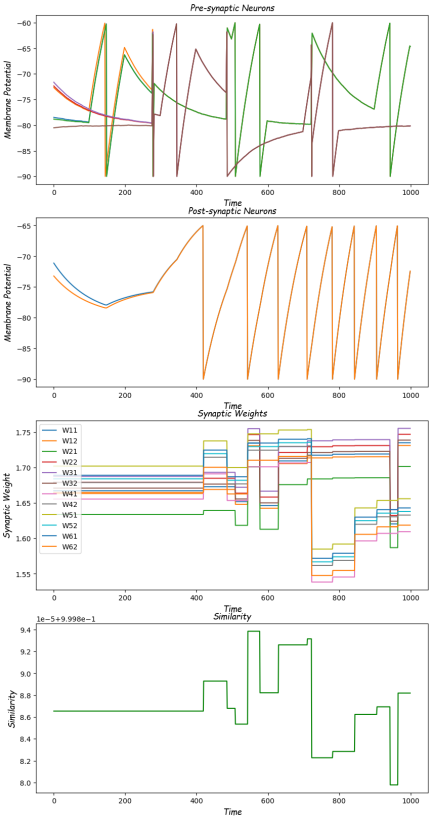

Figure 21. $pattern_1 = [50, 50, 50, 0, 0, 0]$ *and* $pattern_2 = [0, 0, 0, 50, 50, 50]$.
The probability of random activity in the input and output layers is 0.1.

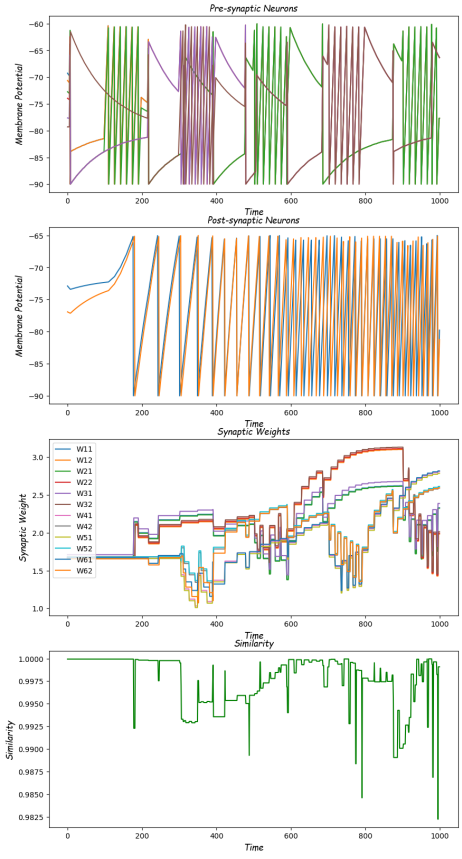

Figure 22.
$pattern_1 = [200, 200, 200, 0, 0, 0]$ *and* $pattern_2 = [0, 0, 0, 200, 200, 200]$.
The probability of random activity in the input and output layers is 0.1.

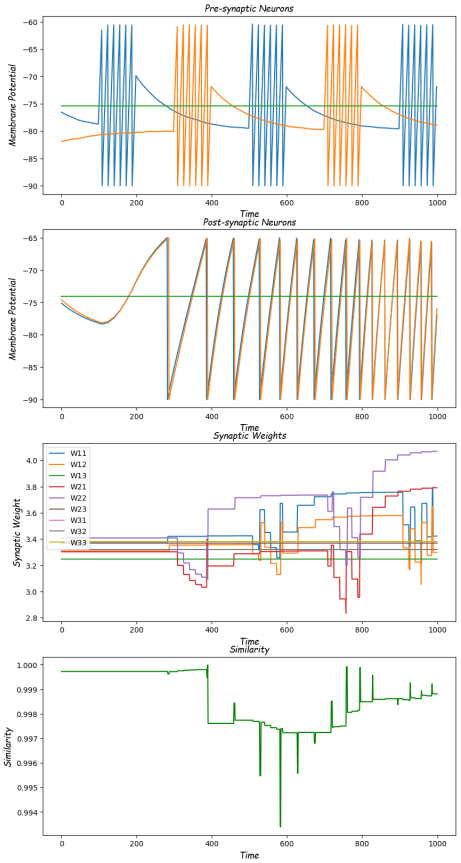

Figure 23. $pattern_1 = [200, 0]$ *and* $pattern_2 = [0, 200]$. *A dummy neuron is added to both layers that does not spike at all.*

3.2. Reward-Modulated Spike-Timing Dependent Plasticity

Continuing from the previous section, adding a reward-modulated learning method to the network can make the network start learning the patterns we want.

Here are the parameters given to the dopamine model in the RSTDP method.

The following figure shows that the network has learned the patterns correctly. The weights of the synapses are updated in a way that the network can distinguish the patterns from each other. The cosine similarity of the output neurons' activities has dropped significantly.

Parameter	Value
$init_d$	0
τ_d	1
$coef$	50

Table 3. *Parameters for the dopamine.*

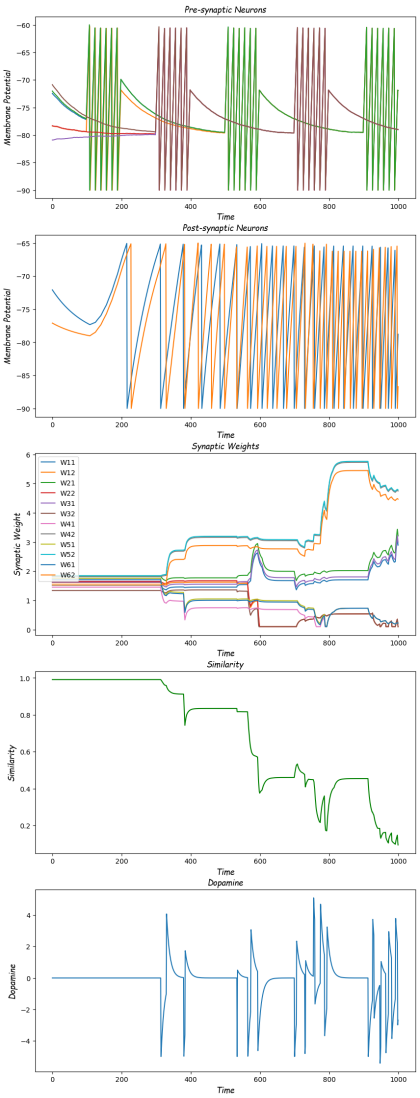

Figure 24.
$pattern_1 = [200, 200, 200, 0, 0, 0]$ *and* $pattern_2 = [0, 0, 0, 200, 200, 200]$

In the next figure, where I added intersection between two patterns, the network distinguishes between the input neurons that are not the intersection part. Because one of the output neurons tries to encourage one the neurons in the intersection part and the other one decreases the dopamine level of the network. Same thing happend for the other input neuron in the intersection.

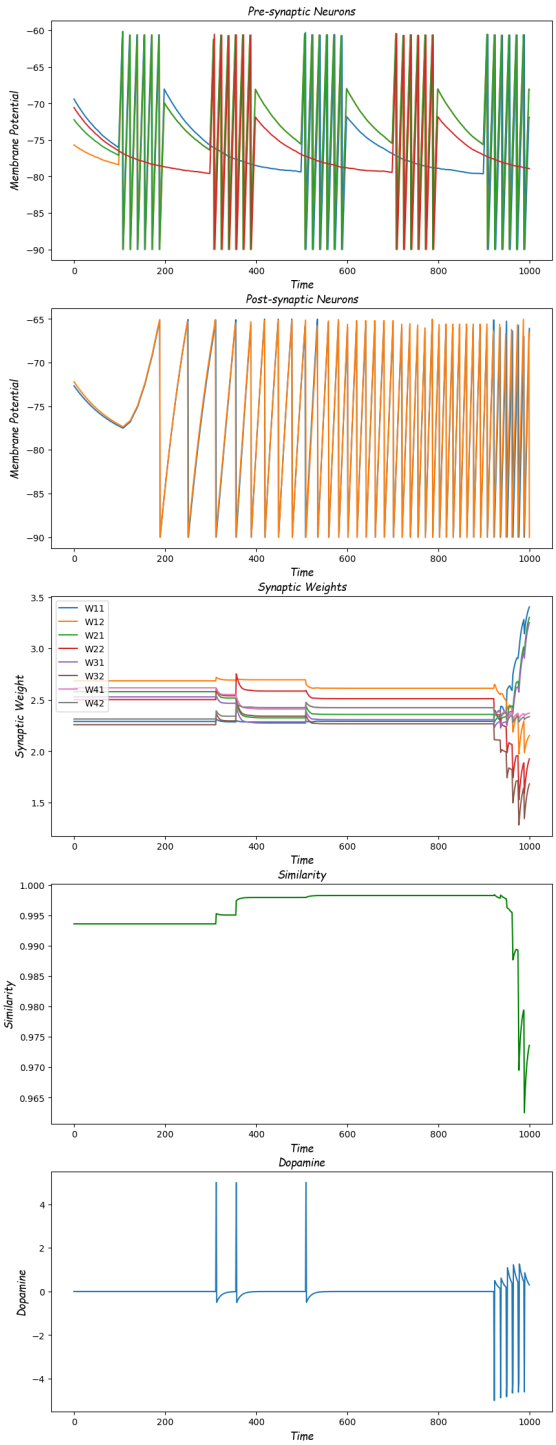

Figure 25. $pattern_1 = [200, 200, 200, 0]$ *and* $pattern_2 = [0, 200, 200, 200]$

In the next figure, I added random activity and spikes to both input and output layers. With the probability being high, the network does not bahave as expected to learn the patterns. But when the probability is low, the network learns the patterns correctly, but not as good as when there is no random activity.

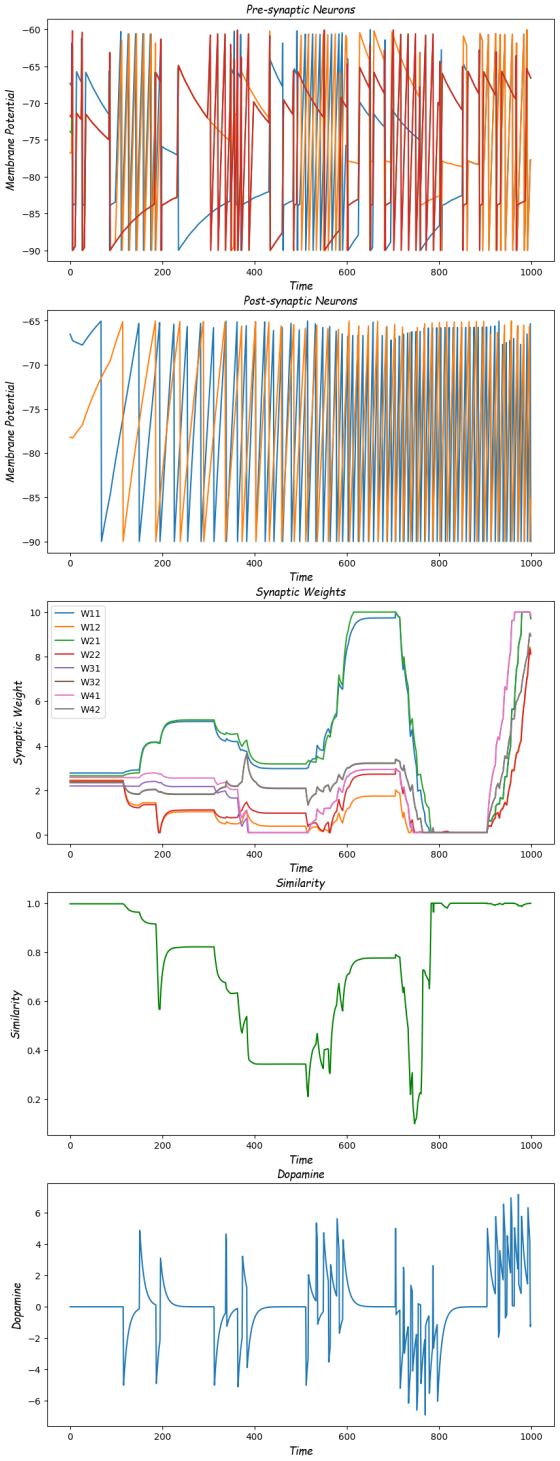

Figure 26. $pattern_1 = [200, 200, 0, 0]$ *and* $pattern_2 = [0, 0, 200, 200]$. *The probability of random activity in the input and output layers is 0.4.*

By adding a dummy neuron to both layers, the network learns the patterns slowly as it can be seen from the figure. The blue curve should increase since it is the neuron responsible for the first pattern. Also, the synaptic weights of the dummy neurons are not changed due the inactivity of the neurons.

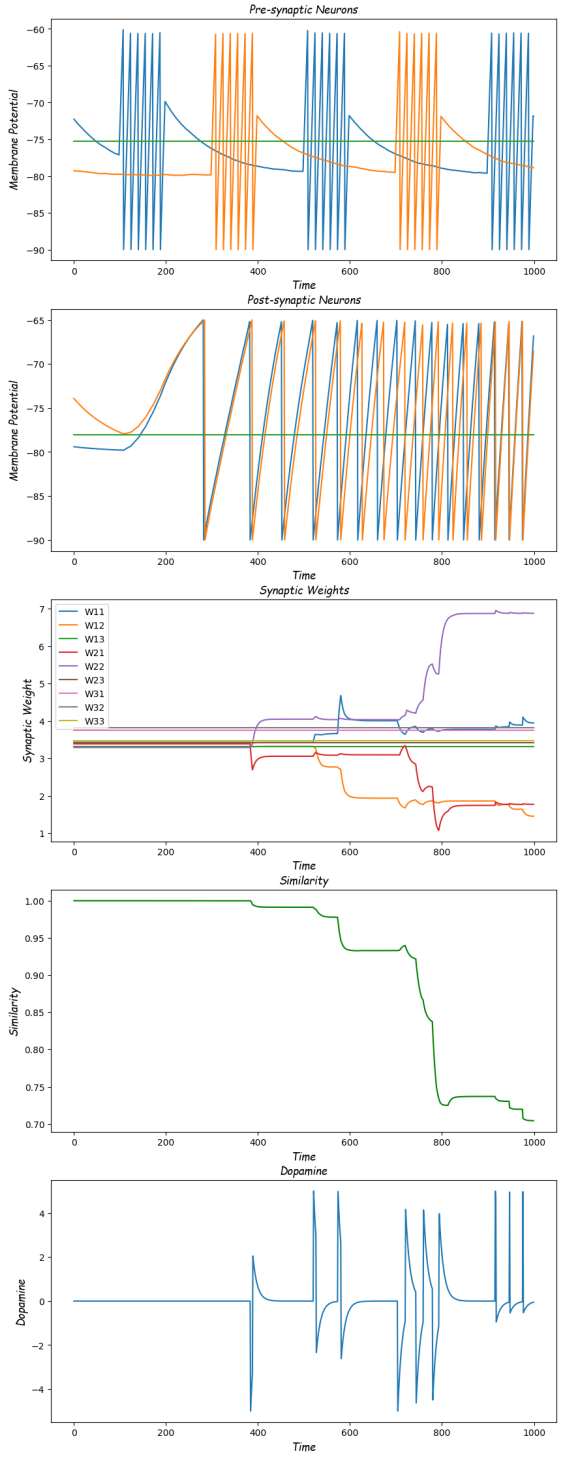

Figure 27. *patterns = [200, 0] and [0, 200], added a dummy neuron to both layer*