project-6

May 18, 2021

Contents

Chapter 1

Introduction

1.1 Pair-based STDP

We implement the following rule for $STDP$:

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

y_i and x_j are spike traces and are determined like this:

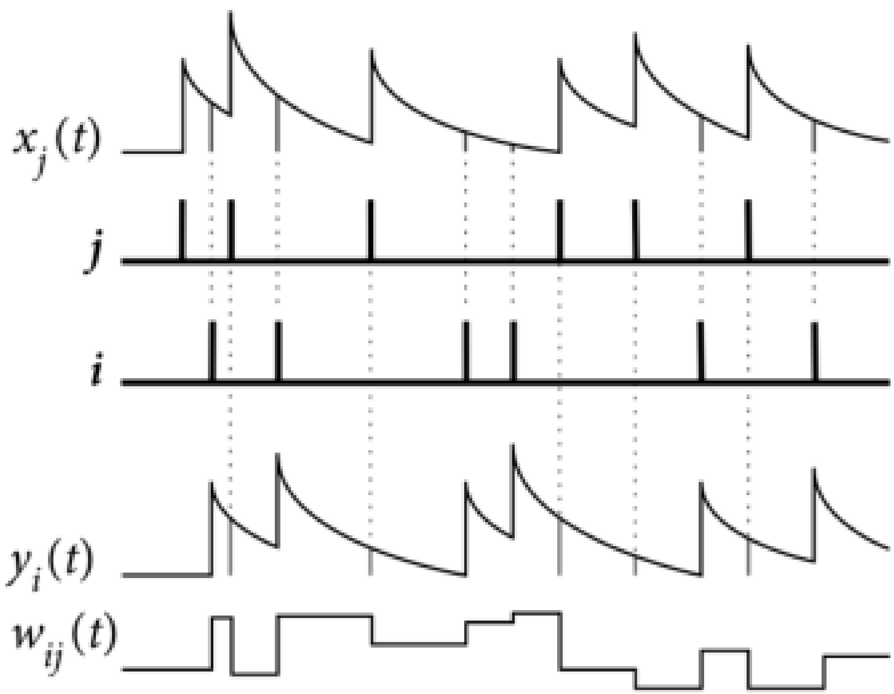
- Presynaptic spike trace x_j:

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

- Postsynaptic spike trace y_i:

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

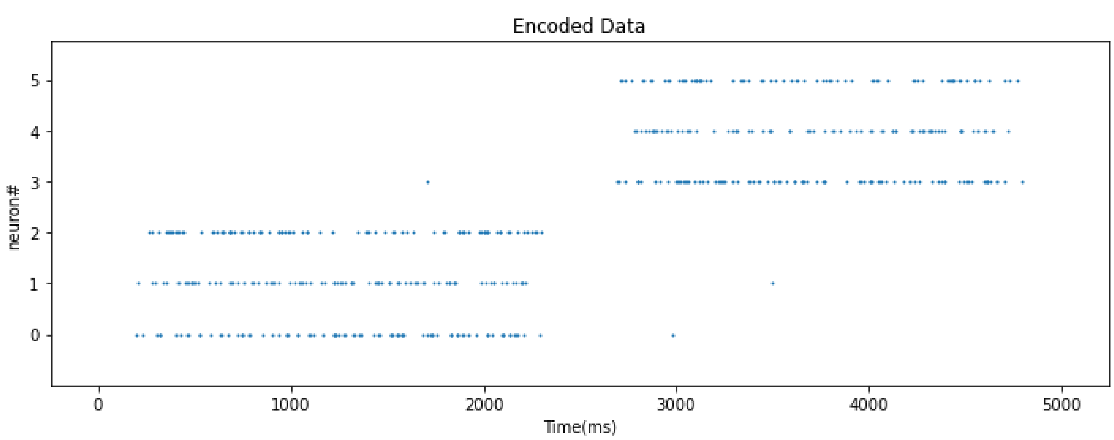
Using the update rule results in the following pattern in the weights:



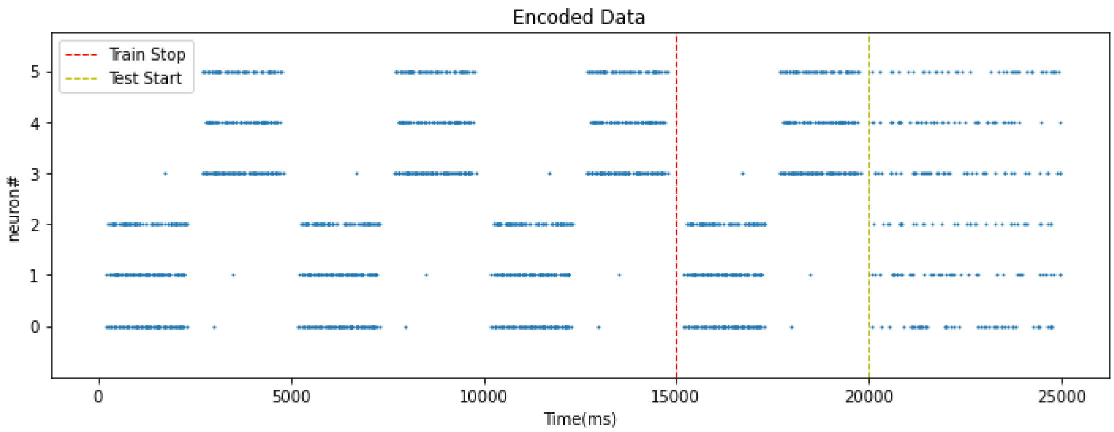
To make the output neurons better distinguish the inputs and make each of them learn only one of the input patterns, we also added a lateral-inhibition in the output layer. The weights of this inhibitory connection is fixed with random initialization.

1.2 Input

The input patterns are created by using the PoissonEncoder that we implemented in the previous project. You can see the patterns below:



By default, the input patterns will be presented to the network four times in our experiments. The last time the training is stopped to check whether the network has learned the patterns correctly. Finally, a random pattern whose average intensity is almost the same as the patterns is fed into the network. This last input is used to show that the network does not activate significantly with an input that it has not seen previously. We call the first three iterations, training iterations, and the 4th step, the evaluation iteration. The last step is the testing iteration. So, the complete input is as follows:



Note that the PoissonEncoder is working **randomly**, so the inputs are a bit different from one simulation to the other. The connection weights are also **randomly** drawn from a normal distribution. To keep the comparisons fair, we fix the **random-seed** when investigating the effect of each parameter in order to have exactly the same input for each case.

By running the training process numerous times, we found a couple of **good** random seeds that make the network show the correct output of the learning rules. We use these good seeds for the following experiments.

1.3 How do we know the network is learning the data?

The co-variance of the input patterns and output neurons' activations indicate that the learning is happening. In other words, the first time the network sees the input, the output activation is low; if the data is presented to the network for more iterations, each time the activation of the neurons tend to increase. This means that the network is learning the input pattern because its activation is increasing each time that it sees the exact same data. In our experiments, we interpret the dependency between the network output and input visually using the output neurons' raster plot.

Chapter 2

STDP

2.1 Default Parameters:

Train Params:

$Time_{simulation} = 25000ms$

$LearningRates = [0.03, 0.03]$ →The first learning-rate refers to A_-, and the second learning-rate refers to A_+.

Neurons Params:

$Num(PresynapticNeurons) = 6$
$\tau_s = 10ms$
$Threshold = -52mv$
$U_{rest} = -60mv$

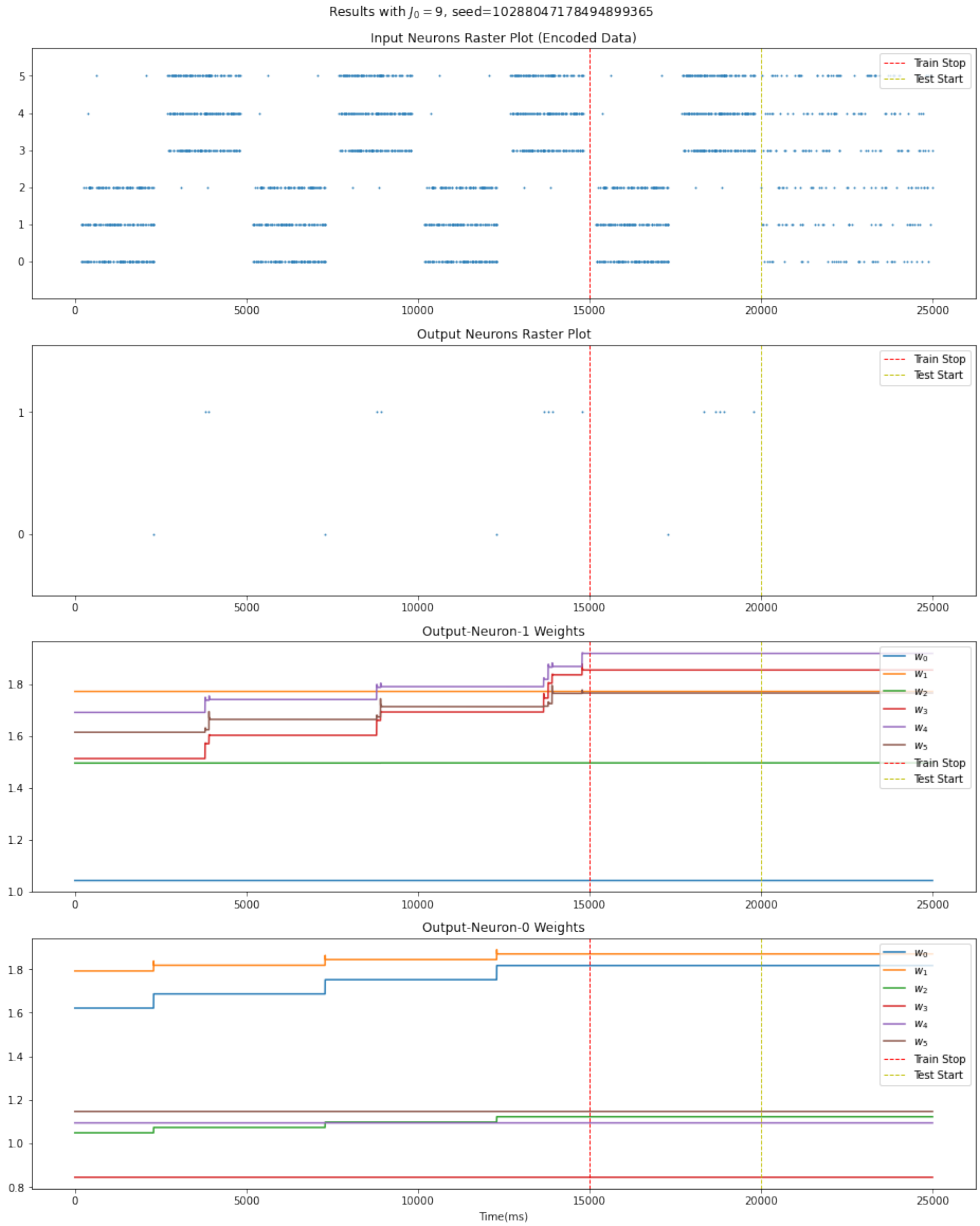
Connection Params:

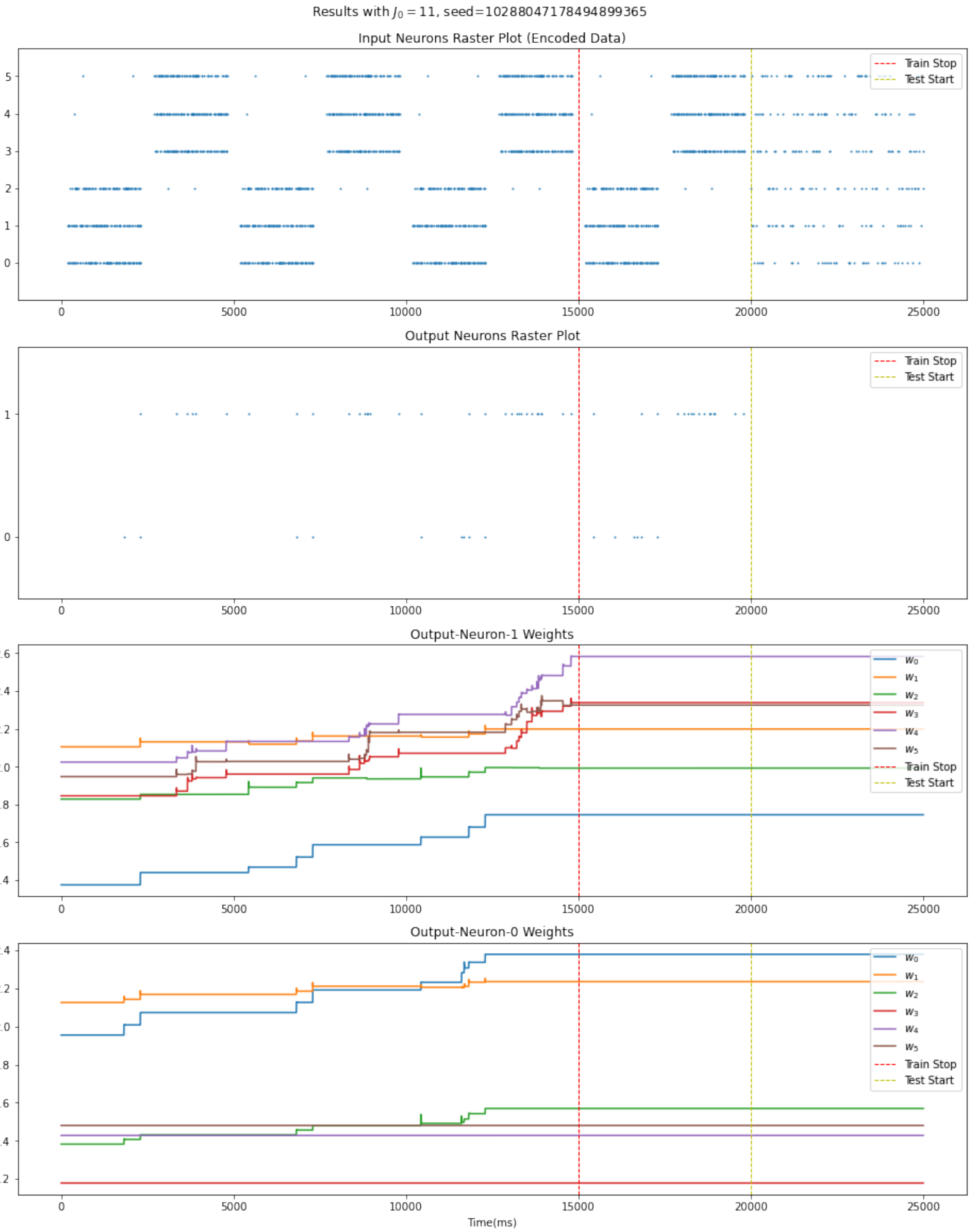
$J_0 = 11$
$\sigma_0 = 2.5$
$Weight_{min} = 0$
$Weight_{max} = 7.5$

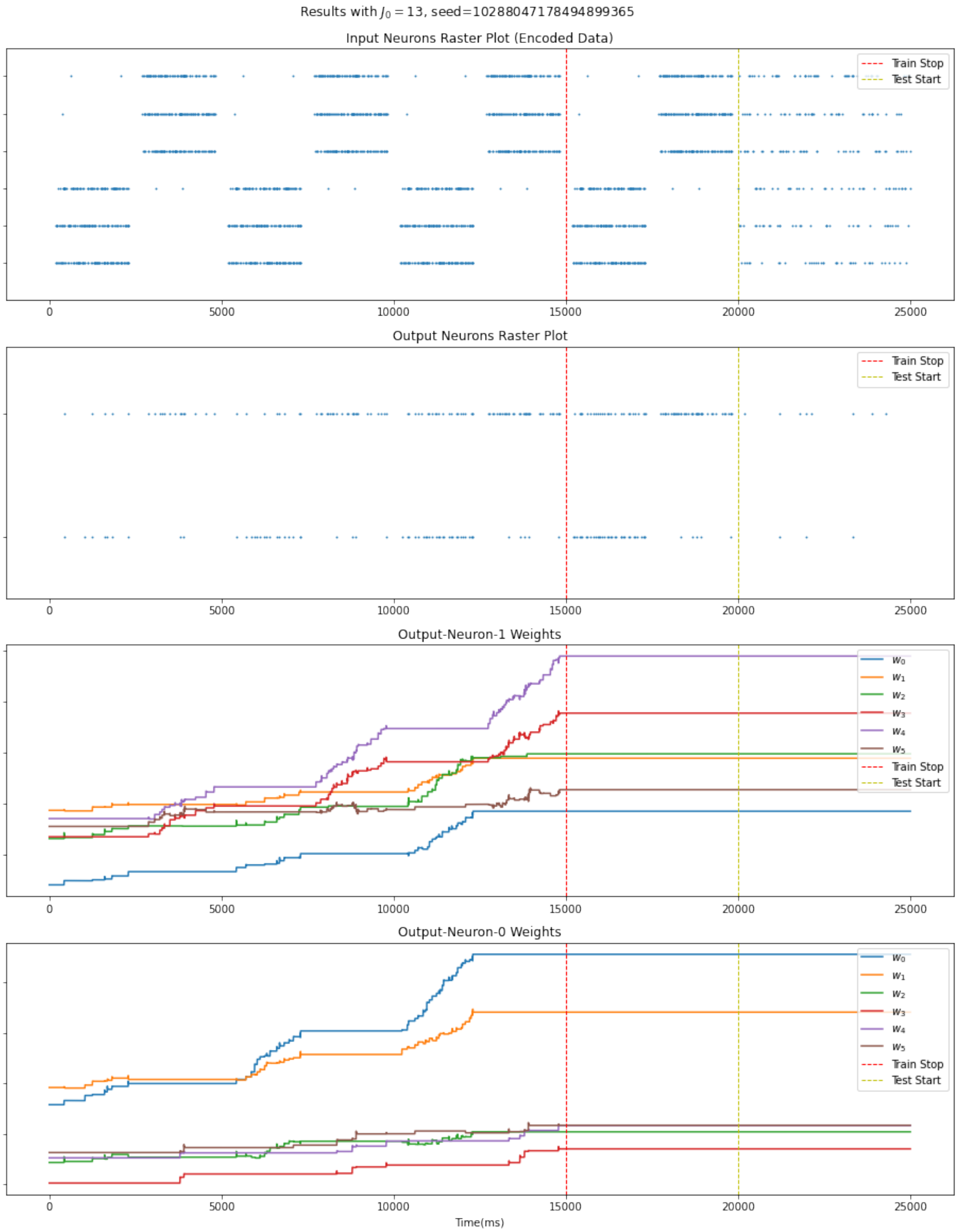
2.2 Experiment #1 (Initial Weights)

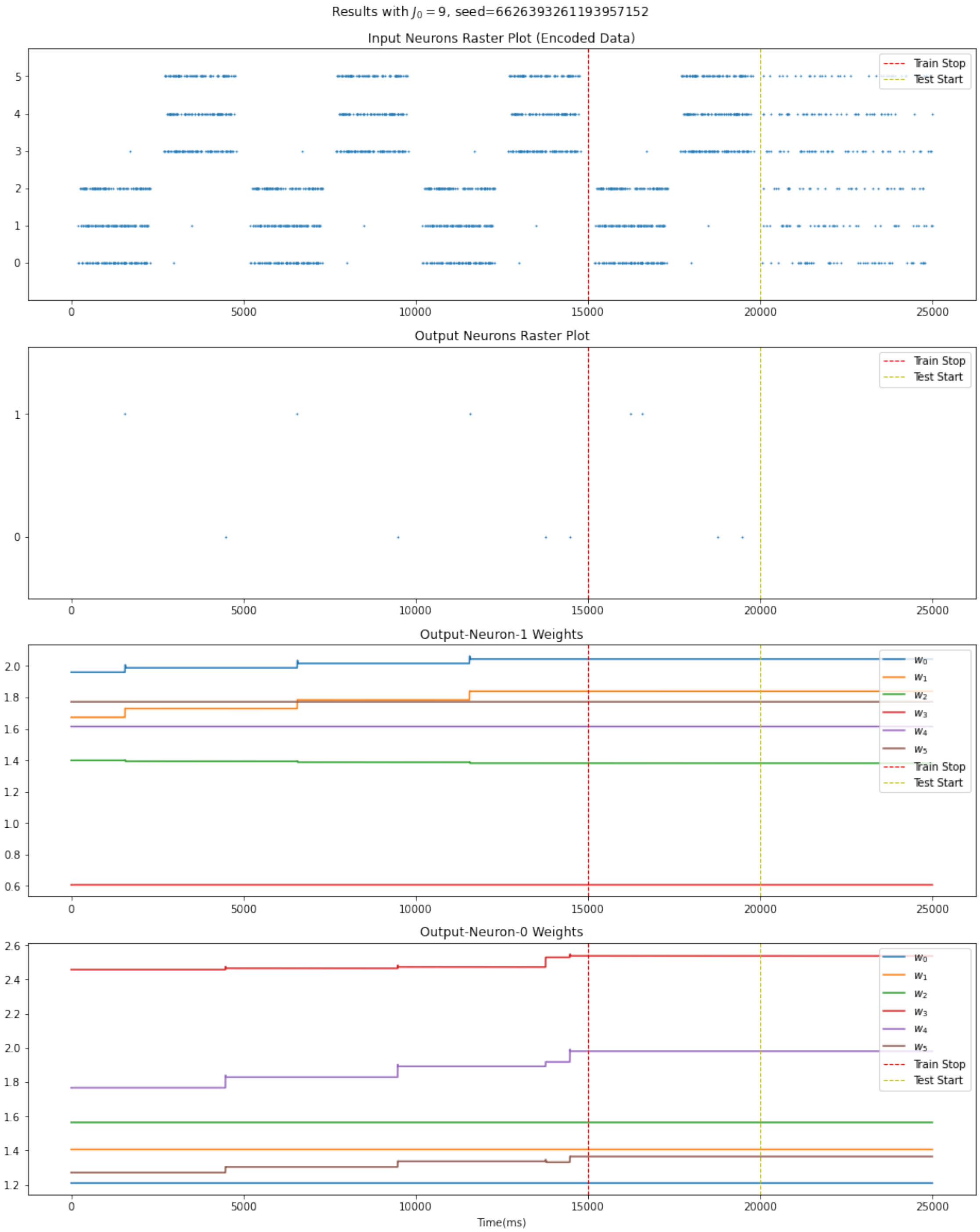
2.2.1 J_0

Higher values of J_0 indicate higher initial weights for connections' weights.

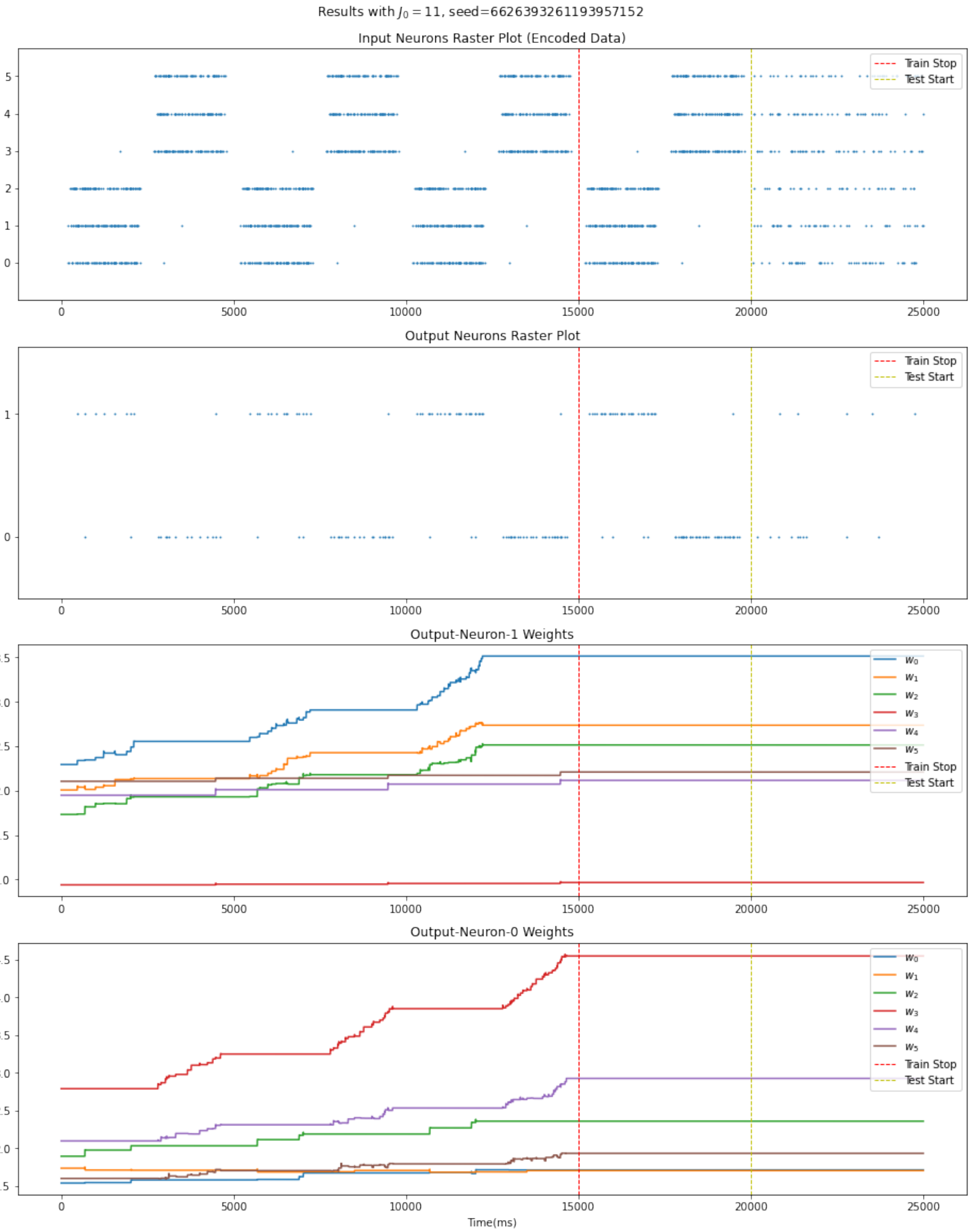

Results with $J_0 = 9$, seed=10288047178494899365
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)

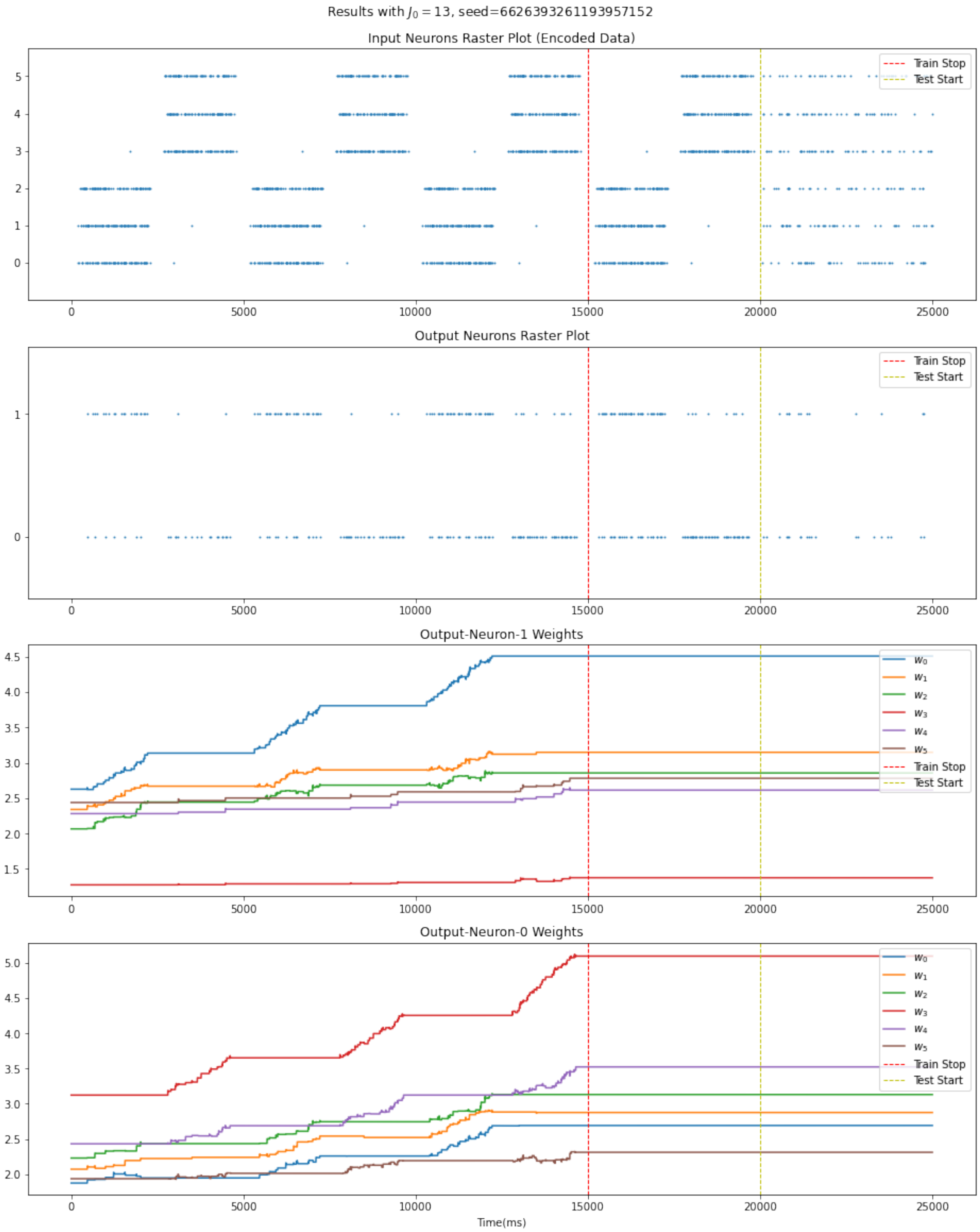

Results with $J_0 = 11$, seed=10288047178494899365
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
0
5000
10000
15000
20000
25000
Time(ms)


Results with $J_0 = 13$, seed=10288047178494899365
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)
0
5000
10000
15000
20000
25000

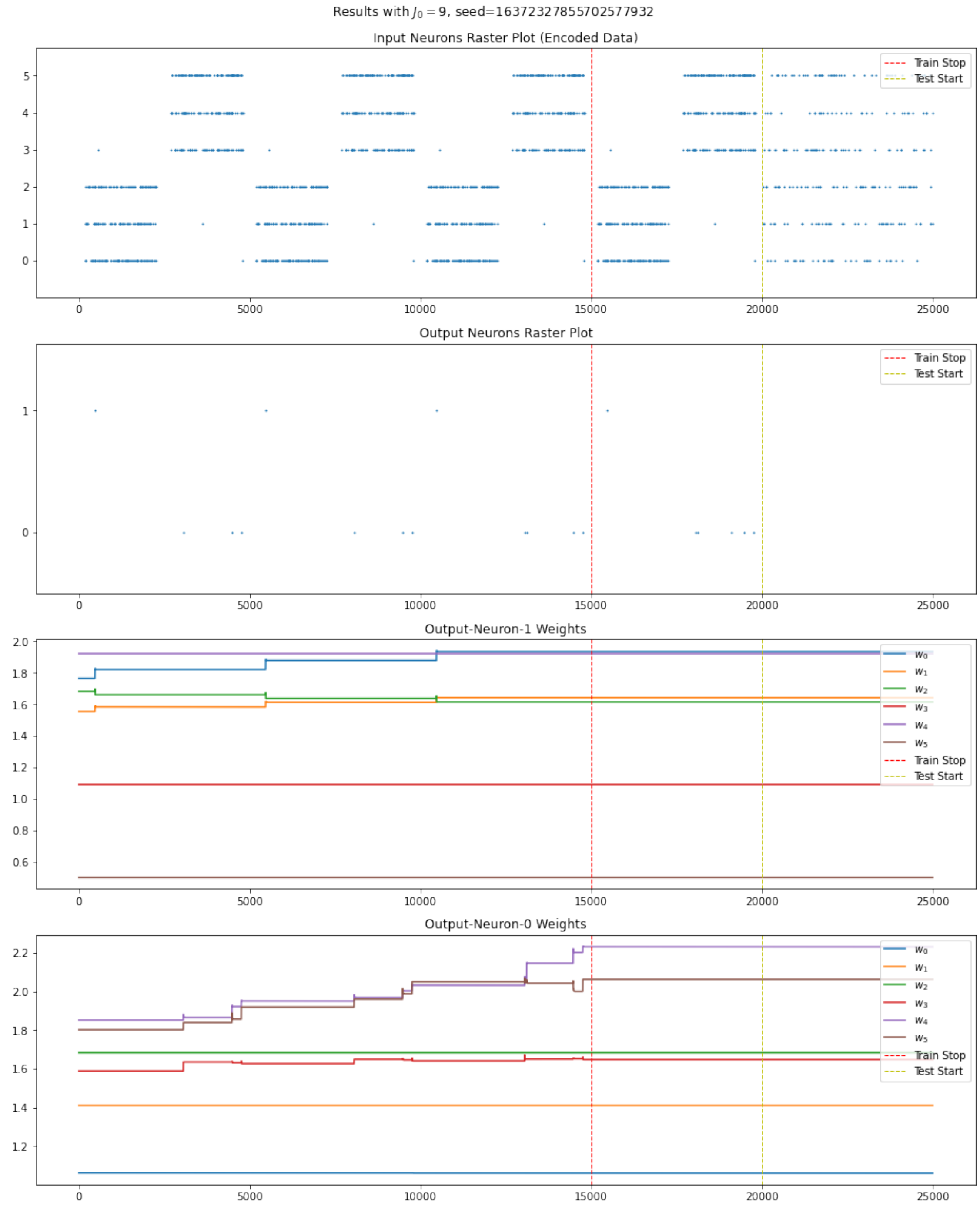


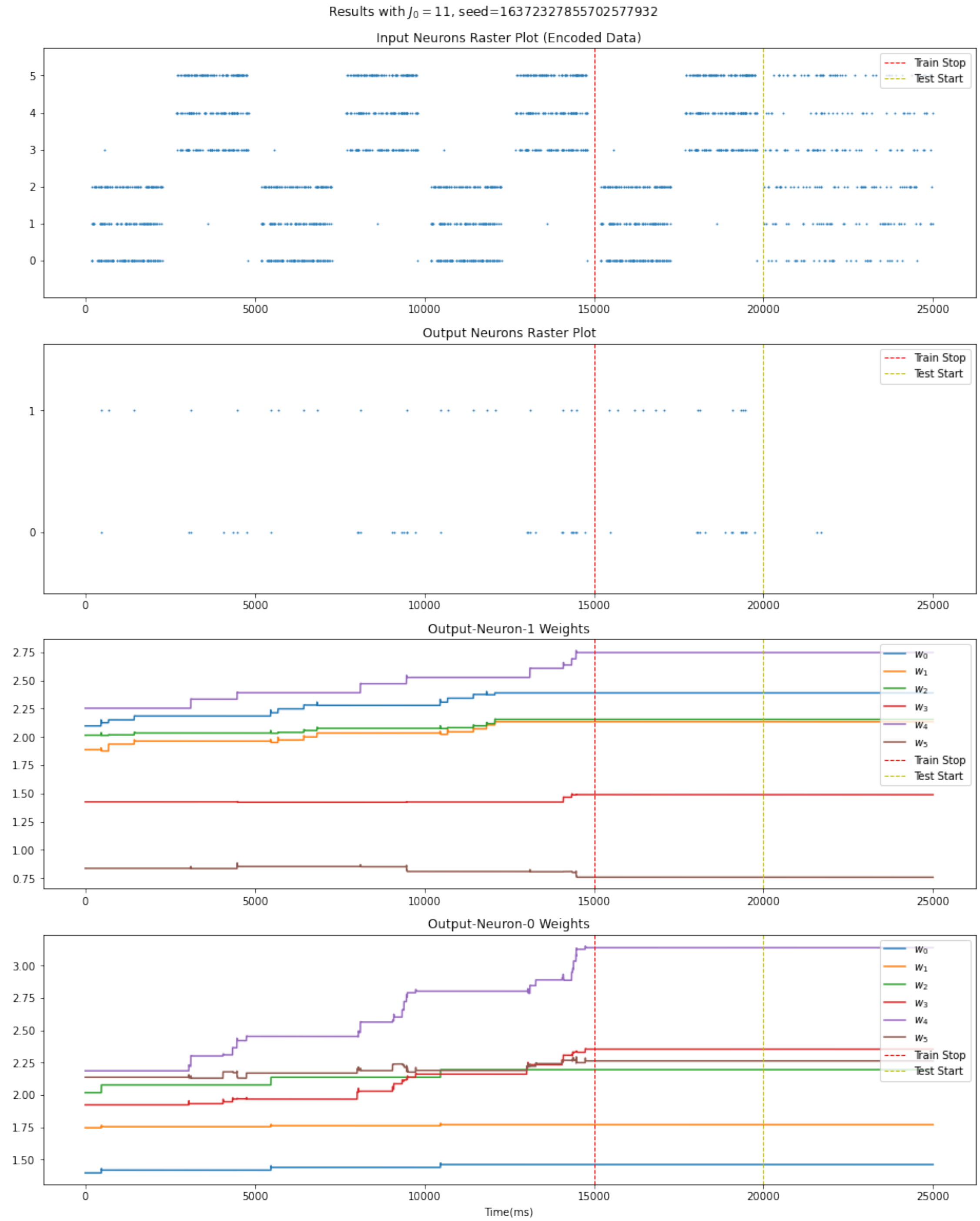
Results with $J_0 = 9$, seed=6626393261193957152
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Train Stop
Test Start
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Time(ms)

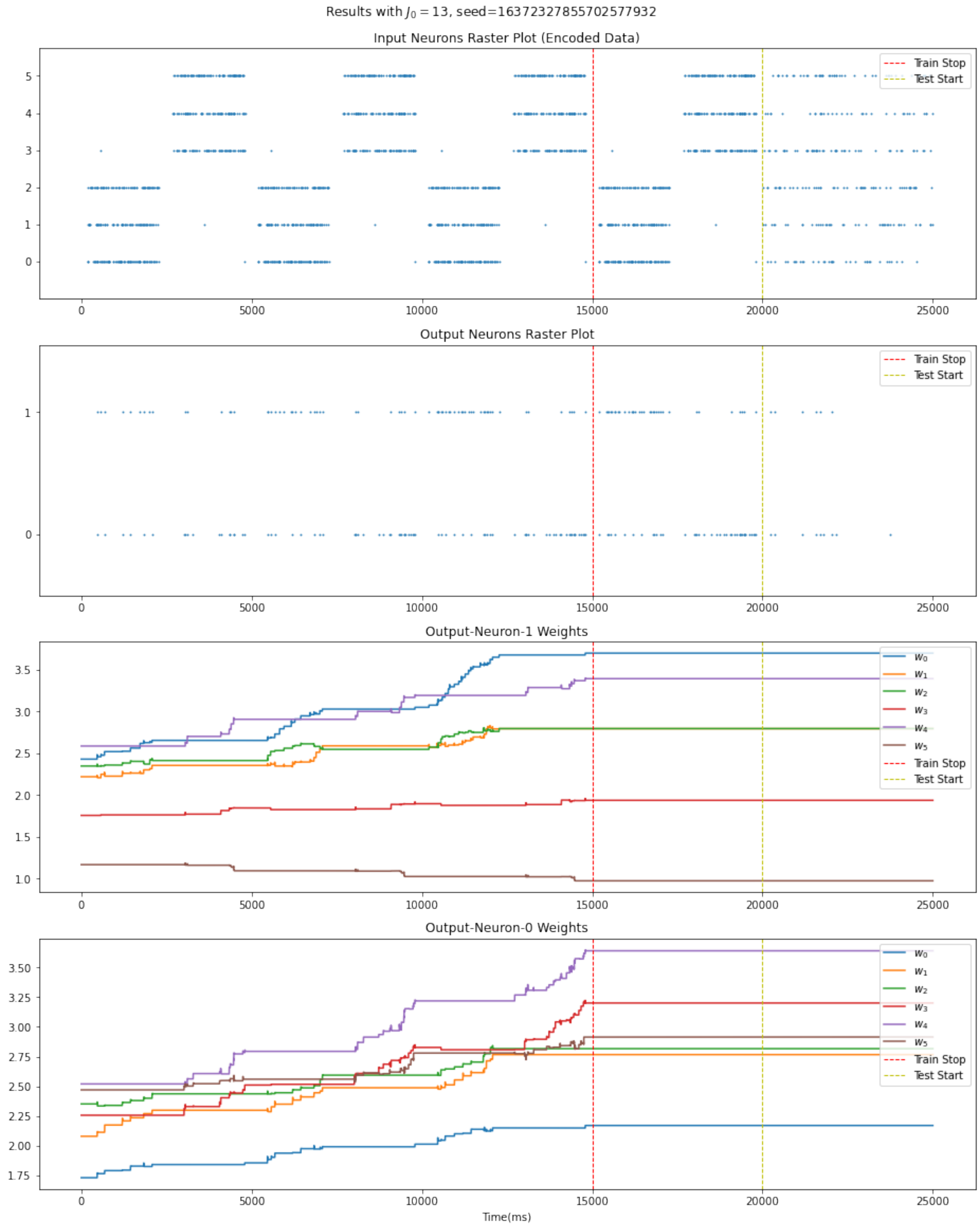

Results with $J_0 = 11$, seed=6626393261193957152
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Train Stop
Test Start
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Time(ms)



Results with $J_0 = 13$, seed=6626393261193957152
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)


Results with $J_0 = 9$, seed=16372327855702577932
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)


Results with $J_0 = 11$, seed=16372327855702577932
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)



From the above plots we conclude the following:

1. If the initial weights are too low (all less than 2), the network activation will be too low to let the training process start. This is due to the fact that STDP is based on neurons' activations; so, without any activation the training process cannot happen, and the network will not **learn** the input patterns.

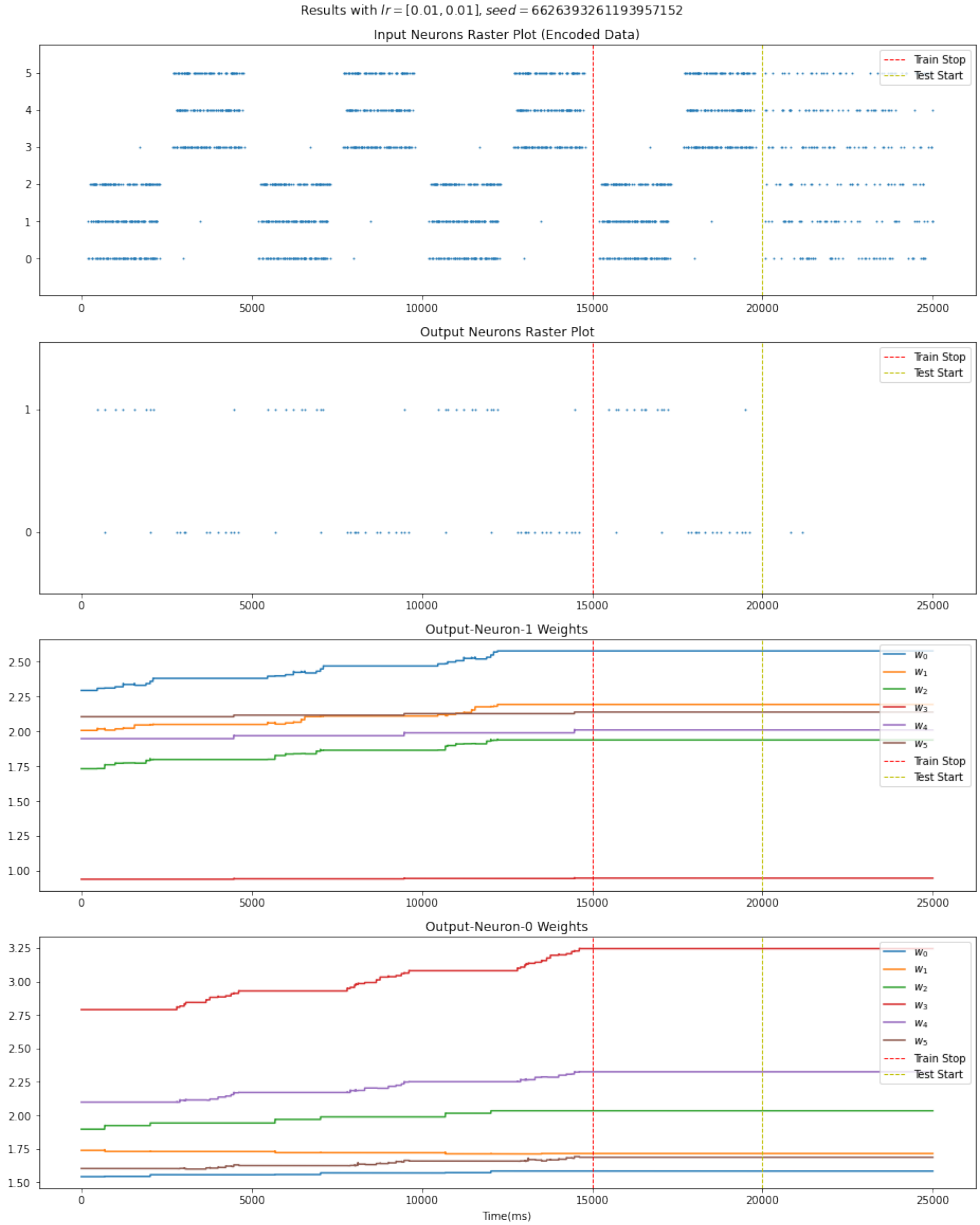
2. If the initial weights are too high, the network initial activation is too high; therefore, the connections' weights tend to increase with any input not specifically with the patterns. In other words, high initial

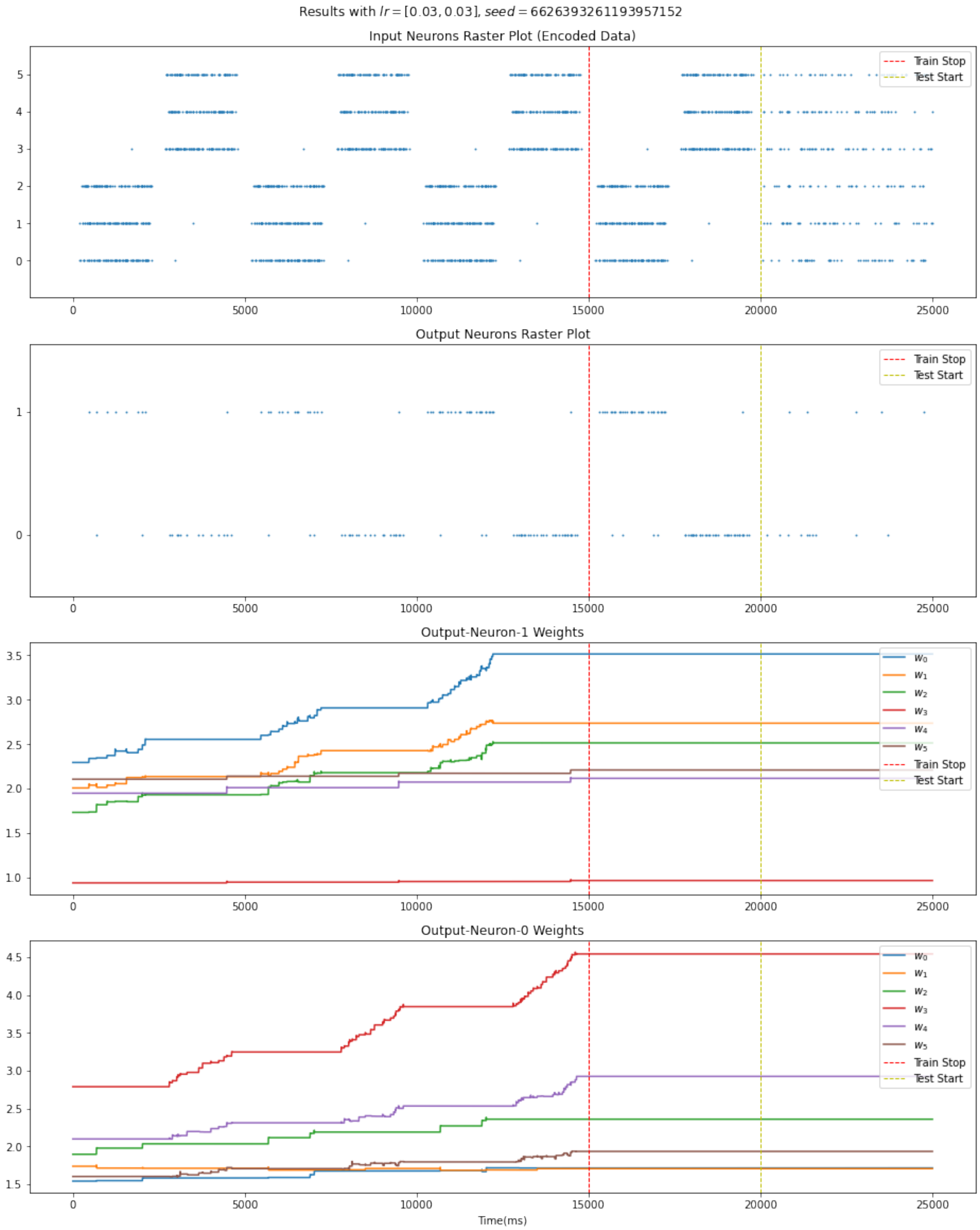
weights have a bad effect on network's sensitivity on the input patterns. Also, the network activity tends to increase with unseen data when value of J_0 is high.

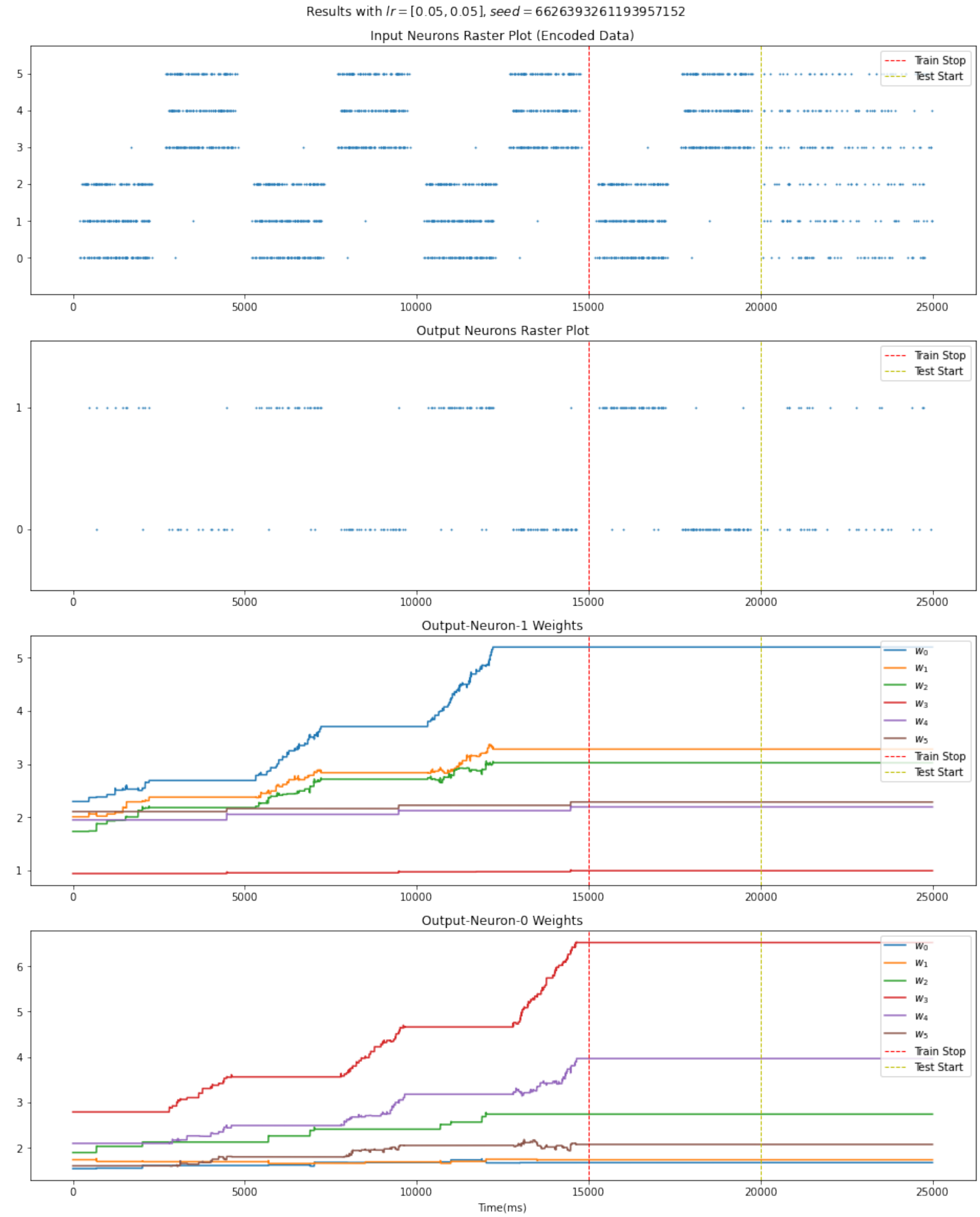
3. When we set $J_0 = 11$, the learning process is perfectly visible in the network's output pattern. With the second random seed, the first neuron is learning the second pattern, and the second neuron is learning the first input pattern. With the first random seed, the second neuron is learning both patterns, but with more concentration on the second pattern. The first neuron is slowly learning the first pattern only.

2.3 Experiment #2 (Learning Rate)

2.3.1 Equal Learning Rates




Results with $lr = [0.03, 0.03]$, $seed = 6626393261193957152$
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Train Stop
Test Start
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Time(ms)

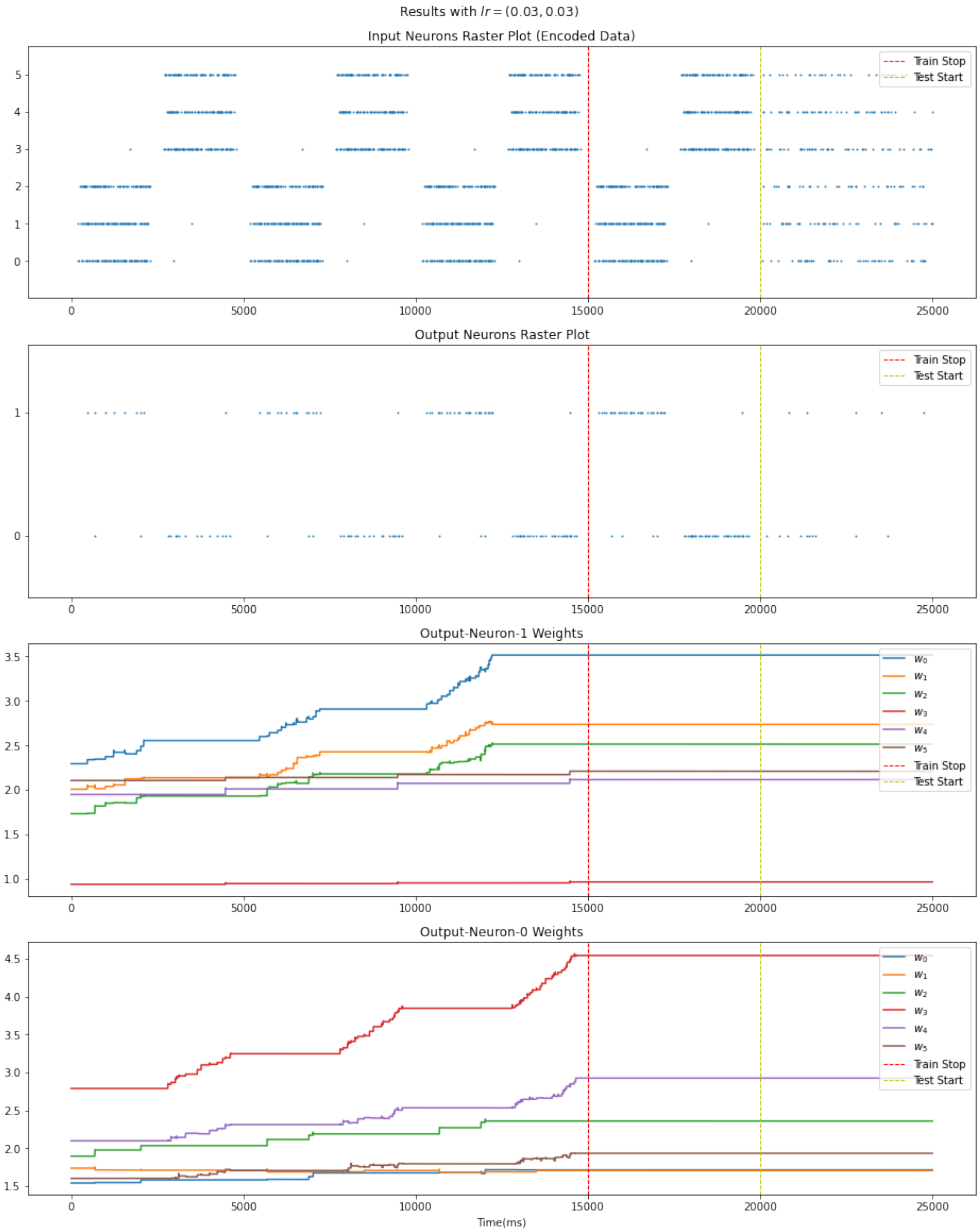


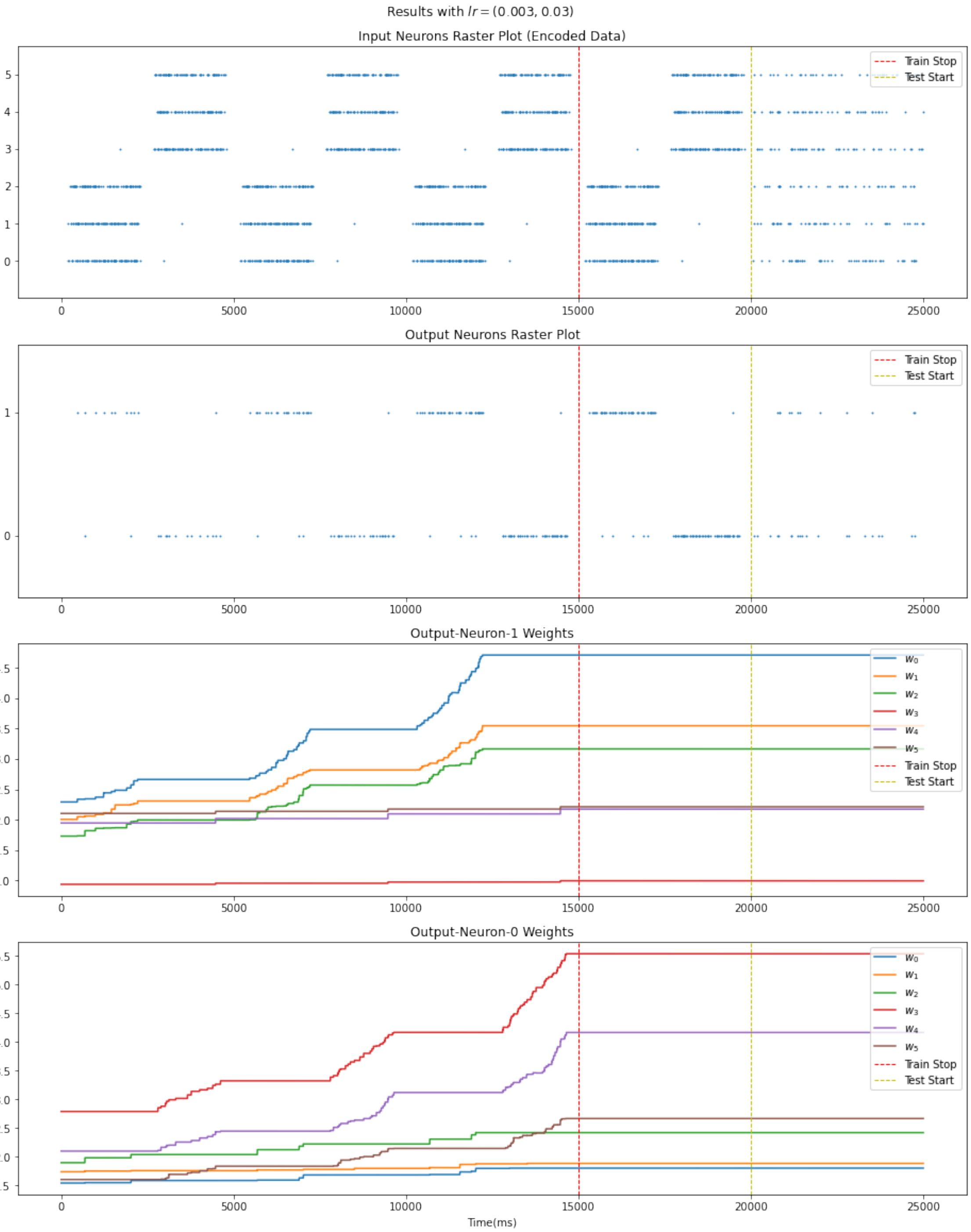
As expected, the higher the learning-rates, the faster the training occurs. The learning rate could be seen in the slope of weights' plots. By increasing the learning rates, the slope of weight-plots tend to increase.

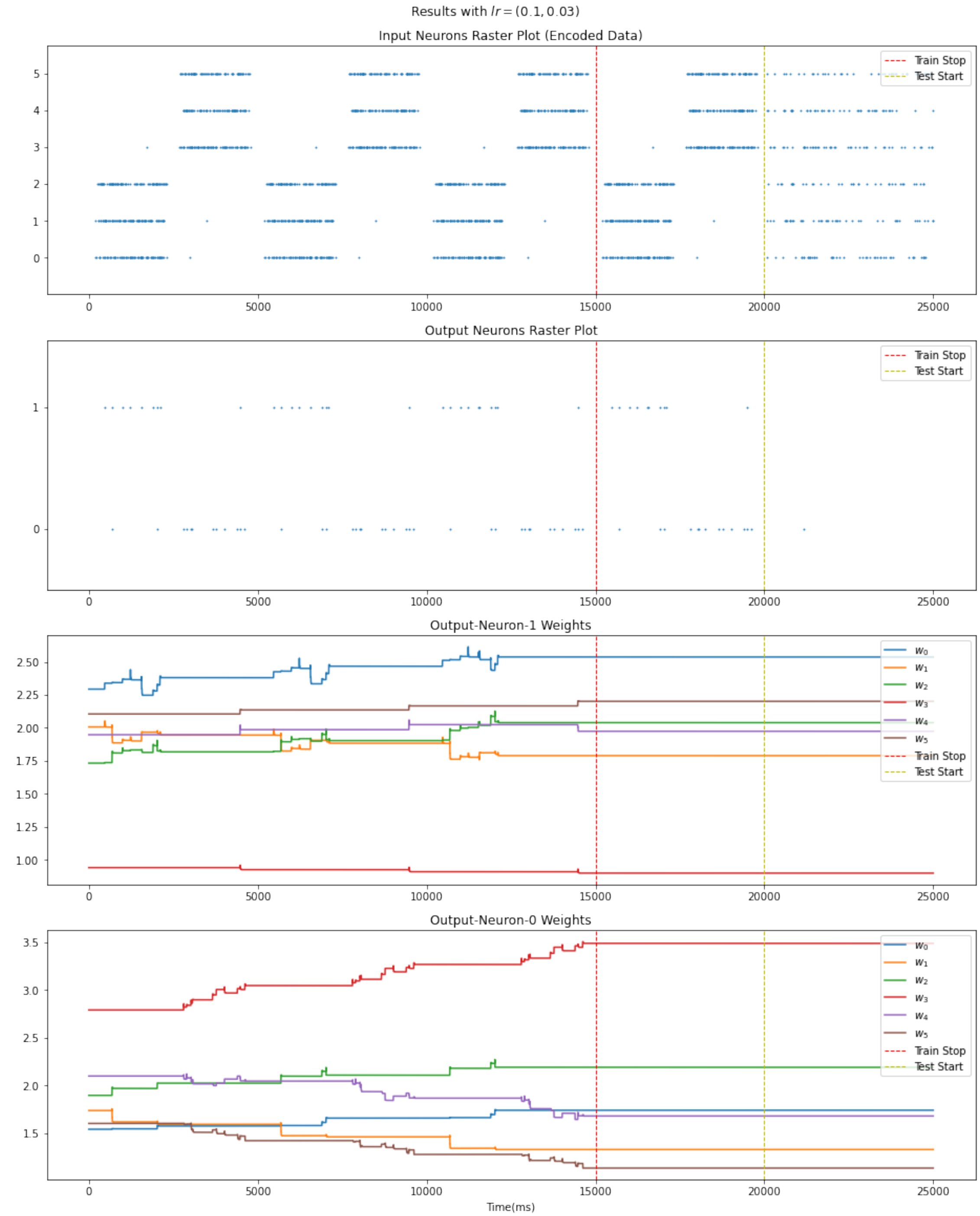
2.3.2 Unequal Learning Rates

In this case, to make the results more visible we only demonstrate extreme inequality between learning rates.

The first plot is drawn to have a baseline for comparison.


Results with $lr = (0.03, 0.03)$
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Output-Neuron-0 Weights
Time(ms)

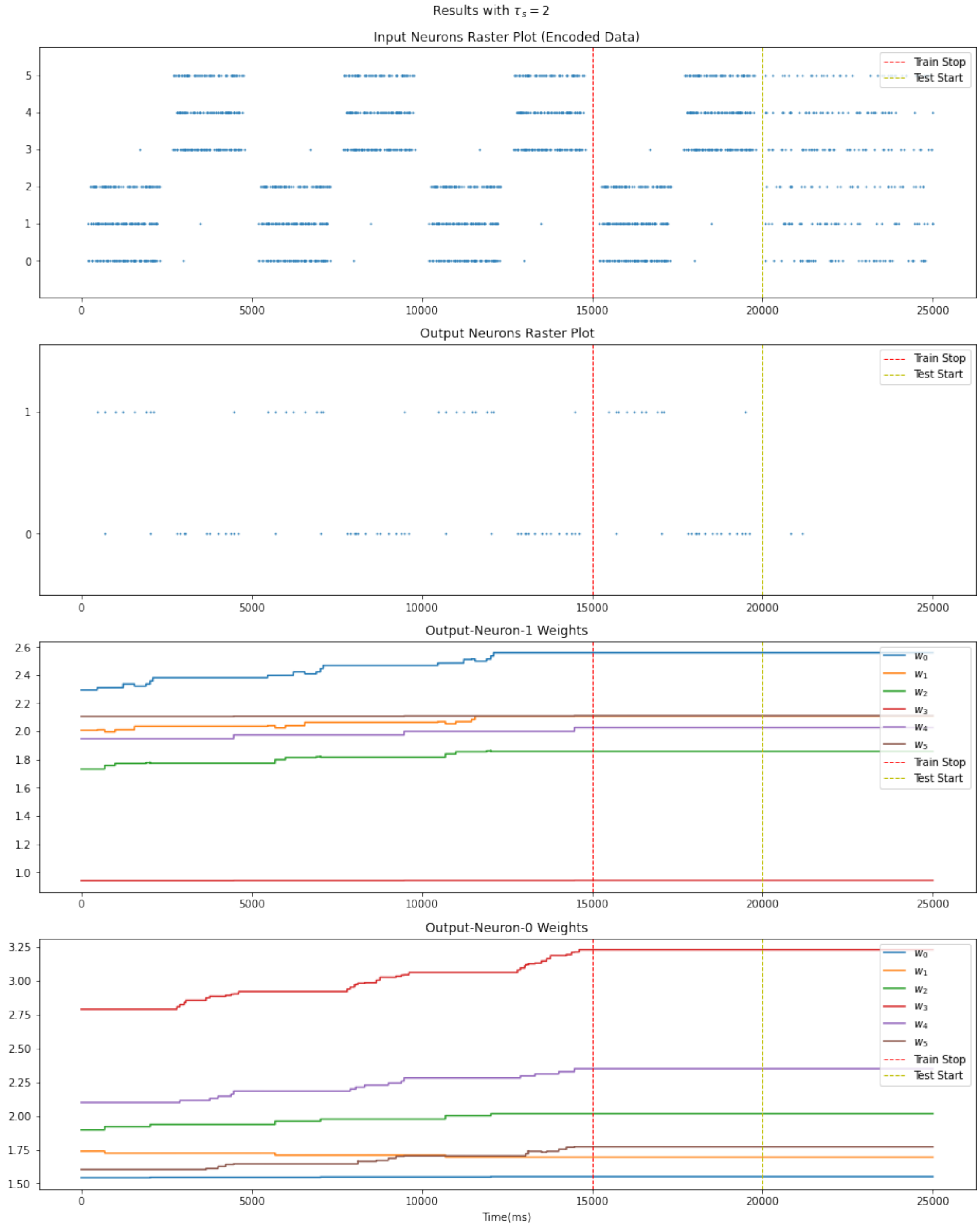

Results with $lr = (0.003, 0.03)$
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)

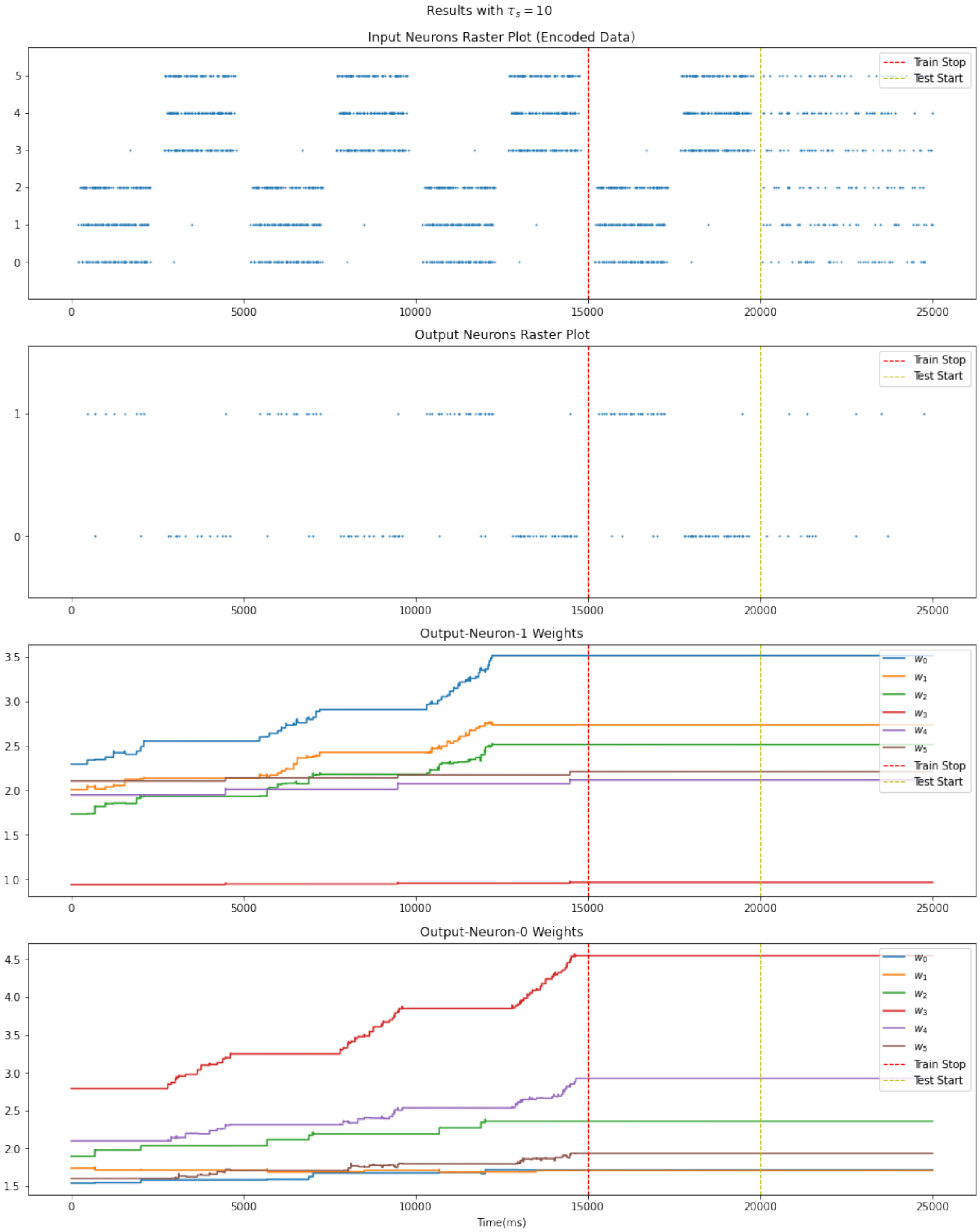


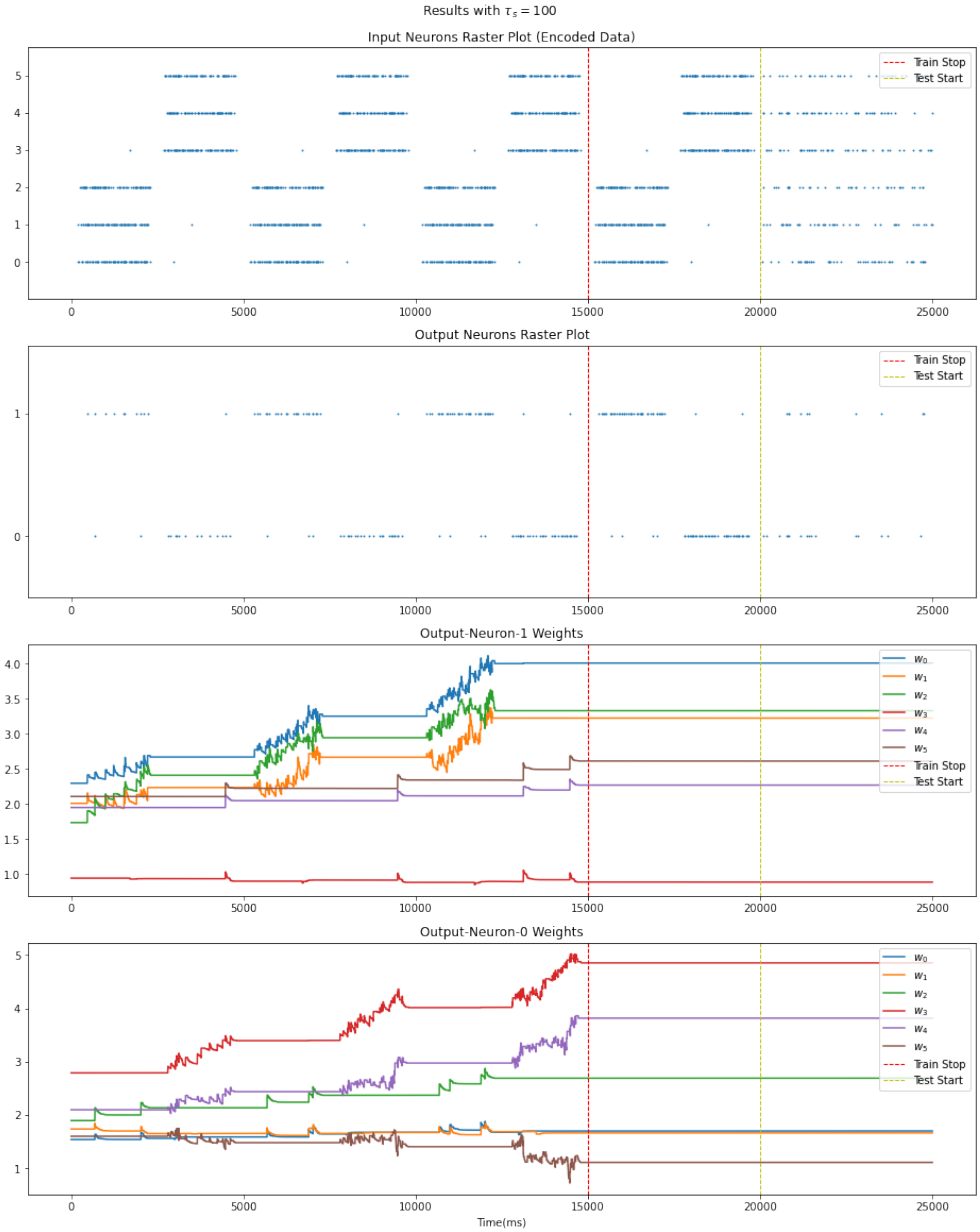
If A_+ is greater than A_- by a large margin, the weights tend to increase with higher slope, and they will not be corrected sufficiently when correction is needed (when the pre-synaptic neuron spikes after the post-synaptic neuron). As a result, the network's activation with random input increases as well. This is visible in the network output in $[20s, 25s]$ section of the plots. In contrast, if A_- is greater than A_+ by a large margin, the network only learns the desired patterns as non-ideal activations will be suppressed. This finding hints us to use different learning rates with A_- being a little higher ($e.g. A_- = 0.05, A_+ = 0.03$). But we should take it into account that if the difference between the two learning rates becomes very high, the network will lose its ability to learn completely; either its weights suddenly saturate at the maximum value, or the weights decay to zero.

2.4 Experiment #3 (τ_s)

Low amounts of τ_s makes the weight updates more local in time. *i.e.*, the spikes that happened in a smaller time period affect the weights. This is the opposite for high values of τ_s. Intuitively it is better to use relatively small values for τ_s since STDP should use short time periods for updating the weights. If τ_s becomes too high, the learning process becomes problematic since very old spikes change the weights. This is mostly destructive with non-wanted spikes (post-synaptic neuron spikes before pre-synaptic neuron).


Results with $\tau_s = 2$
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Train Stop
Test Start
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Time(ms)


Results with $\tau_s = 10$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Time(ms)
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5



As stated above lower τ_s, performs more reasonably as only spikes that occurred very recently affects the learning. More reasonable means that the network is not responding to random input in the testing phase. But notice that with $\tau_s = 2$, the total amount of output neurons' activation has become lowered. This is because the weight updates are happening more slowly; therefore, the learning takes longer time.

Chapter 3

Flat-STDP

To implement Flat-STDP, we binarized the spike-traces with a fixed threshold of 0.05. If the traces are greater than 0.05, they will be set to 1 and 0 otherwise.

3.1 Default Parameters

Train Params:

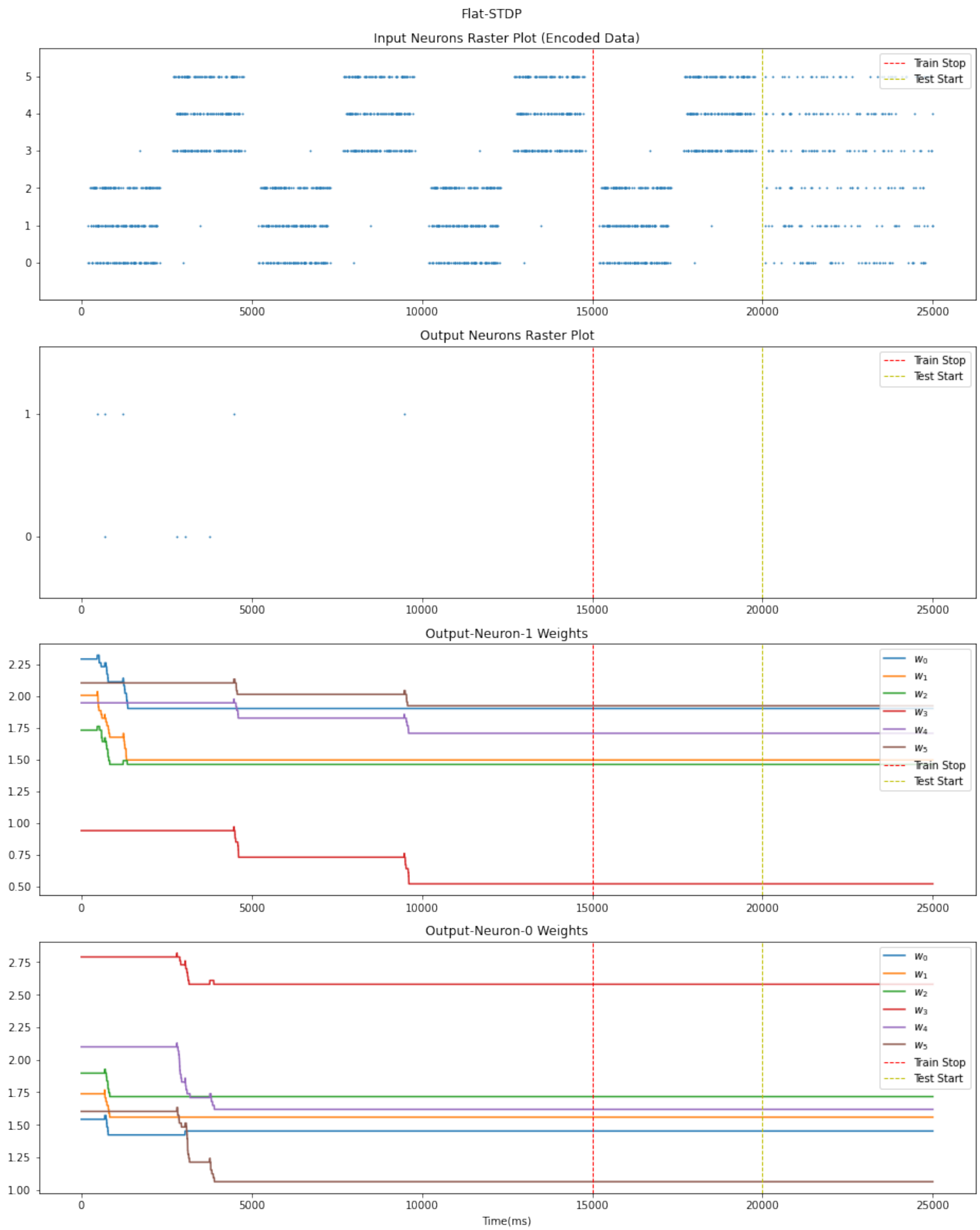
$Time_{simulation} = 25000ms$
$LearningRates = [0.025, 0.075]$

Neurons Params:

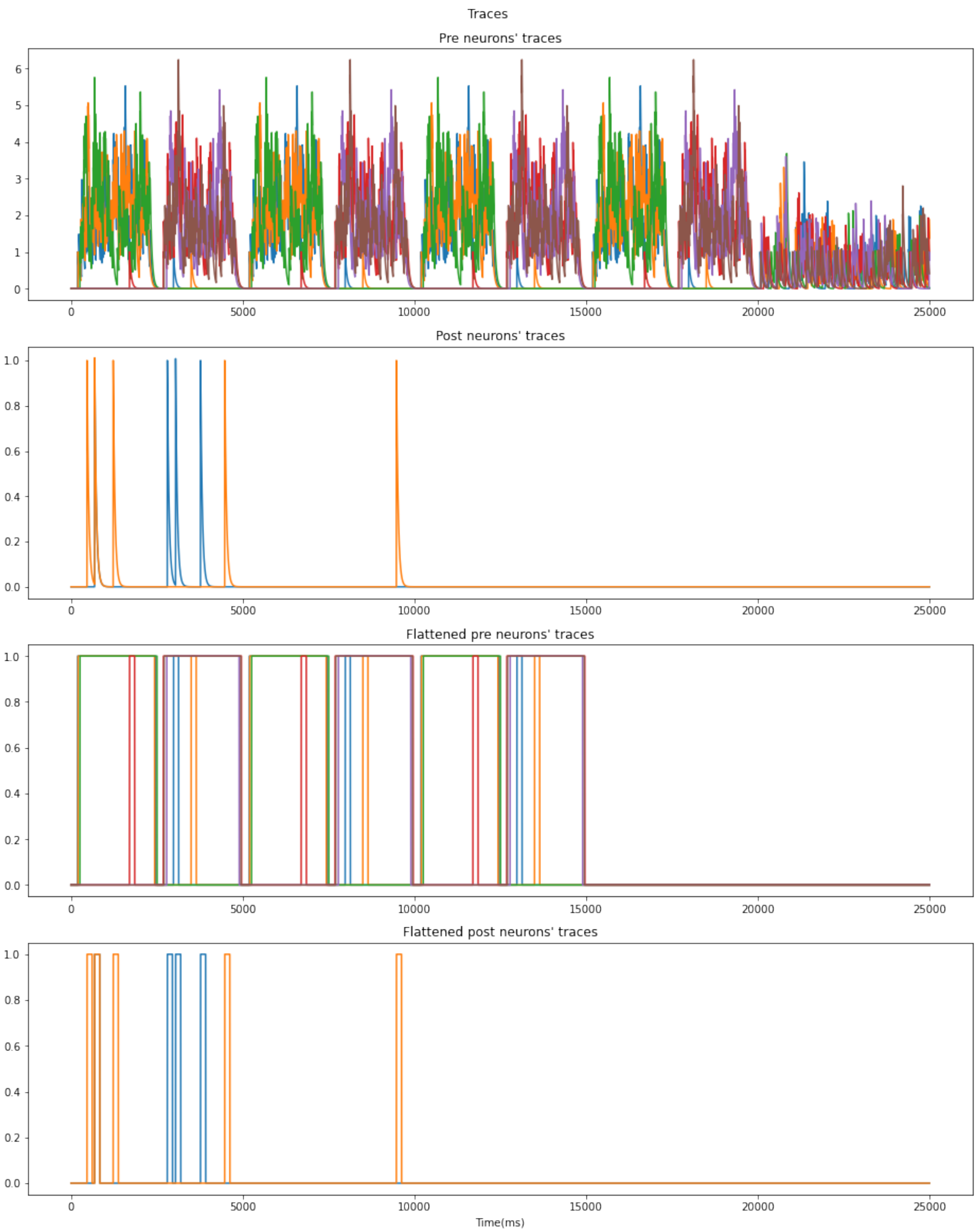
$Num(PresynapticNeurons) = 6$
$\tau_s = 10ms$
$Threshold = -52mv$
$U_{rest} = -60mv$

Connection Params:

$J_0 = 11$
$\sigma_0 = 2.5$
$Weight_{min} = 0$
$Weight_{max} = 7.5$

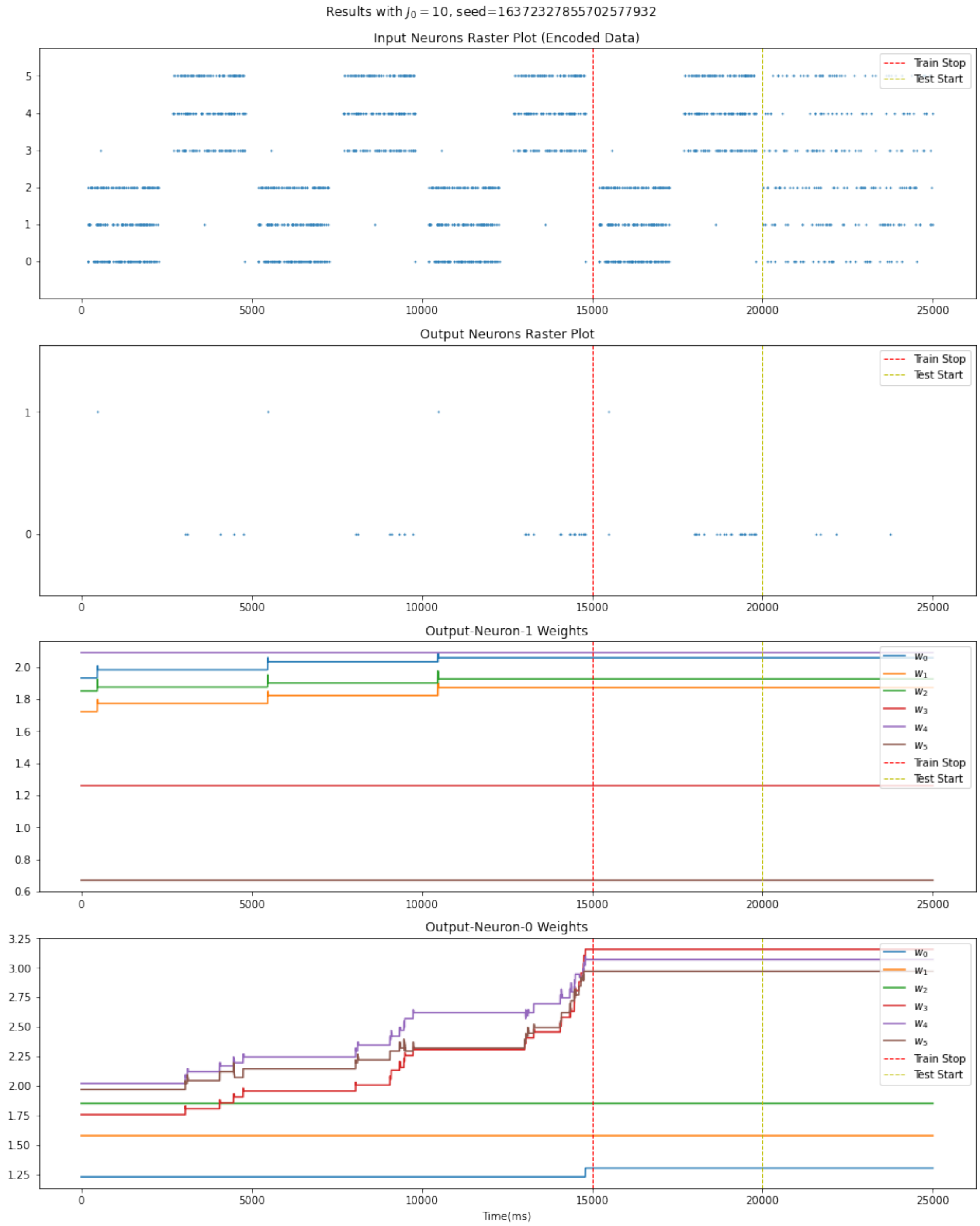


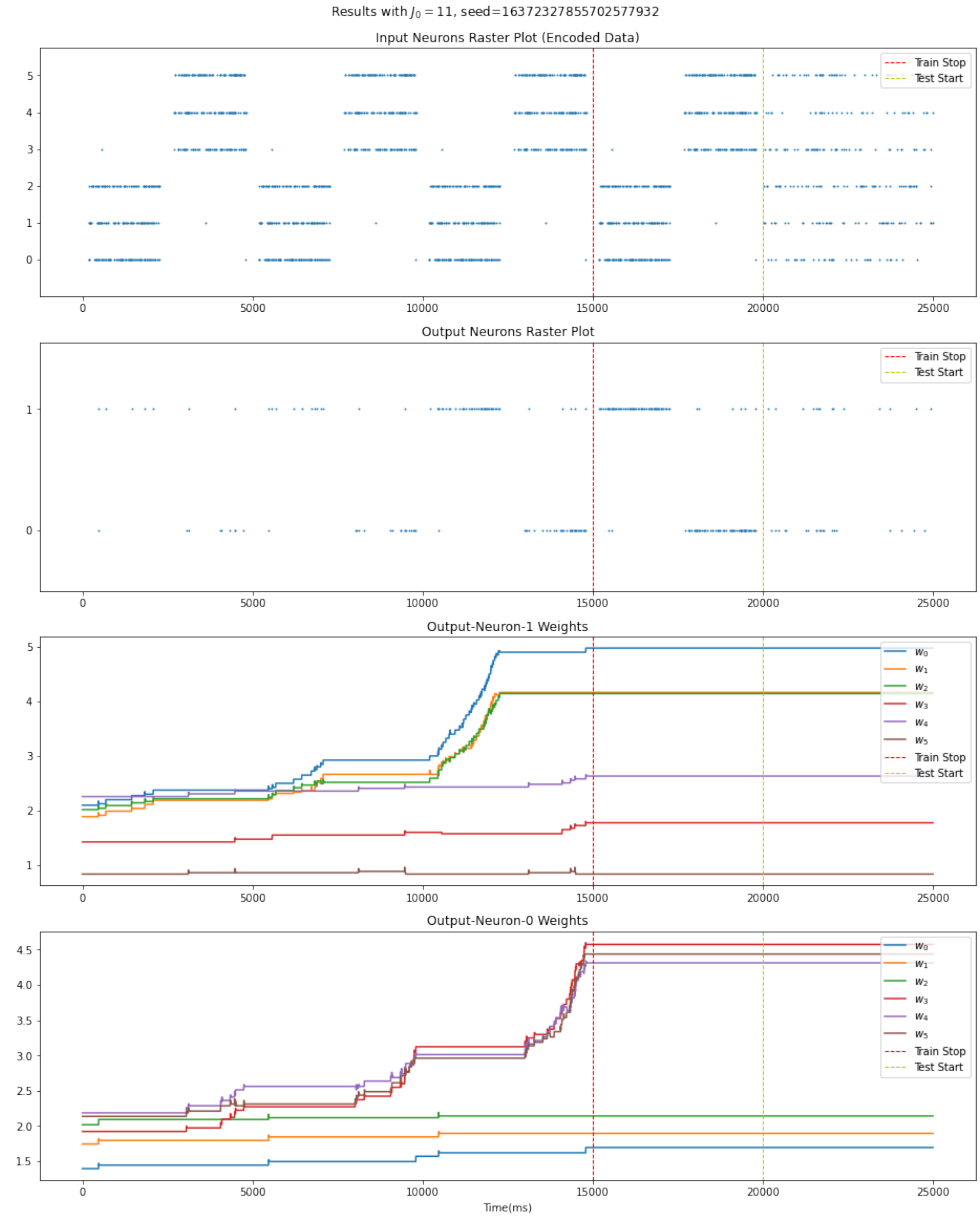
Flat-STDP
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Output-Neuron-0 Weights
Time(ms)



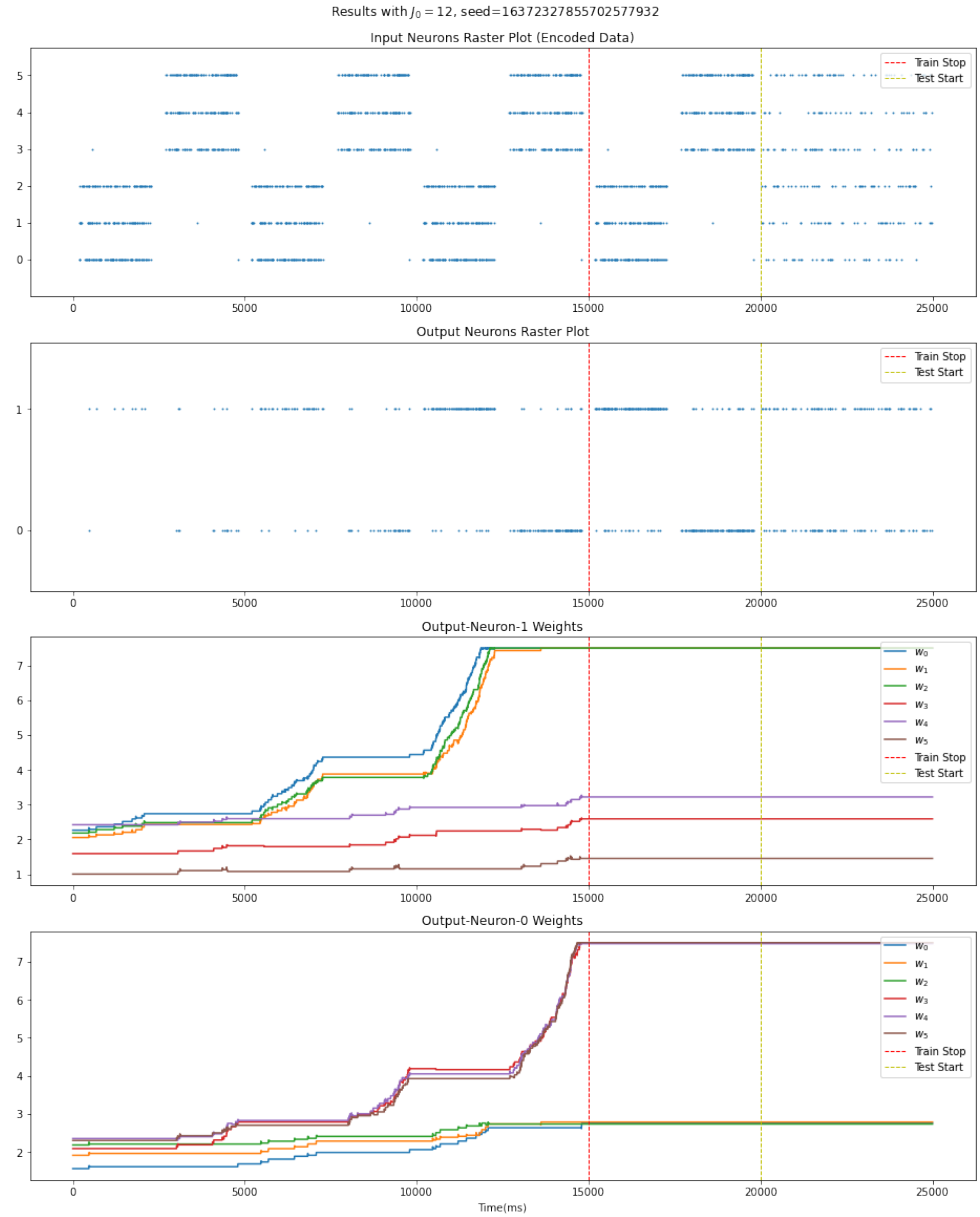
In the above plots we have set $\tau_s = 50$ to make the changes in the traces more visible. Due to the fact that the traces are now flattened and τ_s is increased compared with the previous experiments, the weights are decreasing instead of increasing, so the network is not learning anything. In the following experiments we don't plot the traces anymore and better parameters for the network.

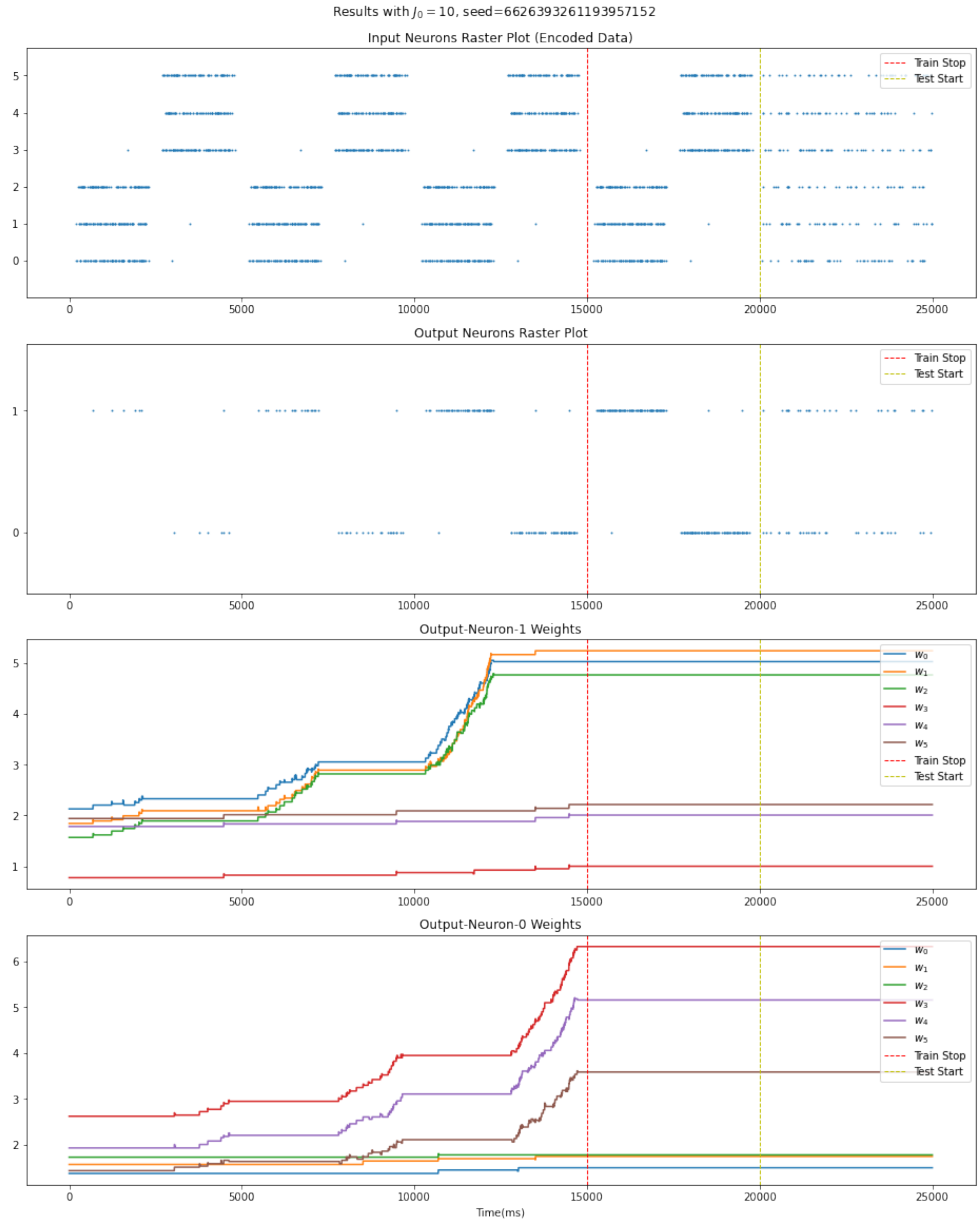
3.2 Experiment #1 (Initial Weights)



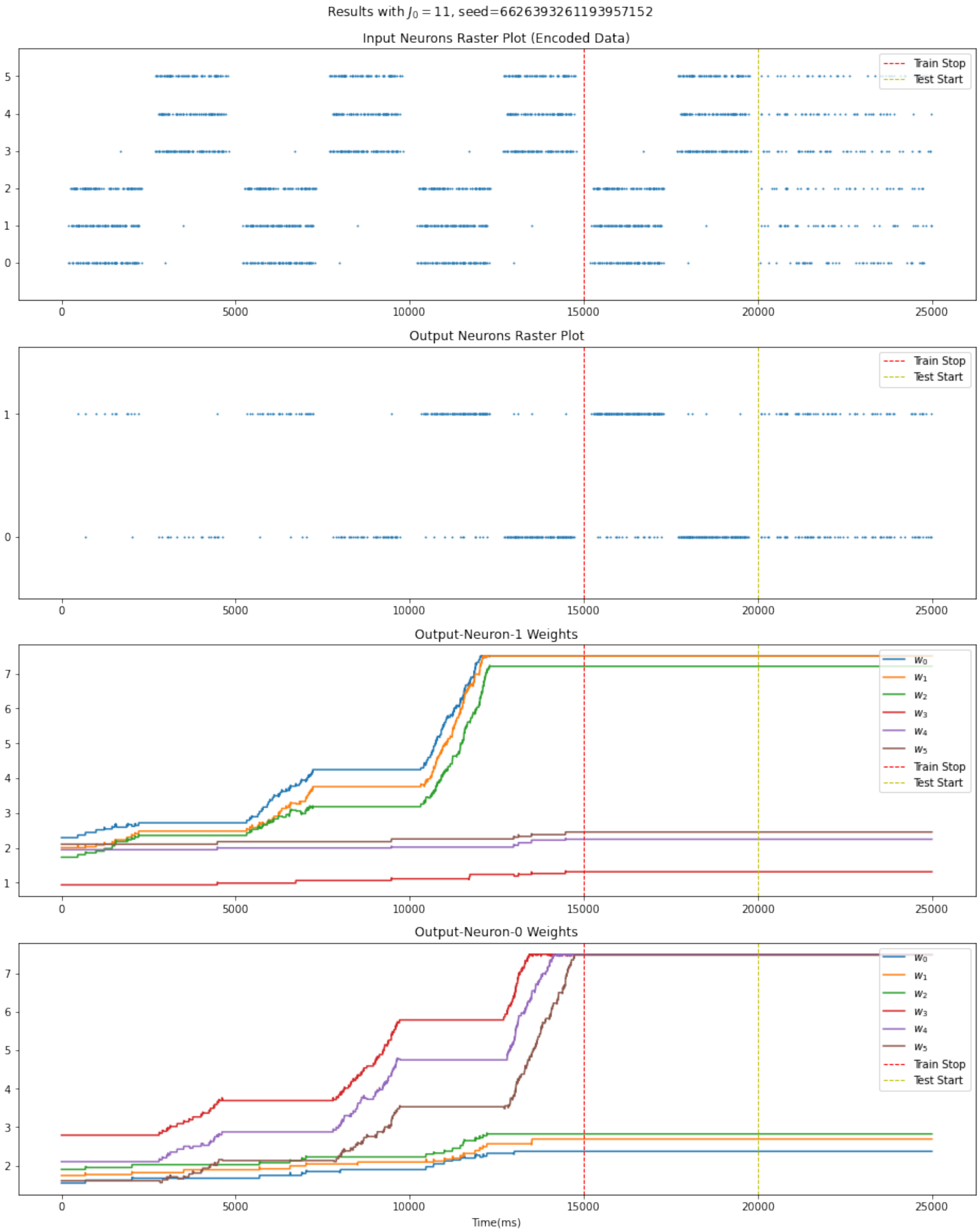


Results with $J_0 = 11$, seed=16372327855702577932
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Output-Neuron-0 Weights
Time(ms)

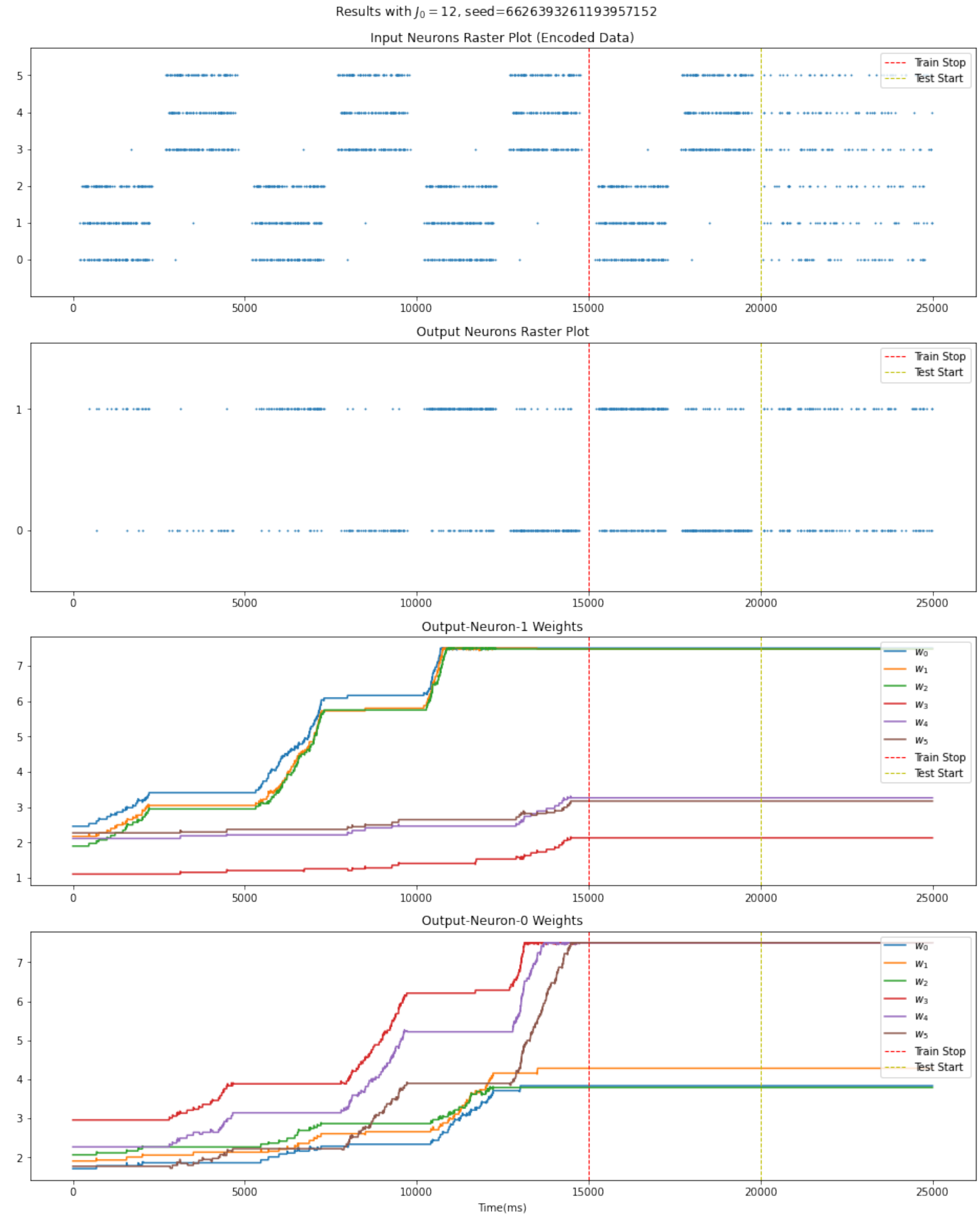

Results with $J_0 = 12$, seed=16372327855702577932
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Time(ms)



Results with $J_0 = 10$, seed=6626393261193957152
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
Time(ms)

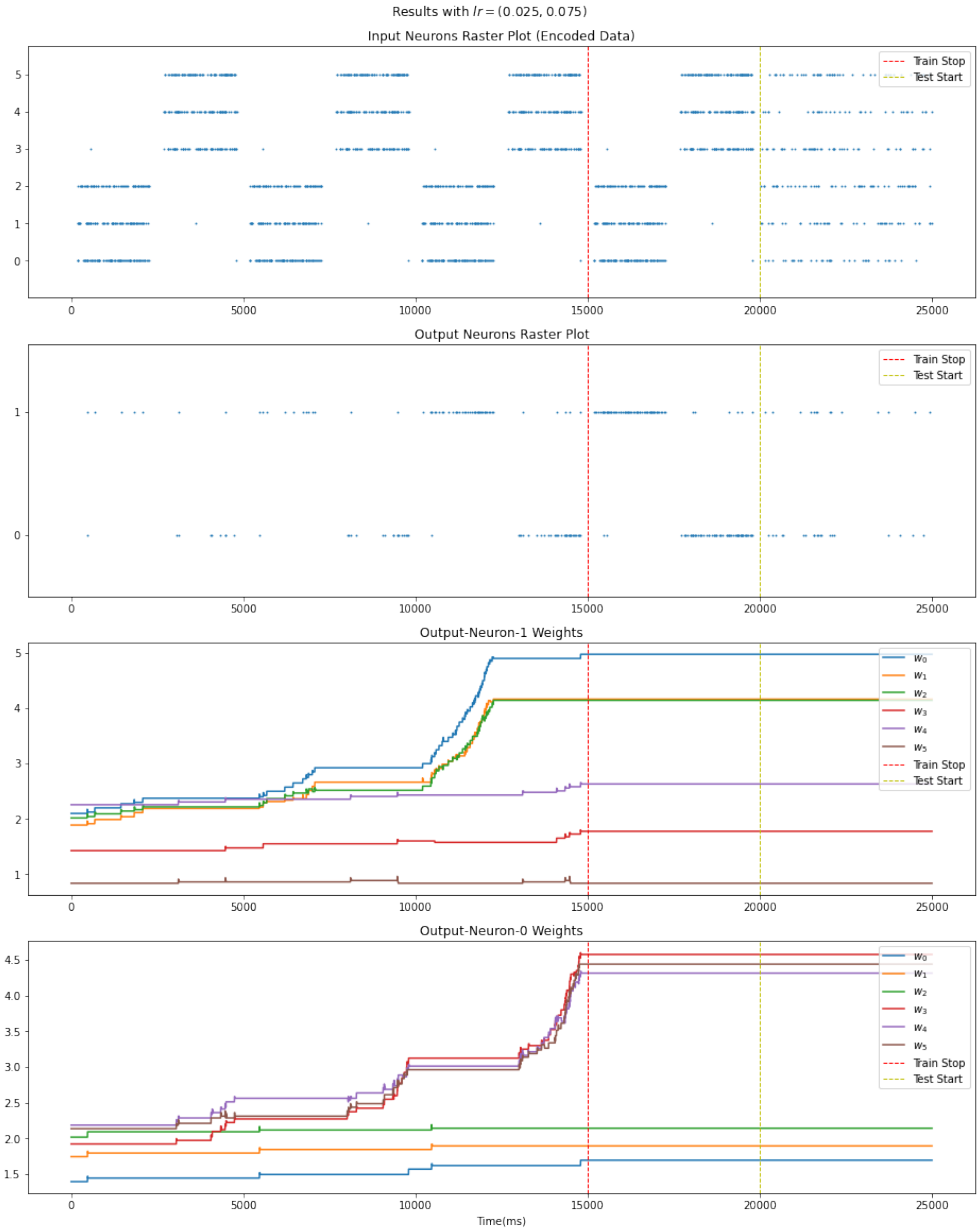


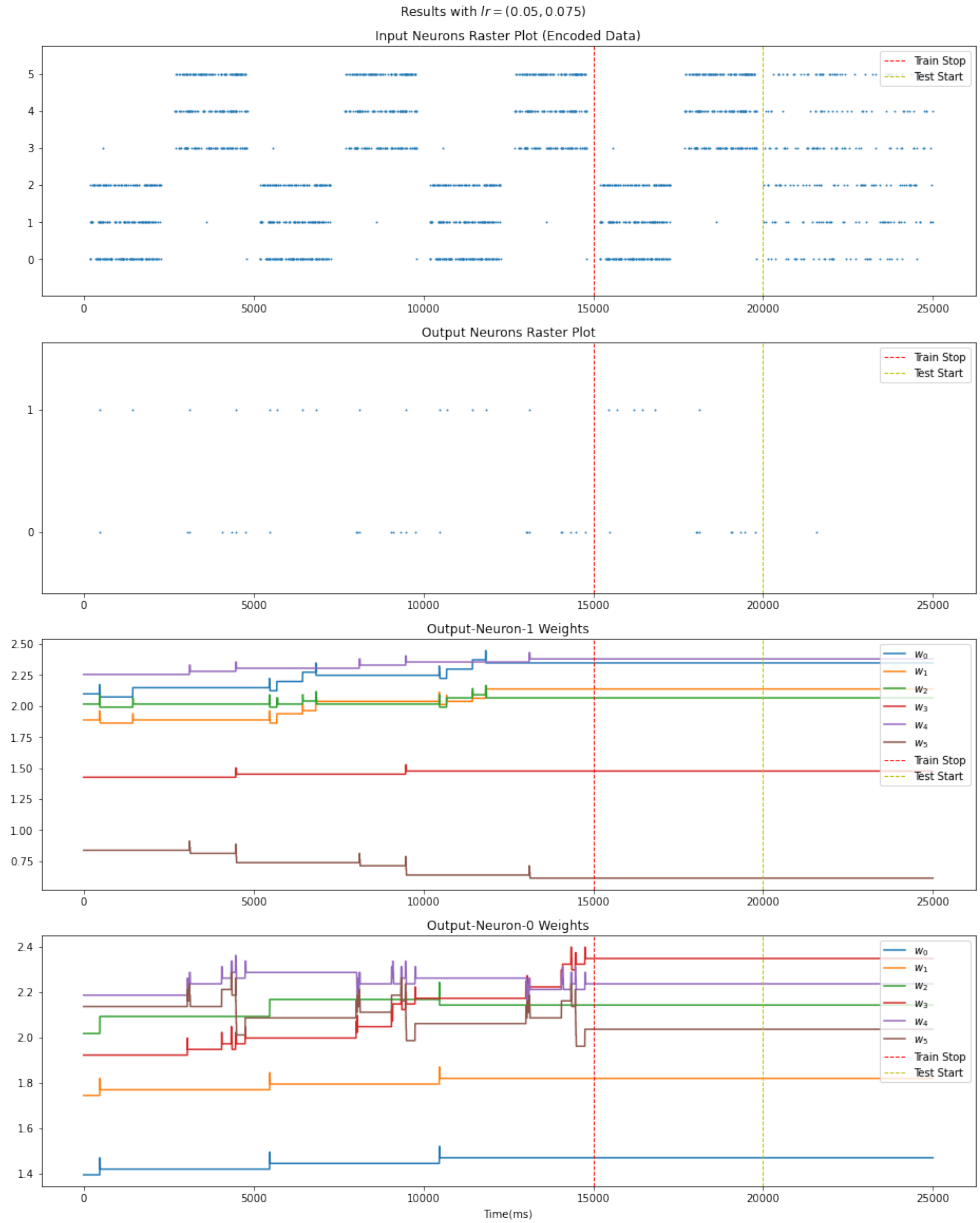
Results with $J_0 = 11$, seed=6626393261193957152
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Train Stop
Test Start
Time(ms)

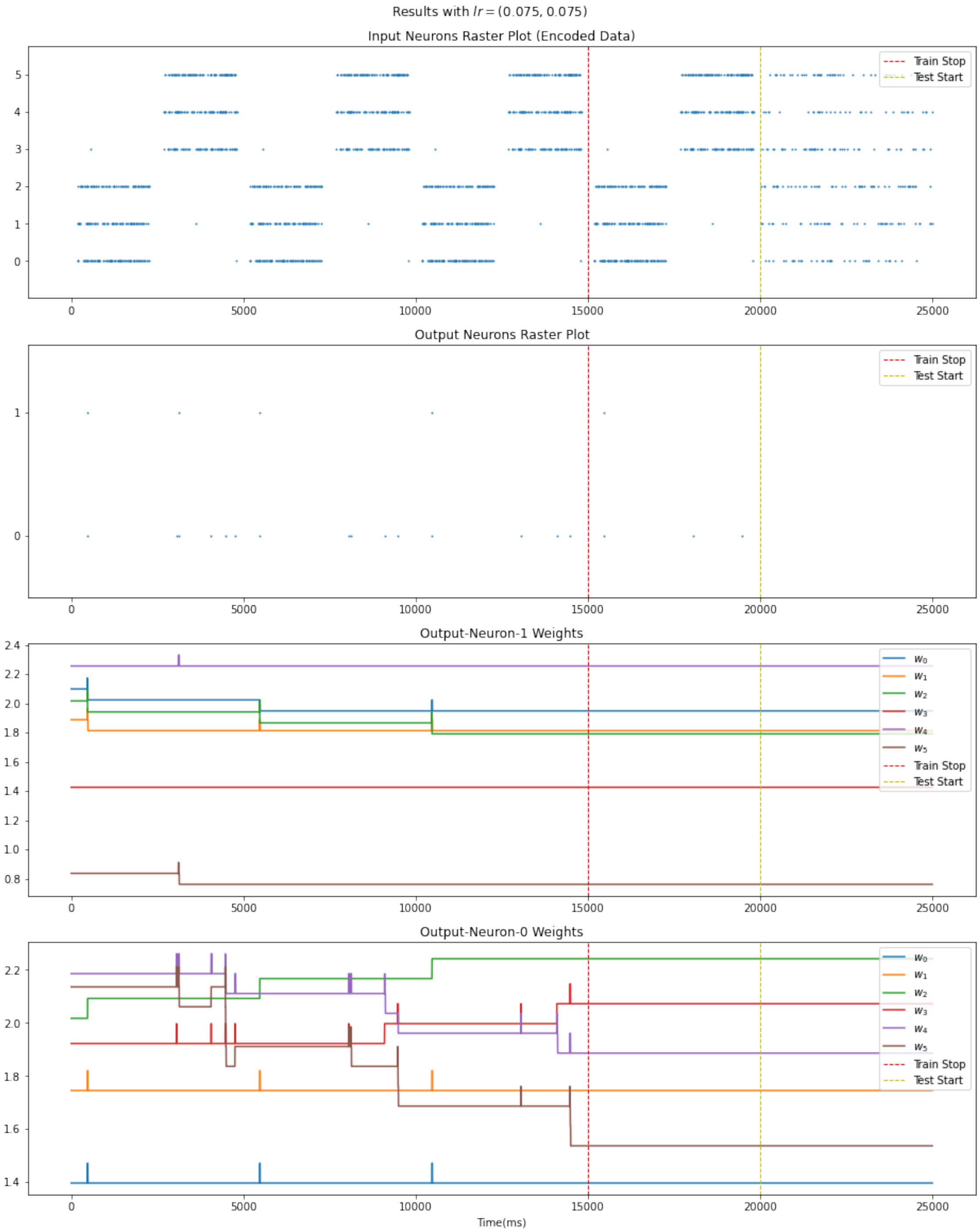


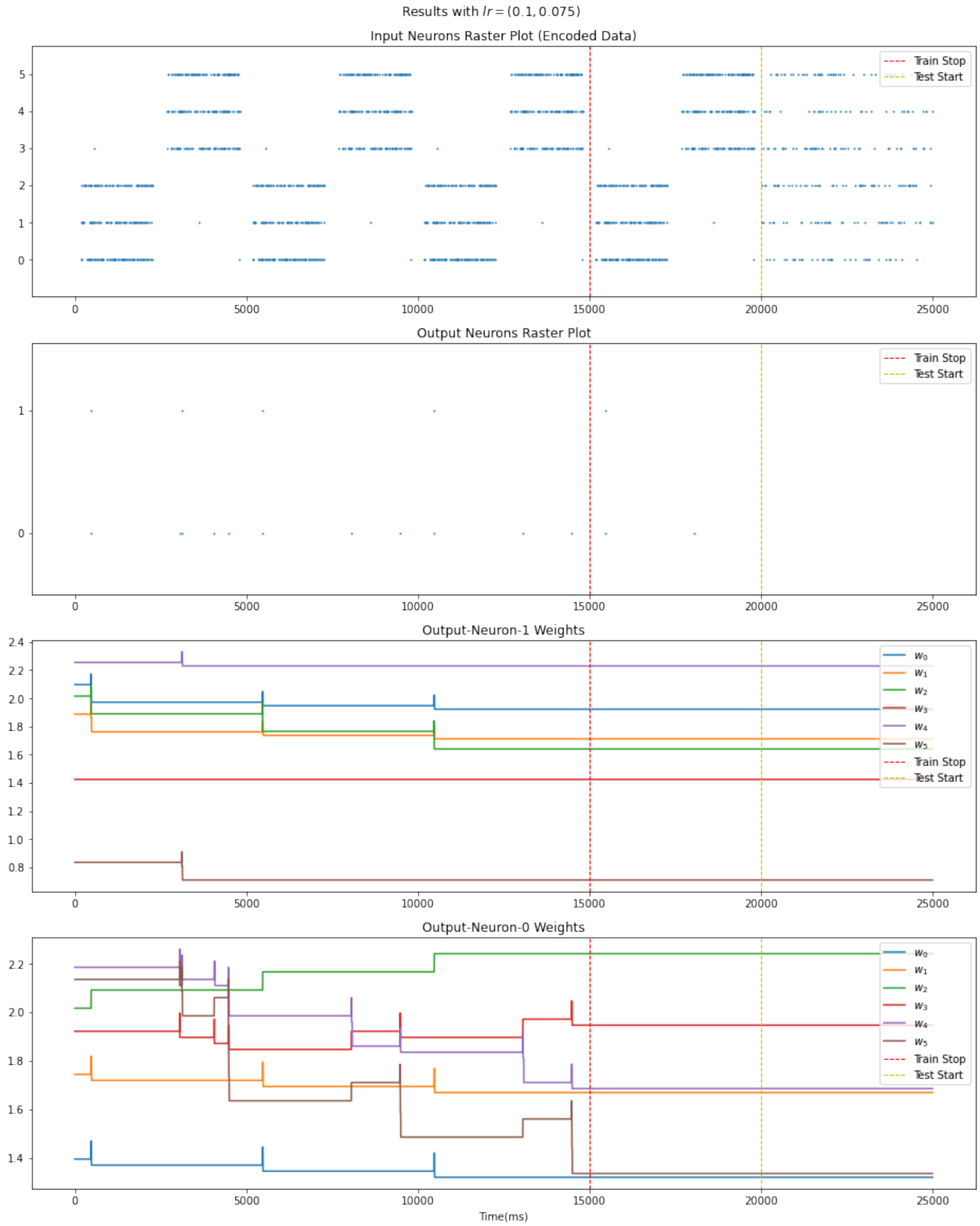
The conclusion of the above plots is the same as we made in STDP section.

3.3 Experiment #2 (Learning Rates)




Results with $lr = (0.05, 0.075)$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)

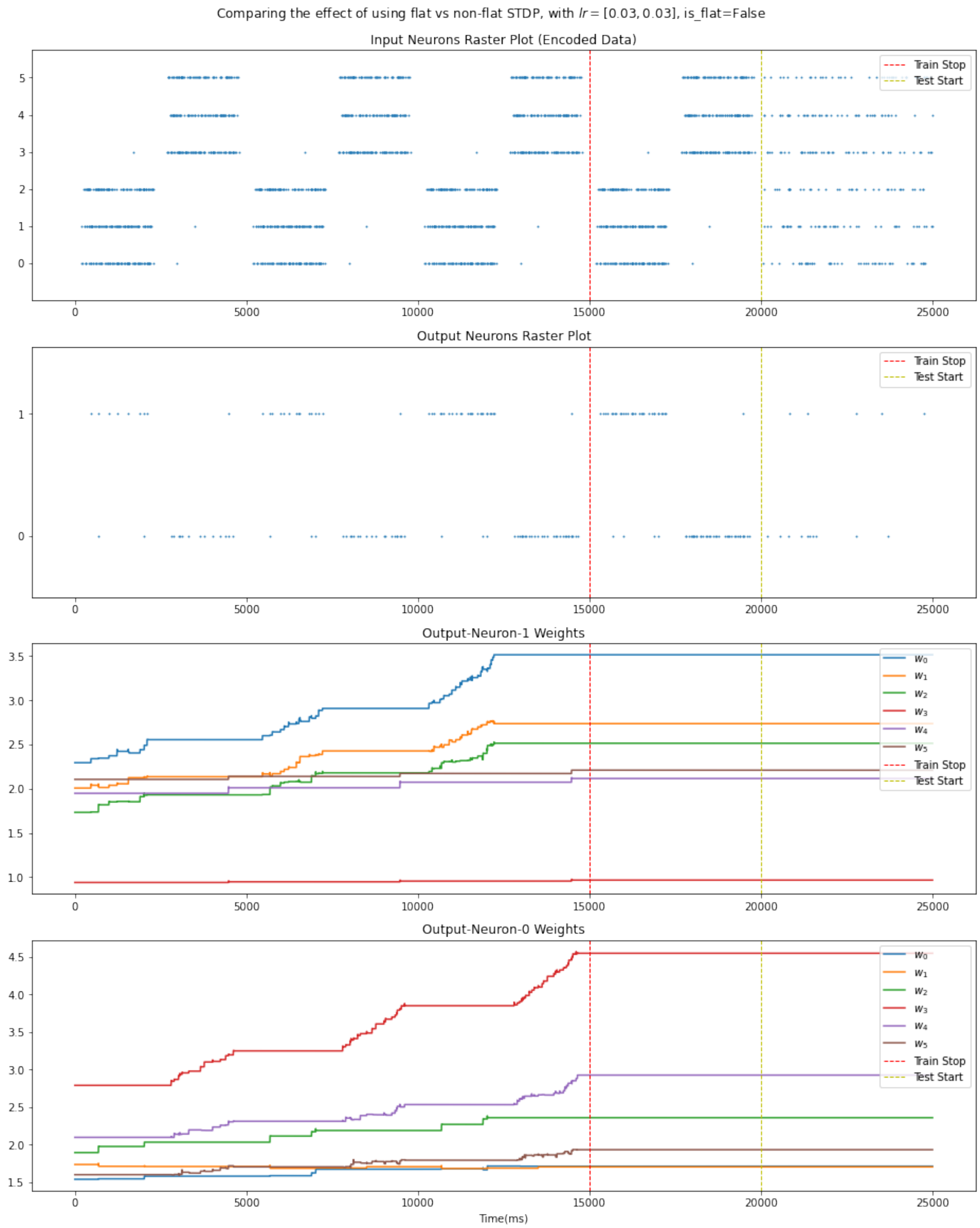

Results with $lr = (0.075, 0.075)$
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)

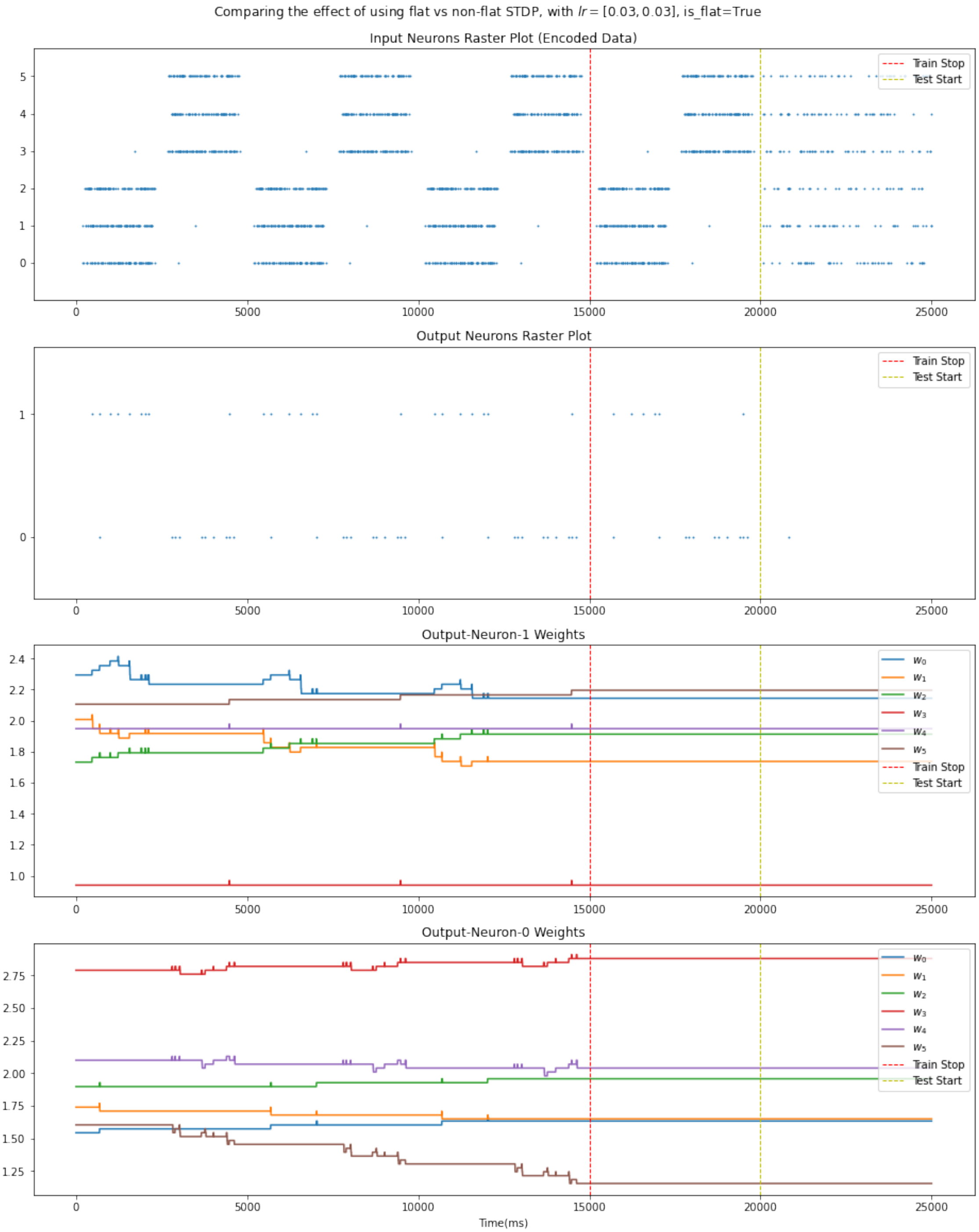


The difference between this experiment, and the one for STDP is that the same learning rates with the same input results in different learnt weights and different network activity, which is not unexpected. Another difference is that in STDP learning rule, it was better to keep A_- a little higher than A_+; but here, it is the opposite in most of the cases (random seeds). The reason is that higher A_- in Flat-STDP do not let the weights to be learned efficiently, and the amount of decrease in the weights would be higher than the amount of increase if the input's frequency is a bit high (like our inputs).

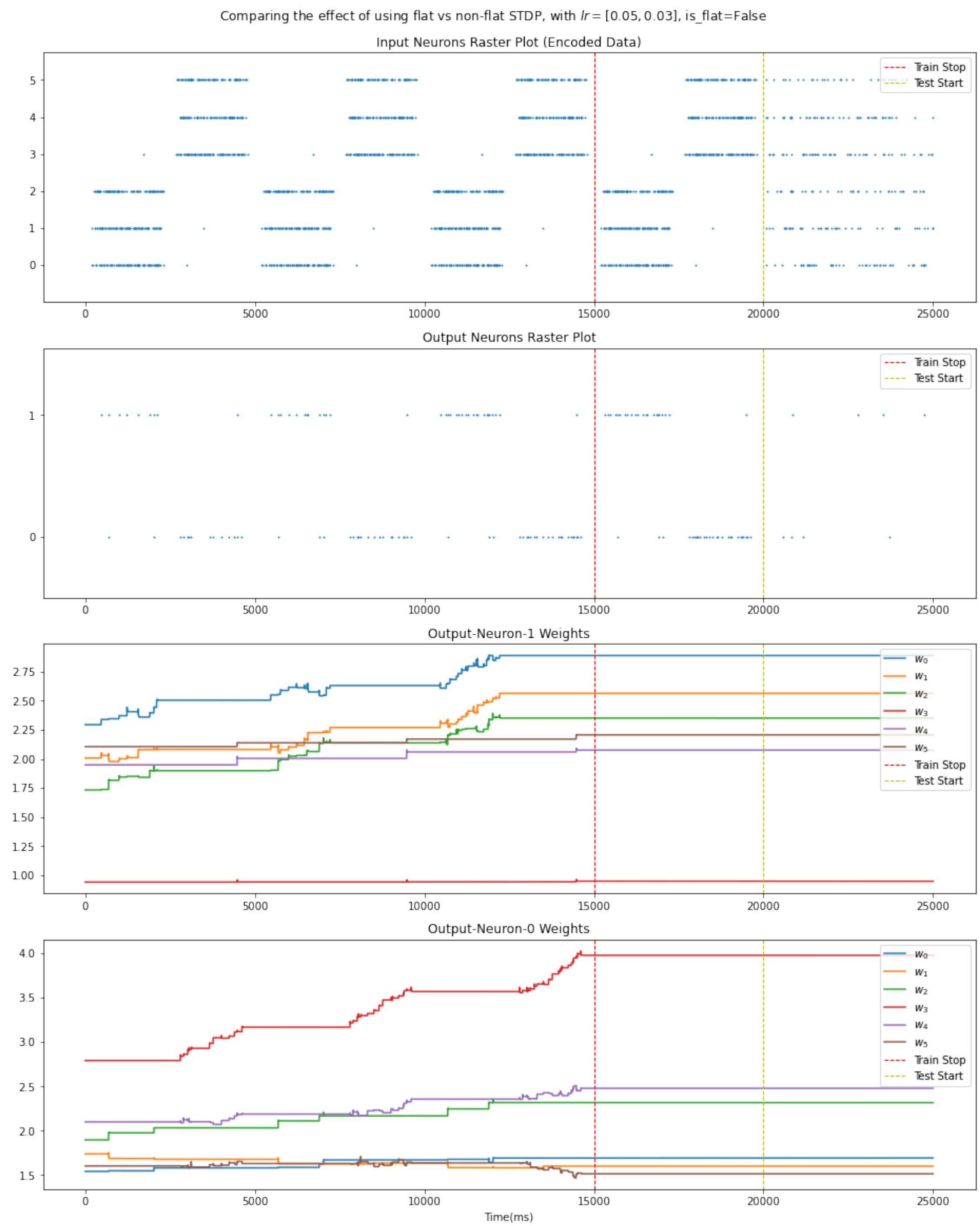
3.3.1 Comparing Flat-STDP with STDP

The below plot uses the same learning rates as we used for STDP.

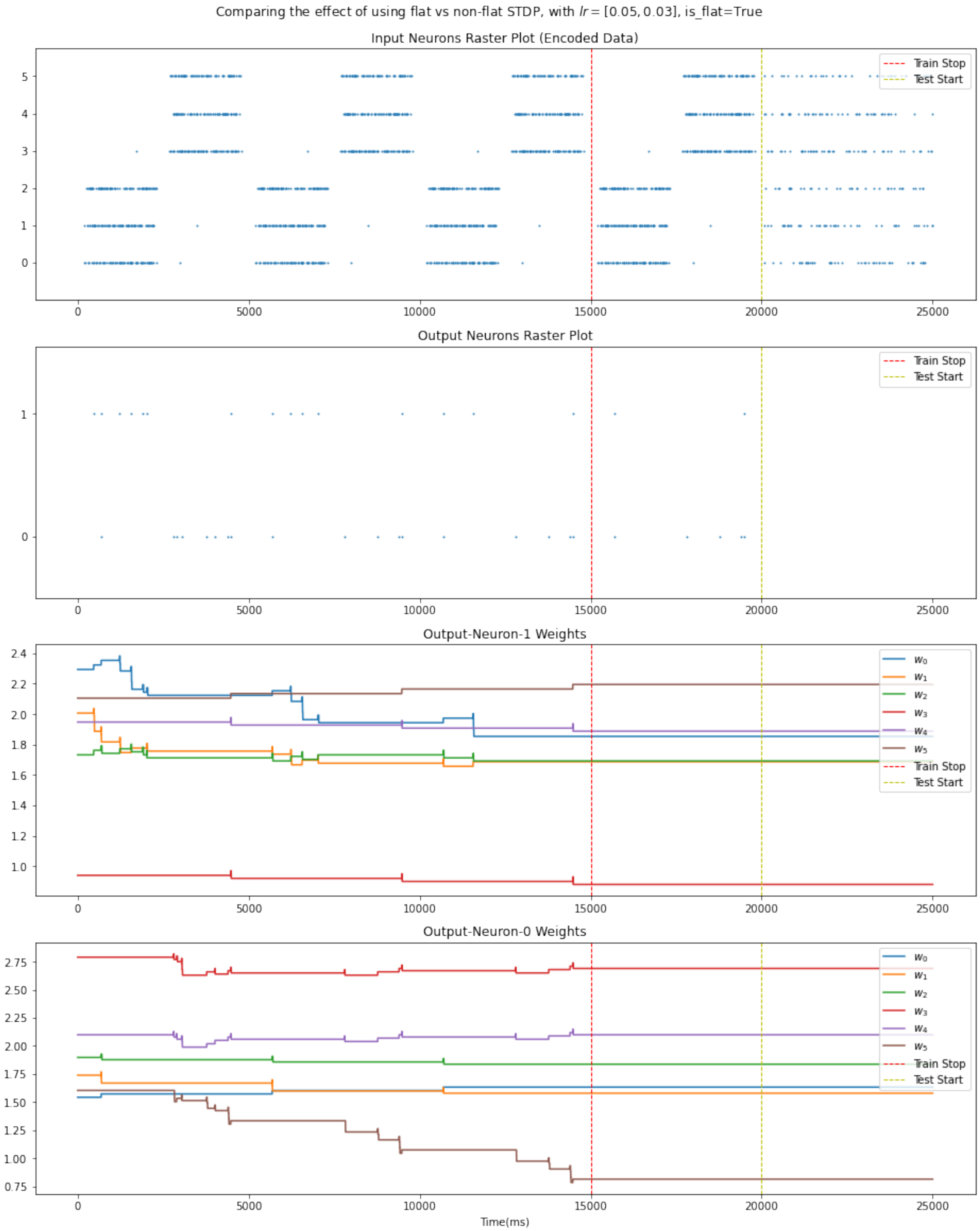




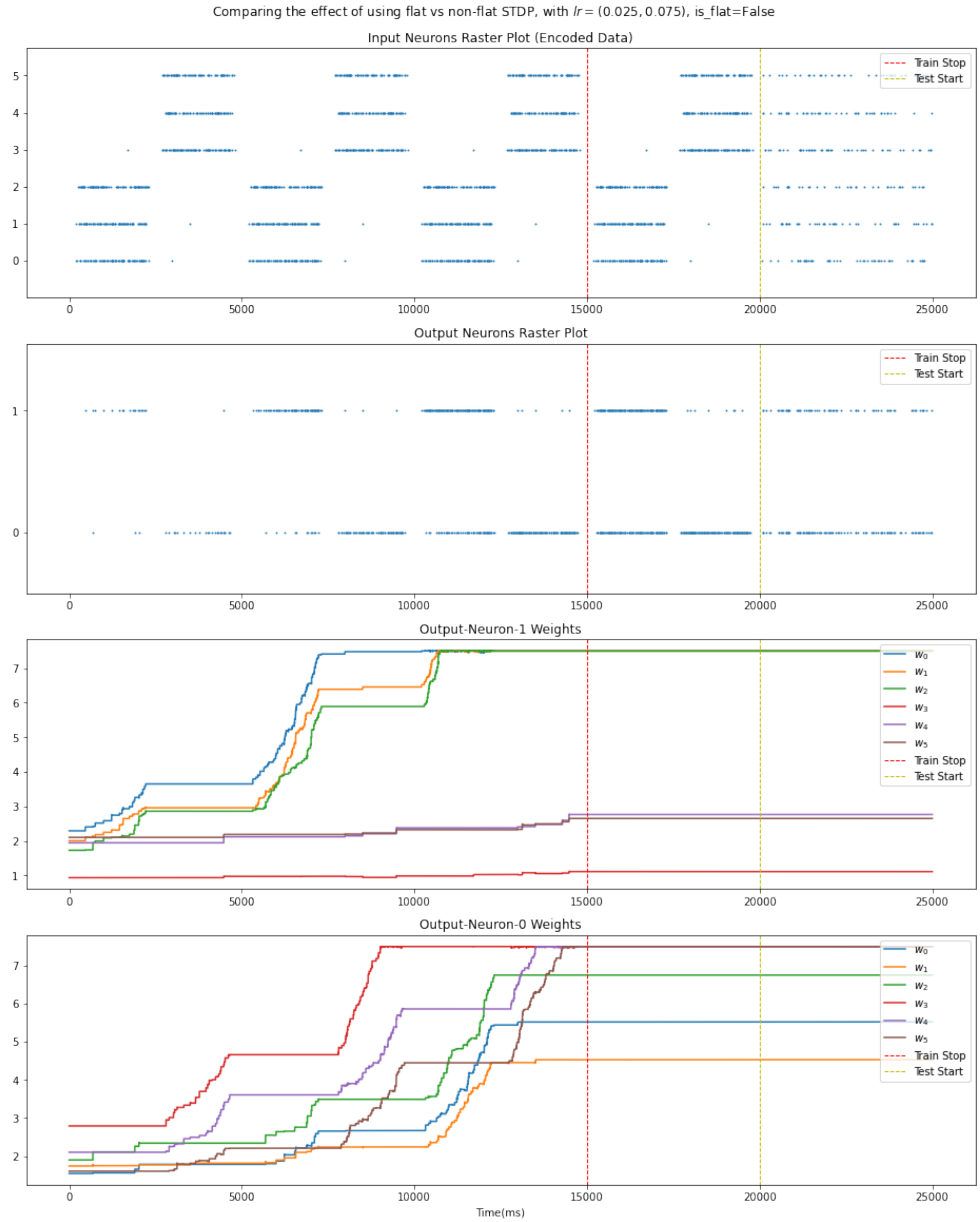
Comparing the effect of using flat vs non-flat STDP, with $lr = [0.03, 0.03]$, is_flat=True
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
Time(ms)

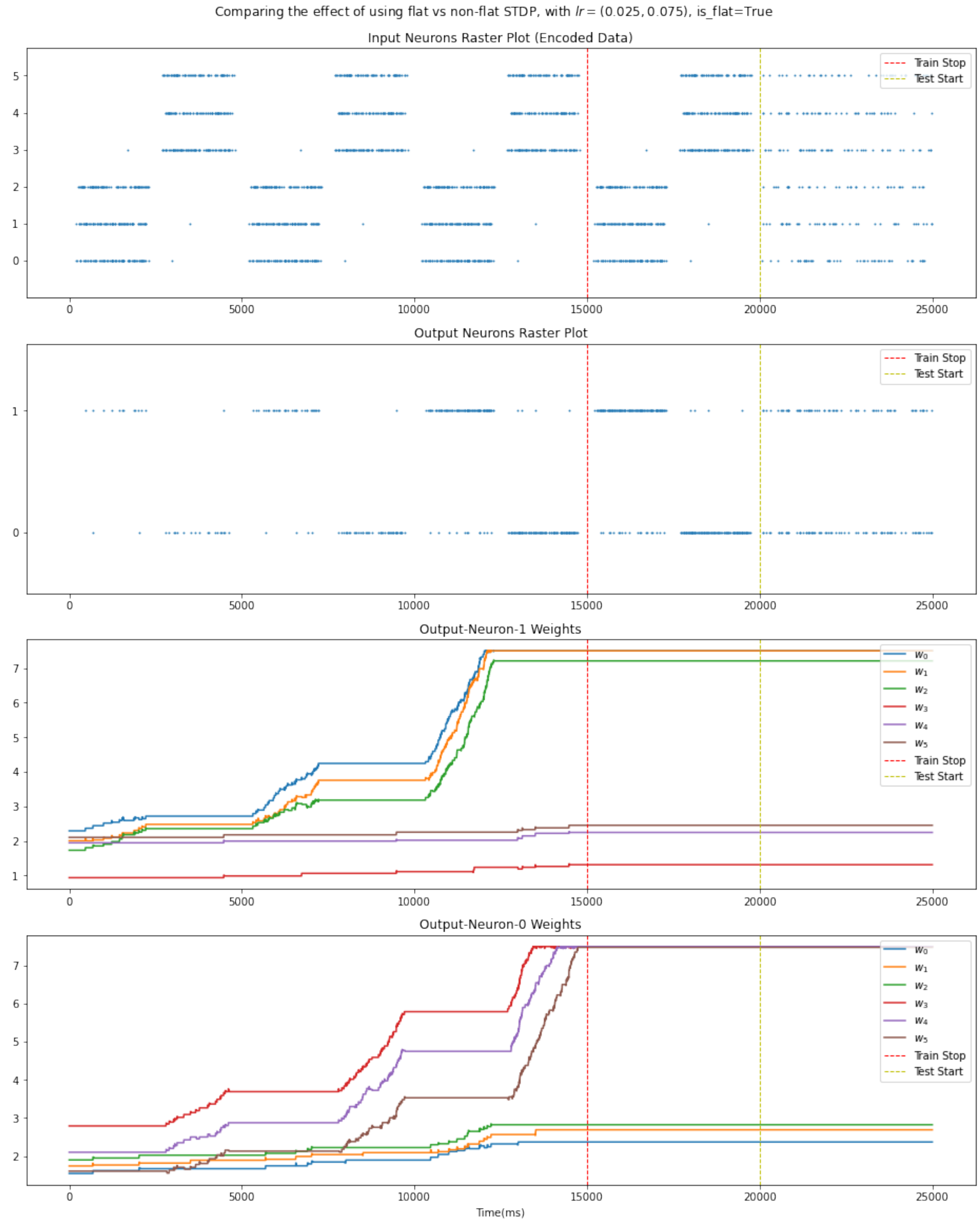


Comparing the effect of using flat vs non-flat STDP, with lr = [0.05, 0.03], is_flat=False
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w0
w1
w2
w3
w4
w5
Time(ms)



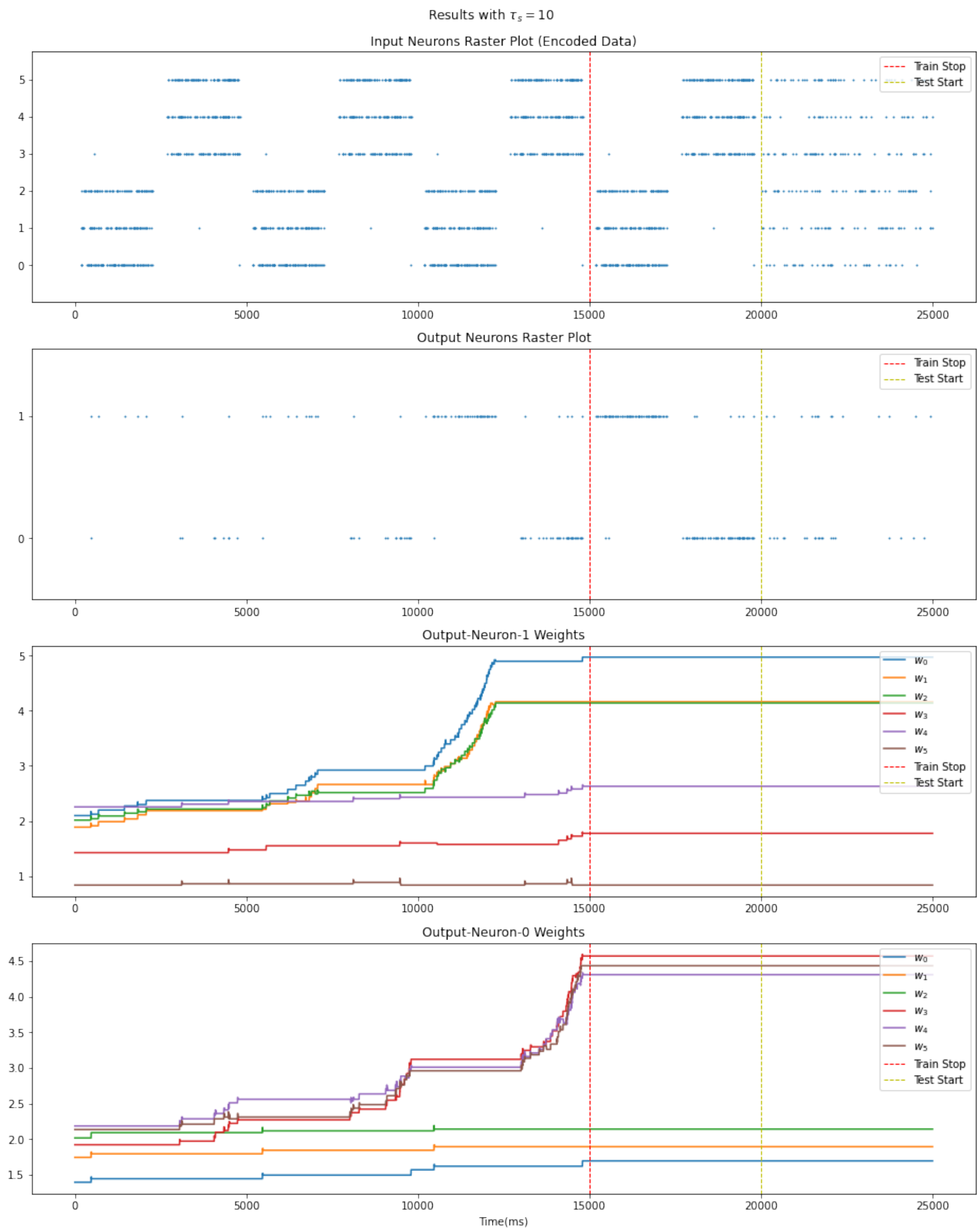
Comparing the effect of using flat vs non-flat STDP, with $lr = [0.05, 0.03]$, is_flat=True
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Output-Neuron-0 Weights
Time(ms)

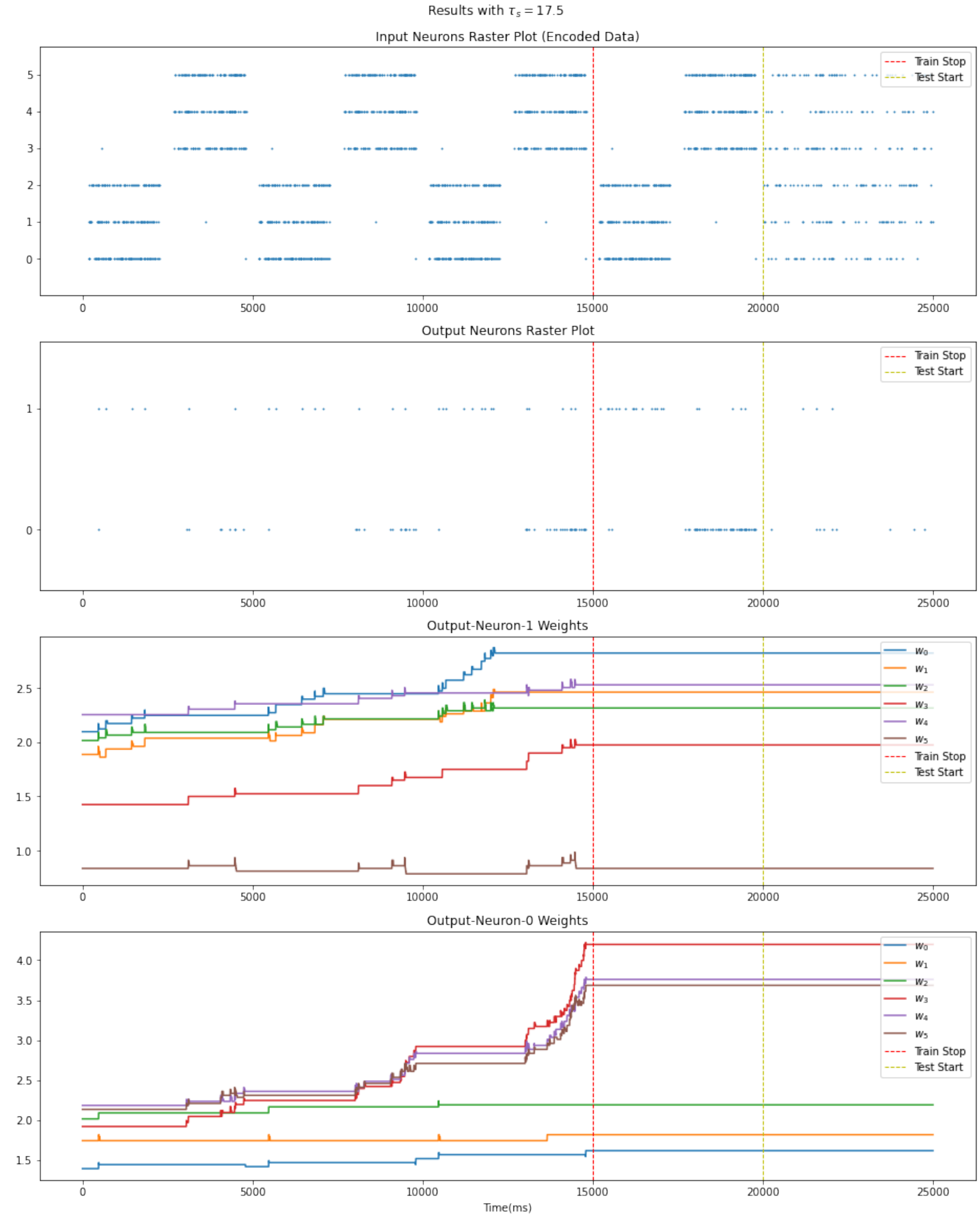

Comparing the effect of using flat vs non-flat STDP, with $lr = (0.025, 0.075)$, is_flat=False
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)



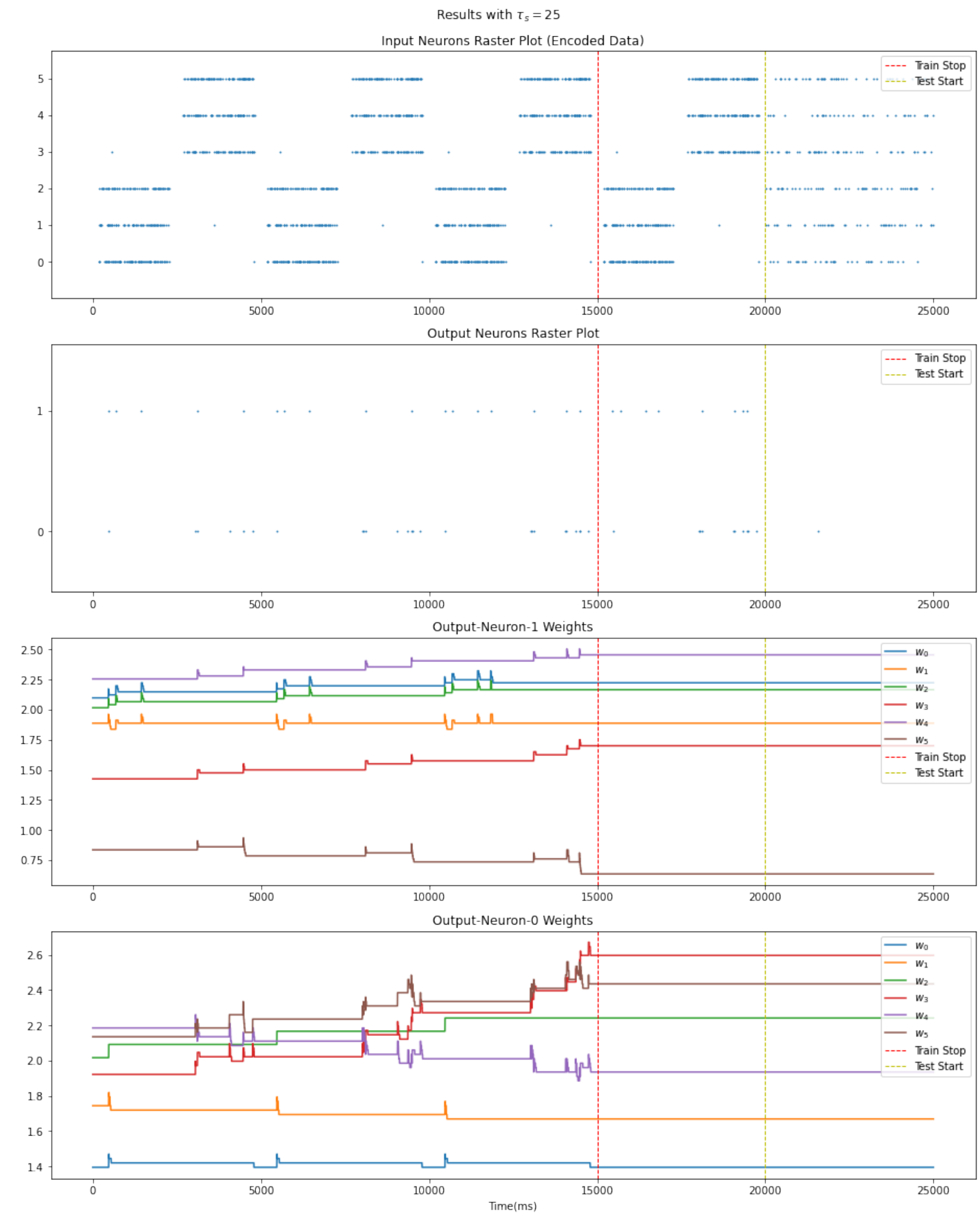
We see that in some cases despite the fact that F-STDP is simpler, it works as good as non-flat-STDP and even better in some other cases ($lr = [0.025, 0.075]$). The only thing to consider is to change the learning rate when switching from STDP to F-STDP. So, it is favorable to use F-STDP because it works as good as STDP but is simpler.

3.4 Experiment #3 (τ_s)





Results with $\tau_s = 17.5$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)

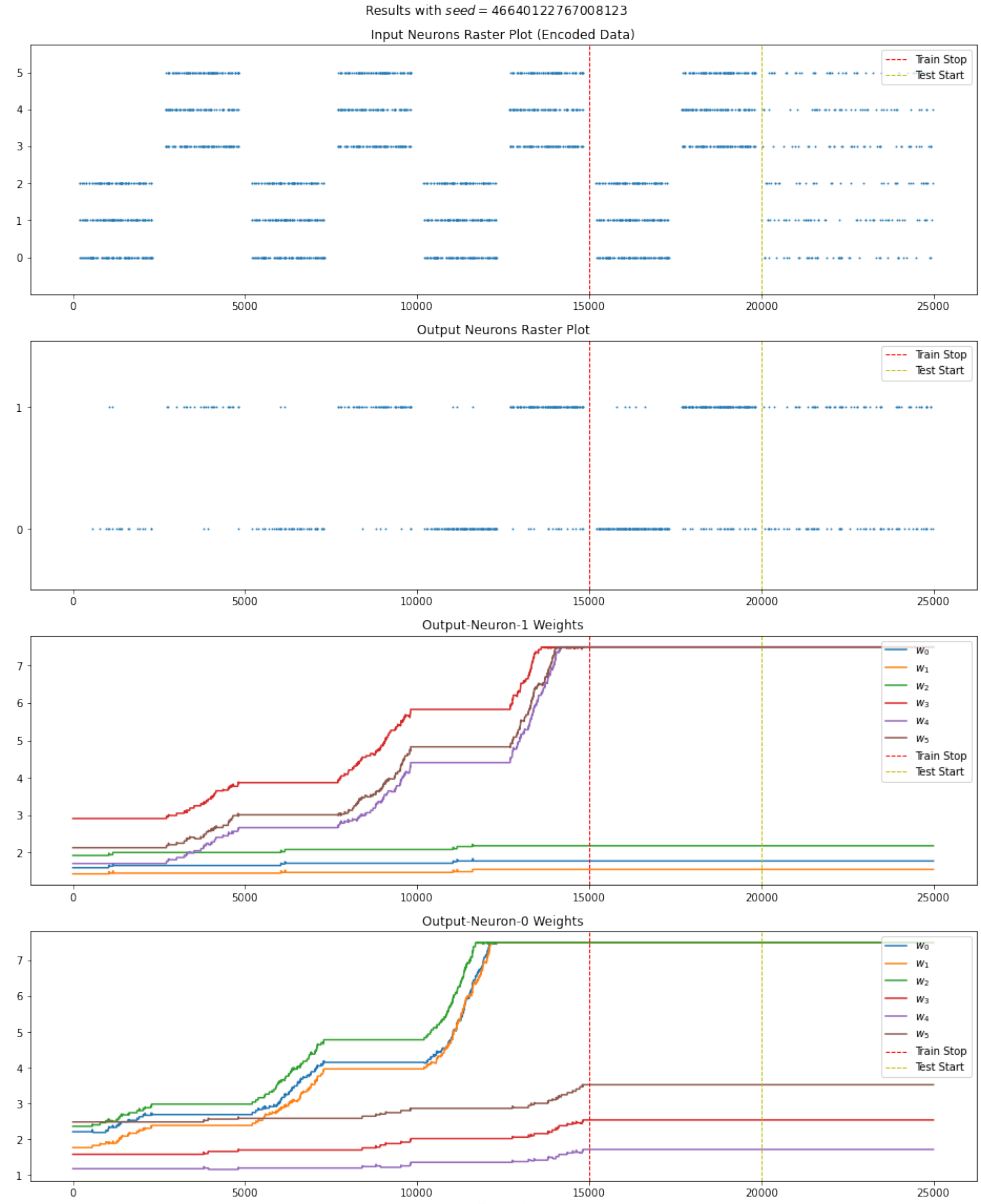


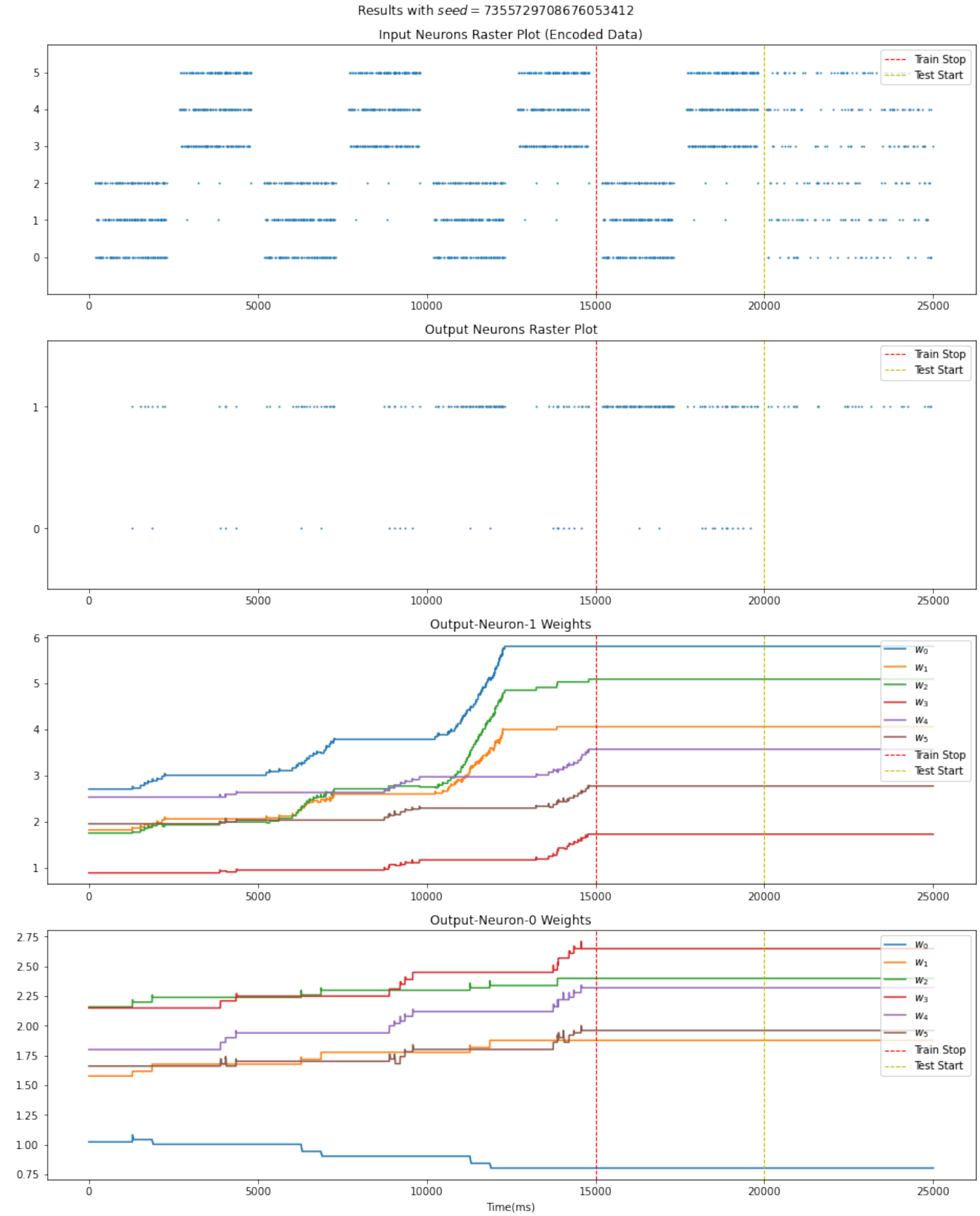
By increasing the value of τ_s, the network's ability to learn properly is declining because older spikes are taken into account for updating the weights. These old spikes contain un-wanted spikes and have a bad impact on the update rule.

Chapter 4

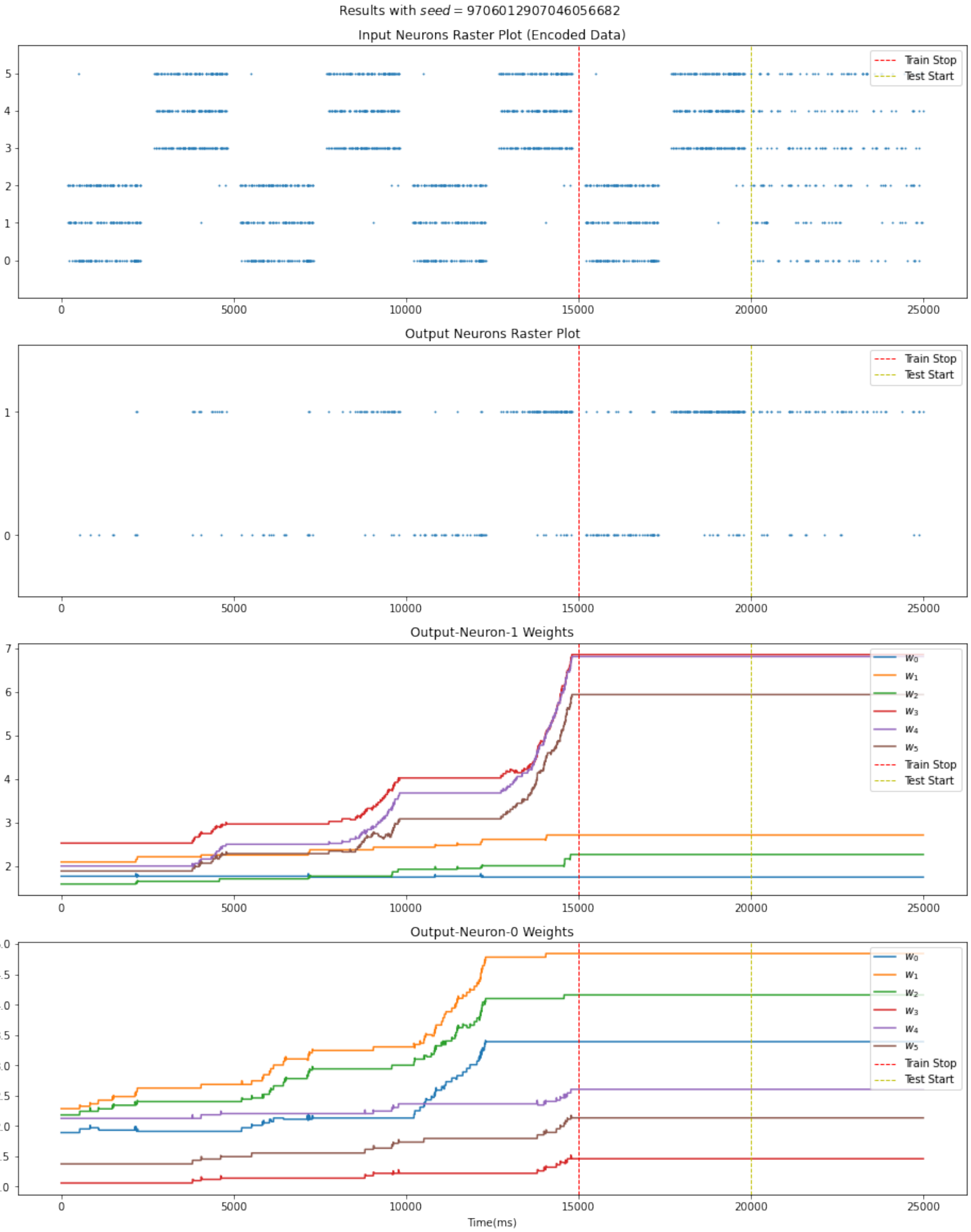
Some Random Simulations

The following simulations are presented to show the output of Flat-STDP without manually selecting the random seeds resulting in favorable results.

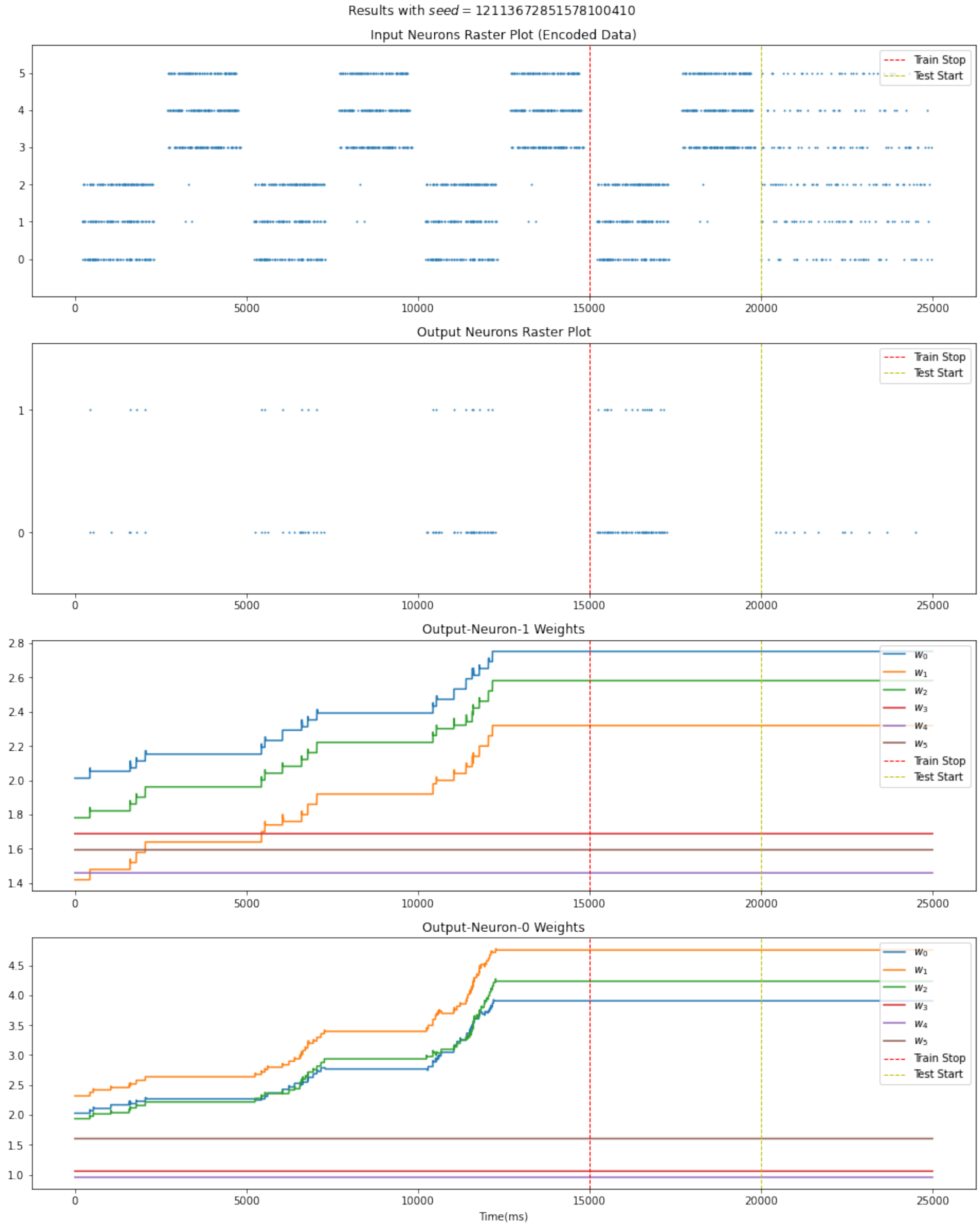

Results with $seed = 46640122767008123$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Train Stop
Test Start
Time(ms)

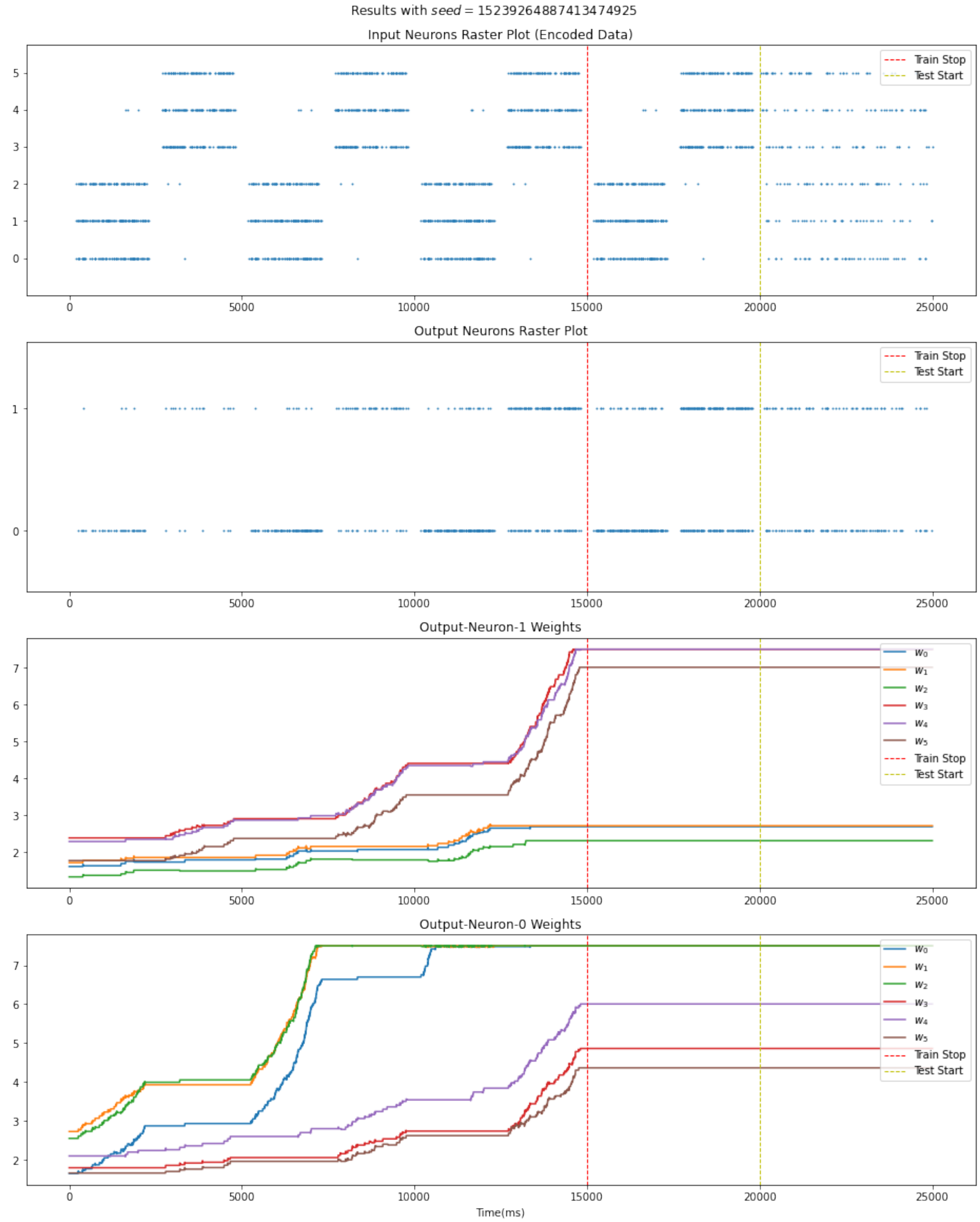


Results with $seed = 7355729708676053412$
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)



Results with $seed$ = 9706012907046056682
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Output-Neuron-0 Weights
Time(ms)


Results with seed = 12113672851578100410
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
Output-Neuron-0 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Time(ms)



Results with $seed$ = 15239264887413474925
Input Neurons Raster Plot (Encoded Data)
Train Stop
Test Start
Output Neurons Raster Plot
Output-Neuron-1 Weights
w_0
w_1
w_2
w_3
w_4
w_5
Output-Neuron-0 Weights
Time(ms)

Chapter 5

Summary

1. We can achieve almost the same results using `Flat-STDP` and `STDP`. So, it's better to use `Flat-STDP` because it is simpler.

2. If the initial weights are too high, the network's initial activation is high; therefore, the weights tend to saturate at the maximum value faster. Also, the network sensibility only to the input patterns decreases, and the output neurons will be activated frequently even with unseen data. In contrast, if the initial weights are set too small, they cannot be activated using the input, and the learning will not happen altogether.

3. It is better to keep A_- a little higher than A_+ in `STDP`. The opposite is true in `Flat-STDP`. By increasing both of these parameters, the learning will take place faster.

4. It is better to keep τ_s small. High values for this parameter have a bad impact on the learning process, because unwanted and old spikes will affect the update terms. However, very low values also, might prevent the training to begin or make it too slow.

5. `STDP` is highly dependent on the initial weights and in many cases the output is not what we expect from it. Sometimes both neurons learn the same input, sometimes none of them learn anything, and sometimes one learns both patterns, and the other only learns one of them or neither of them. The expected output (each neuron learns only of the patterns) could be achieved by running the simulation multiple times and saving the random-seed of initializations. One thing that will help to achieve these favorable results more often is to add `lateral-inhibition` between output neurons. We used this technique in this project. Another method is to use `homostatis`, which we did not implement here.