project-7

May 21, 2021

Contents

Chapter 1

Introduction

1.1 Reward-Modulated STDP (R-STDP)

We implement two more formulas (in addition to previous project STDP) in order to add rewarding functionality to STDP.

We update the `synaptic weights (s)` according to a `variable c (synaptic tag)` and the amount of extracellular concentration of dopamine.

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

where $\tau = t_{post} - t_{pre}$, d describes the extracellular concentration of DA and $\delta(t)$ is the Dirac delta function.

Typically, τ_c has a big value *e.g.*, $500ms$. Synaptic tag stores the activity of the neurons in a longer time interval compared with spike traces that are used in `STDP`.

The variable d describes the concentration (μM) of extracellular DA:

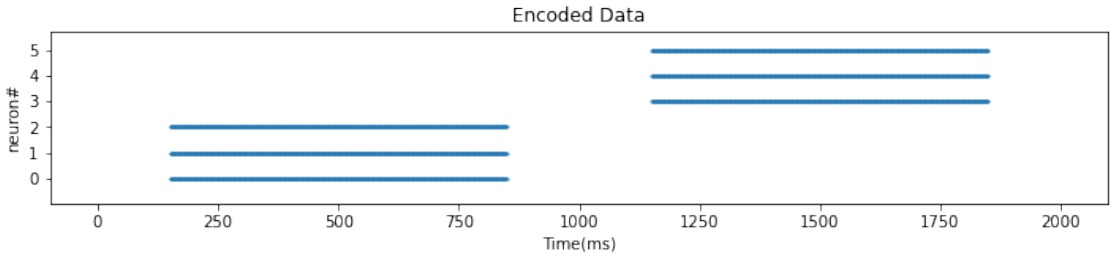
$$\frac{dd}{dt} = -\frac{d}{\tau_d} + DA(t),$$

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).

$DA(t)$ represents the reward/punish signal.

1.2 Input

The input patterns are created by using the PoissonEncoder. Note that the intensity of the input is set to a high value. This is the reason it seems that the input neurons are spiking continuously. Before and after each input pattern, exists a time period that the input is cutoff.



Note: The random seed is fixed (not selected manually though) when testing the effect of each parameter to have a fair evaluation.

Chapter 2

R-STDP

2.1 Default Parameters

Train Params:

$Time_{simulation} = 10000ms$

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

$\tau_c = 75ms$

$\tau_d = 10ms$

Neurons Params:

$Num(PresynapticNeurons) = 6$

$\tau_s = 10ms$

$Threshold = -42.5mv$

$U_{rest} = -60mv$

Connection Params:

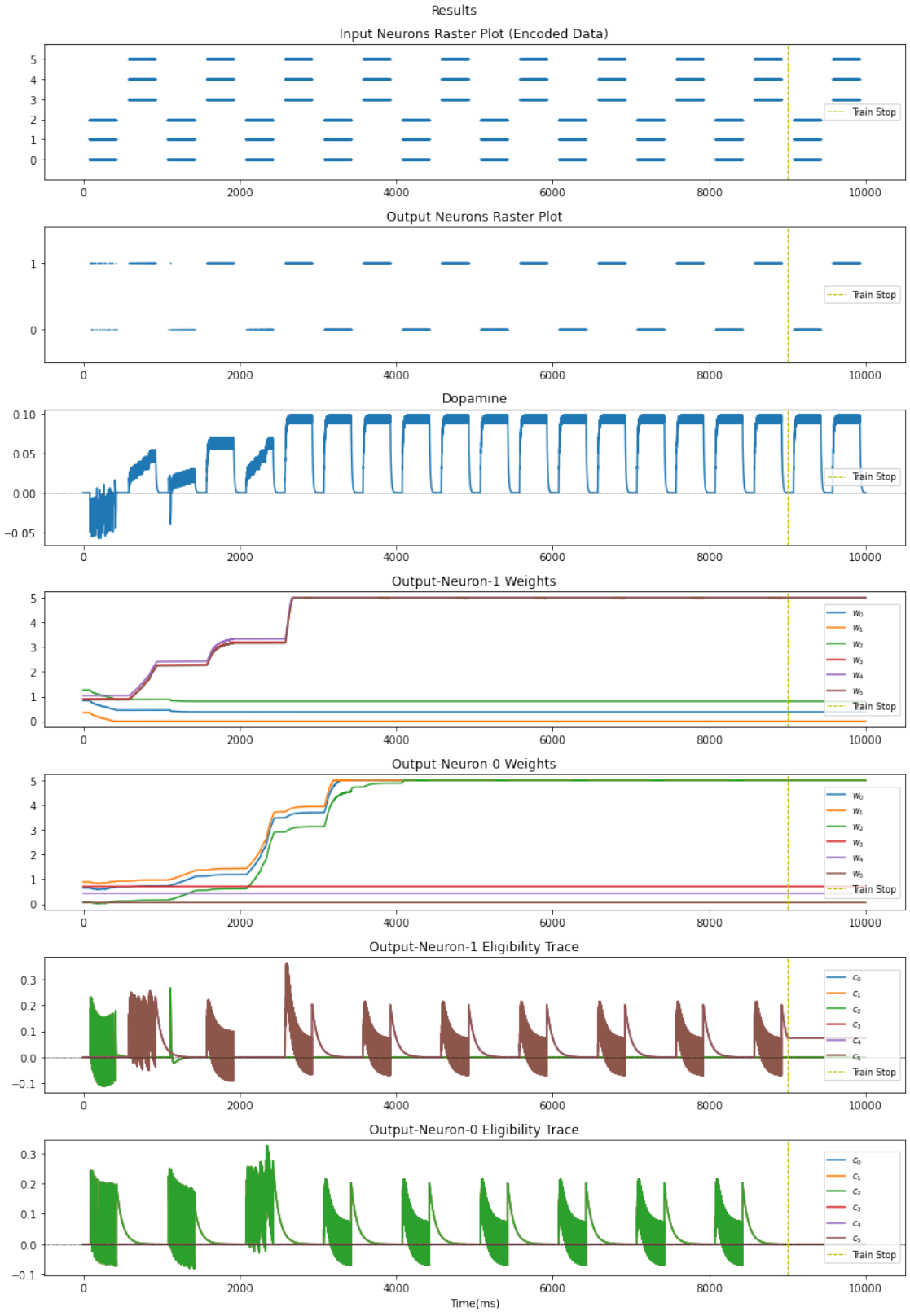
$J_0 = 5$

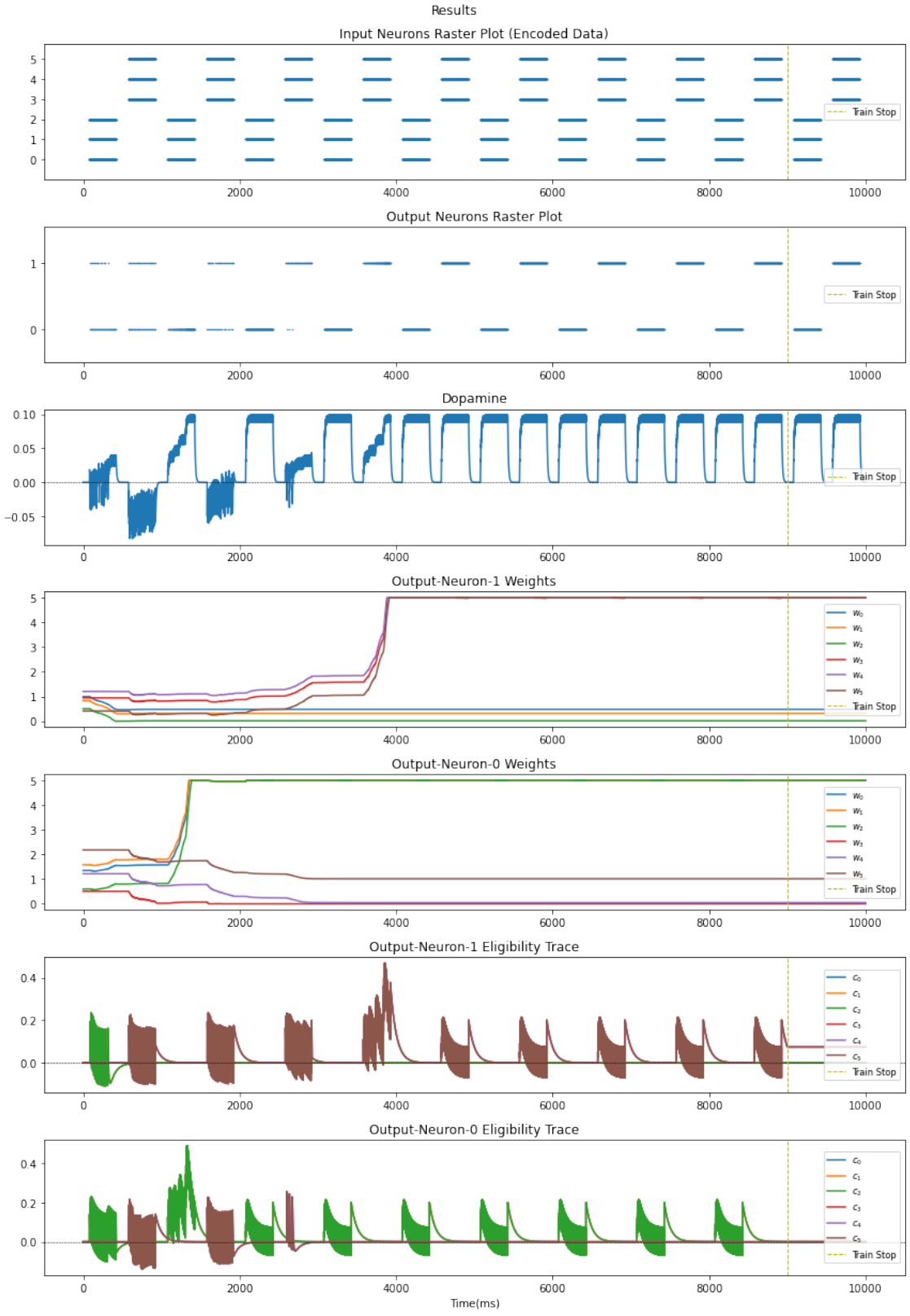
$\sigma_0 = 3$

$Weight_{min} = 0$

$Weight_{max} = 5$

Here you can see the simulation result using default parameters with two different random weight initializations.


Results
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

2.2 Experiment #1 (Punish/Reward Policy)

When the first pattern is active, we want only the first neuron to only. Likewise, when the second pattern is active we want only the second neuron to respond. For this to happen, we have to reward each of them when responding correctly and punish them when responding incorrectly. Additionally, when both of the neurons are responding to one pattern we need to send a **punish** signal to the network because this behaviour is also unwanted. One important thing that we considered is that the amount of reward/punish is not equal in each case. You can see the rewarding policy clearly in the table below:

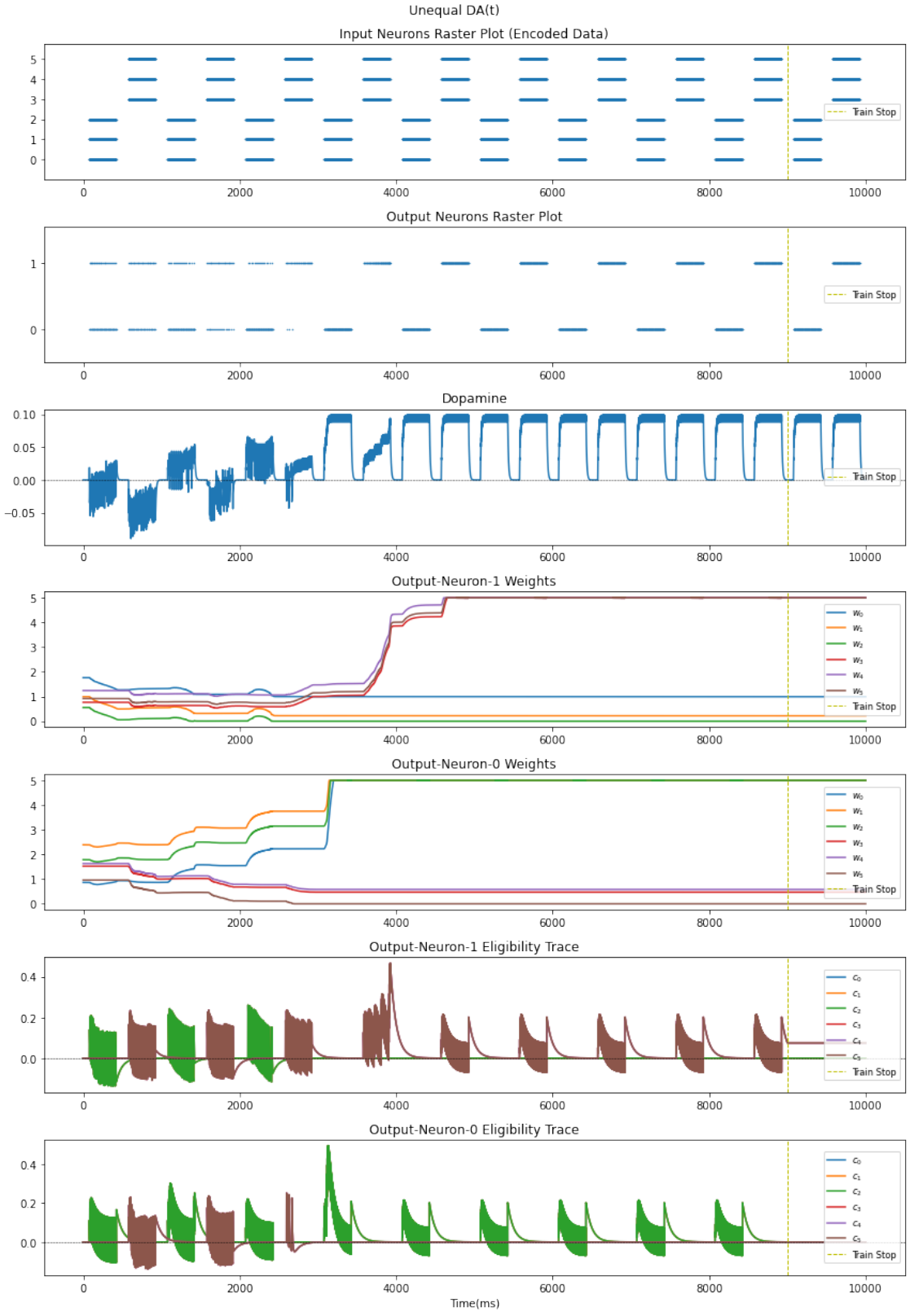
Pattern	Neuron-0 Spiked	Neuron-1 Spiked	Result
0	False	False	NoOp → DA(t) = 0
0	False	True	Punish → DA(t) = -2
0	True	False	Reward → DA(t) = +0.75
0	True	True	Punish → DA(t) = -0.25
1	False	False	NoOp → DA(t) = 0
1	False	True	Reward → DA(t) = +0.75
1	True	False	Punish → DA(t) = -2
1	True	True	Punish → DA(t) = -0.25

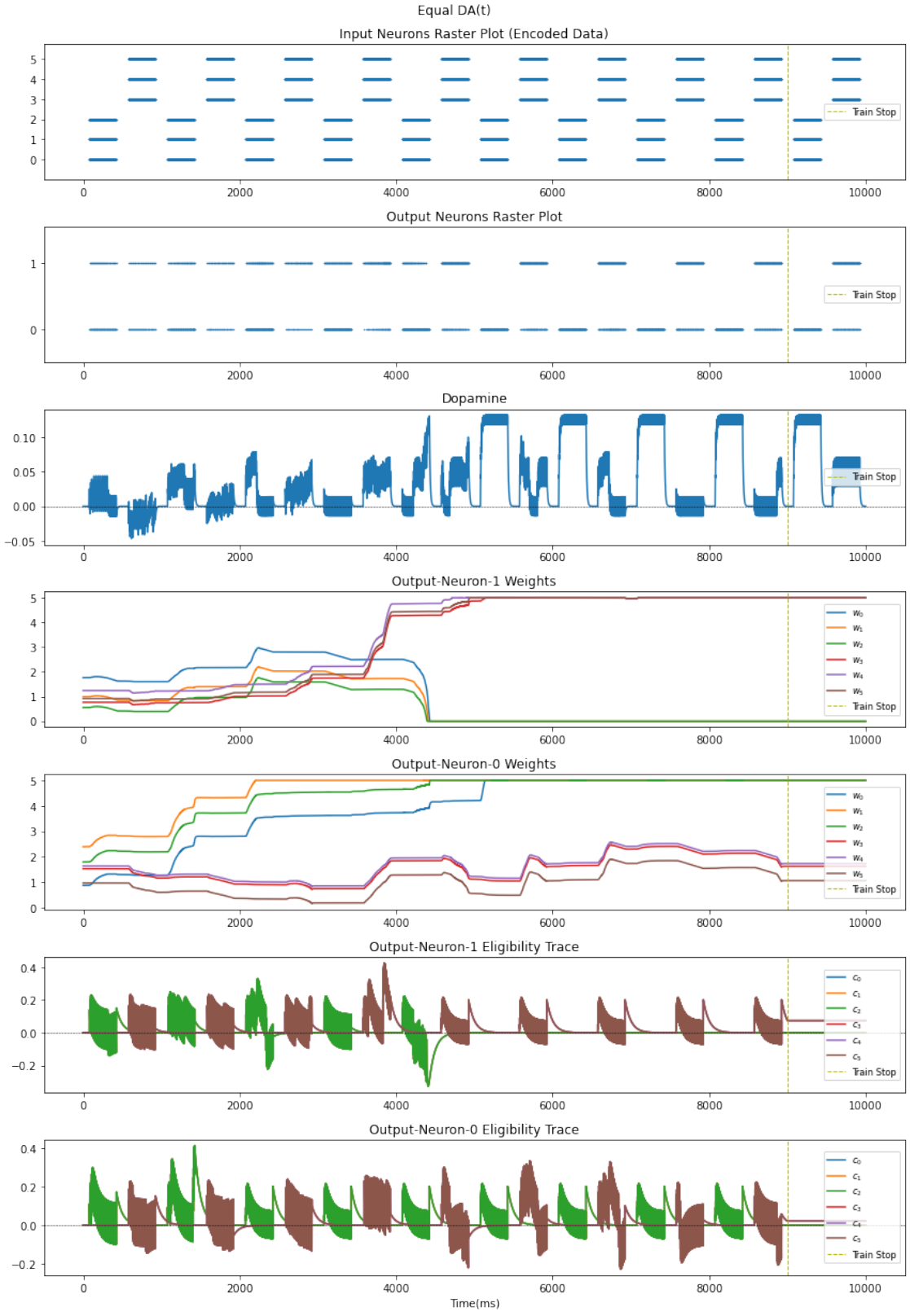
We found empirically that these ratios between DA updates will result in a desirable outcome with more probability. The intuition is that if both neurons spike together we should not punish them significantly, because if we do, the weights will decay very fast, so the network activation will be vanished, and the learning process stops altogether. Also, the rate of releasing dopamine should not be very high because fast dopamine release might increase the weights faster than it should.

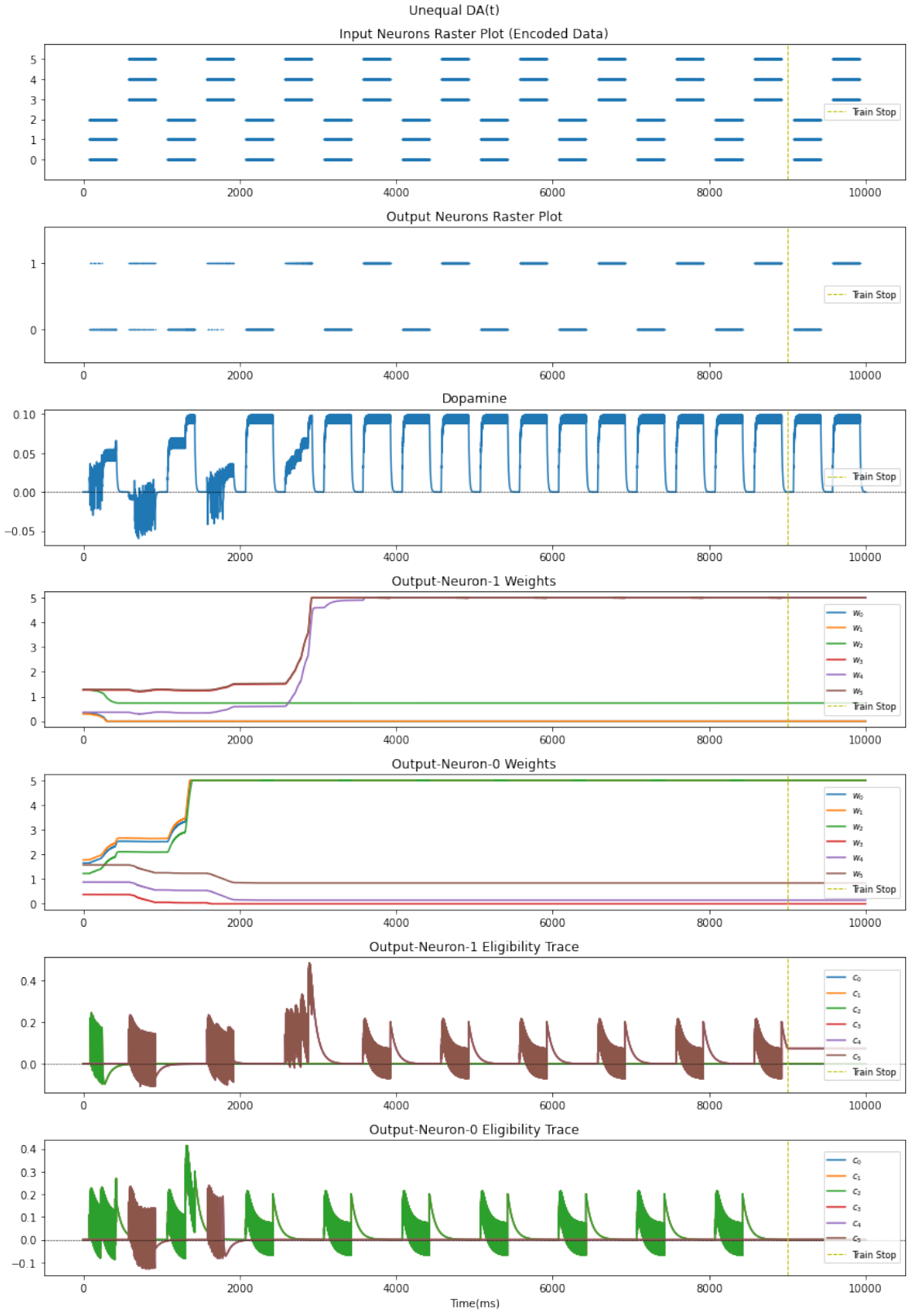
The DA(t) term is also multiplied by a small coefficient (0.025) before being used. The reason is to reduce the effect of each iteration. The effect is the same as multiplying both learning rates by this coefficient because: $\frac{dw}{dt} = c * d$

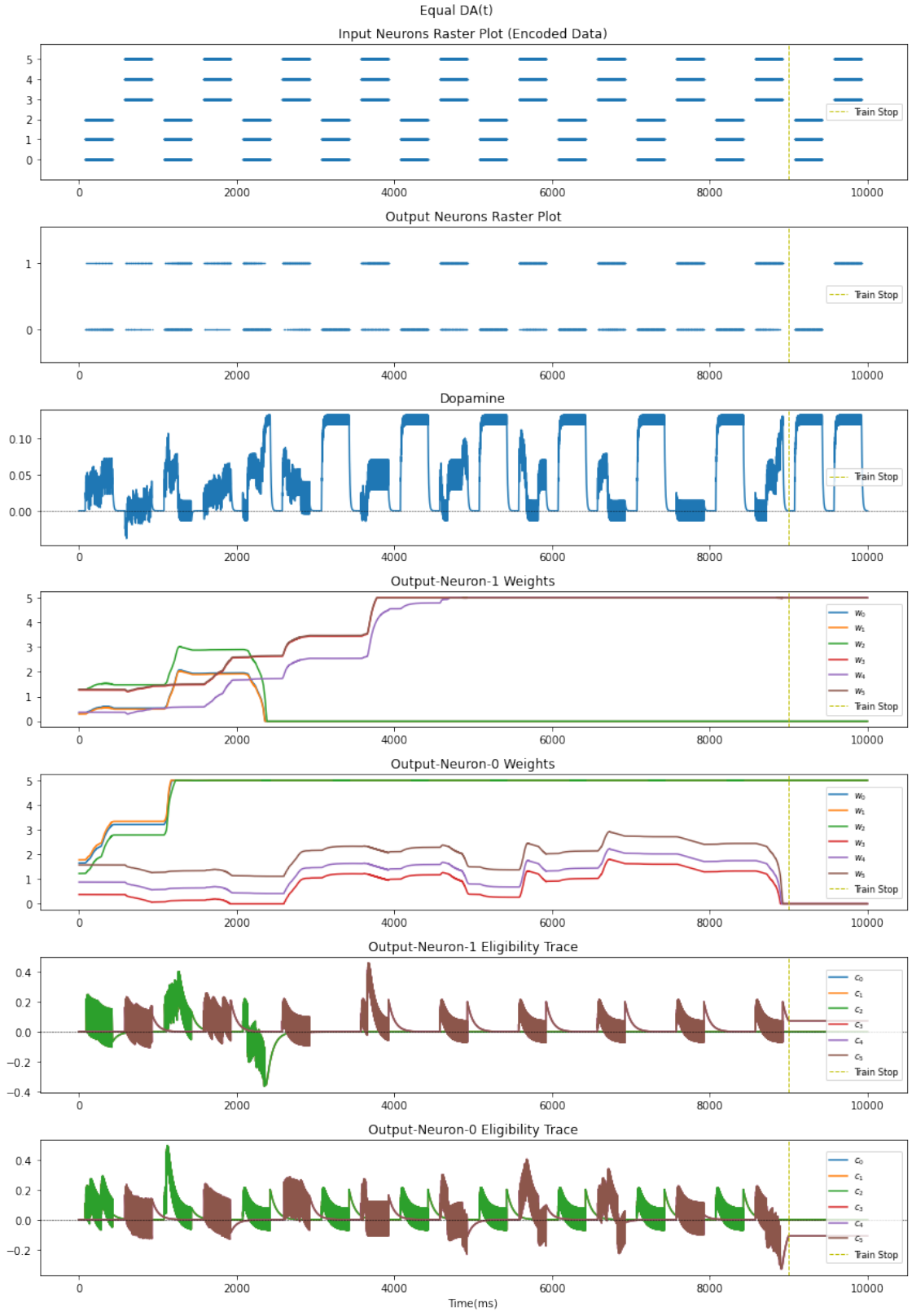
2.2.1 Robustness Test

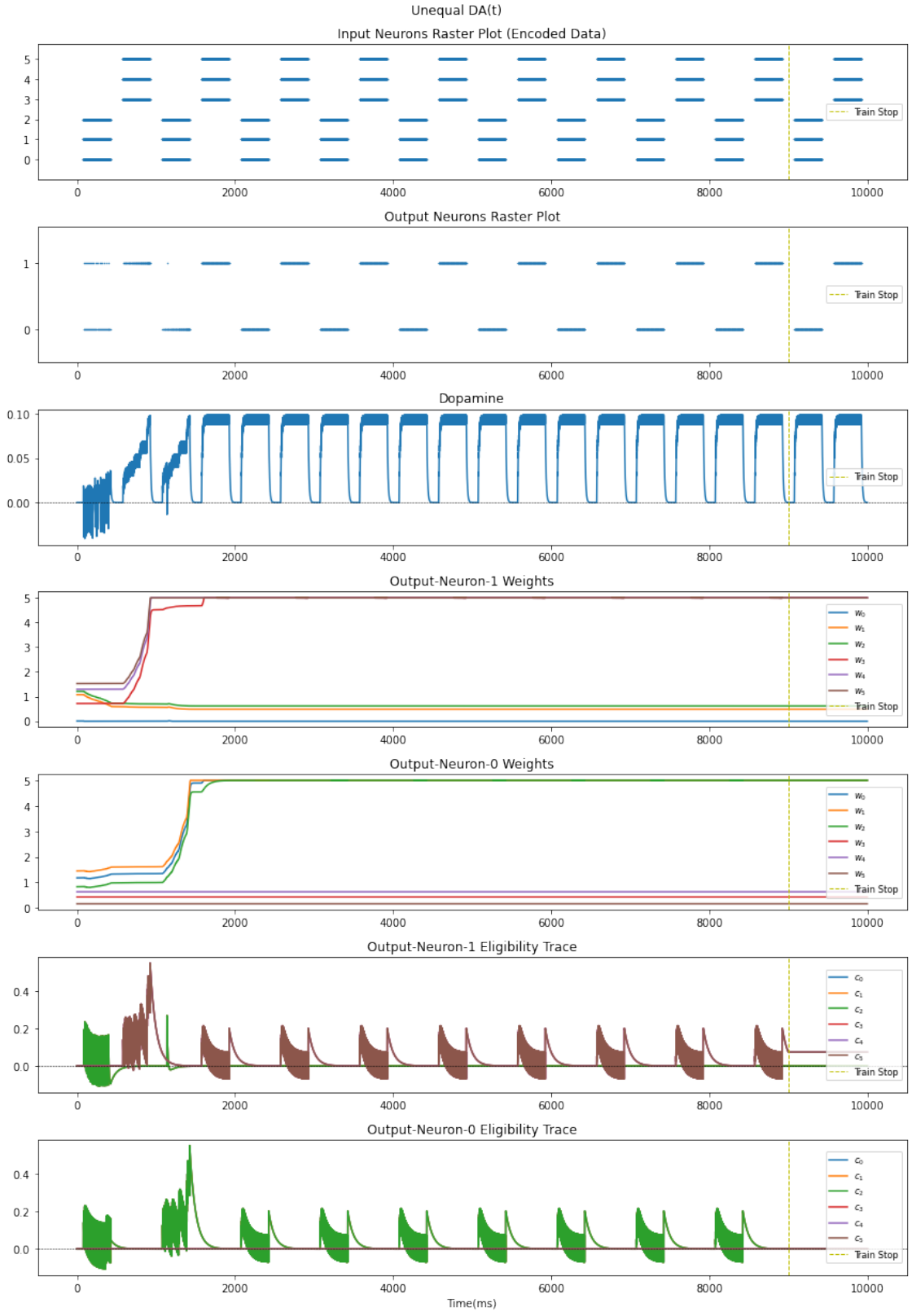
Here we run the simulation multiple times with Equal DA(t) values and with values that was presented in the above table to show that the ratios are actually helping the learning process.

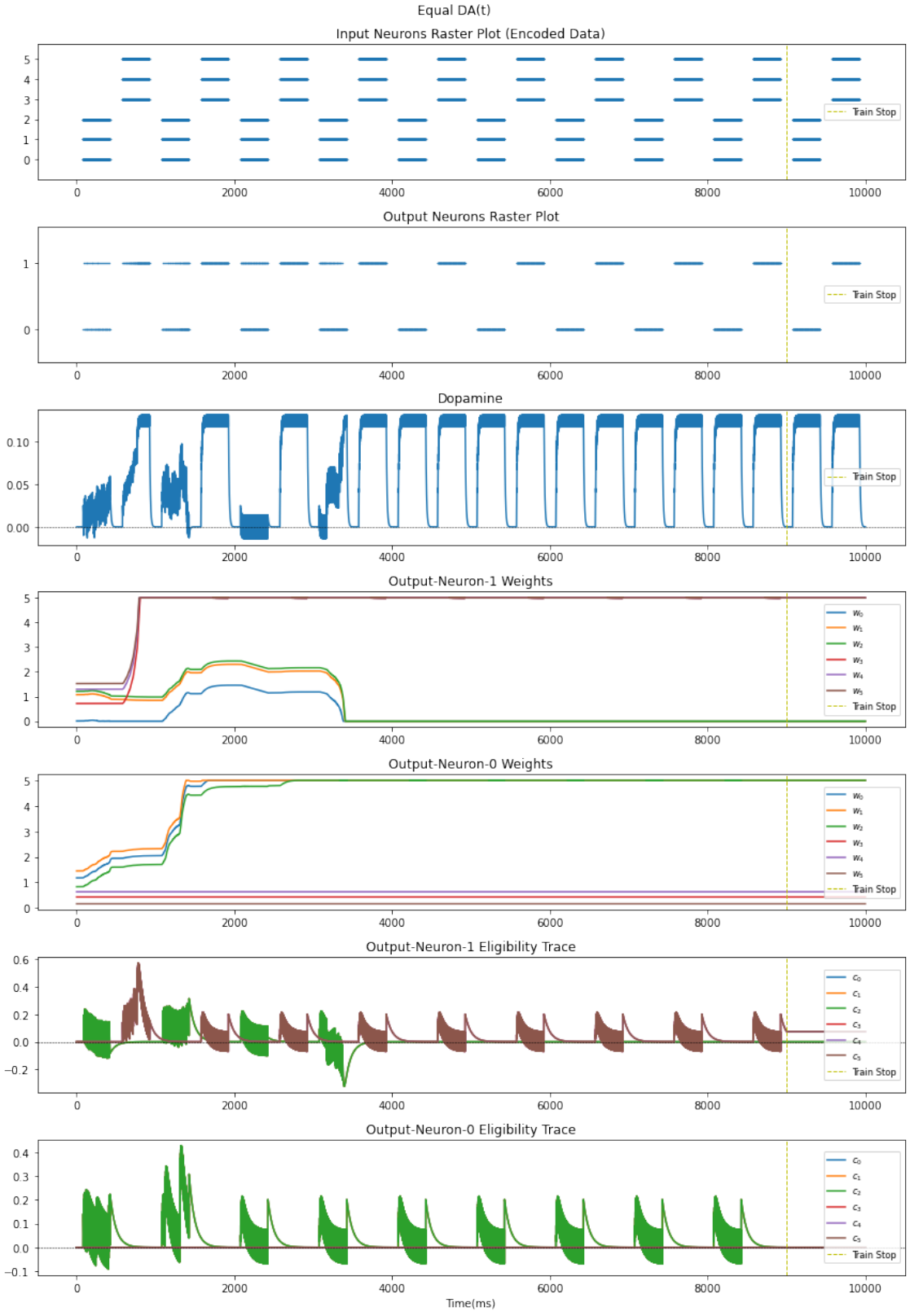

Unequal DA(t)
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Equal DA(t)
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Unequal DA(t)
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Equal DA(t)
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

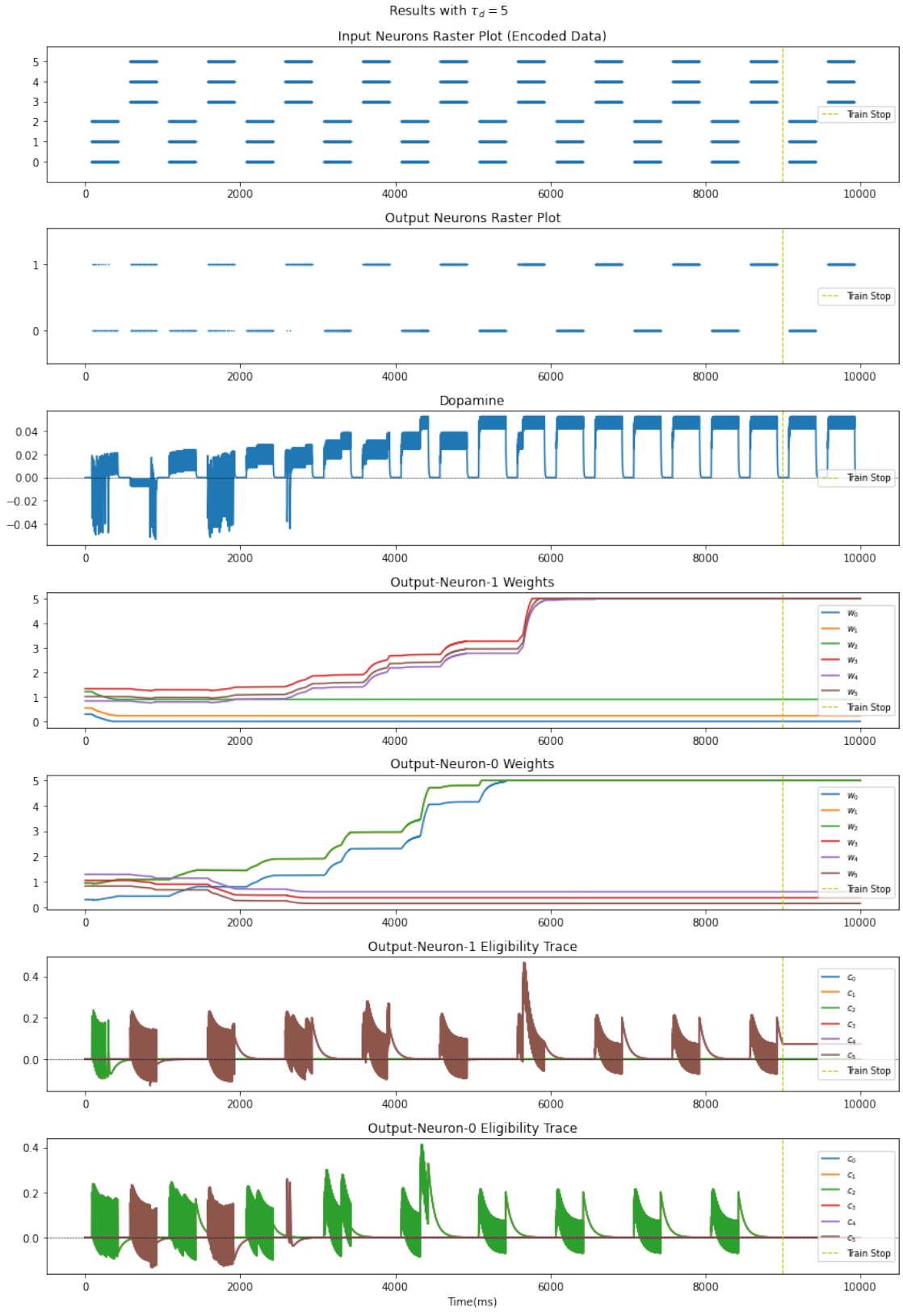

Unequal DA(t)
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

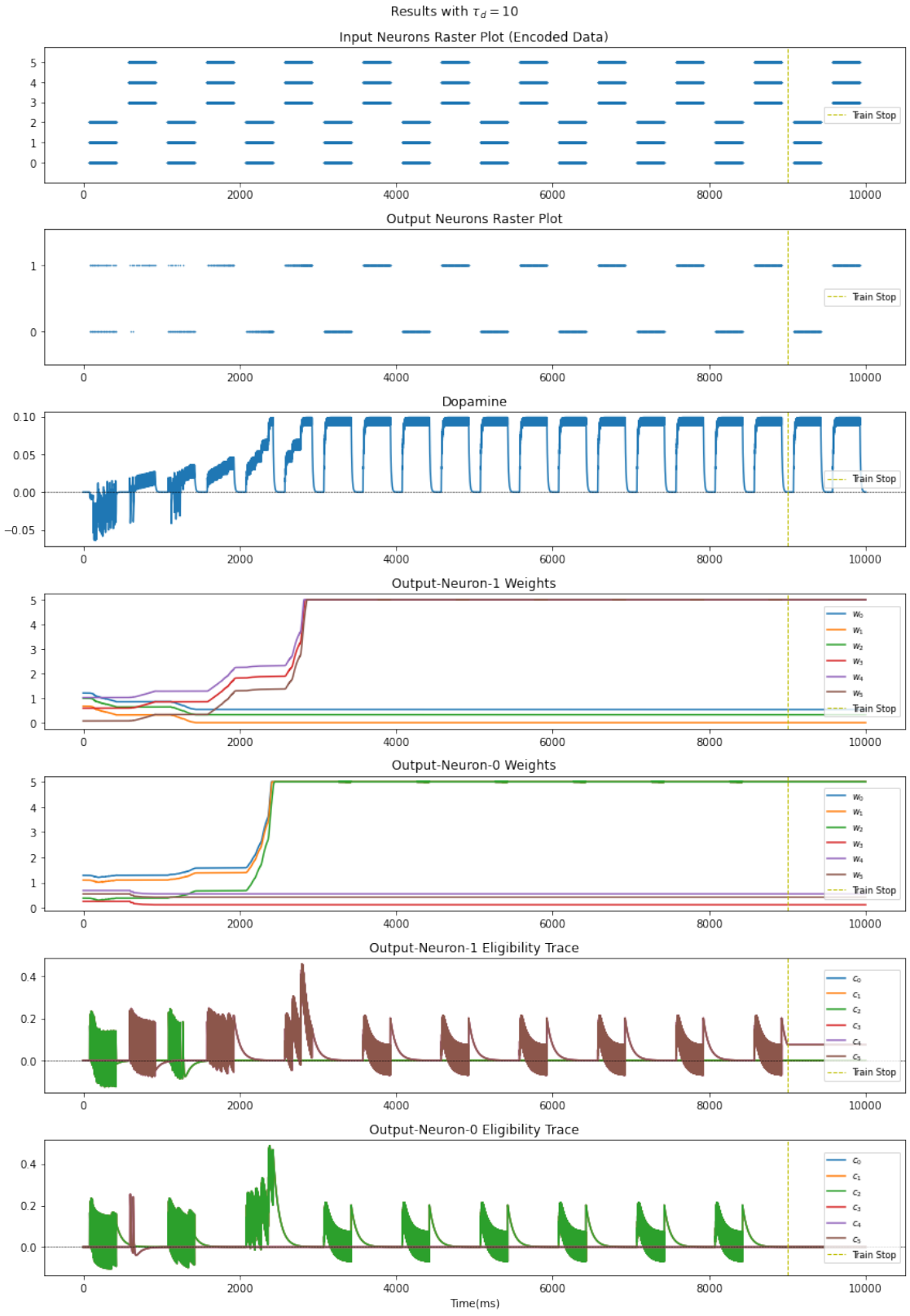

Equal DA(t)
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

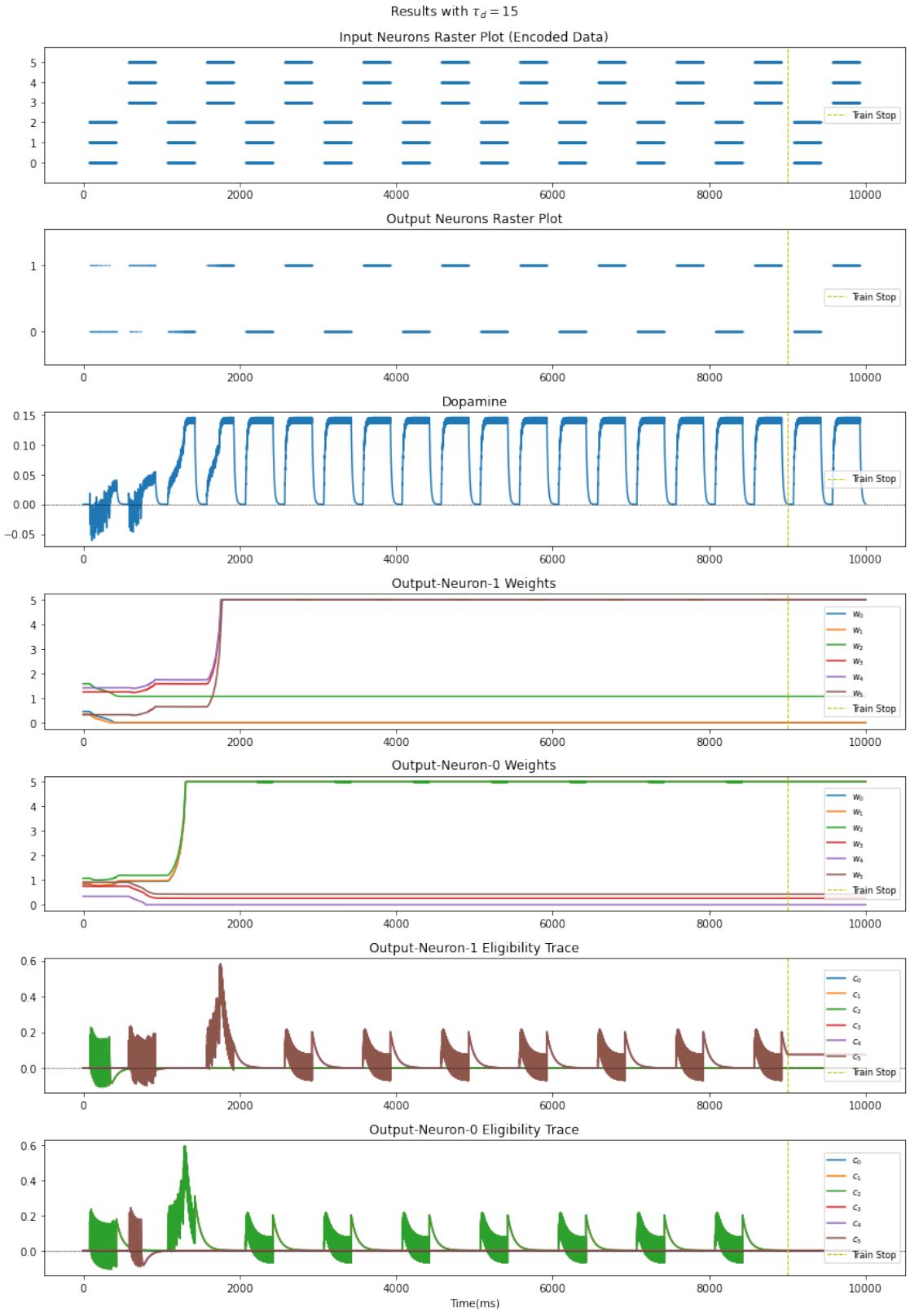
With first weight initialization, network with `Equal DA(t)` fails to learn the patterns correctly. With the second and third weight initialization, both strategies result in a correct outputs; however, the network with `Unequal DA(t)` learns the patterns quickly (with lower iterations). We ran the simulations numerous times in addition to the ones presented here, and the same pattern has been seen. Using `Unequal DA(t)` the network always learns the input patterns correctly.

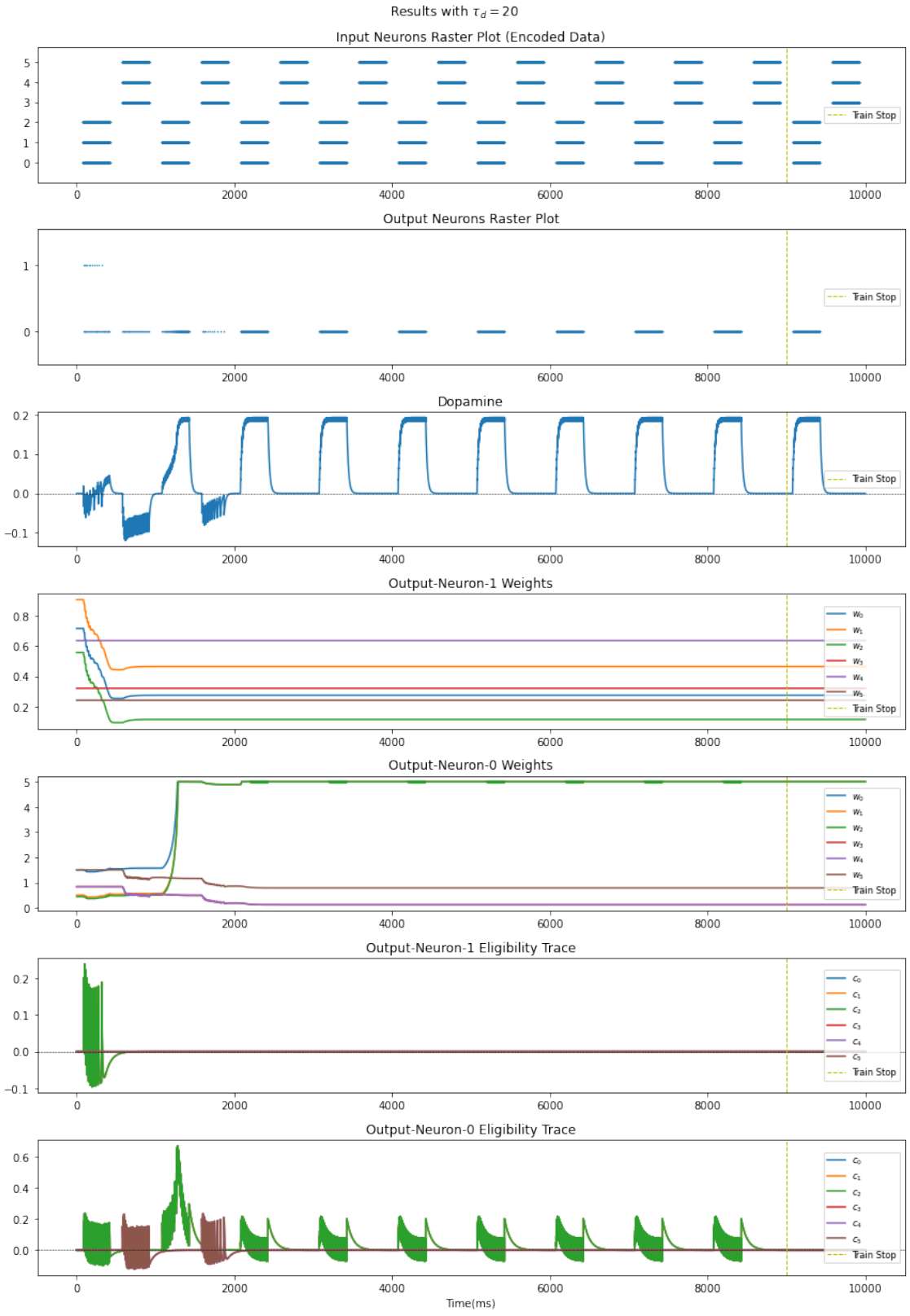
2.3 Experiment #2 (τ_d)

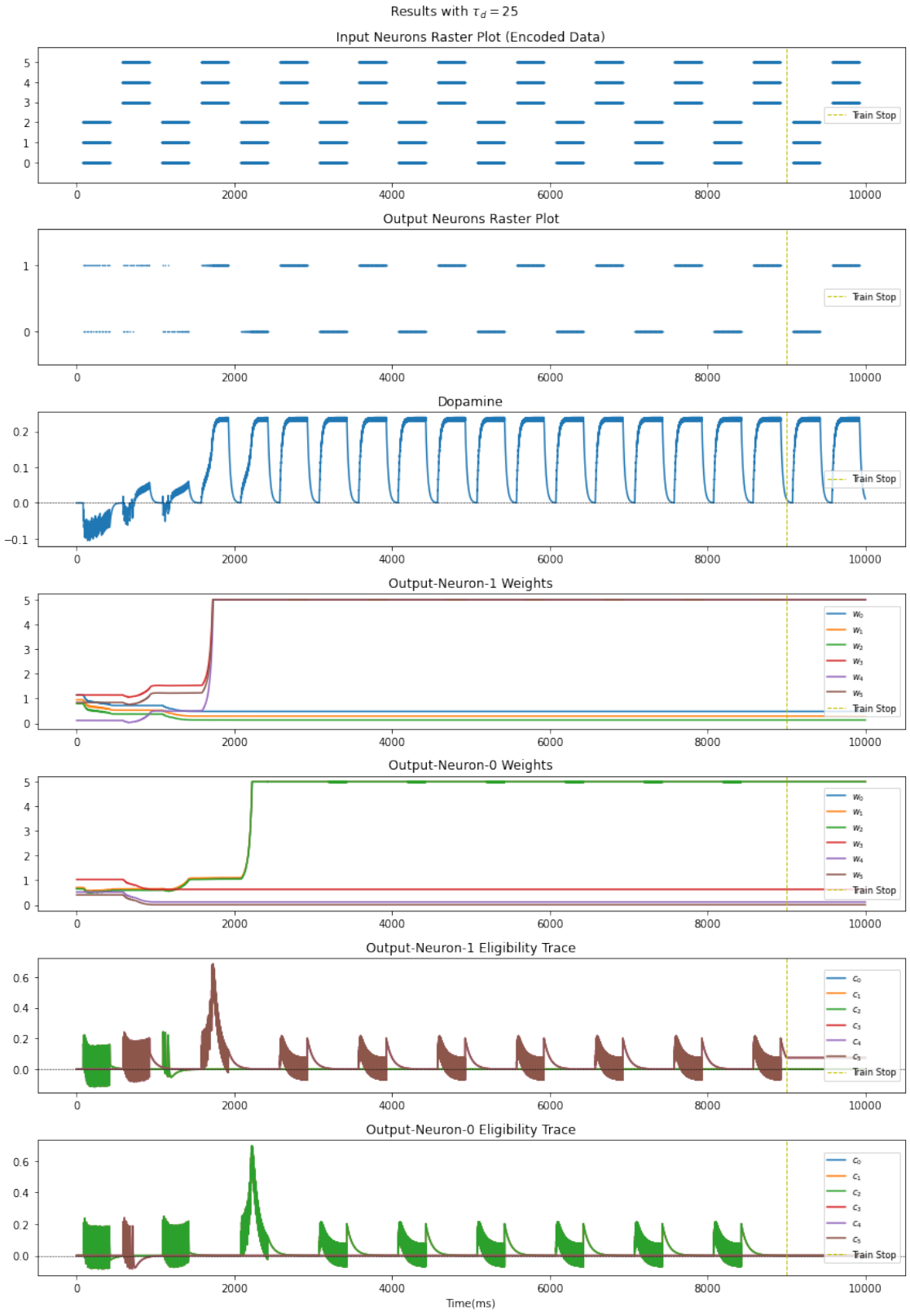
We test different values of τ_d in $[5, 10, 15, 20, 25]$

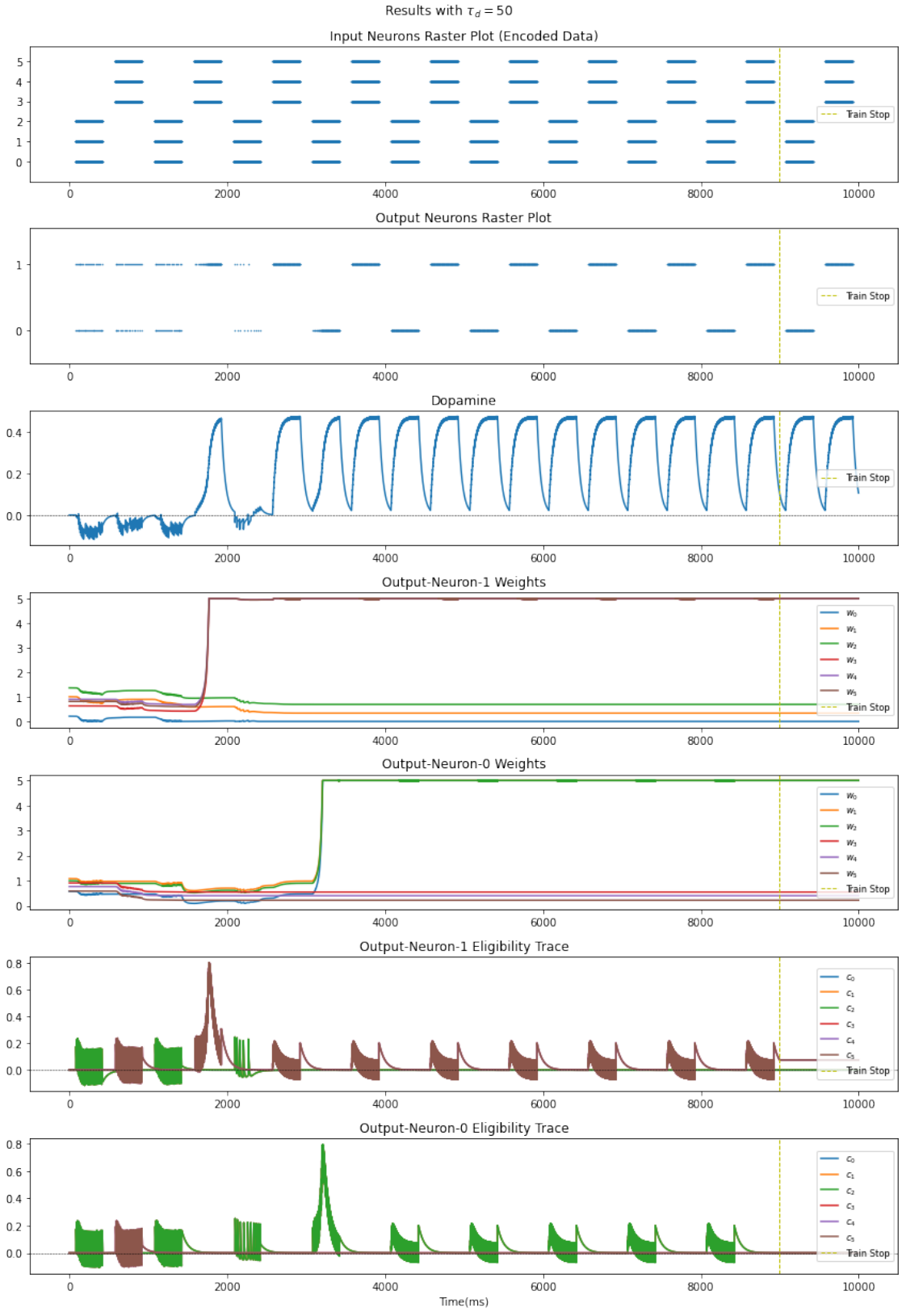

Results with $\tau_d = 5$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_d = 10$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_d = 15$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_d = 20$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_d = 25$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

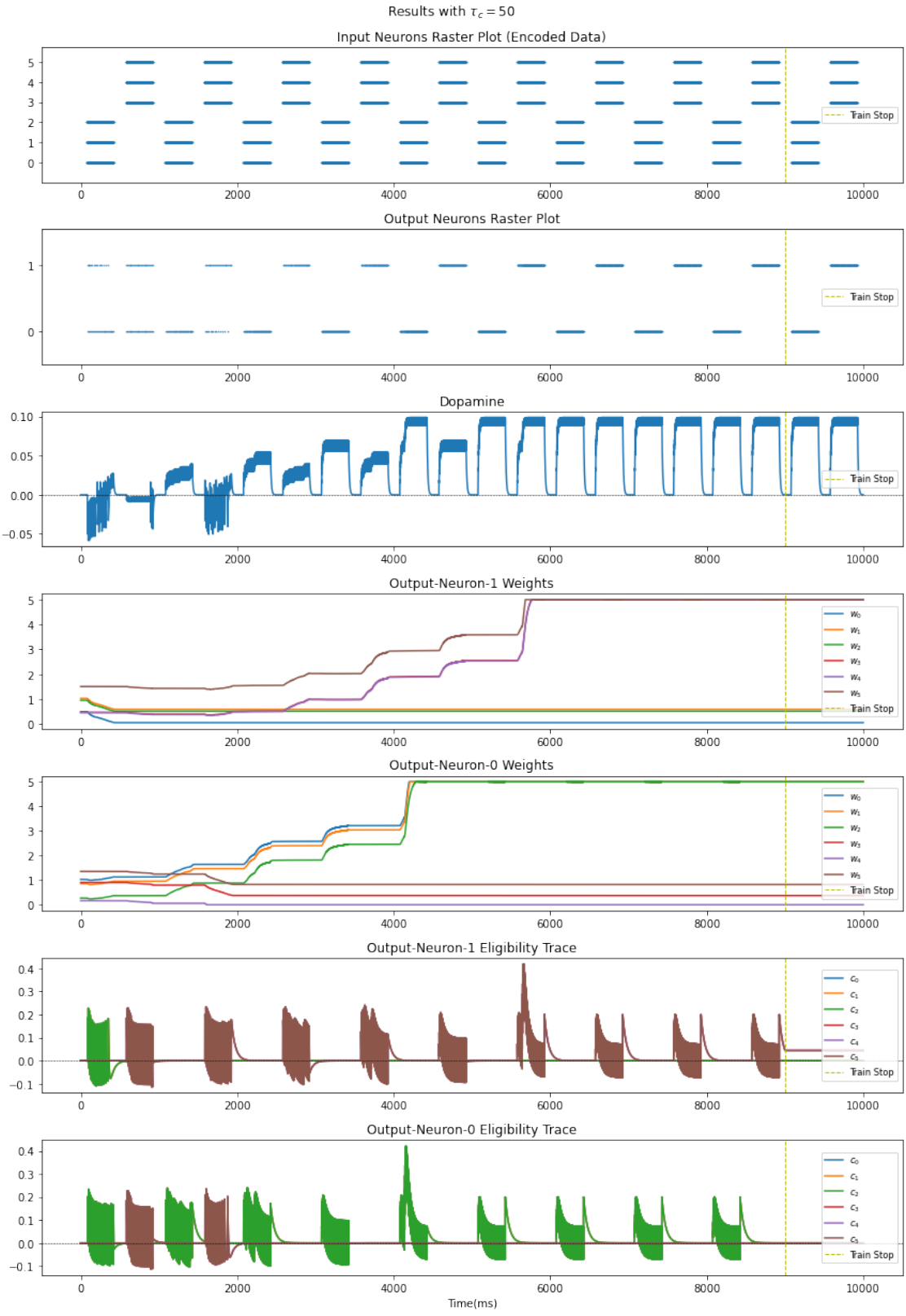

Results with $\tau_d = 50$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
w_0
w_1
w_2
w_3
w_4
w_5
c_0
c_1
c_2
c_3
c_4
c_5
Time(ms)

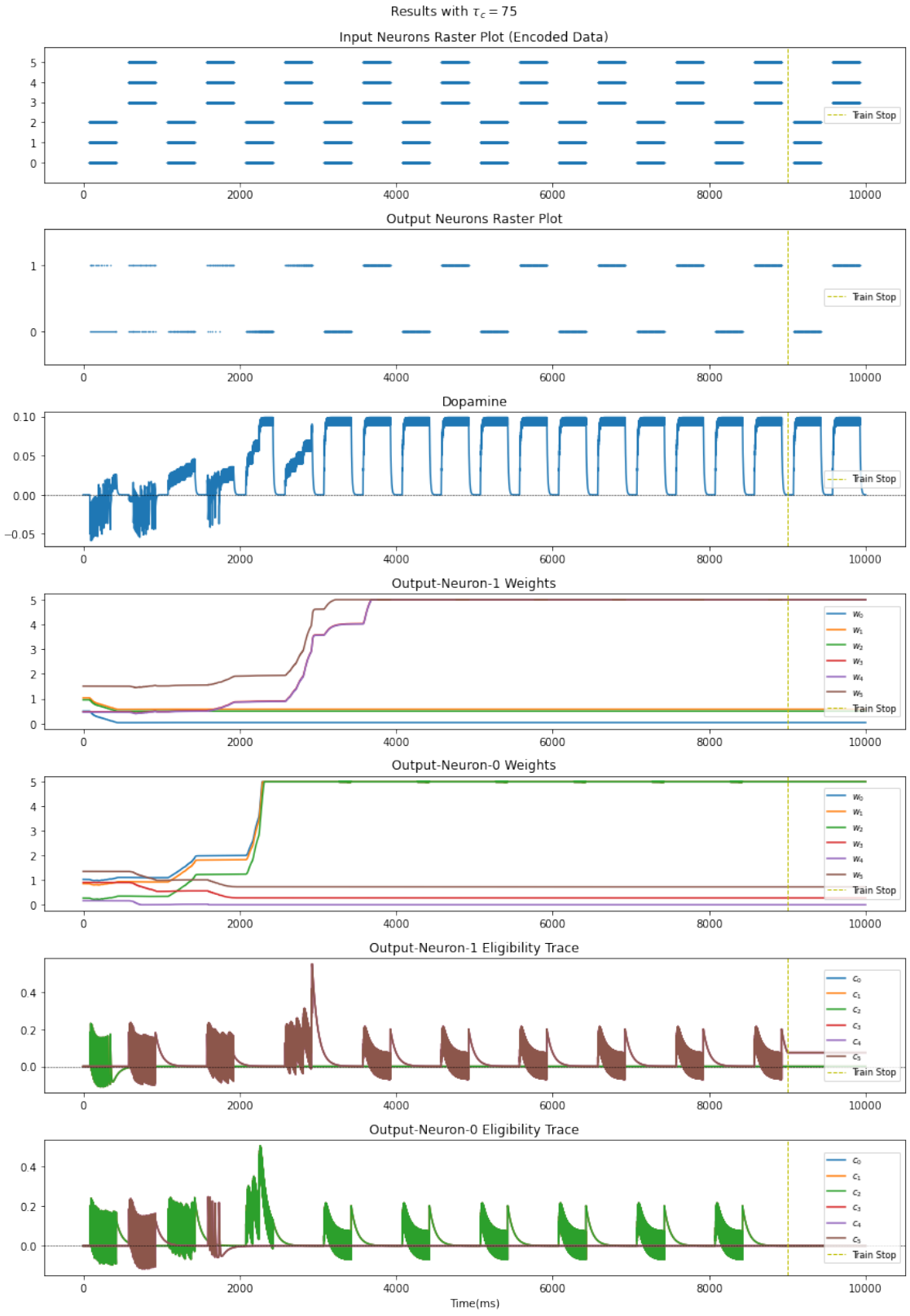
We see that only when $\tau_d = 20$, the learning process is not doing perfectly. In this case only one of the neurons learn its pattern, and the other learns nothing. The reason is that by increasing τ_d, the effect of older dopamine updates might change the weight when it should not. (*e.g.*, the input pattern has been changed). The reason that the weights are not being corrected until the end of the simulation is that the neuron-1 weights has been decreased too much that the neuron cannot be activated anymore. If incorrect weight updates do not make the neuron dead, the weights will be corrected in the next iterations. This effect is clearly visible in the last plot where $\tau_d = 50$.

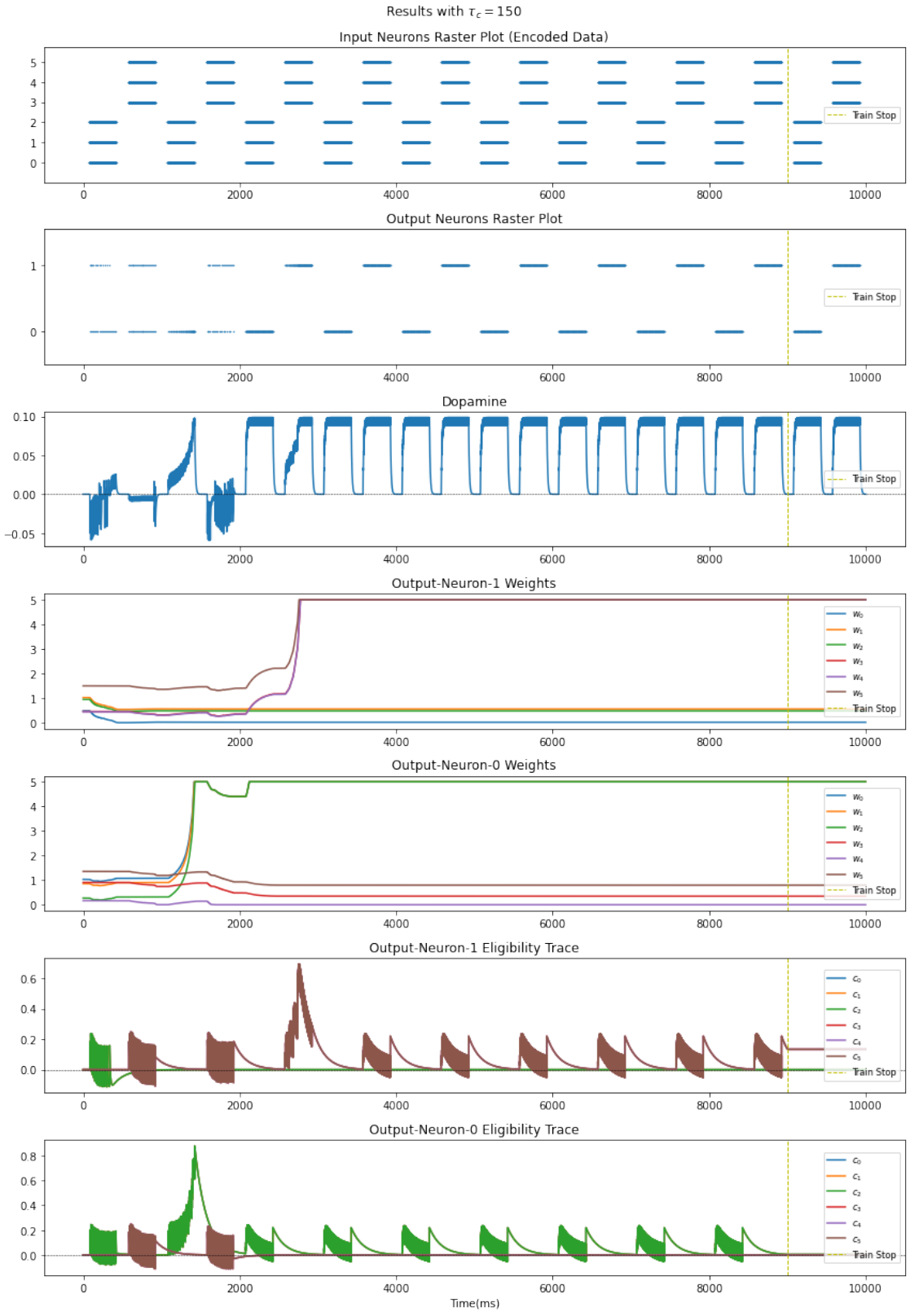
The conclusion is that the lower values of τ_d are better in general. The reason is that the release of dopamine for a single event (seeing one of the patterns for example) should not affect the network's response to another event. When τ_d is low, the value of DA decays quickly and this unwanted situation is prevented.

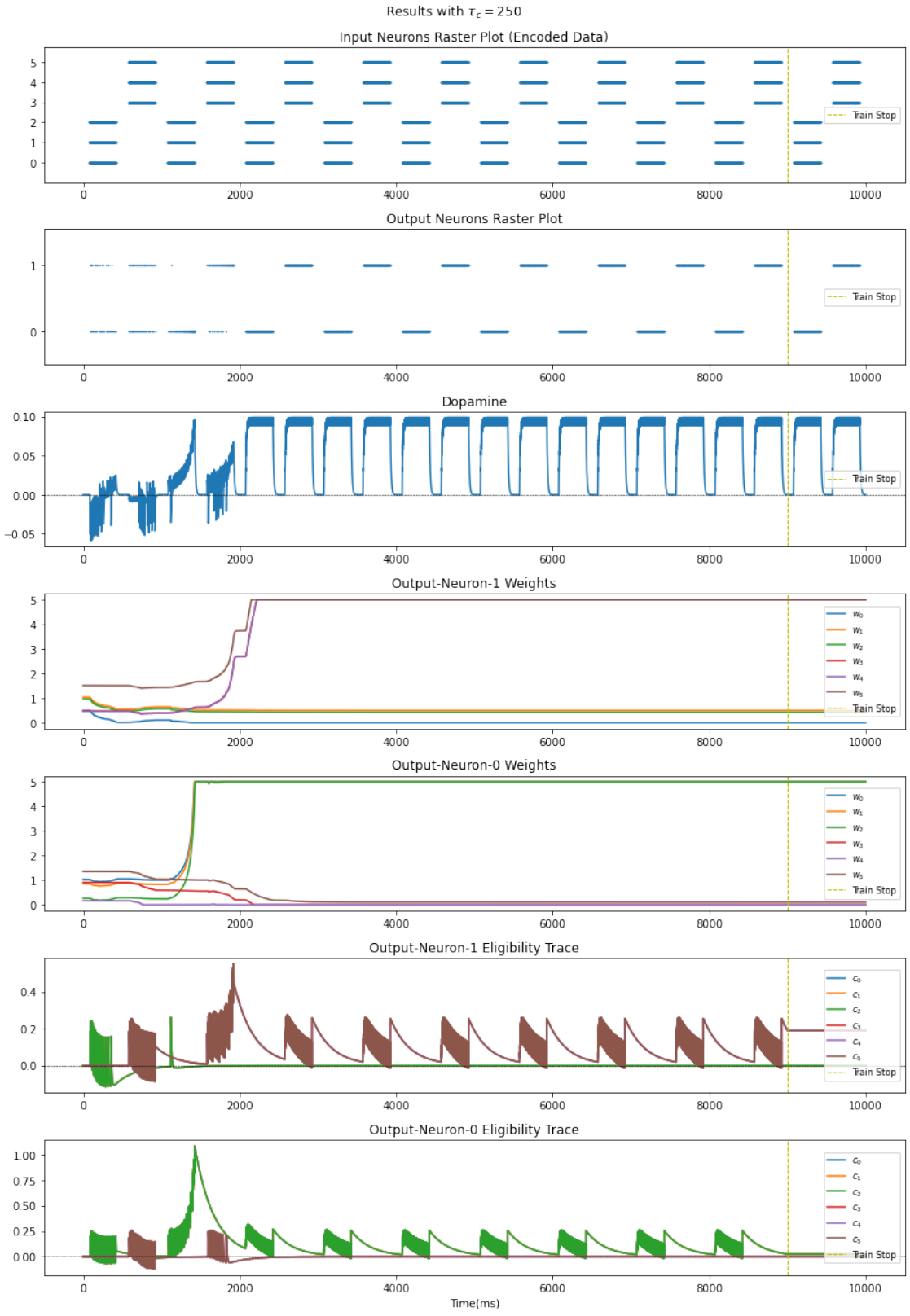
2.4 Experiment #3 (τ_c)

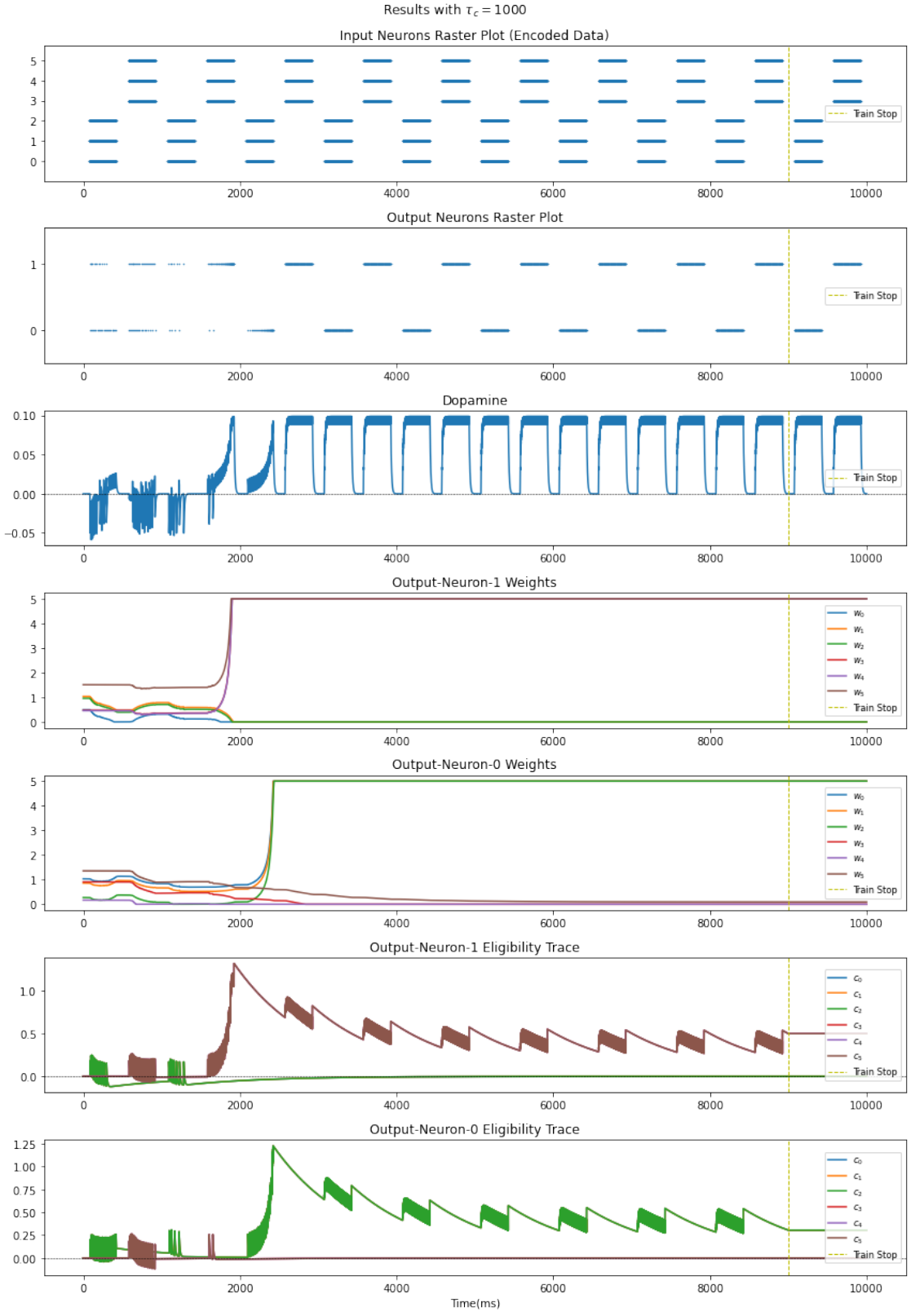
We test different values of τ_c in $[50, 75, 150, 250, 500]$


Results with $\tau_c = 50$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_c = 75$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

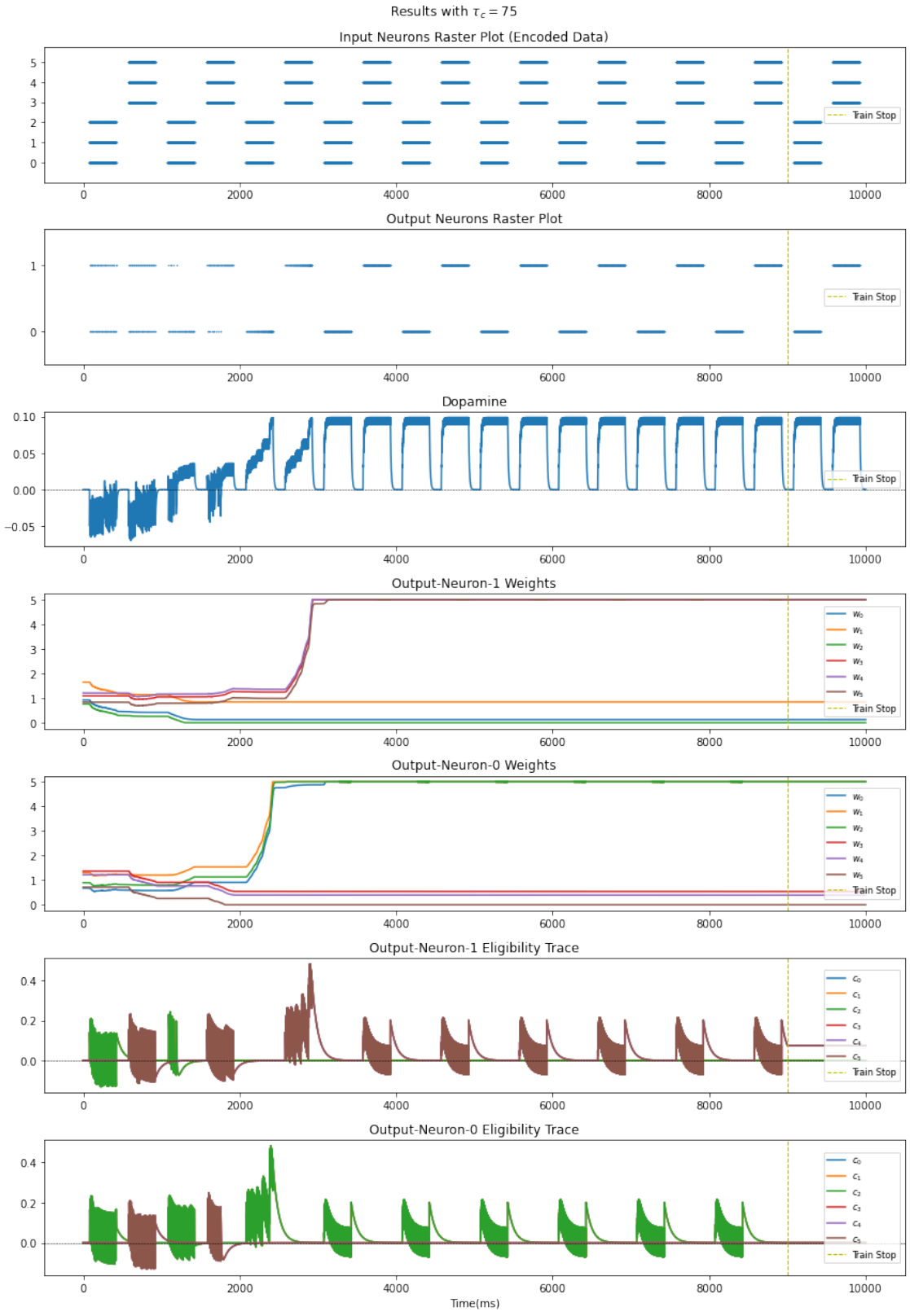

Results with $\tau_c = 150$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

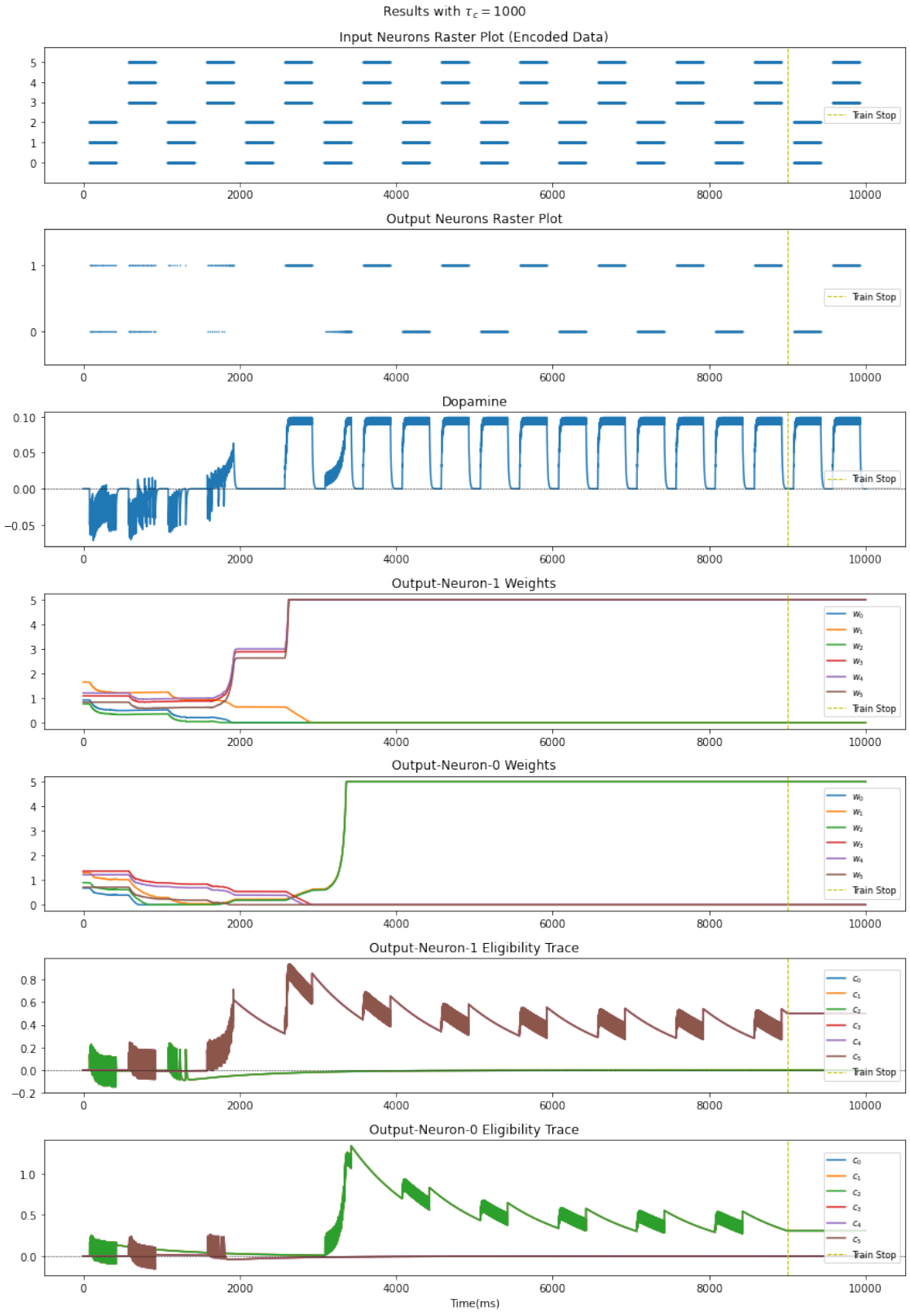

Results with $\tau_c = 250$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

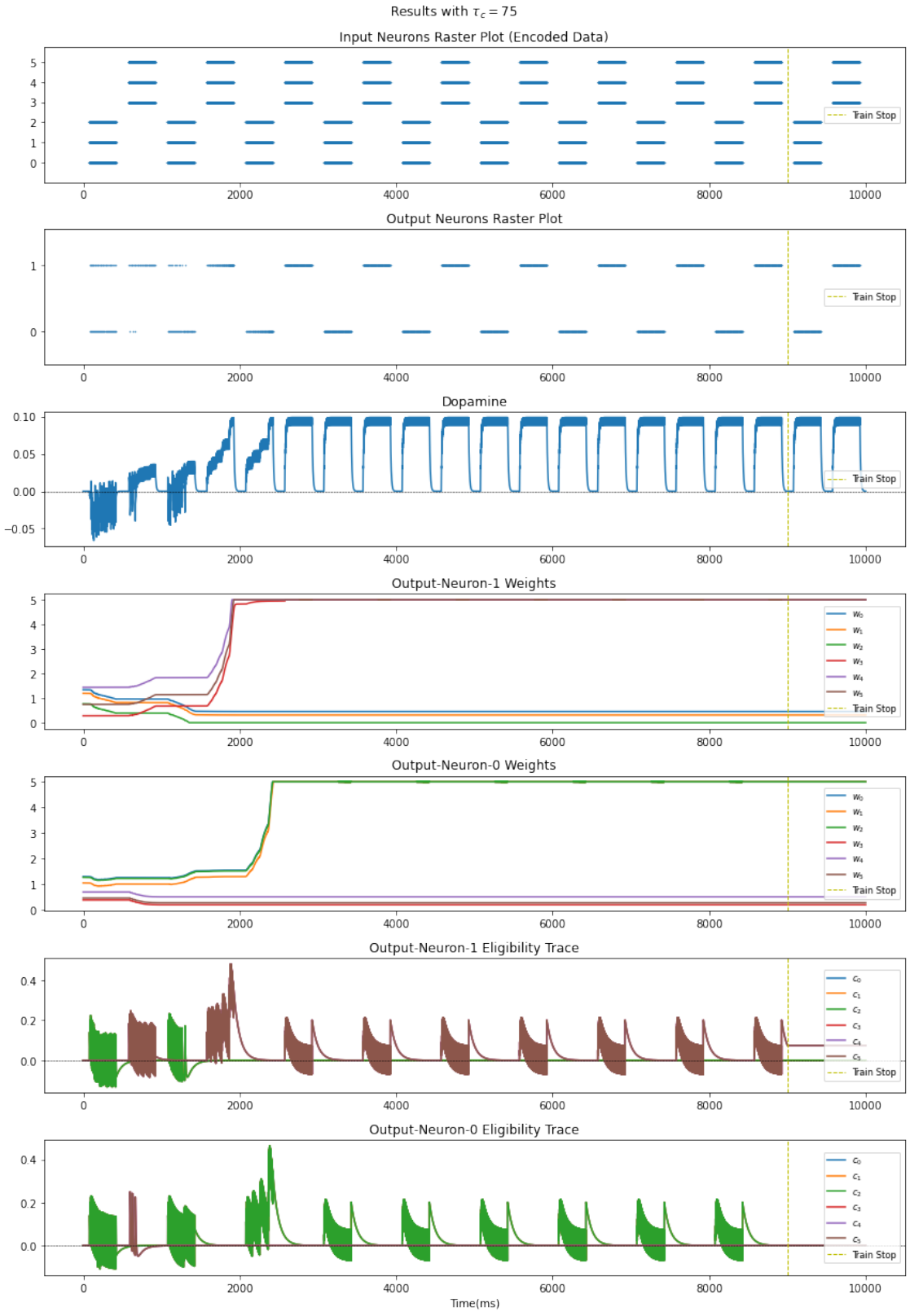

Results with $\tau_c = 1000$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

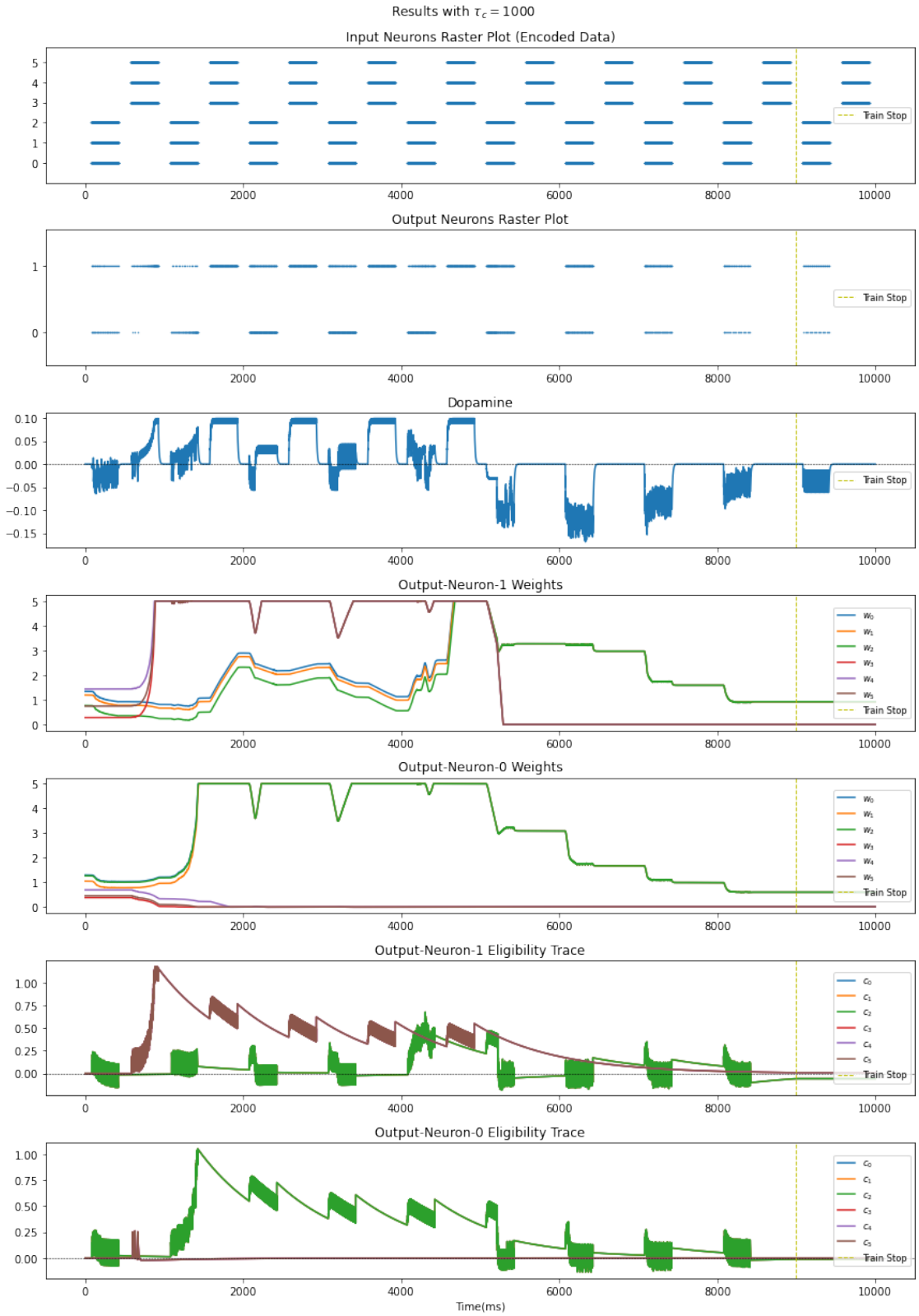
We see that all the outputs are correct. But with repeating the simulation for numerous times we have reached to this conclusion: When $\tau_c >= 250ms$, the possibility of network's fault will fluctuate, but if $\tau_c < 250ms$ the network's output is correct almost certainly. Here is the explanation: each of the patterns are presented to the network for a period of about 500ms, then it switches to next pattern after some delay. When $\tau_c = 250$, the effect of the previous pattern is partially available in `Synaptic Tag`. This unwanted information in eligibility traces might make the learning rule to update the weights incorrectly. Since this pattern is repeated in all the iterations, the wrong direction will not be fixed. But if we increase τ_c is 1000, the eligibility trace has the information of neuron's activity from the last $1000ms$ and in $1000ms$ both of the input patterns are present and may help the learning to perform as expected. However, it is best that τ_c is set to a value that only accumulate the information of just one pattern. Now consider when $\tau_c < 250ms$, in this case, most of the information that is present in `eligibility trace` is relevant to the current input pattern (note that between input patterns we have periods of no-input). This situation helps the learning process to update the weights without error.

Below, you see the simulation repeated to show the above point:


Results with $\tau_c = 75$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_c = 1000$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_c = 75$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

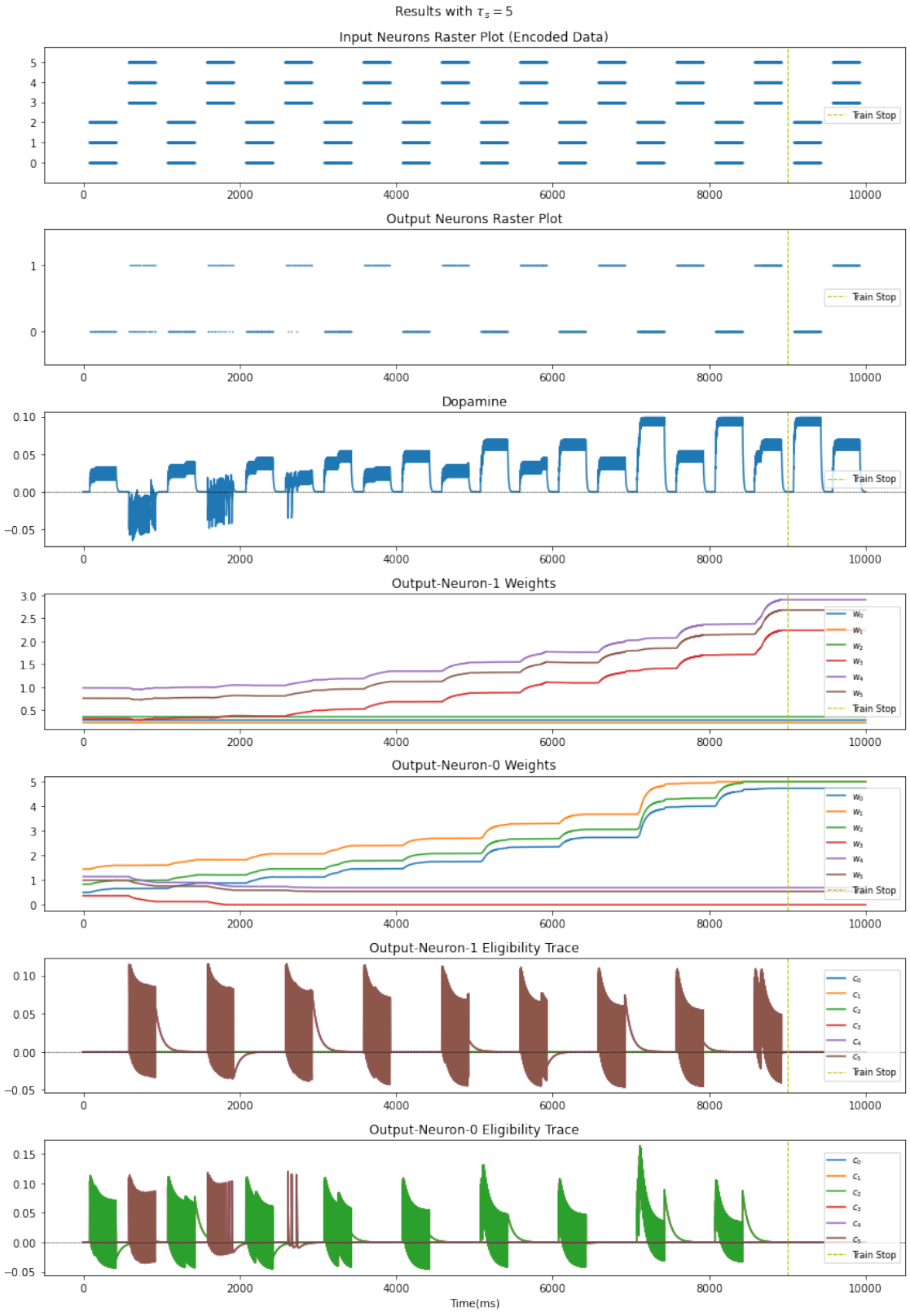


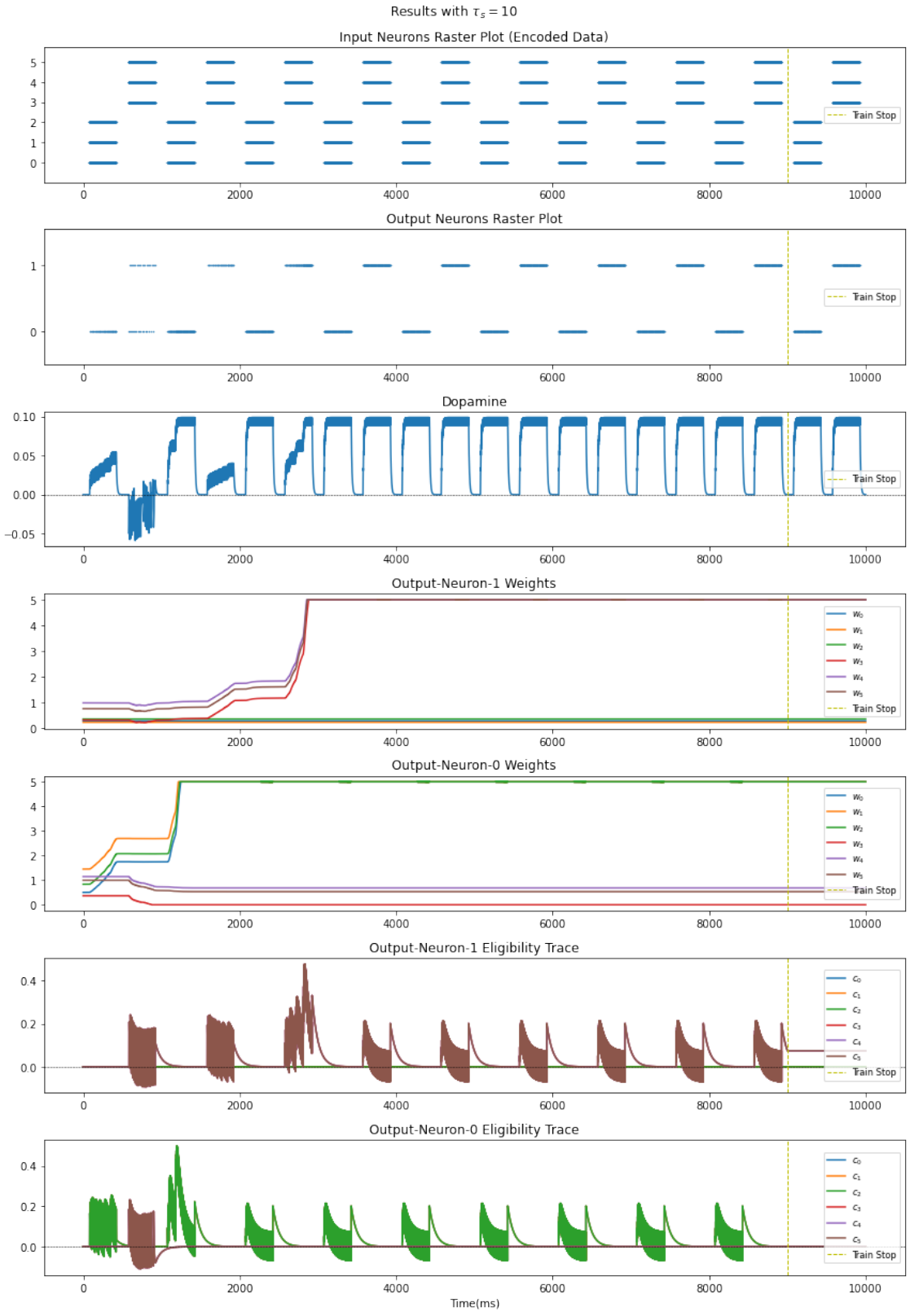
With the first weight initialization both τ_c values work correctly. With the second initialization however,

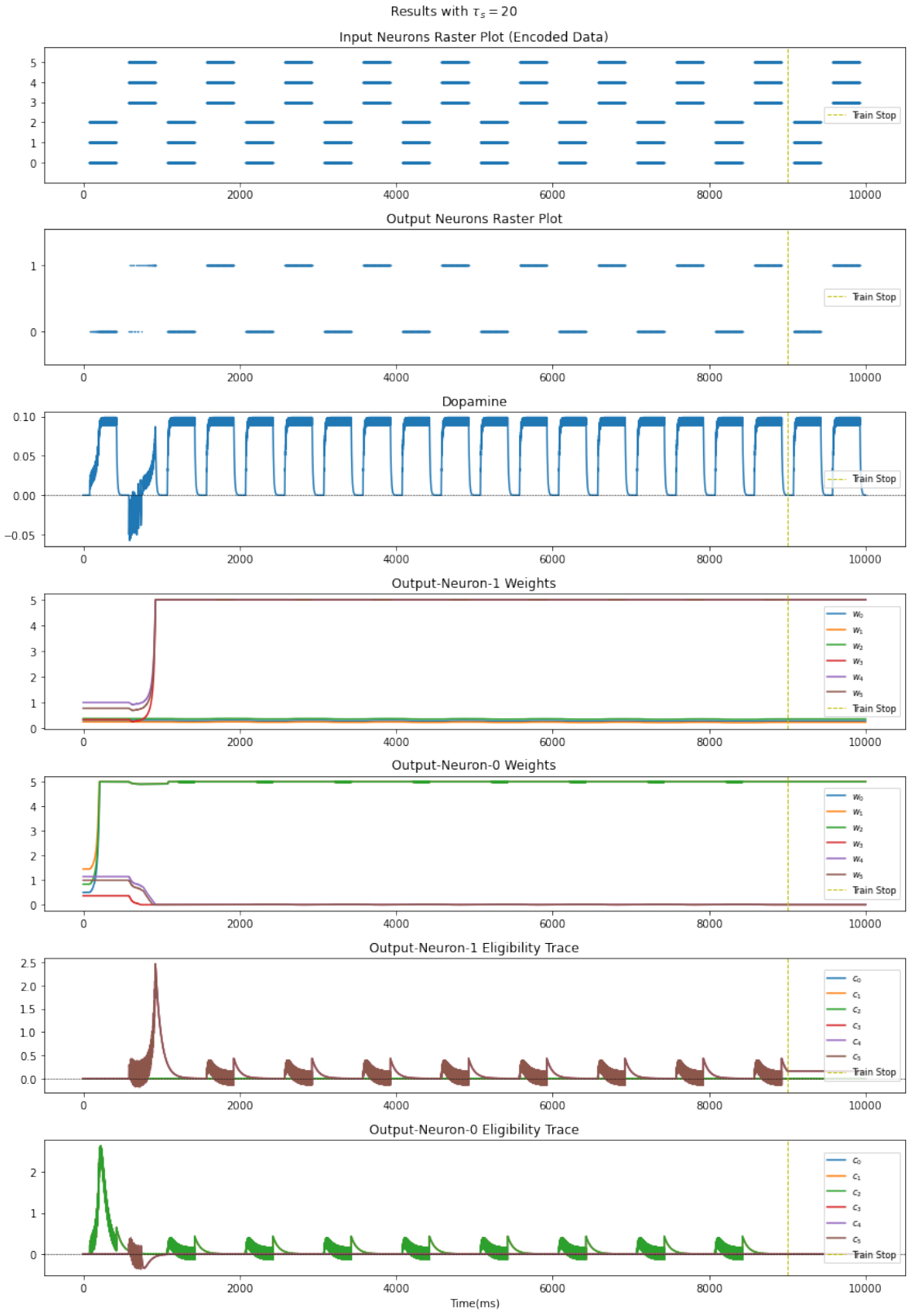
$\tau_c = 1000ms$ fails.

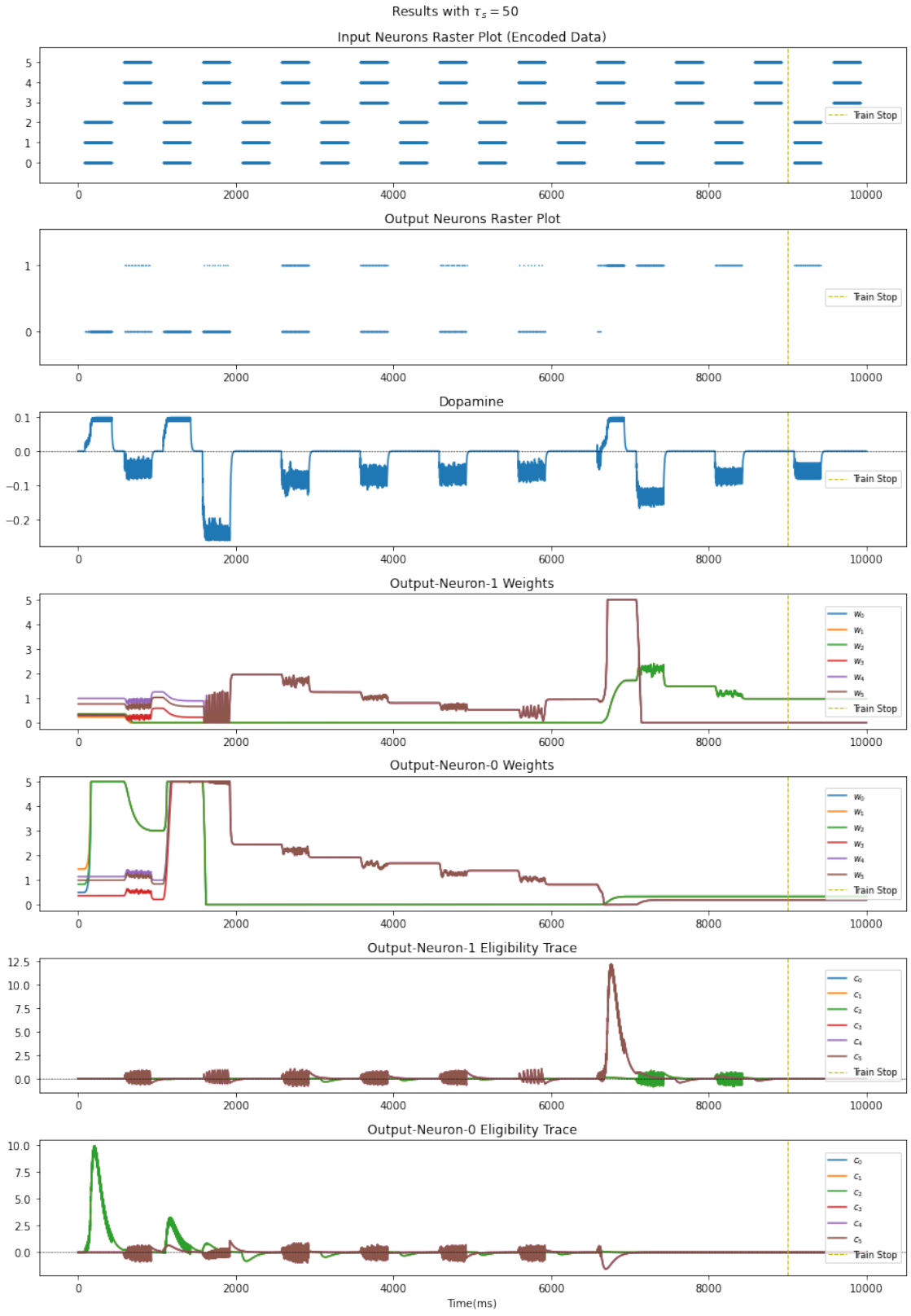
2.5 Experiment # (τ_s)

We test different values of τ_s for network's neurons in $[10, 20, 50]$


Results with $\tau_s = 5$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_s = 10$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_s = 20$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Results with $\tau_s = 50$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)

We can observe that by increasing τ_s, the maximum value of `Synaptic Tag` increases as well. *e.g.*, when $\tau_s = 50$ the maximum value of C is about 10 but when $\tau_s = 10$ the maximum value of C is about 0.5. When C grows to an abnormally high value, $\frac{dw}{dt} = cd$ will have a high value as well. This large amount of weight update potentially disturbs proper learning as the weights might grow or decay unacceptably fast. This parameter could somewhat act as learning rate here because by growing it, weight updates become more intense. Therefore, it's better to set τ_s to a low value such as 10.

Chapter 3

Flat-R-STDP

For Flat Reward Modulated STDP we implement the following update rule for synaptic weights:
$\frac{dw}{dt} = FlatSTDP(\tau)\delta(t - t_{pre/post}) * d$

3.1 Default Parameters

Train Params:

$Time_{simulation} = 10000ms$

$LearningRates = [0.03, 0.3], \tau_d = 10ms \rightarrow$ DA Policy #1 $\quad LearningRates = [0.03, 0.3] \rightarrow$ DA Policy #2

Neurons Params:

$Num(PresynapticNeurons) = 6$
$\tau_s = 10ms$
$Threshold = -42.5mv$
$U_{rest} = -60mv$

Connection Params:

$J_0 = 5$
$\sigma_0 = 3$
$Weight_{min} = 0$
$Weight_{max} = 5$

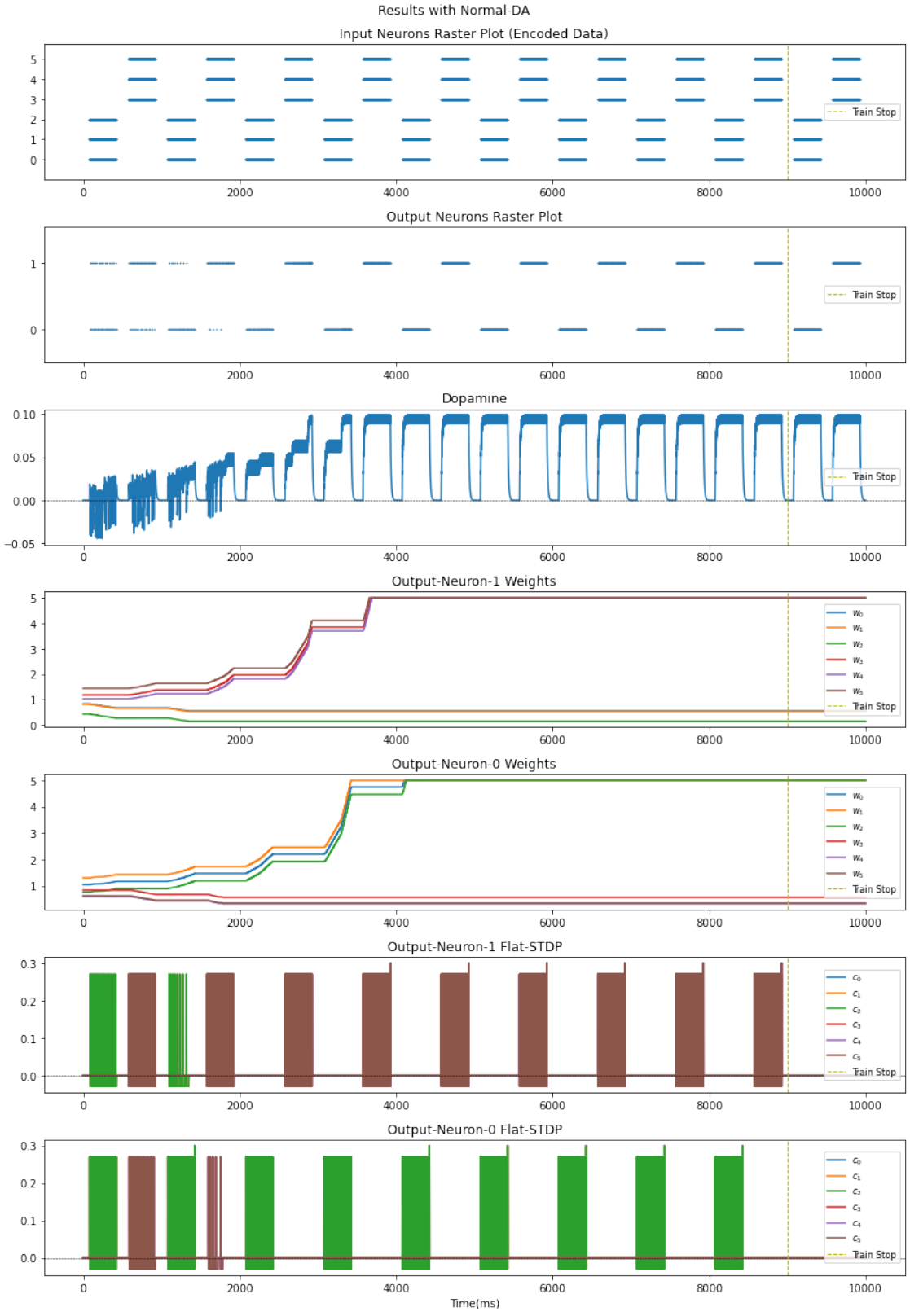
3.2 Experiment #1 (DA policies)

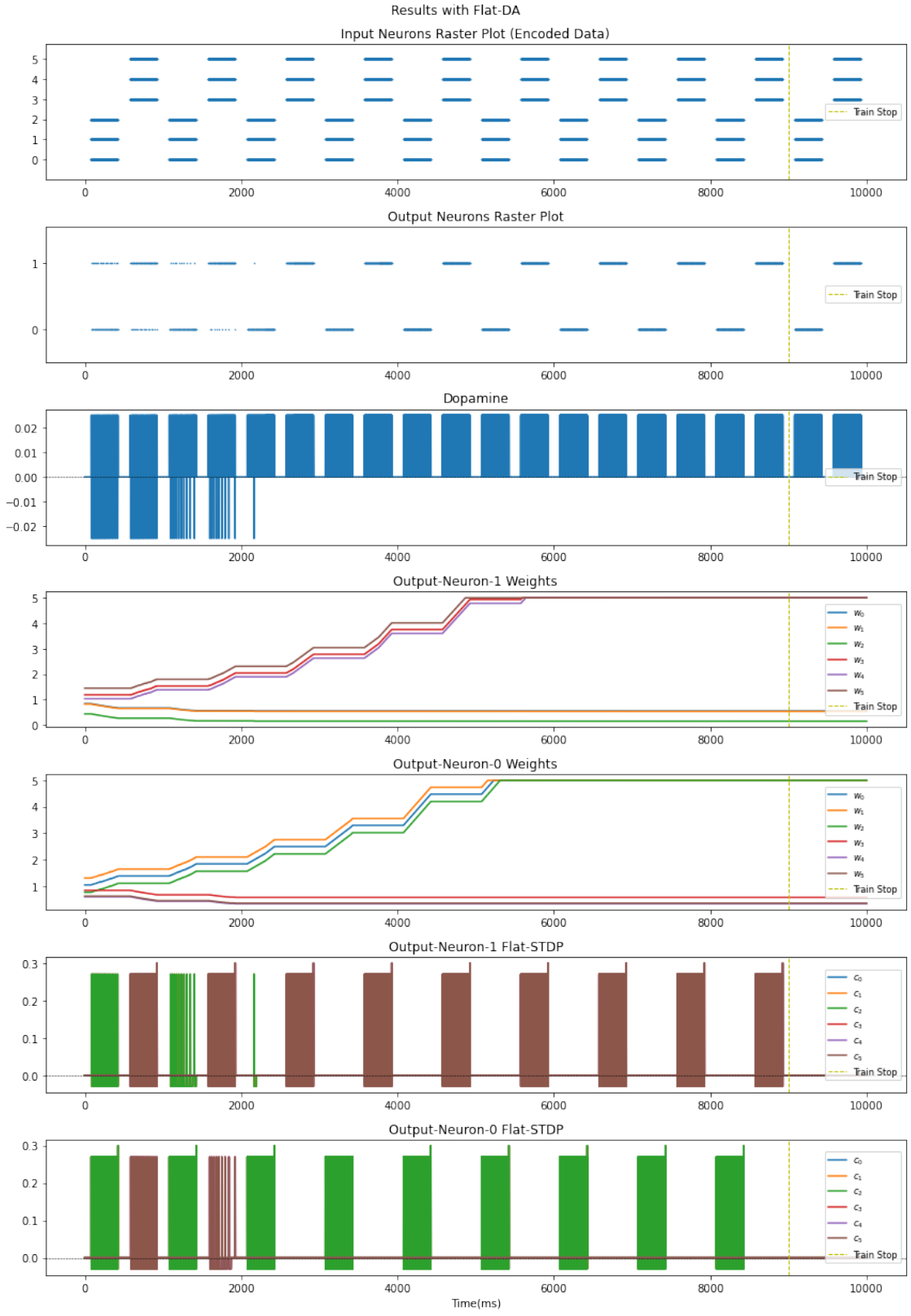
For DA term `d`, we investigate two different approaches:

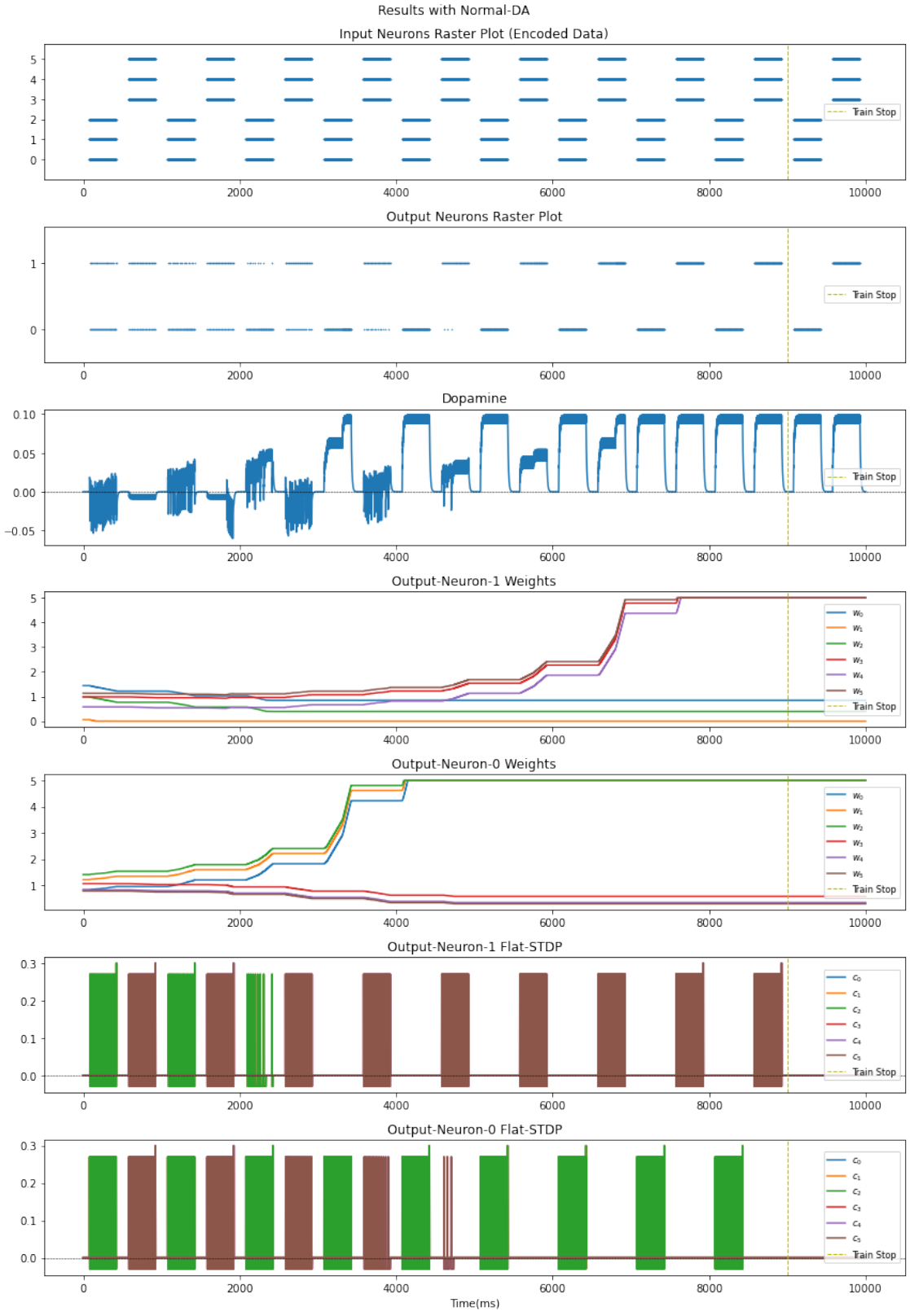
1. DA acts as before and the dynamic and policy of DA(t) updates are not changed.
2. We also apply flatting strategy to DA, so the decay is removed and $d = DA(t)$ is used. In this case, unlike `R-STDP`, we use equal punish/reward $DA(t)$ values; *i.e.*, $DA(t) = 1$ if the neurons activate as expected, $DA(t) = -1$ if the activities are not what we expect (both neurons are firing, or the opposite neuron is firing), $o.w. DA(t) = 0$.

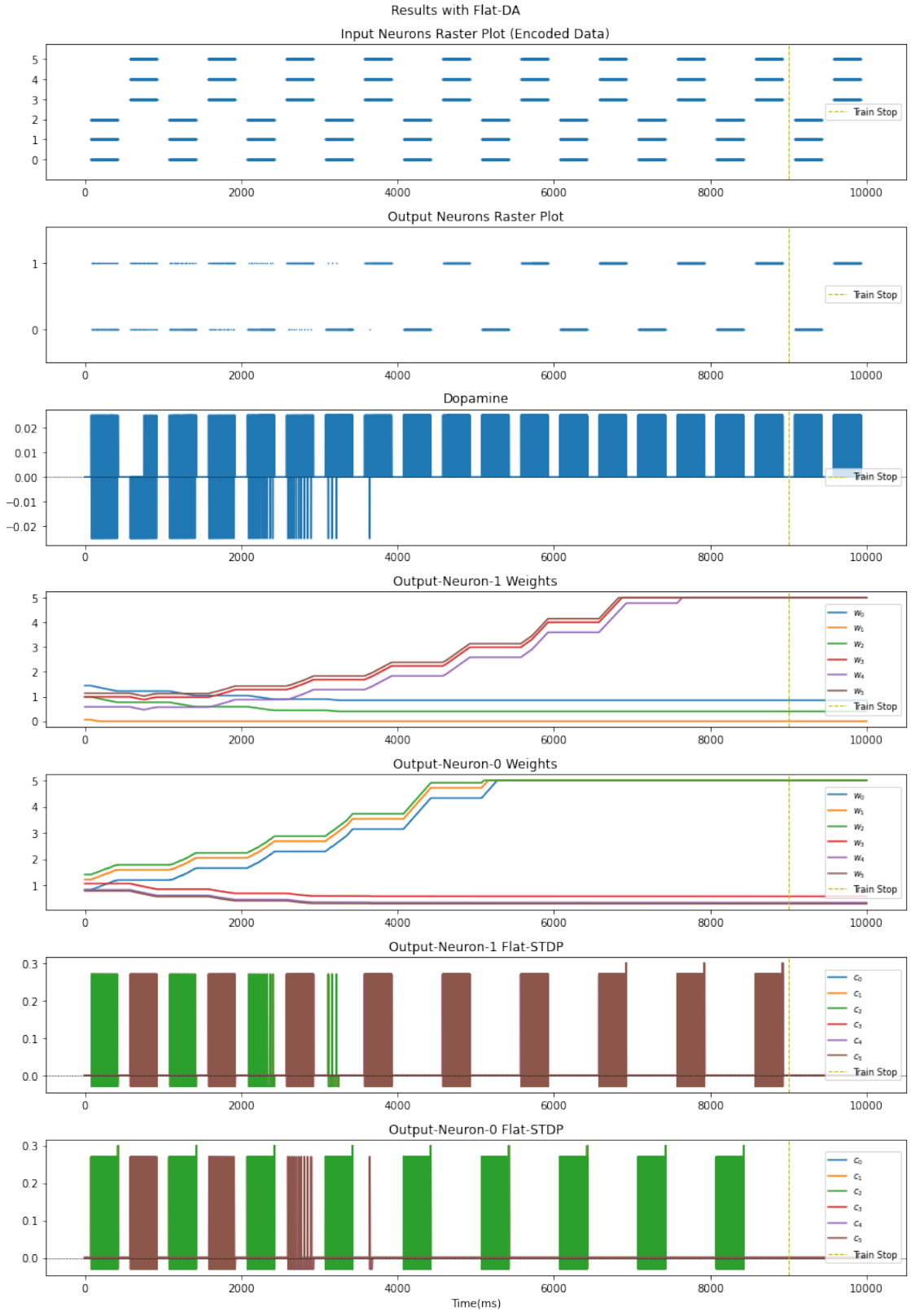
The second option only changes the polarity of STDP; when dopamine is positive, Flat-STDP is applied, when dopamine is negative the reverse Flat-STDP is applied.

We run the simulation with both policies with two different weight initializations. $A_- = 0.03, A_+ = 0.3$ in both cases.


Results with Normal-DA
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)


Results with Flat-DA
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)

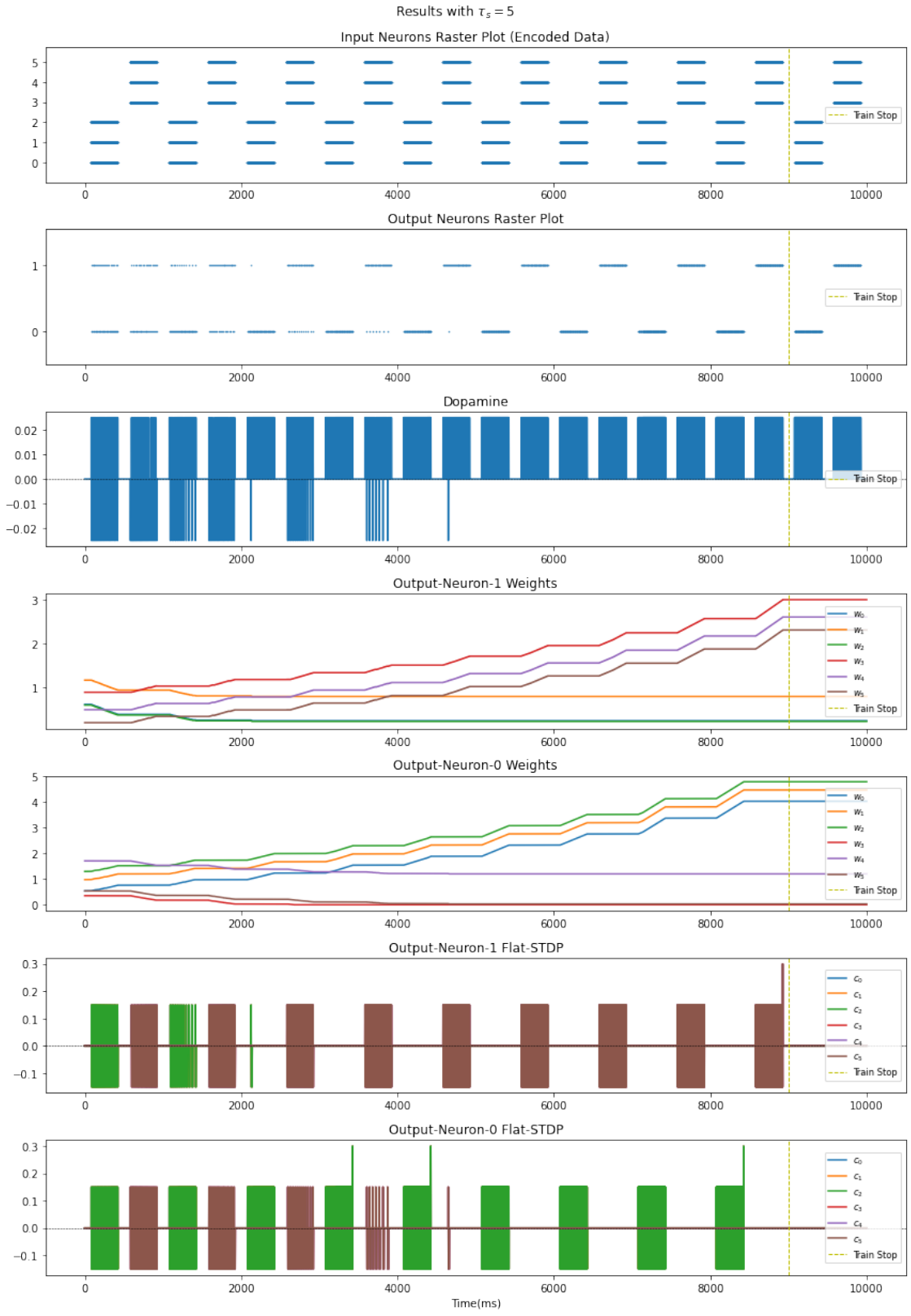

Results with Normal-DA
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)

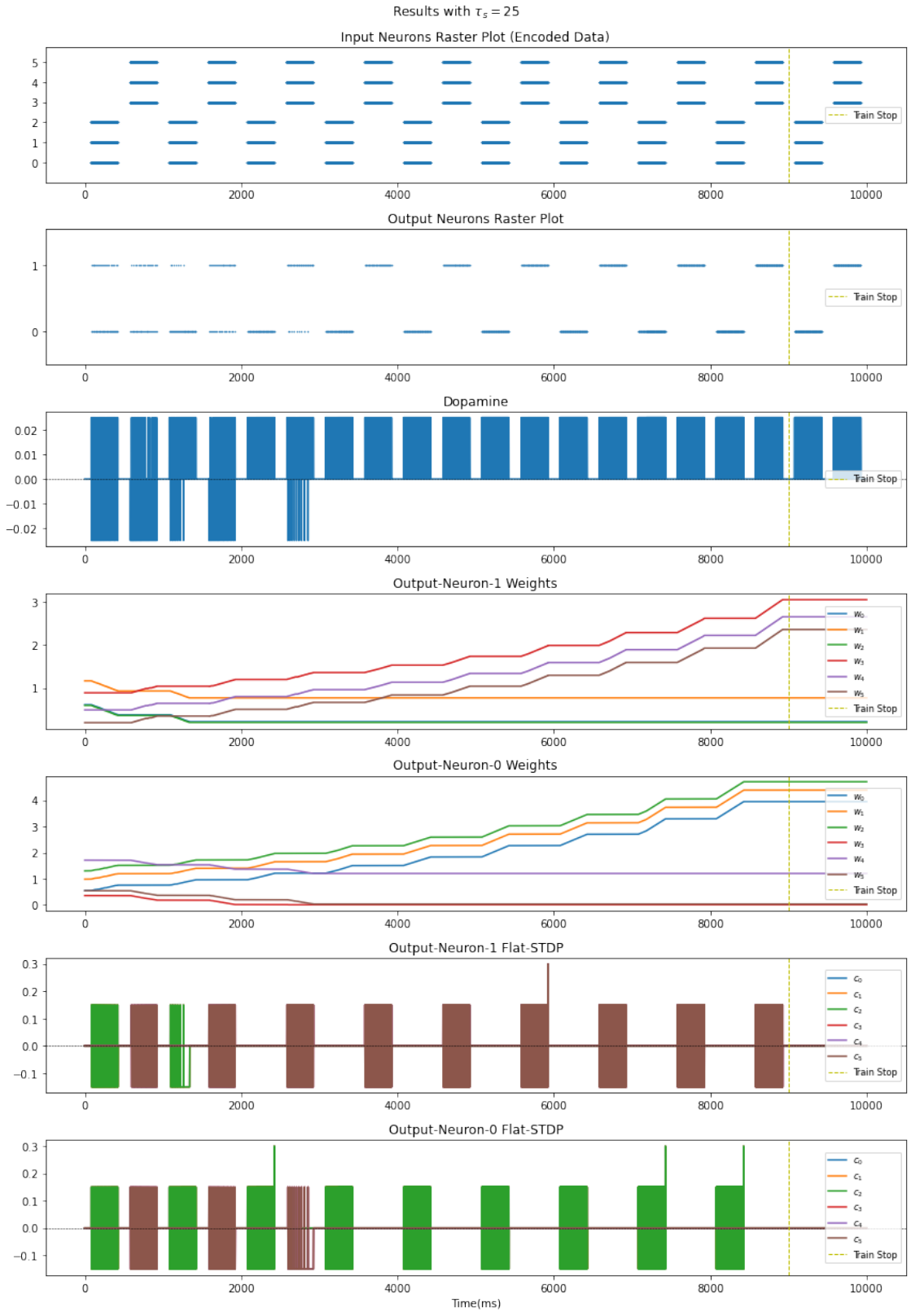

Results with Flat-DA
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)

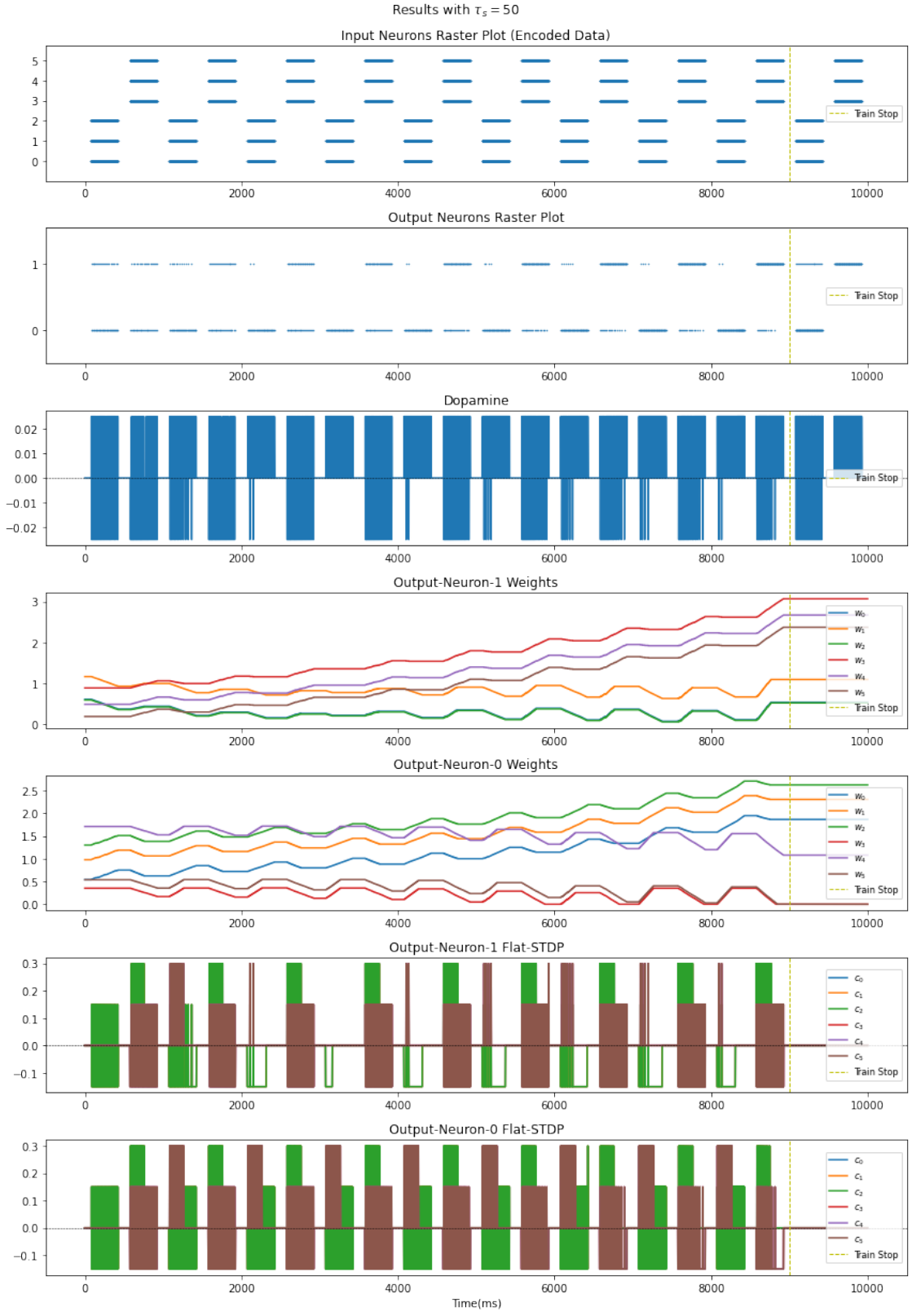
By observing the above plots we conclude that the Flat-DA policy performs very similar to Normal-DA. So the dynamics of Normal-DA are unnecessary here and could be ignored completely. The following simulations will use Flat-DA as it is simpler and its performance is just as good as Normal-DA.

3.3 Experiment #2 ($WindowSize_{FlatSTDP}$)

Due to our implementation of FlatSTDP (binarizing the traces before applying), we manipulate the $WindowSize_{FlatSTDP}$ using τ_s of neurons. The bigger τ_s is, the bigger $WindowSize_{FlatSTDP}$ becomes.


Results with $\tau_s = 5$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)


Results with $\tau_s = 25$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)

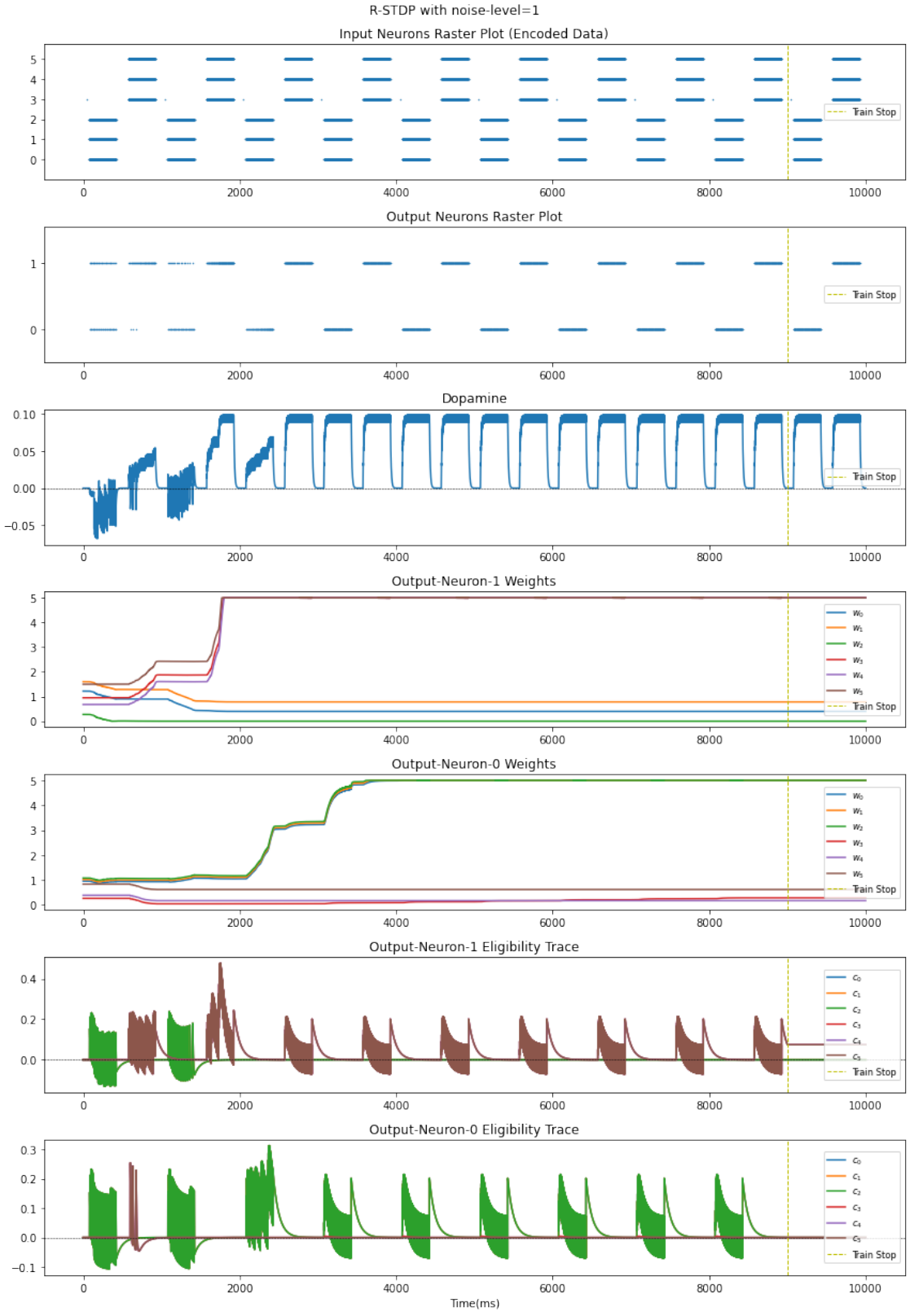

Results with $\tau_s = 50$
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)

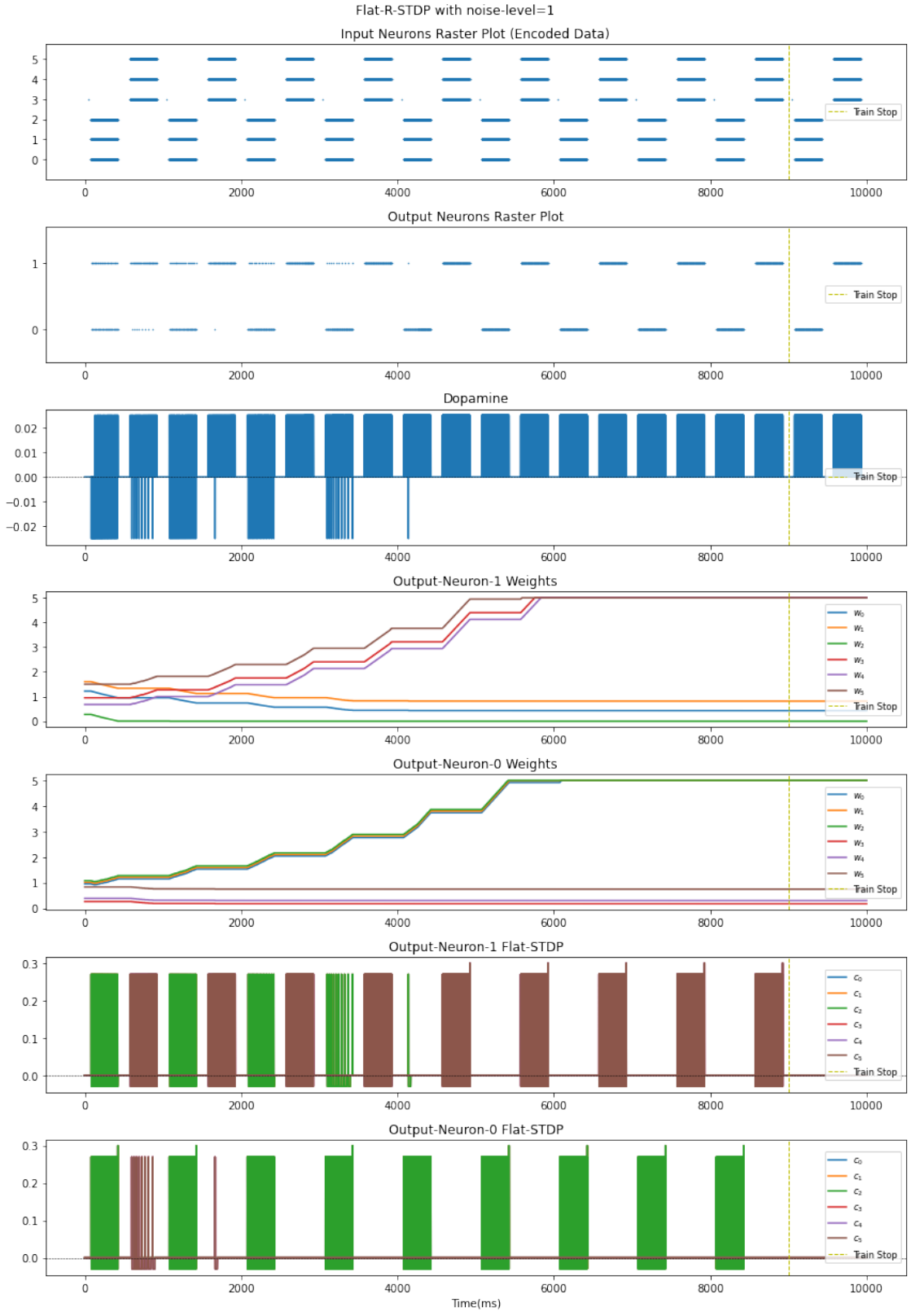
The results match with the ones we found with R-STDP. Only when $\tau_s = 50$ the learning has failed to converge to the desired output. This is because old spikes of neurons are affecting the weight updates which is not desired. Before that point, the learning is working flawlessly.

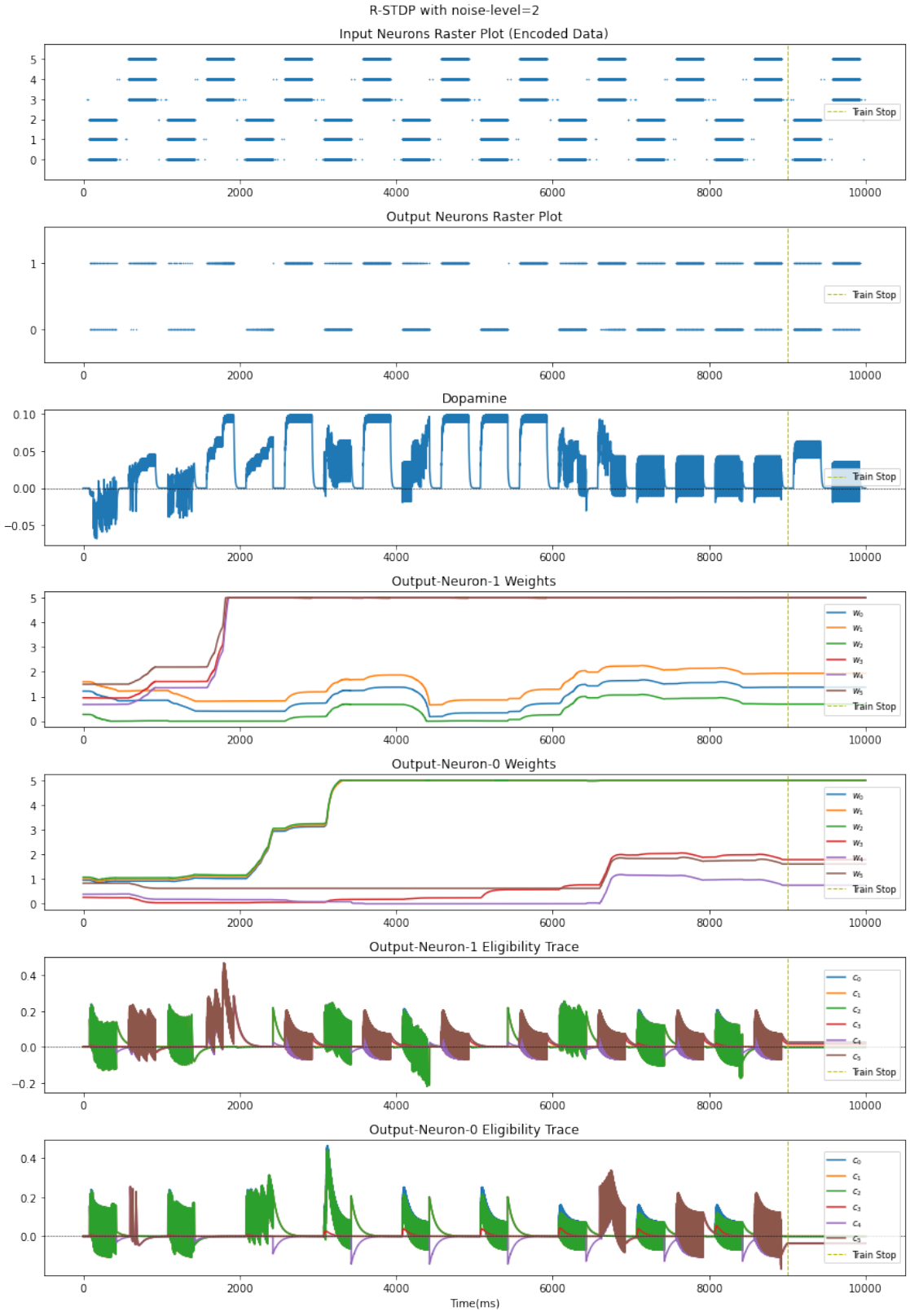
Chapter 4

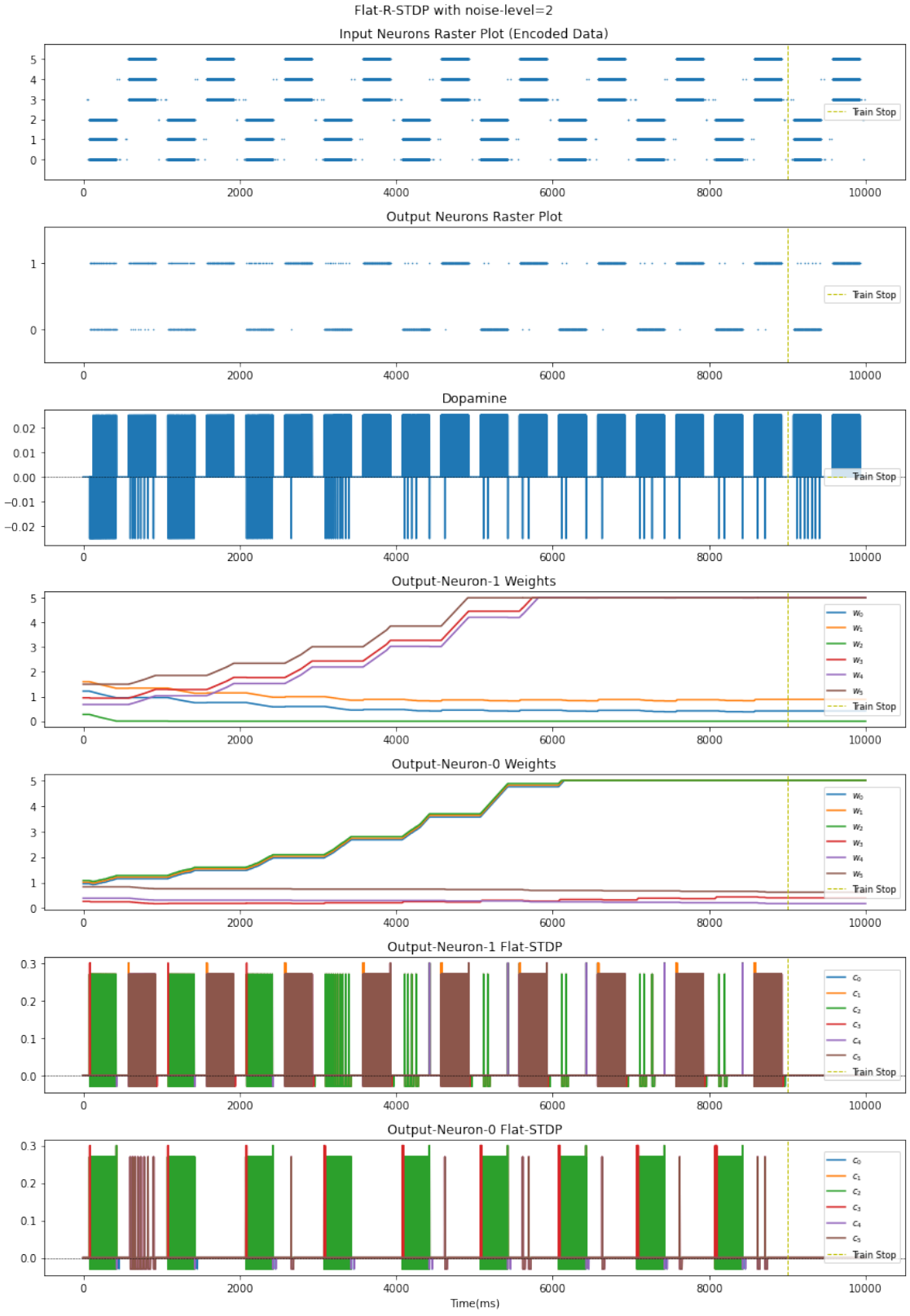
Comparison

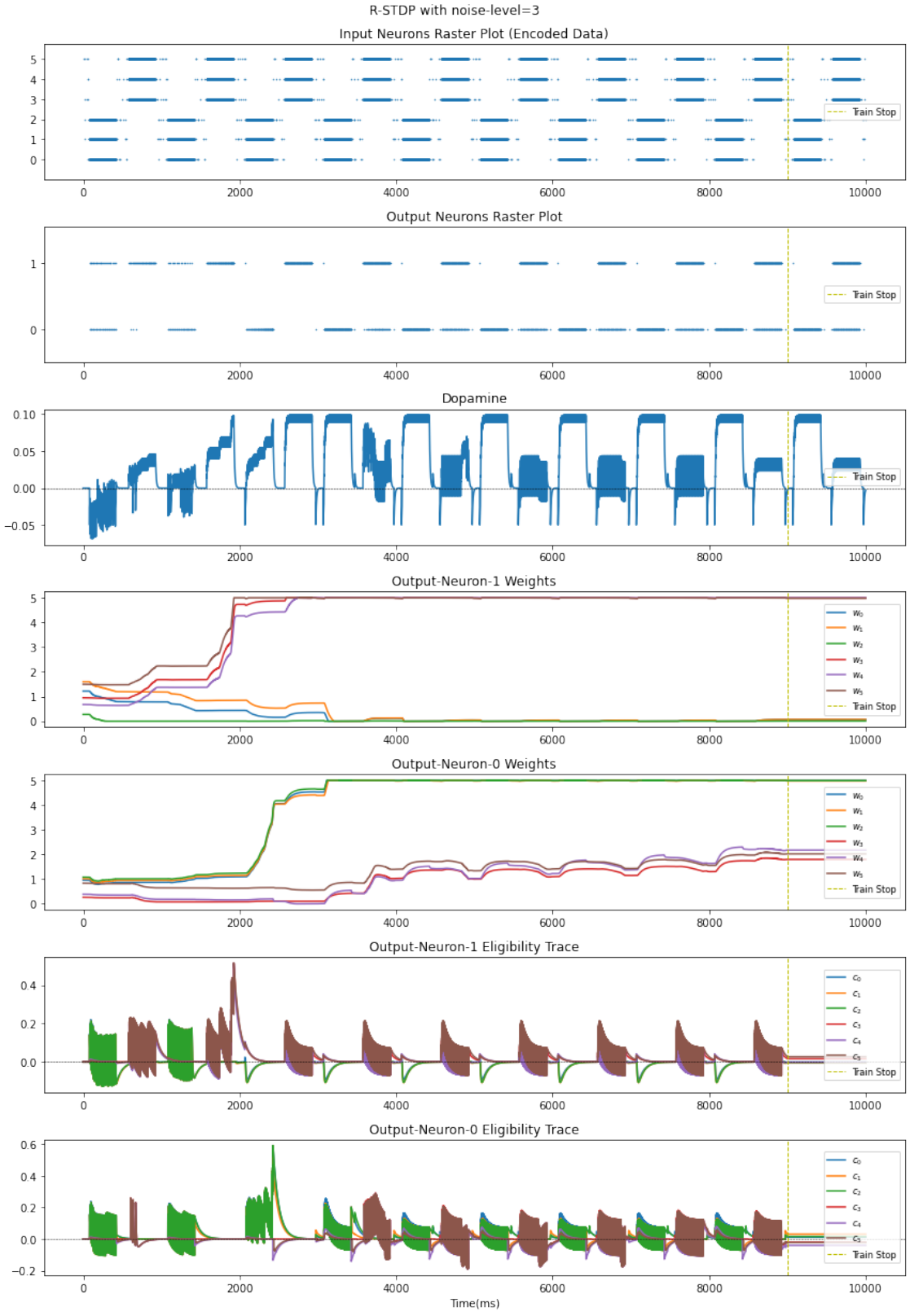
In this section, we add noise in three levels to the input and run the learning simulation using both `R-STDP` and `Flat-R-STDP` to test their robustness to noise.

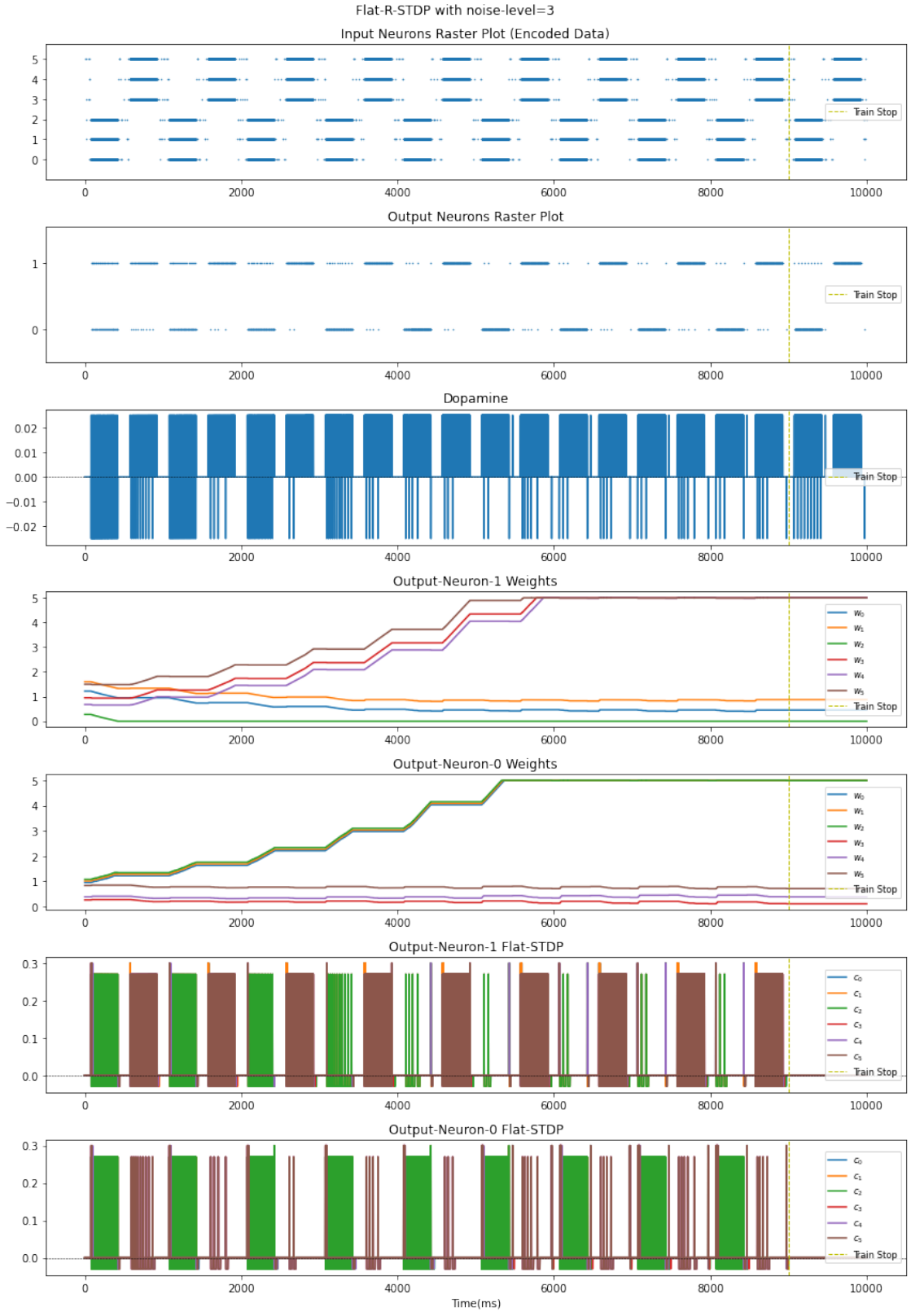

R-STDP with noise-level=1
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Flat-R-STDP with noise-level=1
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)


R-STDP with noise-level=2
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Flat-R-STDP with noise-level=2
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)


R-STDP with noise-level=3
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Eligibility Trace
Output-Neuron-0 Eligibility Trace
Train Stop
Time(ms)


Flat-R-STDP with noise-level=3
Input Neurons Raster Plot (Encoded Data)
Output Neurons Raster Plot
Dopamine
Output-Neuron-1 Weights
Output-Neuron-0 Weights
Output-Neuron-1 Flat-STDP
Output-Neuron-0 Flat-STDP
Train Stop
Time(ms)

It is visible that **Flat-R-STDP is much more robust to noise**. When $NoiseLevel = 1$, both rules are performing correctly. When $NoiseLevel = 2$ and $NoiseLevel = 3$, `R-STDP`'s results are completely ruined but `Flat-R-STDP` results are still acceptable because only by measuring their activity we can predict which input pattern is presented to the network.

Chapter 5

Summary

1. If the dopamine update-terms ($DA(t)$) update the Dopamine concentration level in an unequal manner (presented in a table in the introduction section), the `R-STDP` performs better in most of the cases with more success rate.

2. In `R-STDP`, τ_d should be set to a small value; *e.g.*, 5 or 10. If this parameter becomes too high, dopamine updates from old activities will affect the weight updates, and this might ruin the learning process.

3. In `R-STDP`, τ_c should be set to a value that encodes the neurons' previous activity information. It is best that the information mostly consists of the pattern the network is currently learning; *o.w.*, the learning might get corrupted.

4. `Flat-R-STDP` with Flat-DA performs as good as Normal-DA; so, it is better to use Flat-DA since it is simpler, faster, and it does not require unequal $DA(t)$ for best performance.

5. `Flat-R-STDP` with Flat-DA does not need any setting for τ_c and τ_d.

6. `Flat-R-STDP` is much more robust to noisy input compared with `R-STDP`.

7. `Flat-R-STDP` is undoubtedly the better option in many use-cases as it requires very little parameter tuning and is easier to work with.

8. Both `R-STDP` and `Flat-R-STDP` performs considerably better than their non-reward-modulated counterparts with much less need for parameter tuning. It is because simple `STDP` is an unsupervised learning rule, in contrast `R-STDP` is a reinforcement learning rule and do not act as blindly as `STDP`.