P03 report file

Darya Ansaripour

April 24, 2024

Abstract

This report consists of an overview on the third project.

1 INTRODUCTION

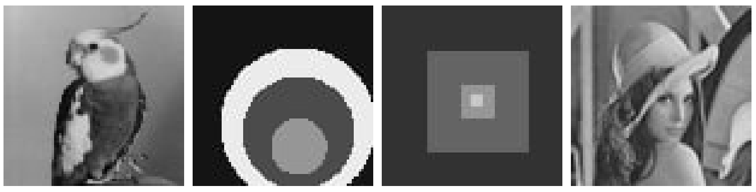
As requested, three encodings have been implemented. STDP and RSTDP behaviors have been implemented and tested on a two-layer network.

2 ANALYSIS AND BEHAVIORS

2.1 Encodings

4 images were selected and for each encoding, their encoded counterpart is shown.

Figure 1: Images used for encodings.



2.1.1 Time to First Spike

Results are presented in Figure 1 and 2. This encoding is implemented by mapping each pixel to a neuron and mapping each neuron's first spike time to it's corresponding pixel value. It is observable that by increasing the number of iterations, the encoding is more detailed.
Neurons raster plot is also available in Figure 2. Density of spikes at a specific time shows that there is a lot of pixels corresponding to mapping to that time (In other words, there is a lot of plot with the mapped color.), for instance there are 3 main lines in the 3rd raster plot, representing three main colors of the picture. In the fourth image, the pixels exhibit a wide range of colors, resulting in numerous neurons spiking at each time step, as evident from the raster plot. However, the third and second raster plots display only a few time steps with a high spike count. This discrepancy arises because the corresponding images for these plots contain only a limited set of distinct colors.
When comparing results obtained from different training durations, we observe that increasing the training time leads to more detailed encodings. This increased level of detail becomes evident in the image representation after encoding.
Obviously after a certian amount of iterations the quality of encoding won't change due to the specific mapping that have been used: (It is assumed that images are flattened.)

$$ttfs_i = \frac{iterations \times pixel_i}{255} \tag{1}$$

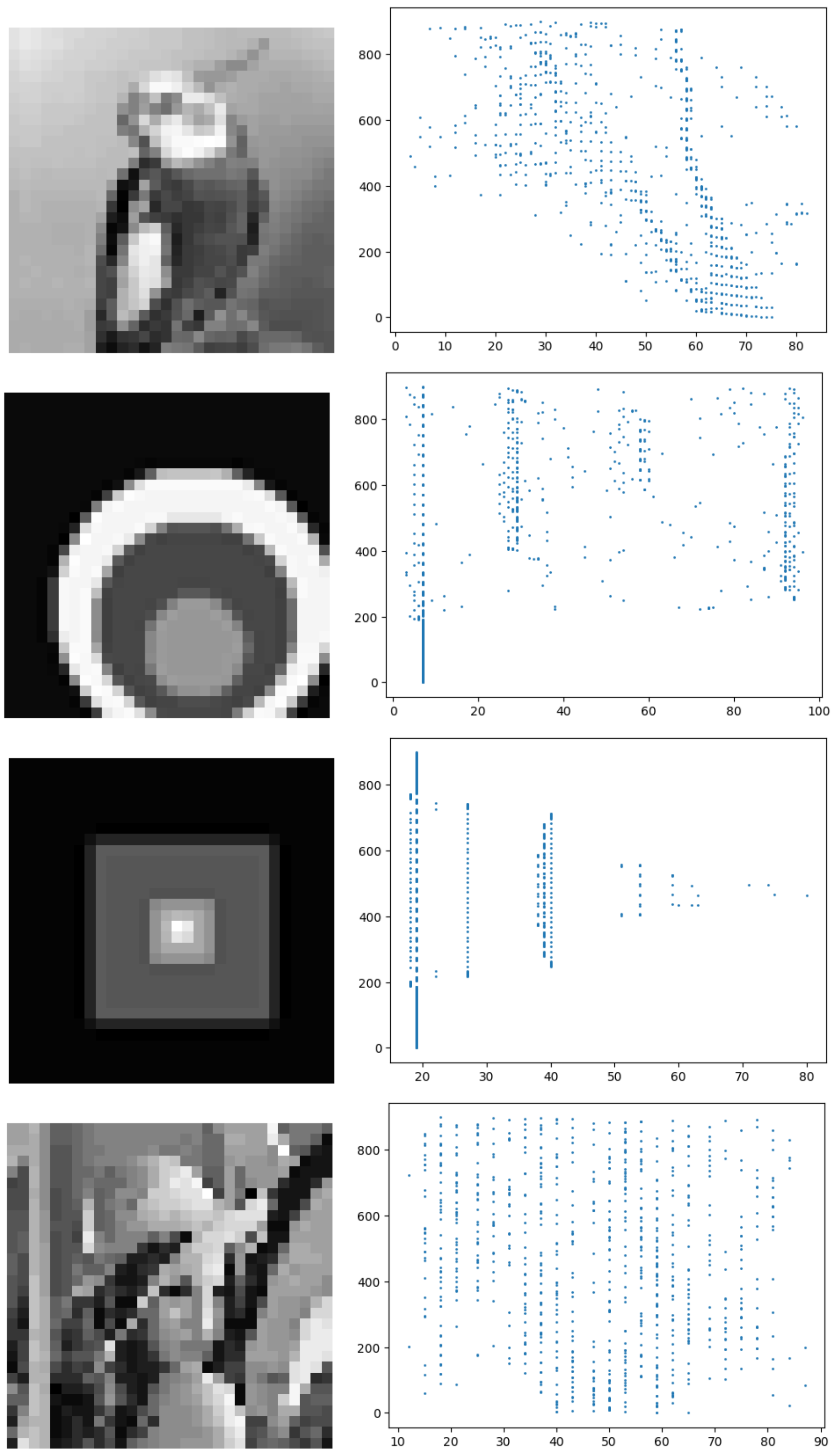
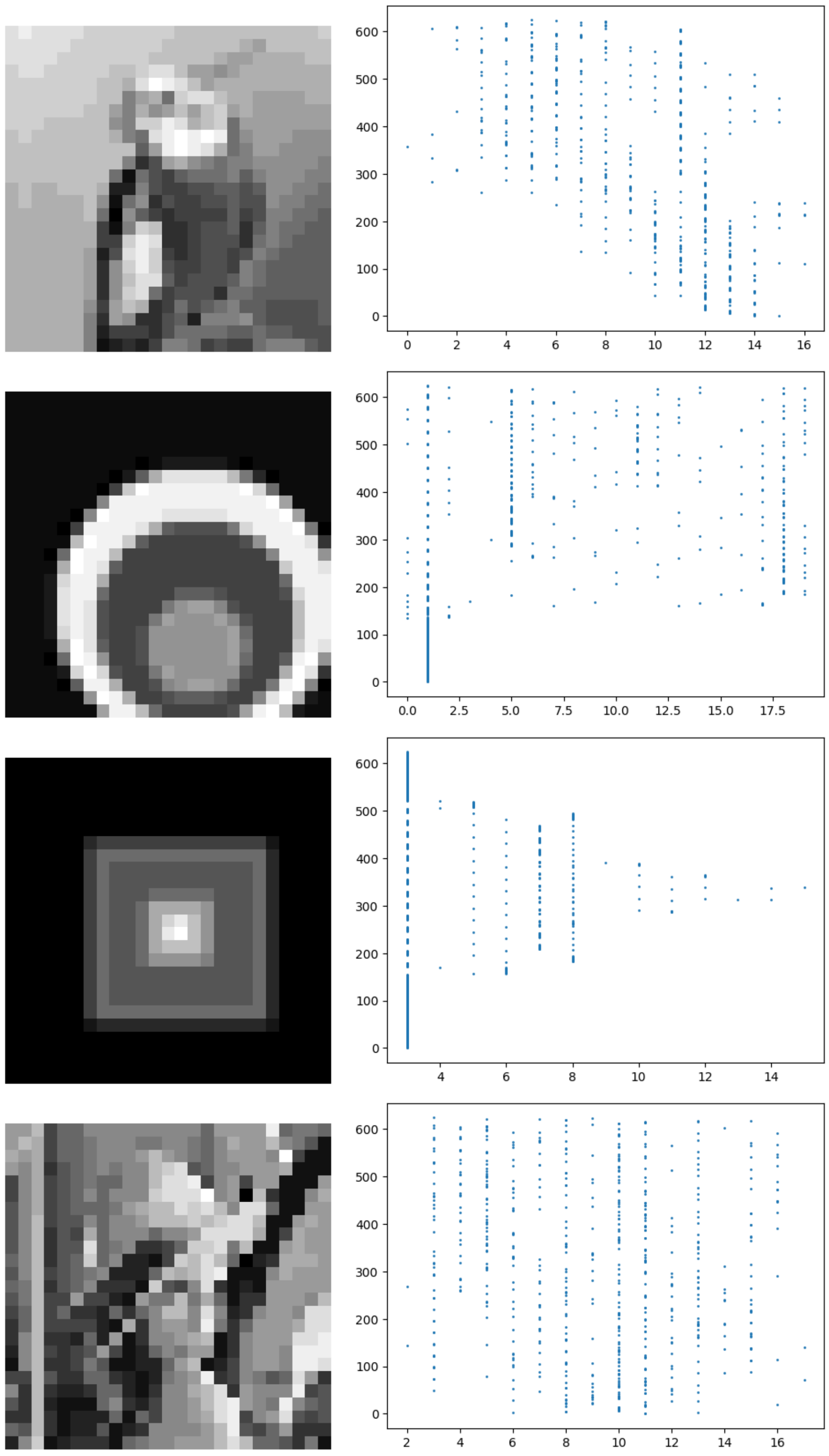


Figure 2: Time to First Spike encoding results for total iterations of 100dt

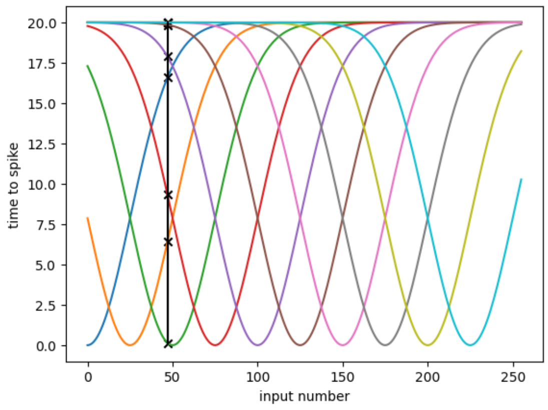
Figure 3: Time to First Spike encoding results for total iterations of 20dt



2.1.2 Number Representing

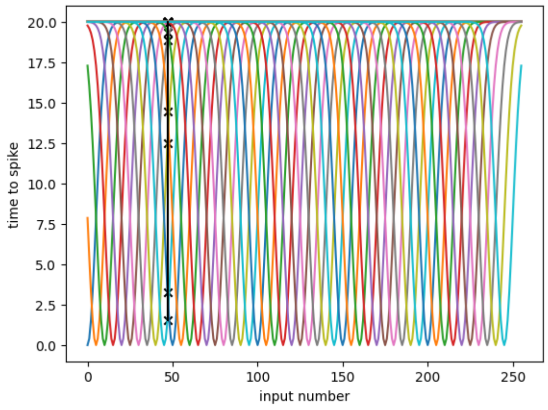
As previously discussed in lectures, this encoding is a method for representing numbers, Given a number one can make N neurons and map the number to a vector consisting of each neurons spike time. So input would be a number (or an image) and It will be coded into a vector (or a vector per pixel) which represents neurons spike times.
It is observable that increasing the number of neurons, results in a better estimation of input number (since there would be more neurons mapped to numbers.) Plots are drawn vice versa because when a neuron is stimulated by its best stimulus, It spikes sooner. In figure 4 it is observable that the 3rd neuron which its best stimulus is 50 has the sooner spike time and after that we have second neuron which is corresponded to 25. Decoded images have been created by integrating each neuron spikes during time and ploting it in a 25*25 shape.

Figure 4: 10 neurons tuning curves and input number 47



neurons spike times: [16.5837, 6.4208, 0.1434, 9.3182, 17.8861, 19.8461, 19.9958, 19.9999, 19.9999, 19.9999]

Figure 5: 50 neurons tuning curves and input number 47



neurons spike times: [20.0, 19.9999, 19.9999, 19.9999, 19.9999, 19.9987, 19.9382, 18.8773, 12.4937, 1.5376, 3.2945, 14.4395, 19.3190, 19.9693, 19.9994, 19.9999, 19.9999, 19.9999, 20.0, ..., 20.0]

Figure 6: Number representation encoding neurons raster plot for images. number of neurons=10, Time=100.

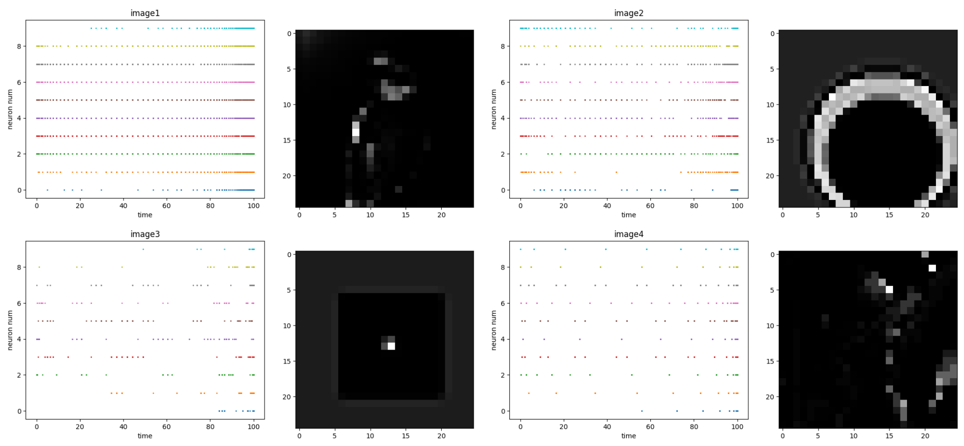
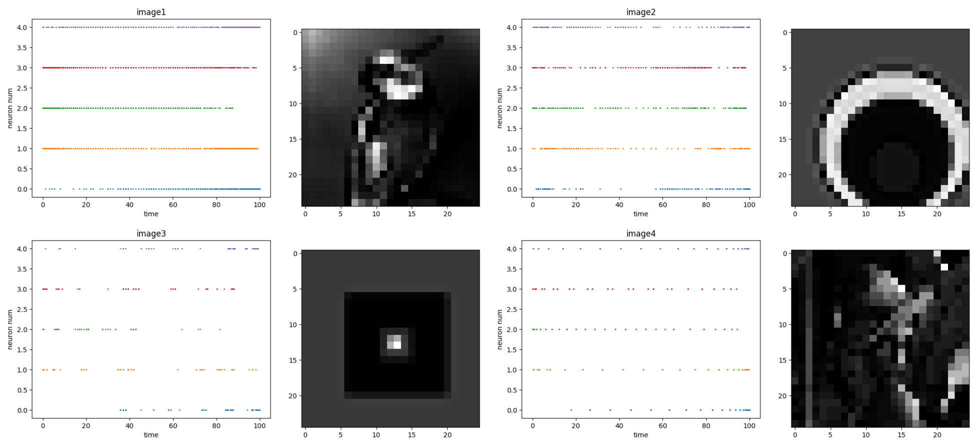


Figure 7: Number representation encoding neurons raster plot for images. number of neurons=5, Time=100.



Increasing the number of neurons does not necessarily improve the results, that is because of the way I decode the results, when number of neurons increase, by integrating the encoded results for each pixel, the number grows and so the pixel become darker, this leads to a poor color detection.
I have thought about other ways to decode, I got nothing but only an idea to map the first neuron activity time to the pixel, but it would be similar to time to first spike at some levels, which is why I omitted.

2.1.3 Poisson Encoding

I mapped a parameter λ to each neuron, which is the parameter for its poisson distribution. The pararmeter is relative to the value of its corresponding pixel, The brighter the color the higher probability of spike! Decoded images have been created by integrating each neuron spikes during time and ploting it in a 25*25 shape.

$$P(Spike_i(dt) = k) = \frac{e^{-\lambda dt}(\lambda dt)^k}{k!} \tag{2}$$

Which means that the probability of getting k spikes during a period of time dt follows above equation. In the raster plots, we notice distinct patterns:
Image 2: The initial rows in picture appear dark, indicating that neurons corresponding to those pixels exhibit minimal spike activity.
Image 3: The central portion of picture is bright, suggesting that neurons in the center exhibit higher activity compared to those at the periphery.
These observations provide valuable insights into neural behavior and activity patterns.

Figure 8: poisson encoding raster plots and their corresponding decodings. k=2, time=100.

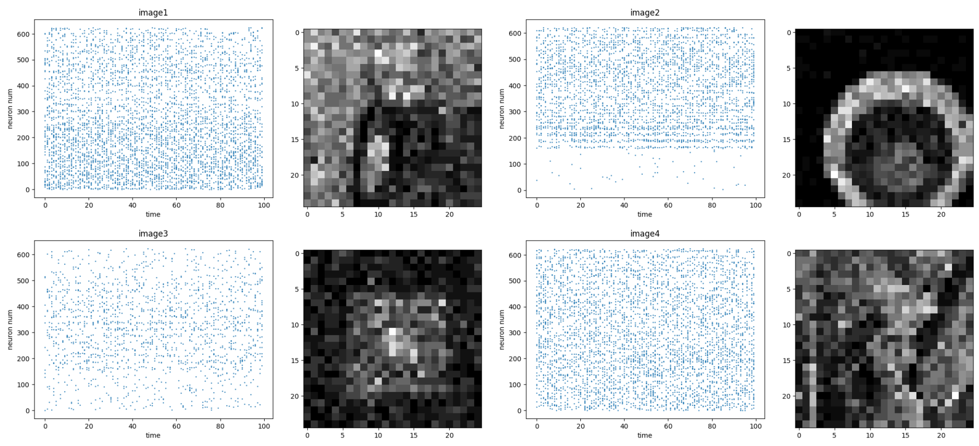
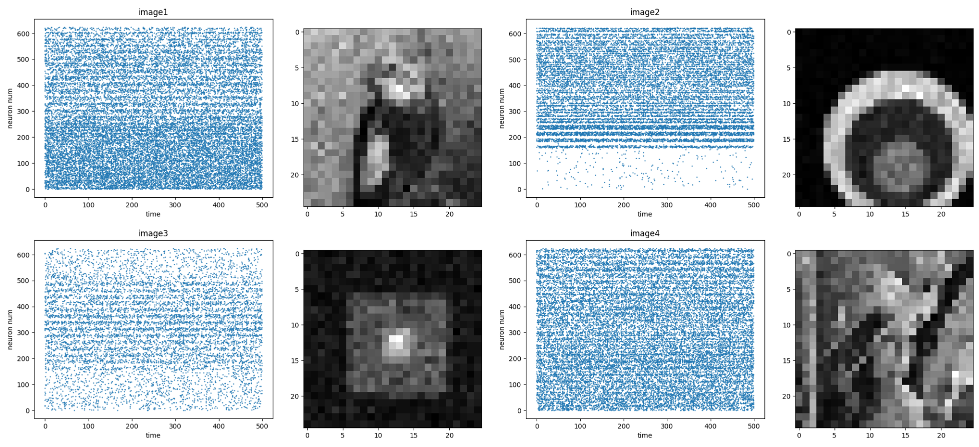


Figure 9: poisson encoding raster plots and their corresponding decodings. k=2, time=500.



2.2 STDP and RSTDP

From now on, we assume that the input and output layers have parameters as Table1 and their connection scheme is full connectivity.
Patterns are a vector of input currents, they are declared in such a way that only specific neurons become active. We map the left half of neurons to the left output neuron, and similarly, the right half of input neurons to the right output neuron. This approach helps create specialized connections and facilitates targeted neural responses.
Consider the patterns as shown in the table below. We simultaneously apply pattern 1 and pattern 2. Additionally, we introduce a third pattern consisting entirely of zeros between the application of the first and second patterns. This approach enhances our understanding of neural behaviors. At first, patterns has no neurons in common but eventually we increase their intersection.

Table 1: Neuron groups parameters

Model	**Parameters**				
	thresh	**R**	**u rest**	**u reset**	**tau**
LIF (input)	-37	5	-67	-75	10
LIF (output)	-65	5	-67	-75	10

Table 2: Synapse parameters

Scheme	**Parameters**	
	J_0	N
Full	2	6×2

Table 3: Learning parameters

Model	**Parameters**					
	τ_P	τ_M	W_{Max}	γ	β	τ_C
STDP	4	3	4	2	2	-
RSTDP	4	3	4	2	2	1

Table 4: Patterns

Figure	**Patterns**		
	$Pat1$	$Pat2$	$Pat3$
Fig 8,11	[50,50,50,0,0,0]	[0,0,0,50,50,50]	[0,0,0,0,0,0]
Fig 9	[50,50,50,50,0,0]	[0,0,50,50,50,50]	[0,0,0,0,0,0]
Fig 10	[50,50,50,50,50,0]	[0,50,50,50,50,50]	[0,0,0,0,0,0]

2.2.1 STDP

2.2.1.1 Random Applying

STDP approach does not have an observable learning, weights do not converge to a specific pattern, since there is no superviser. We can observe that every time a presynaptic neuron fires before the postsynaptic one, the corresponding weight would increase and it decreases when the postsynaptic neuron fires before presynaptic. Cosine similarity has a random behavior too, since model does not distinguish between two output neurons.

Figure 10: STDP behavior (patterns have no intersection)

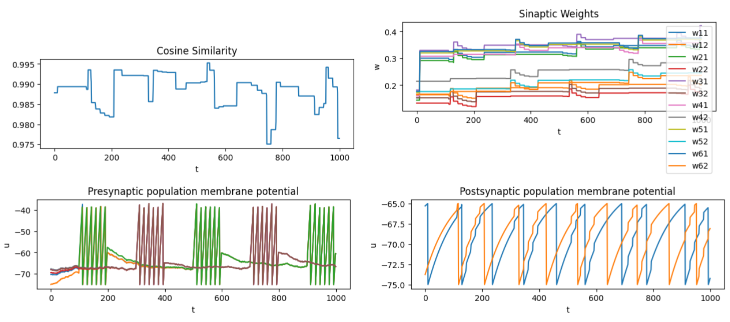


Figure 11: STDP behavior (patterns have more intersection)

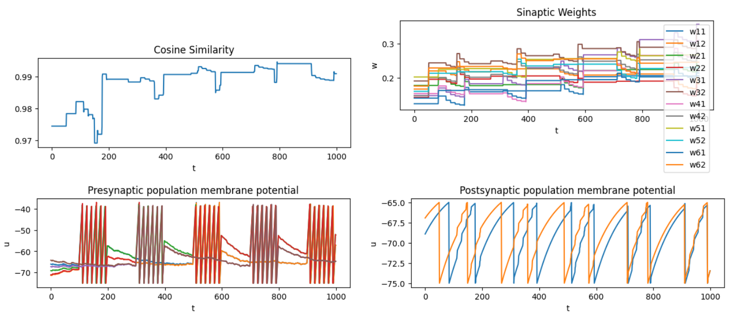
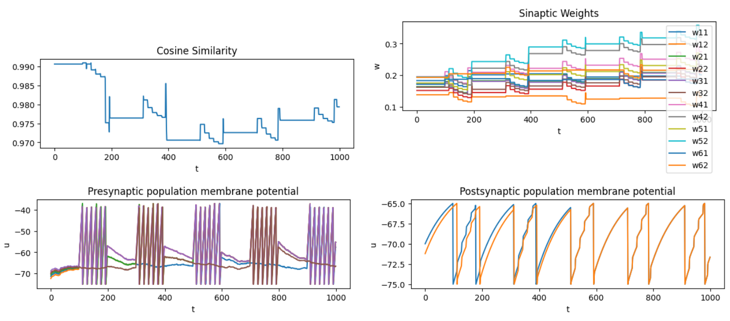


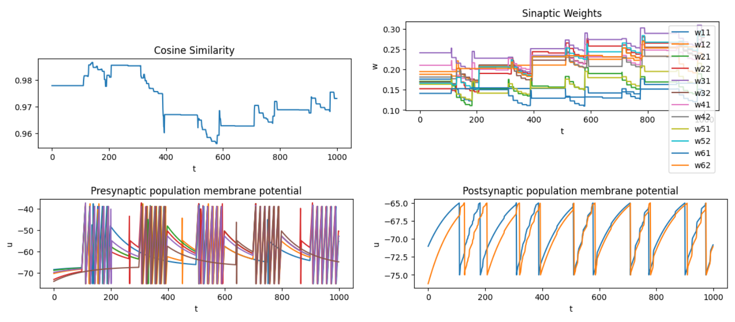
Figure 12: STDP behavior (patterns are same except for 2 neurons)



2.2.1.2 Random Applying With Background Activity

In this part I managed some random spikes of each neuron independent of input pattern during the simulation to see if it affects the learning procedure. Since STDP does not learn the correspondence of a target neuron and its pattern, It'll suffice providing results for only one pair of patterns.

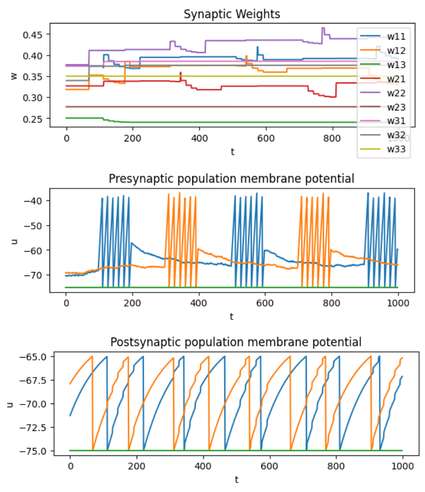
Figure 13: STDP behavior (patterns have no intersection)



2.2.1.3 Adding Dummy Neurons

For a better understanding of what's going on, I decremented the number of input neurons to 2 in addition of a dummy neuron with no spike. By observing each synapse weight we get differnet results. Synapse connecting two dummy neurons does not have any change in its weight because traces do not change when there is no spike in pre and postsynaptic neurons. Synapse connecting dummy input neuron too first and second target neurons (weights shown by w_{31}, w_{32} in plots.) would have an increase in its weight because its postsynaptic neuron is activated after the presynaptic one, but this would eventually vanish due to the increasing time difference of activities. In contrary synapse connecting dummy output neuron too first and second input neurons (weights shown by w_{13}, w_{23} in plots.) would have a decrease in its weight because its presynaptic neuron is activated after the postsynaptic one, but this also would eventually vanish due to the increasing time difference of activities. Other weights act normal. The reader can check STDPs behavior easily. Considering plot below, blue graphs refer to first pre/postsynaptic neurons and oranges are related to the second ones.

Figure 14: STDP behavior (patterns have no intersection, adding dummy neurons)



2.2.2 RSTDP

Note that reward function is declared in such a way that network is awarded when first (last) 3 input neurons and first (second) output neuron have spike, and it is punished when first (last) 3 input neurons and second (first) output neuron have spike, and if both output neurons fire, no reward or punishment is involved.

2.2.2.1 Random Applying

There are a lot of interesting results here! Consider the first pair of patterns with no neurons in common, It is expected that model matches the patterns with target neurons due to the supervise of reward function, as we can see, in the weight graph, $\{w_{ji} | i = 1 \; and \; j = 1,2,3 \; or \; i = 2 \; and \; j = 4,5,6\}$ increase and $\{w_{ji} | i = 1 \; and \; j = 4,5,6 \; or \; i = 2 \; and \; j = 1,2,3\}$ decrease. Since increasing synapse weights are related to different input neurons for each output neuron, the vectors of weights become very different during the time (for one, first 3 increase and for other last 3 elements increase.) so the cosine similarity will decrease as expected. After learning those patterns each pattern activation leads to more activity of it's target neuron.
Now consider the second pair of patterns with two neurons in common, we expect that synapse weights related to uncommon neurons change like the first case and synapse weights related to common ones don't change because of reward and punishments that occur simutanously due to the activation of neuron in both patterns. As we can see in the graph, $\{w_{ji} | i = 1,2 \; and \; j = 3,4\}$ do increase or decrease strictly. The cosine similarity decrease since there are four elements of weight vectors that differ from each other during time. Since common neurons weights do not increase or decrease, the effect of input pattern on activation of output neurons decreases.
In the end consider the pair of patterns with most intersection, having a lot of neurons in common results in fewer learning, because of the activation of each output neuron's related neurons in both patterns, the network cannot learn whether the first pattern is activated or the second one, thats because of weights that do not converge. As we can see in the cosine similarity plot, weight vectors are almost the same so pattern activations do not affect a target neuron specifically.

Figure 15: RSTDP behavior (patterns have no intersection)

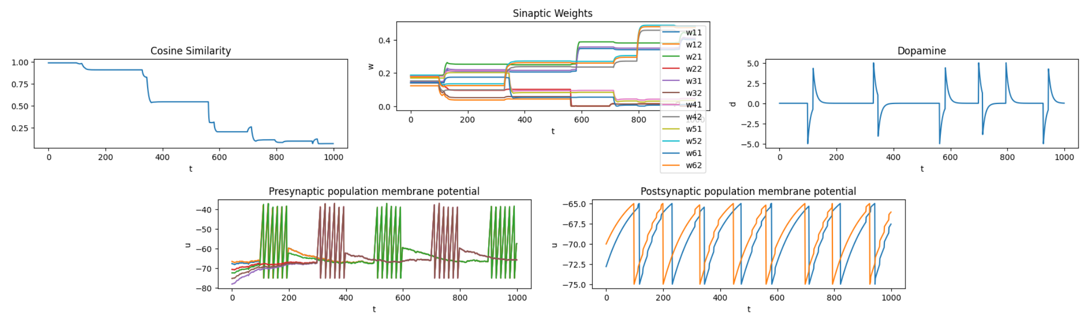


Figure 16: RSTDP behavior (patterns have more intersection)

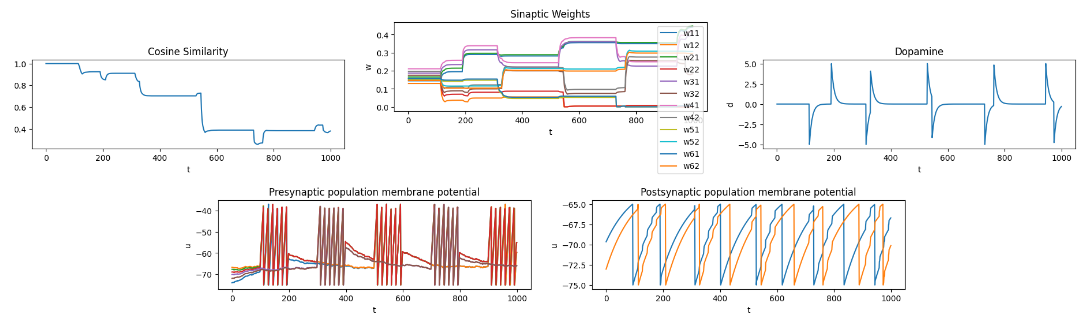
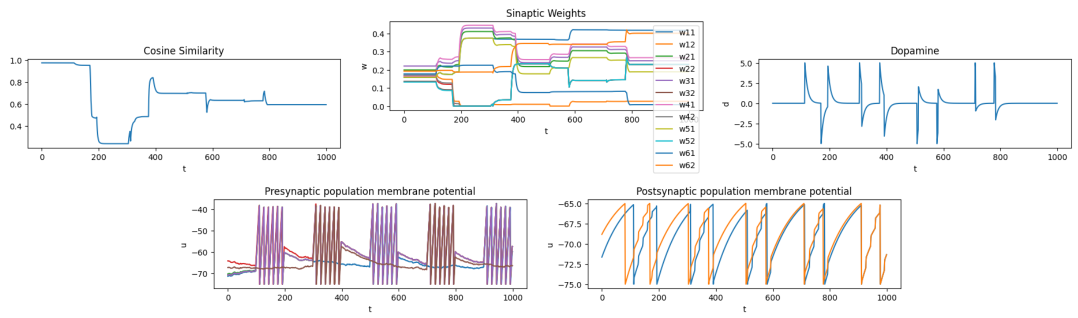


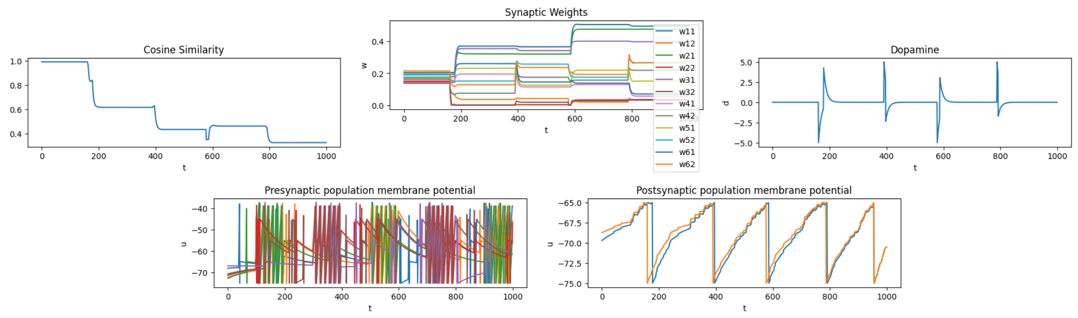
Figure 17: RSTDP behavior (patterns are same except for 2 neurons)



2.2.2.2 Random Applying With Background Activity

In this part I managed some random spikes of each neuron independent of input pattern during the simulation to see if it affects the learning procedure. Since I want to discuss learning procedure It'll suffice providing results for only one pair of patterns.
As it is observable, having random spikes during the simulation, cause an unexpected increase or decrease of dopamine, for example assume that pattern 1 is activatec and mean while neuron 4 has its own random spike due to the background activity we defined, this results in a punishment since pattern 1 was activated but activation of neuron 4 may result in activation of second target neuron and this leads to a punishment, the other case is also probable. So it's reasonable to find some weights with not much change.

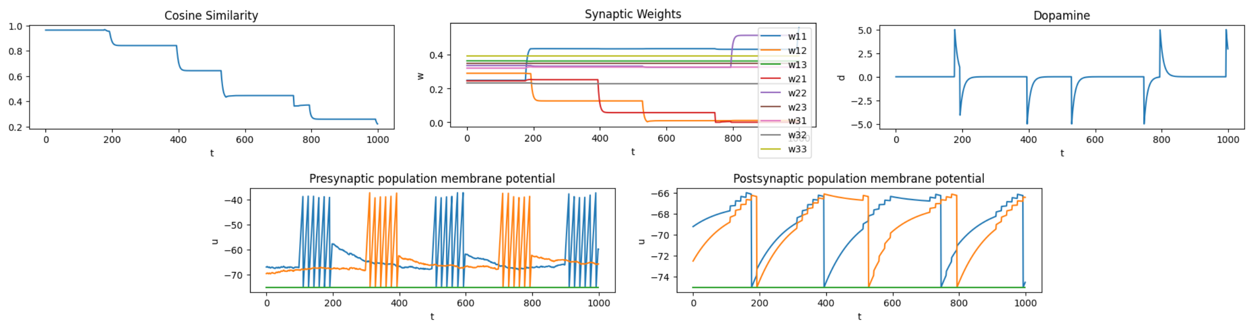
Figure 18: RSTDP behavior (patterns have no intersection)



2.2.2.3 Adding Dummy Neurons

For a better understanding of what's going on, I decremented the number of input neurons to 2 in addition of a dummy neuron with no spike. By observing each synapse weight we get differnet results. Synapse connecting two dummy neurons does not have any change in its weight because traces do not change when there is no spike in pre and postsynaptic neurons. Synapse connecting dummy input neuron too first and second target neurons (weights shown by w_{31}, w_{32} in plots.) would have an increase in its weight because its postsynaptic neuron is activated after the presynaptic one, but this would eventually vanish due to the increasing time difference of activities. In contrary synapse connecting dummy output neuron too first and second input neurons (weights shown by w_{13}, w_{23} in plots.) would have a decrease in its weight because its presynaptic neuron is activated after the postsynaptic one, but this also would eventually vanish due to the increasing time difference of activities. Other weights act normal. The reader can check RSTDPs behavior easily. Considering plot below, blue graphs refer to first pre/postsynaptic neurons and oranges are related to the second ones.

Figure 19: RSTDP behavior (patterns have no intersection, adding dummy neurons)


Cosine Similarity
Synaptic Weights
Dopamine
Presynaptic population membrane potential
Postsynaptic population membrane potential
w11
w12
w13
w21
w22
w23
w31
w32
w33
t
w
d
u