NEURAL CODING SCHEMES AND LEARNING RULES

Bita Moharrami
University of Tehran

Supervised by: Mohammad Ganjtabesh

May 2024

Contents

Chapter 1

Neural coding schemes

1.1 Time-To-First-Spike coding

In this section, we are going to study Time-To-First-Spike (TTFS) encoding method. In this method each neuron spikes at most once. In order to encode our data using TTFS method, we must convert input pixels to the first-spike patterns.

Now, we are going to give several photos as inputs and analyse the raster plot of encoded pixels.

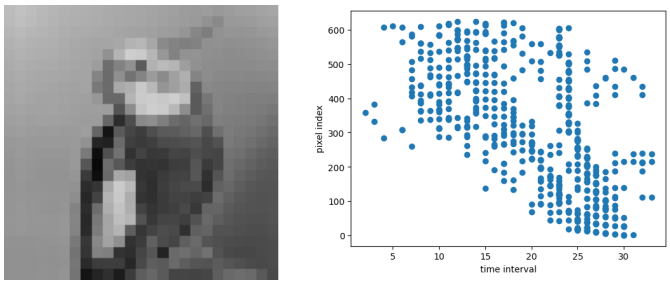

Figure 1.1: Time to first spike encoding example on 40 intervals

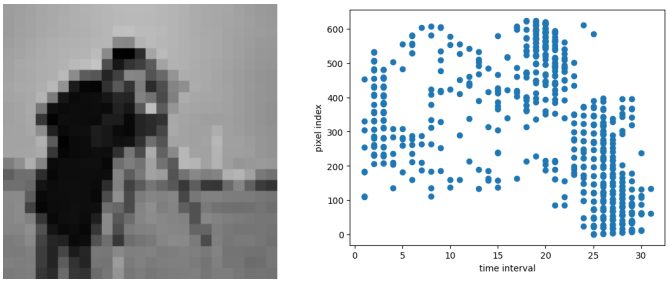

Figure 1.2: Time to first spike encoding example on 40 intervals

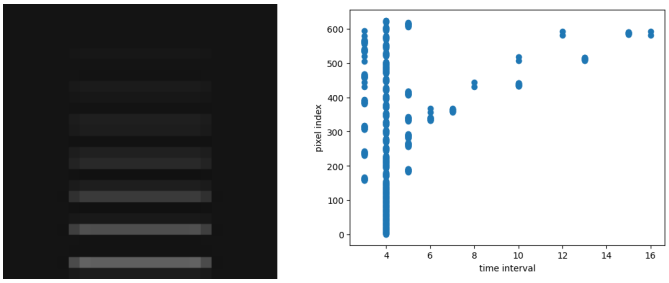

Figure 1.3: Time to first spike encoding example on 40 intervals

As we can see in the Figure 1.1, Figure 1.2, and Figure 1.3, the pixels which have more pixel value (whiter areas) give us later spikes. Also once a neuron related to a pixel spiked, it doesn't spike anymore (see figure 1.3 as a more explicit example for this).
Now we are going to divide the time to smaller intervals and analyse what happens.

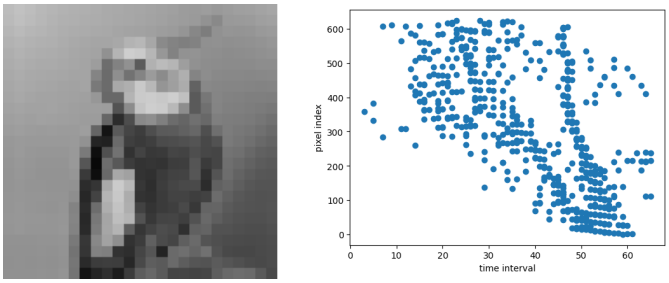

Figure 1.4: Time to first spike encoding example on 80 intervals

As expected, by increasing the number of intervals in a fixed time period, we will get more precise spikes.

1.2 Numerical values coding

In this section, we aim to encode a single value. For doing this, we will use tuning curves. We consider one tuning curve per neuron (with a fixed variance); Then we will encode the input value based on its intersection points with tuning curves.

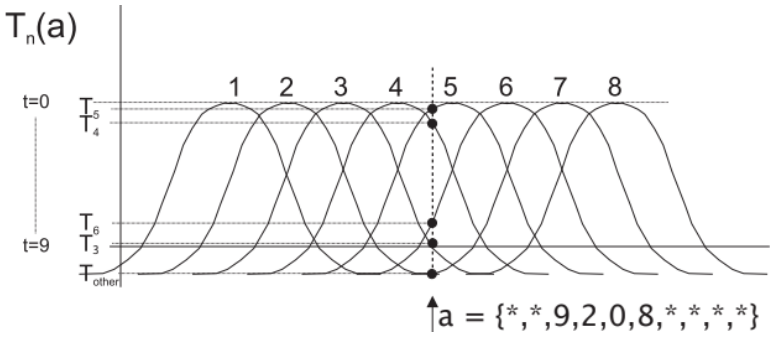

Figure 1.5: Tuning curve presentation of numerical value

As we can see in Figure 1.5, according to the given value, neurons 5, 4, 6 and 3 spike in t = 0, 2, 8, and 9 respectively (We estimate that the encoded value has been between 5 and 4, closer to 5 and so on.).

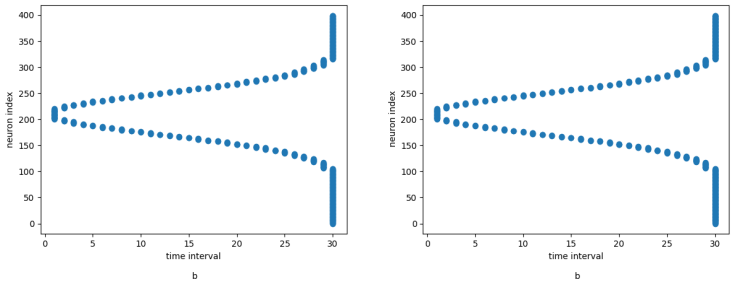

Figure 1.6: a) Encoded input value = 155.6 with variance = 40 on a population with 400 neurons; b)Encoded input value = 210.2 with variance = 40 on a population with 400 neurons.

As we can see in Figure 1.6, the order of neurons' spiking is according to the closeness of the number that they represent to the input value.
Now we are going to decrease the variance and see what happens.

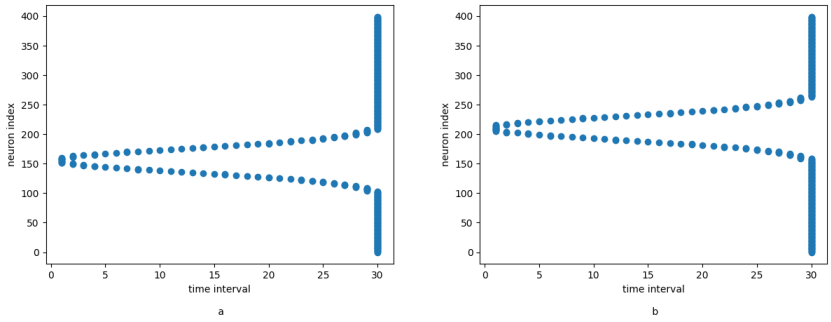

Figure 1.7: a) Encoded input value = 155.6 with variance = 20 on a population with 400 neurons; b)Encoded input value = 210.2 with variance = 20 on a population with 400 neurons.

As expected, by reducing the variance of the tuning curves, the scattering of spikes reduces.

1.3 Poisson distribution coding

Here we are going to encode our input pixels using a Poisson distribution. In order of doing this, we assign a spike probability to every pixel using Poisson distribution to estimate a proper activity schema.

Under a Poisson distribution with the expectation of λ events in a given interval, the probability of k events in the same interval is:

$$\frac{\lambda^k e^{-\lambda}}{k!}$$

and here, we assign the number of our pixels to k; Because each pixel is an input and each input is an event. In fact, every single pixel plays the role of one neuron.

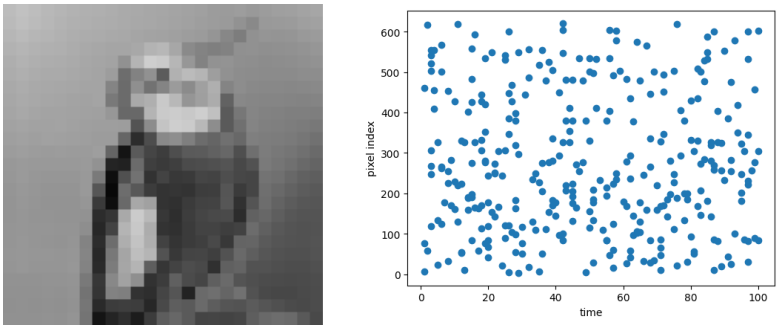

Figure 1.8: Poisson distribution encoding with $\lambda = 150$

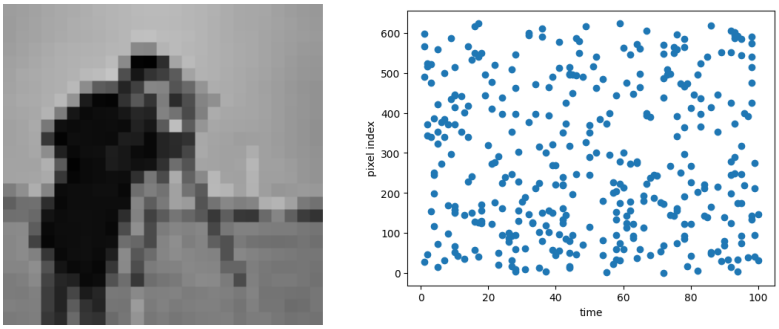

Figure 1.9: Poisson distribution encoding with $\lambda = 150$

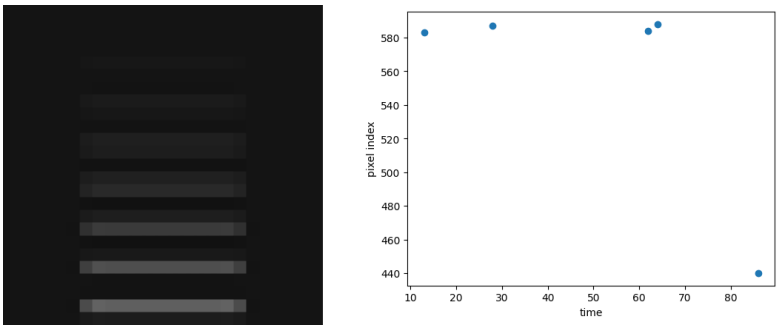

Figure 1.10: Poisson distribution encoding with $\lambda = 150$

Now, we can generate spikes according to the time using Poisson distribution; In this way we expect more spikes as the time gets closer to λ.

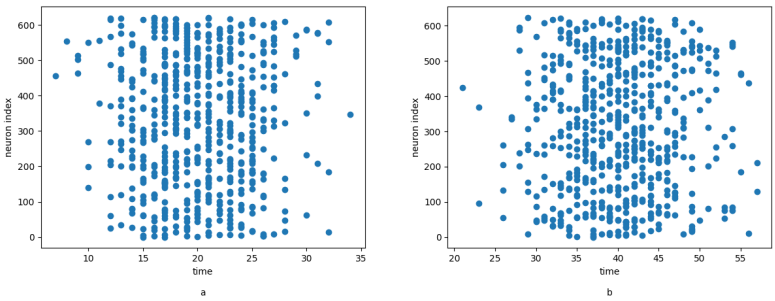

Figure 1.11: a) Poisson generator encoding with λ = 20; b) Poisson generator encoding with λ = 40.

As expected, in Figure 1.11, by getting closer to $time = \lambda$, we have more spikes and vice versa.

Chapter 2

Spike-Timing-Dependent Plasticity

2.1 Spike-Timing-Dependent Plasticity (STDP)

Under the STDP process, if an input spike to a neuron tends, on average, to occur immediately before that neuron's output spike, then that particular input is made somewhat stronger. If an input spike tends, on average, to occur immediately after an output spike, then that particular input is made somewhat weaker hence: "spike-timing-dependent plasticity". Thus, inputs that might be the cause of the post-synaptic neuron's excitation are made even more likely to contribute in the future, whereas inputs that are not the cause of the post-synaptic spike are made less likely to contribute in the future.

STDP learning rule:

$$\Delta w_+ = A_+(w) \, . \, e^{(t_{pre}-t_{post})/\tau_+} \quad if \;\; t_{post} > t_{pre}$$

$$\Delta w_- = A_-(w) \, . \, e^{-(t_{pre}-t_{post})/\tau_-} \quad if \;\; t_{post} < t_{pre}$$

where Δw denotes the change in the synaptic weight, A_+ and A_- determine the maximum amount of synaptic modification (which occurs when the timing difference between presynaptic and postsynaptic spikes is close to zero), τ_+ and τ_- determine the ranges of pre-to-postsynaptic interspike intervals over which synaptic strengthening or weakening occurs. Thus, $\Delta w > 0$ means that postsynaptic neuron spikes after the presynaptic neuron, and vice versa.

The pair-based STDP can be implemented with two local variables: presynaptic spike trace x_j, and postsynaptic spike trace y_i :

$$\frac{dx_j}{dt} = -\frac{x_j}{\tau_+} + \sum_f \delta(t - t_j^f)$$

$$\frac{dy_i}{dt} = -\frac{y_i}{\tau_-} + \sum_f \delta(t - t_i^f)$$

The values of x_j and y_i determine the weight change:

$$\frac{dw_{ij}}{dt} = -A_-(w_{ij})y_i(t)\sum_f \delta(t - t_j^f) + A_+(w_{ij})x_j(t)\sum_f \delta(t - t_i^f)$$

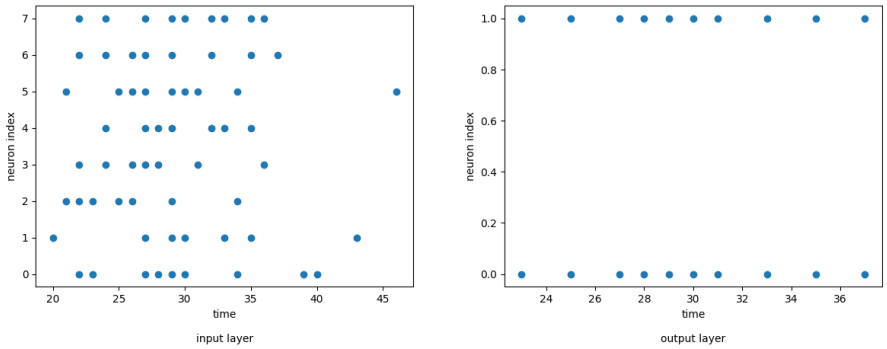

Figure 2.1: STDP learning

In Figure 2.1, we can see that our network is learning the given pattern! During the simulation, the output layer spikes considerably while the spikes of the input layer are decreased. Also the cosine similarity curve of the synaptic weights for output neurons is as following:

As we can see in Figure 2.2, the cosine similarity curve of the synaptic weights for output neurons has the amount of 1 all the time; Because the output neurons receive the same inputs so they have the same changes in synaptic weight and are learning the same pattern.

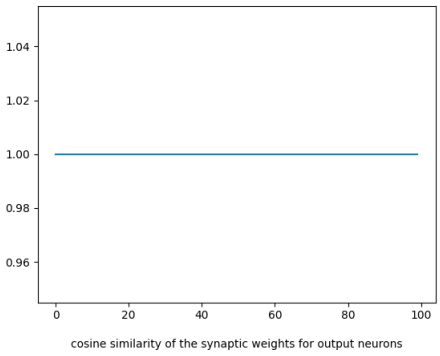

Figure 2.2: Cosine similarity of the synaptic weights for output neurons related to Figure 2.1

And we can see the synaptic weight changes for output neurons in Figure 2.3:

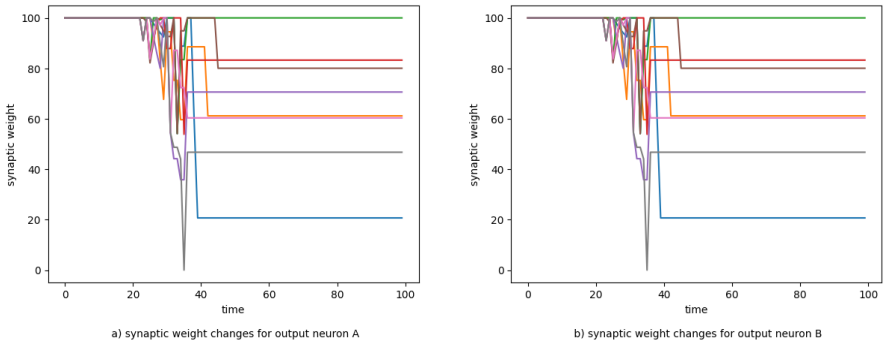

Figure 2.3: Synaptic weight changes related to Figure 2.1; As expected, the fluctuations for output neurons A and B are the same.

2.2 Experiments and analyses

Here, we have one input layer containing two neuron groups in which each neuron group has 4 neurons, and one output layer containing 2 neurons. All of the input neurons are connected to the output neurons. Each input neuron group has its own Poisson distributed activity pattern. We are going to activate input neuron groups one by one to see what happens.

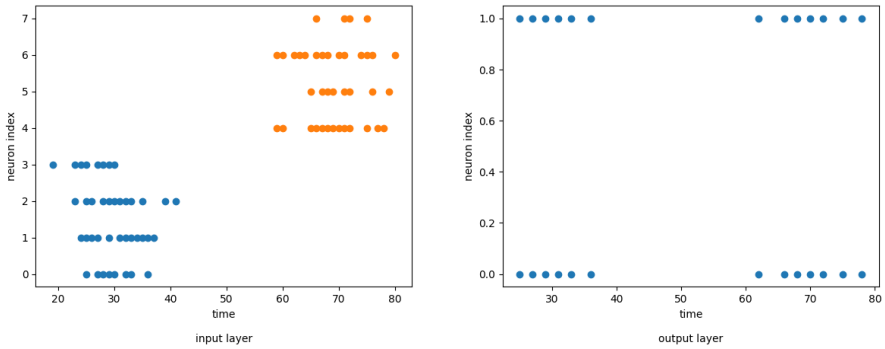

Figure 2.4: STDP learning; Two input neuron groups with different Poisson distributed activities, activated on various times. The output neurons are learning as it is obvious from their spiking pattern.

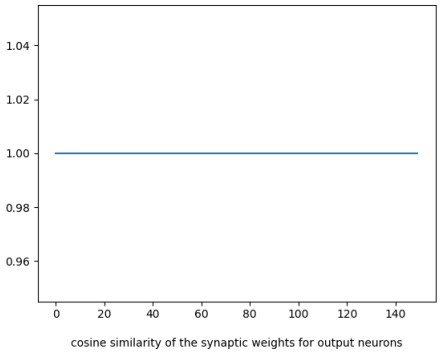

Figure 2.5: Cosine similarity of the synaptic weights for output neurons related to Figure 2.4

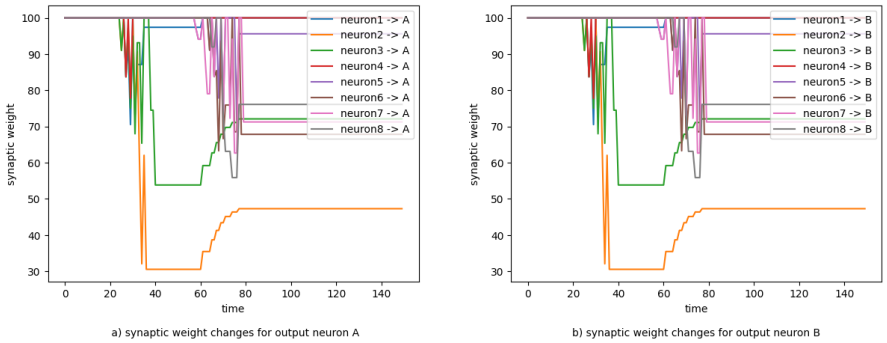

Figure 2.6: Synaptic weight changes related to Figure 2.4; As expected, the fluctuations for output neurons A and B are the same.

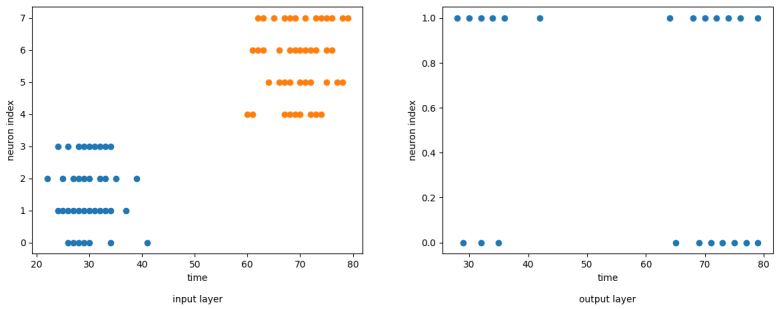

Figure 2.7: STDP learning; Two input neuron groups with different Poisson distributed activities; This time with higher connectivity variance and increased maximum weight of synapses in the STDP process.

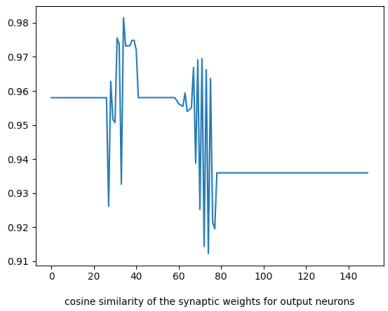

Figure 2.8: Cosine similarity of the synaptic weights for output neurons related to Figure 2.7; In this case, the cosine similarity has some fluctuation; This is mostly because of the increment of the maximum synaptic weight of the process.

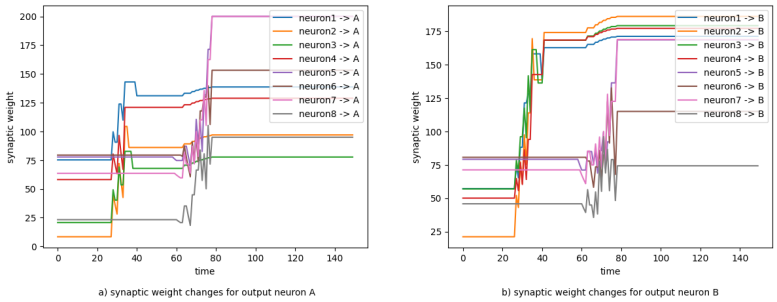

Figure 2.9: Synaptic weight changes related to Figure 2.7; As expected, synaptic weight changes are a bit different for output neurons A and B.

2.3 Input layers with common neurons

Here we are going to add common neurons to our input neuron groups; First, we have 2 neuron groups that each neuron group has 5 neurons, in which 2 neurons are common between the two neuron groups:

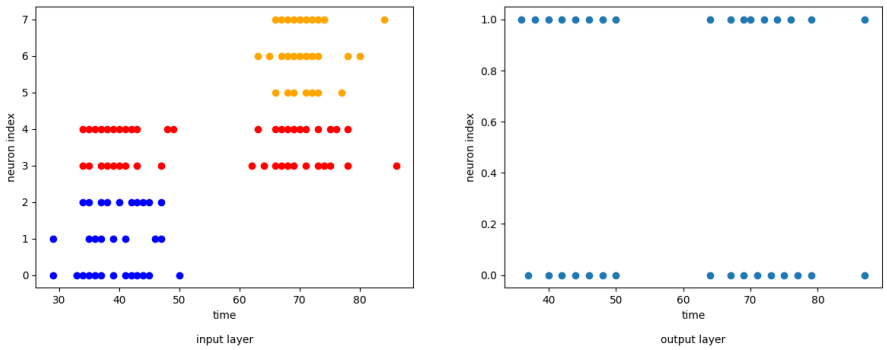

Figure 2.10: STDP learning process with 2 common neurons between input layer's neuron groups (common neurons are shown in red)

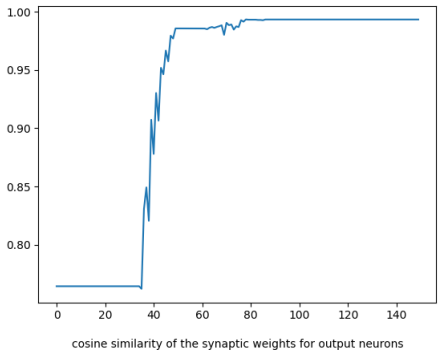

Figure 2.11: Cosine similarity of the synaptic weights for output neurons related to Figure 2.10

As we can see in Figure 2.10, with the presentment of common learning patterns our network is learning faster. Now we will increase the number of common neurons to see what happens:

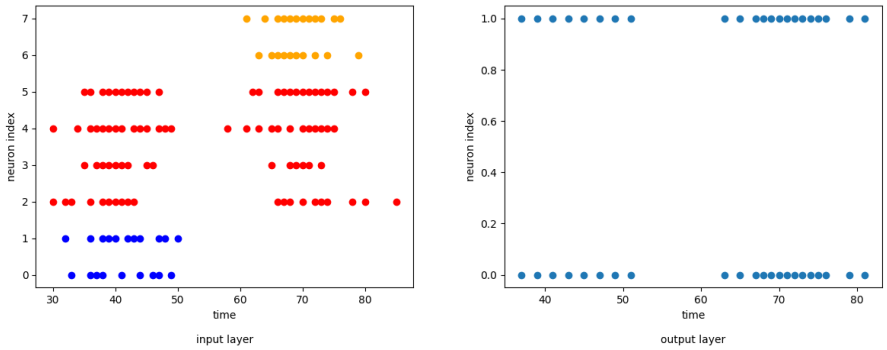

Figure 2.12: STDP learning process with 4 common neurons between input layer's neuron groups (common neurons are shown in red)

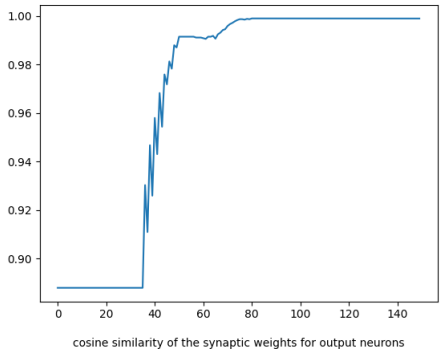

Figure 2.13: Cosine similarity of the synaptic weights for output neurons related to Figure 2.12

In Figure 2.12, it's clear that learning process has got faster and stronger once we increased the number of common neurons in input patterns; Also as we can see in Figure 2.13 the cosine similarity of the synaptic weights is more likely to converge to 1.

2.4 Isolated neurons

In this section we are going to add an isolated neuron to each input and output layer and analyse their activity.

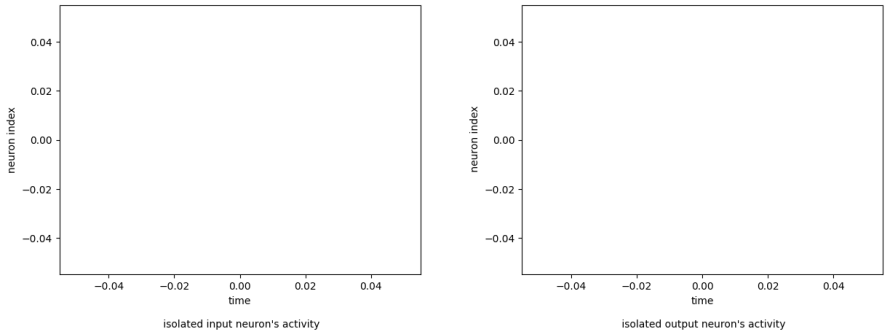

Figure 2.14: Activity of the input and output layers' isolated neurons

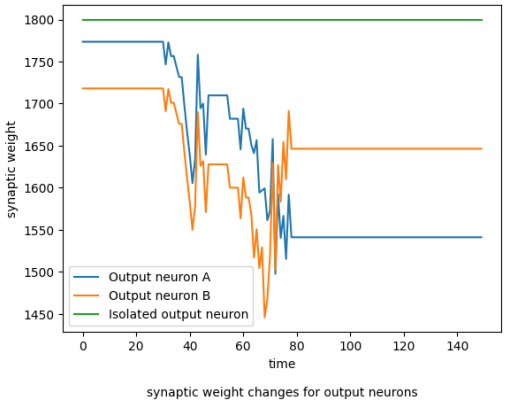

Figure 2.15: Synaptic weight changes for output neurons

As expected, we can see in Figure 2.14 that isolated neurons don't spike. Also it is obvious from Figure 2.15 that synaptic weights related to the isolated output neuron never change.

2.5 Background activity

In this section, we are going to add a background activity to our neuron groups. By this means, sometimes our neurons will spike randomly.

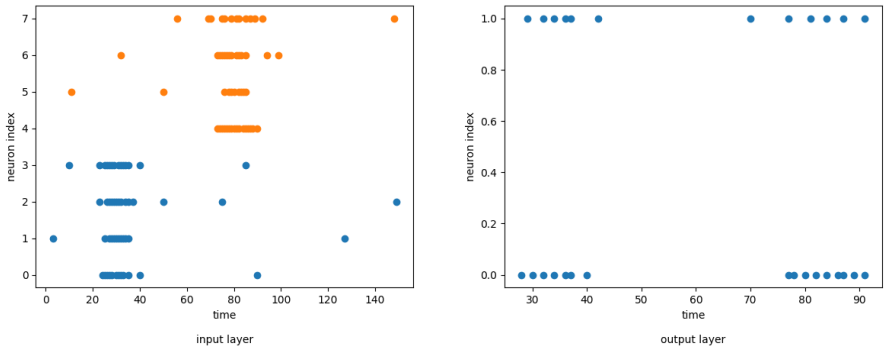

Figure 2.16: STDP learning; Activity of the input and output layers' neurons with a minimum rate of background activity

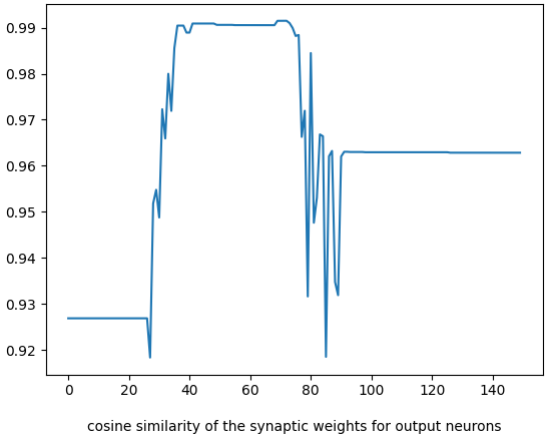

Figure 2.17: Cosine similarity of the synaptic weights for output neurons related to Figure 2.16

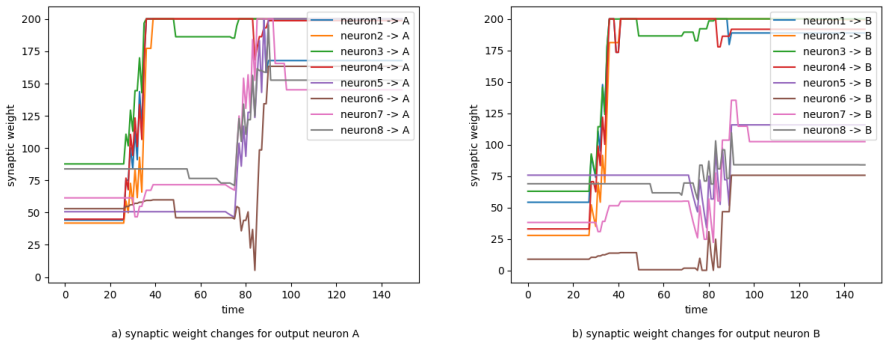

Figure 2.18: Synaptic weight changes related to Figure 2.16

Now we are going to add background activity to the network which has isolated neurons.

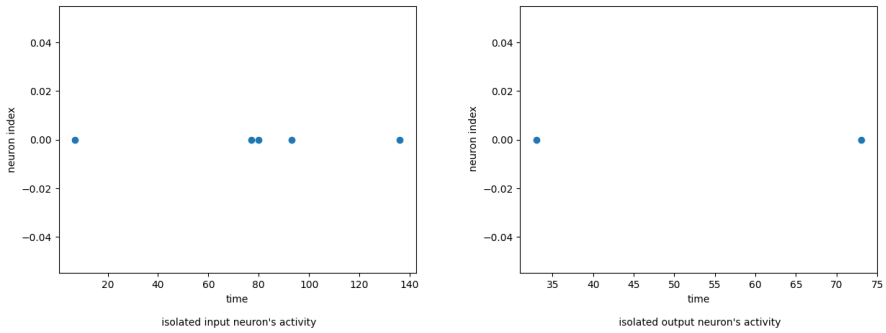

Figure 2.19: Activity of the input and output layers' isolated neurons with a minimum background activity

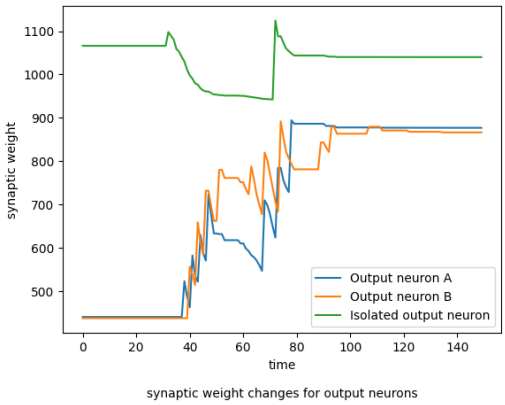

Figure 2.20: Synaptic weight changes for output neurons

As we can see in Figure 2.19, isolated neurons has random spikes as a result of the background activity; Also the synaptic weight curve of the isolated output neuron fluctuates a bit as a result of random spikes.

Chapter 3

Reward-Modulated Spike-Timing-Dependent Plasticity

3.1 Reward-Modulated Spike-Timing-Dependent Plasticity (R-STDP)

In this chapter, we will study Reward-Modulated Spike-Timing-Dependent-Plasticity learning method. In this method, we consider a fixed target; After training the network, if its results are aligned with the considered target, we will reward the network; Vice versa, if the network doesn't meet our expectations, we will punish it. So the network learns from the feedback of each action.

In this method, the state of each synapse is represented by two variables:

1. synaptic strength/weight, s
2. activation of an enzyme important for plasticity, c

The dynamics of these variables follow:

$$\frac{dc}{dt} = -\frac{c}{\tau_c} + STDP(\tau)\delta(t - t_{pre/post})$$

$$\frac{ds}{dt} = cd$$

where $\tau = t_{post} - t_{pre}$, d describes the extracellular concentration of DA and $\delta(t)$ is the Dirac delta function. Firings of pre- and postsynaptic neurons change c by the amount $STDP(\tau)$. Variable c decays to 0 exponentially with the time constant$\tau_c = 1s$ (eligibility trace). The model integrates the millisecond timescale of spike interactions in STDP with the slow trace modulated by global reward signal (behavioral timescale). The variable d describes the concentration (μM) of extracellular DA:

$$\frac{dd}{dt} = -\frac{d}{\tau_d} + DA(t)$$

where τ_d is the time constant of DA uptake and DA(t) models the source of DA due to the activity of dopaminergic neurons (reward minus expected reward).

3.2 Experiments and analyses

Here we are going to train our network with R-STDP learning rule. In the first experiment, we aim the output neuoron B (with index 1) to get activated:

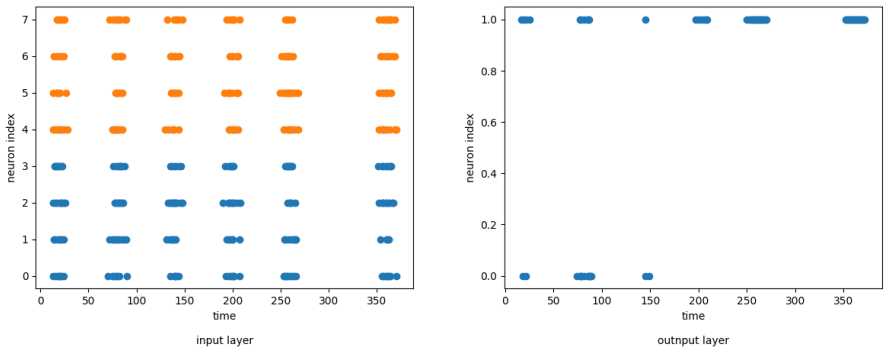

Figure 3.1: R-STDP learning; Target: Neuron B (with index 1)

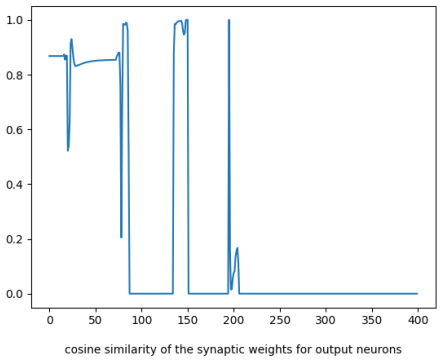

Figure 3.2: Cosine similarity of the synaptic weights for output neurons related to Figure 3.1

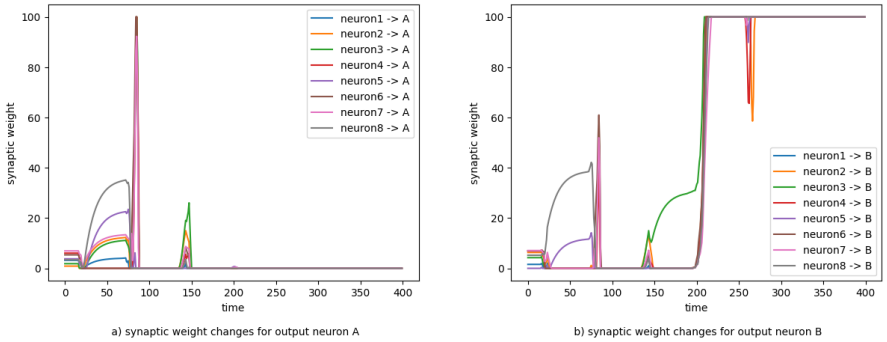

Figure 3.3: Synaptic weight changes related to Figure 3.1

As expected, we can see in Figure 3.1 that the pattern resulting to the activation of output neuron B (with index 1), is being learned; On the other hand, output neuron A (with index 0) first starts to spike but then we punish it (because it's not our target), so its activity converges to 0; Also the synaptic weights related to its activity decrease. (see Figure 3.3)
In Figure 3.2 we see that the cosine similarity of the synaptic weights for output neurons converges to 0 over the time and this is simply because of we rewarded the pattern which results in activation of one neuron, and punished the activation of the other one; So they won't act in the same way.

Now we change the target; This time we aim the pattern which results in the activation of output neuron A (with index 0) to get learned:

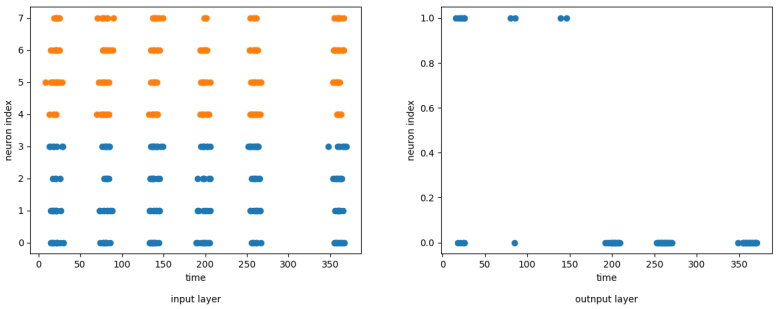

Figure 3.4: R-STDP learning; Target: Neuron A (with index 0)

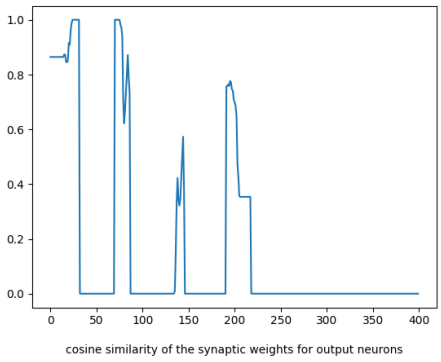

Figure 3.5: Cosine similarity of the synaptic weights for output neurons related to Figure 3.4

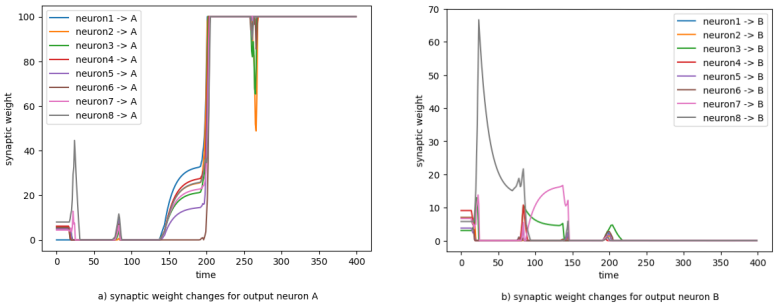

Figure 3.6: Synaptic weight changes related to Figure 3.4

As expected, this time the pattern which results in the activation of neuron A (with index 0) is getting rewarded and learned; Also the synaptic weights related to this process are getting increased till they reach the maximum amount

of synaptic weights (which we set 100.0). Other than that, the pattern is getting punished so after a while, neuron B (with index 1) will not get activated anymore and the synaptic weights resulting to its activation will be 0.

3.3 Input layers with common neurons

Here we are going to add common neurons to our input neuron groups; First, we have 2 neuron groups that each neuron group has 5 neurons, in which 2 neurons are common between the two neuron groups:

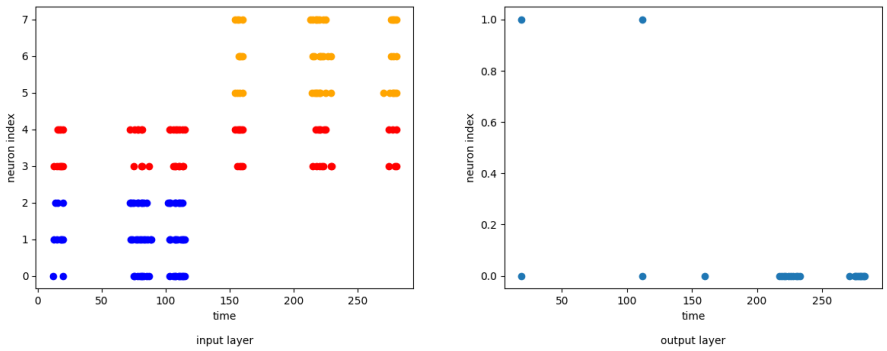

Figure 3.7: R-STDP learning process with 2 common neurons between input layer's neuron groups (common neurons are shown in red); Target: Neuron A (with index 0)

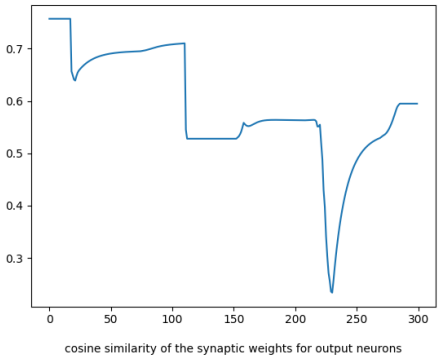

Figure 3.8: Cosine similarity of the synaptic weights for output neurons related to Figure 3.7

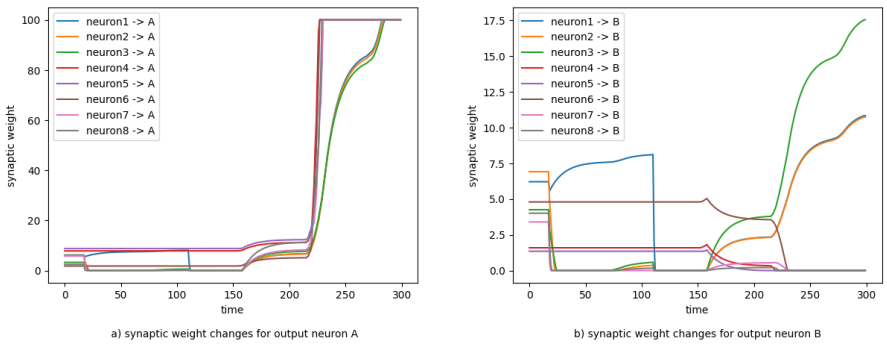

Figure 3.9: Synaptic weight changes related to Figure 3.7

As we can see in Figure 3.7, the network is learning the pattern which results in the activation of the target output neuron, which is neuron A in this experiment; But since there are some common input neurons in the input layer, the cosine similarity of the synaptic weights for output neurons has a considerable (nonzero) value. (see Figure 3.8)
Also the synaptic weights related to the synapses of 2 common neurons with output neuron B, are not 0. Even though our target is not activation of neuron B, but activity of the common input neurons impress the activation of the target neuron, A. So their activation will be rewarded anyway. (see Figure 3.9)

Now, we are going to increase the number of joint neurons.

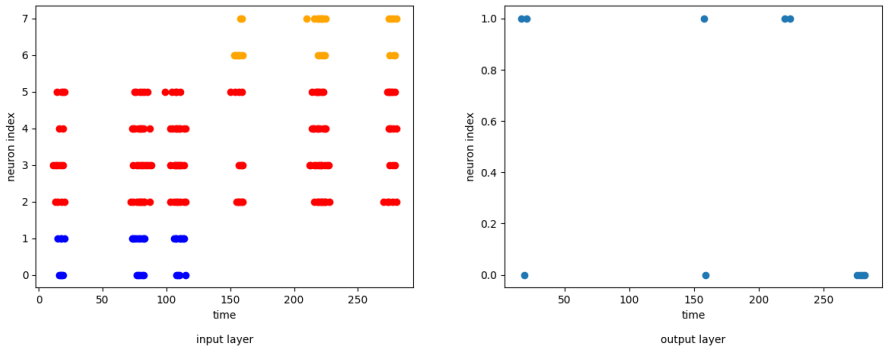

Figure 3.10: R-STDP learning process with 4 common neurons between input layer's neuron groups (common neurons are shown in red); Target: Neuron A (with index 0)

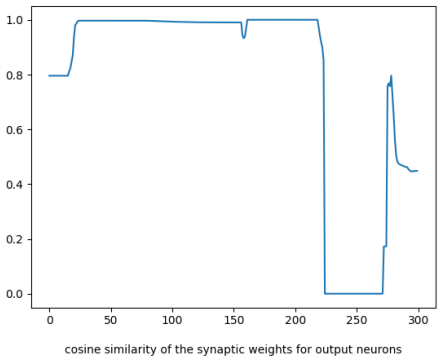

Figure 3.11: Cosine similarity of the synaptic weights for output neurons related to Figure 3.10

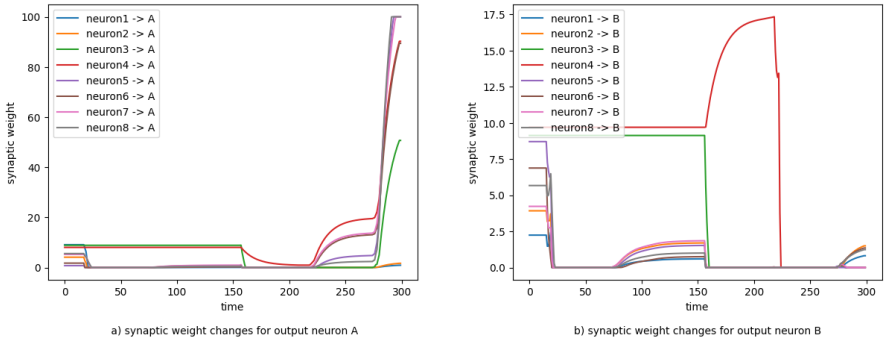

Figure 3.12: Synaptic weight changes related to Figure 3.10

As we can see in Figure 3.11, with increasing the number of common neurons between input neuron groups, the parameter of cosine similarity of synaptic weights for output neurons, relatively has a greater value than the case in which we had fewer joint neurons.

3.4 Isolated neurons

n this section we are going to add an isolated neuron to each input and output layer and analyse their activity. Here the target is activation of output neuron A.

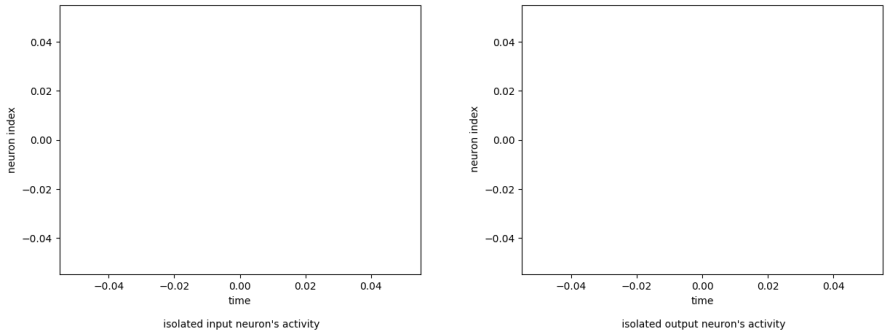

Figure 3.13: Activity of the input and output layers' isolated neurons

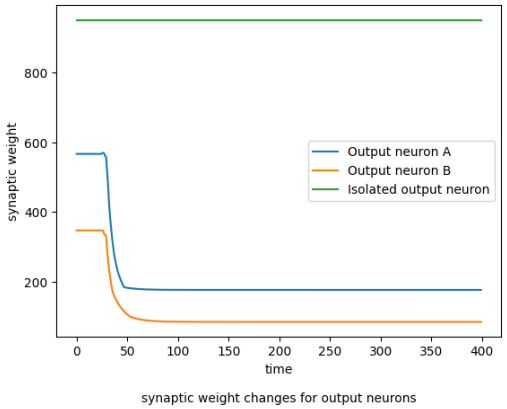

Figure 3.14: Synaptic weight changes for output neurons

As expected, we can see in Figure 3.13 that isolated neurons don't spike. Also it is obvious from Figure 3.14 that synaptic weights related to the isolated output neuron never change despite the changes in the synaptic weights relative to output neurons A and B. We also see in Figure 3.14 that synaptic weights related to output B, decays to 0 as a result of punishment.

3.5 Background activity

In this section, we are going to add a background activity to our neuron groups. By this means, sometimes our neurons will spike randomly.

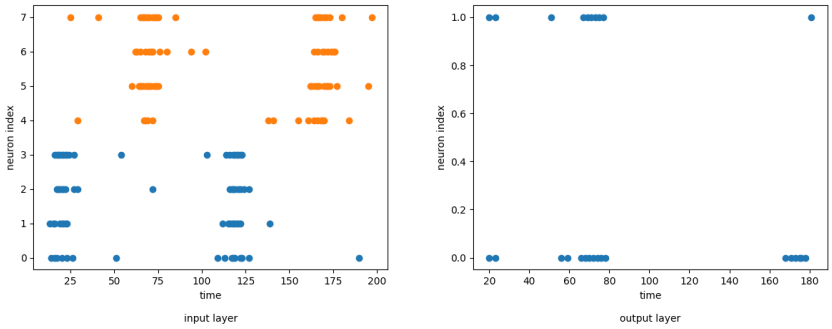

Figure 3.15: R-STDP learning; Activity of the input and output layers' neurons with a minimum rate of background activity; Target: Neuron A (with index 0)

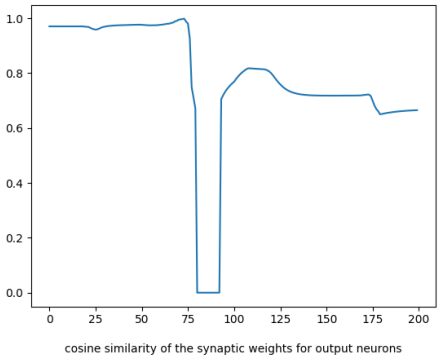

Figure 3.16: Cosine similarity of the synaptic weights for output neurons related to Figure 3.15

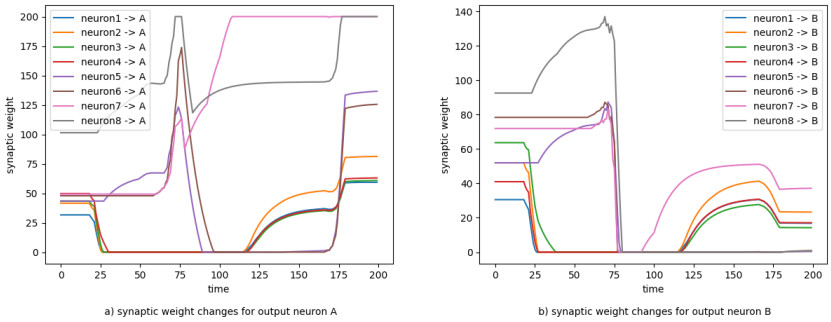

Figure 3.17: Synaptic weight changes related to Figure 3.15

As we can see in Figure 3.17, although the target is the activation of neuron A, but some of the synapses related to neuron B keep nonzero weights, and this is

simply because of the existence of a background activity, which leads to some random spikes. Because of this activity of both output neurons, we expect often nonzero value of the cosine similarity curve for the synaptic weights related to output neurons. (see Figure 3.16)

Now we are going to add background activity to the network which has isolated neurons.

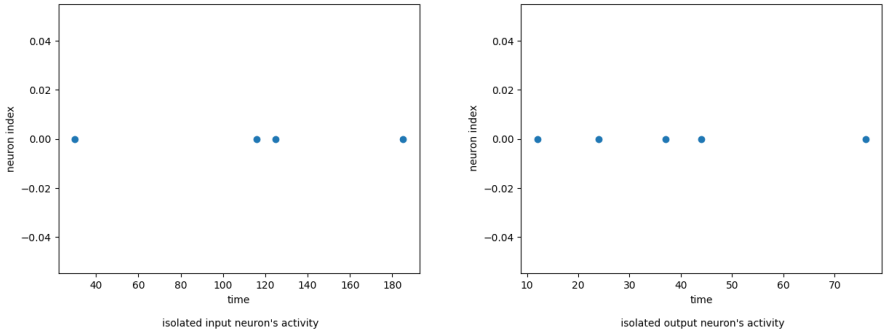

Figure 3.18: Activity of the input and output layers' isolated neurons with a minimum background activity

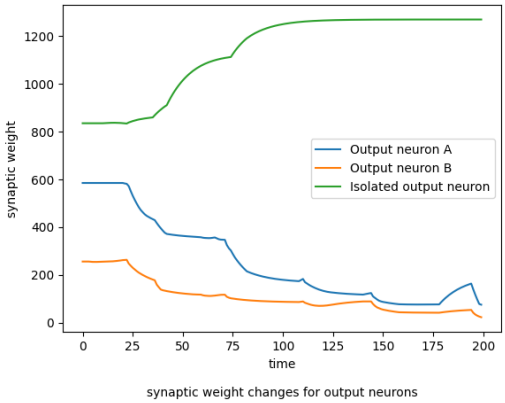

Figure 3.19: Synaptic weight changes for output neurons; Target: neuron A

As we can see in Figure 3.18, isolated neurons has random spikes as a result of the background activity; Also as we can see in Figure 3.19, the synaptic weight curve of the isolated output neuron fluctuates a bit as a result of random spikes. The synaptic weights of the synapses related to the target output neuron, A,

are greater than the synaptic weights related to output neuron B, as we were expecting.

Chapter 4

Conclusions

In this report, we studied three neural coding schemes: Time-To-First-Spike (TTFS), numerical values and Poisson distribution coding methods. We also studied two important learning methods: Spike-Timing-Dependent Plasticity (STDP), and Reward-Modulated Spike-Timing-Dependent Plasticity (R-STDP). We did different experiments and analysed their results. In STDP learning method, a positive change (LTP) occurs if the presynaptic spike precedes the postsynaptic one; For a reversed timing, synaptic weights are decreased. While R-STDP learning method uses a reward-and-punishment paradigm. This method assigns positive values to the desired actions to encourage the agent to use them (rewarding), while negative values are assigned to undesired behaviors to discourage them (punishment).