گزارش پروژه سوم درس علوم اعصاب محاسباتی

محراب کلانتری – 99222088

در این پروژه، هدف پیاده سازی الگوریتم های یادگیری STDP و RSTDP است.

در این پروژه، در ابتدا با توجه به پروژه های قبل، موارد مورد نیاز را پیاده سازی کردیم :

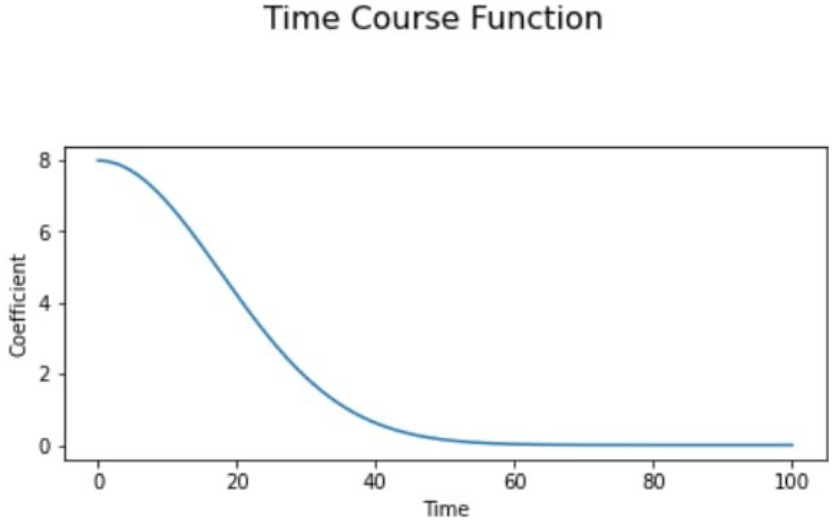
1) کلاس نورون که از مدل LIF برای پیاده سازی آن کمک گرفتیم.

2) کلاس Population که در آن یک جمعیت full connective را پیاده سازی کردیم که با وزن های سیناپسی به هم متصل هستند و قرار است یادگیری STDP بر روی آنها اعمال شود.

3) کلاس SNN که یک شبکه full connective از نورون های LIF است که برای RSTDP آنرا پیاده سازی کردیم و قرار است یادگیری تقویتی بر روی آن انجام شود و سپس بتوان آنرا test کرد.

روش کد کردن اطلاعات :

برای کد کردن اطلاعات از روش Rate Coding استفاده شده است و جریان ورودی به هر نورون post synaptic با استفاده از یک تابع time course و بصورت تجمیعی باتوجه به فعالیت نورون pre synaptic تعیین میشود.



بطور کلی، این روش کدگذاری، فعالیت نورون های pre synaptic را که در طول زمان تجمیع شده و با ضریبی که باتوجه به تابع بالا تعیین میشود، به نورون post synaptic میدهد.

بخش اول پروژه

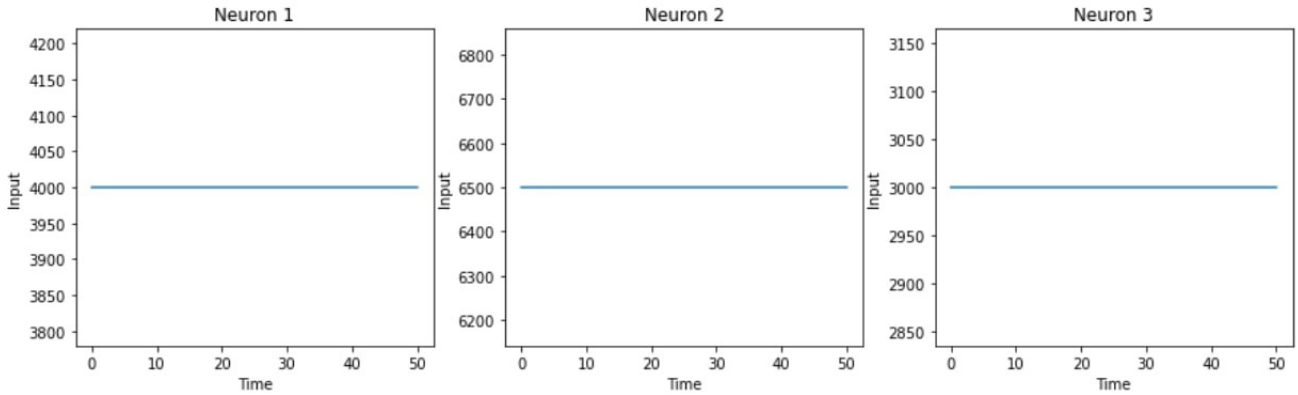
در این بخش میخواهیم STDP Learning را پیاده سازی کنیم.

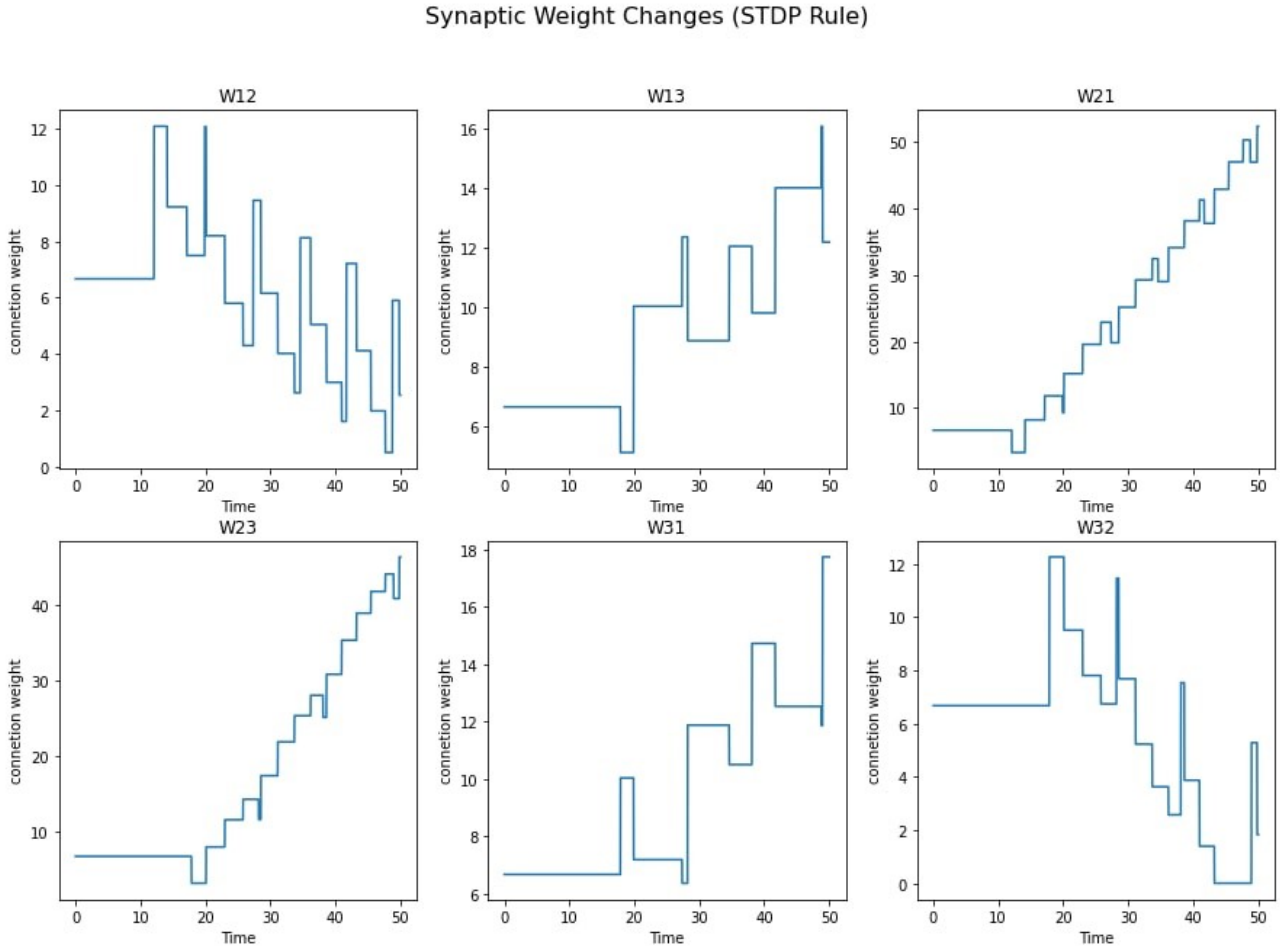
برای پیاده سازی آن از Pair Based STDP استفاده کردیم.

در ابتدا سه نورون را به صورت Full Connective بهم متصل کرده و سه نوع جریان مختلف به این مدل میدهیم که بصورت زیر آمده است.

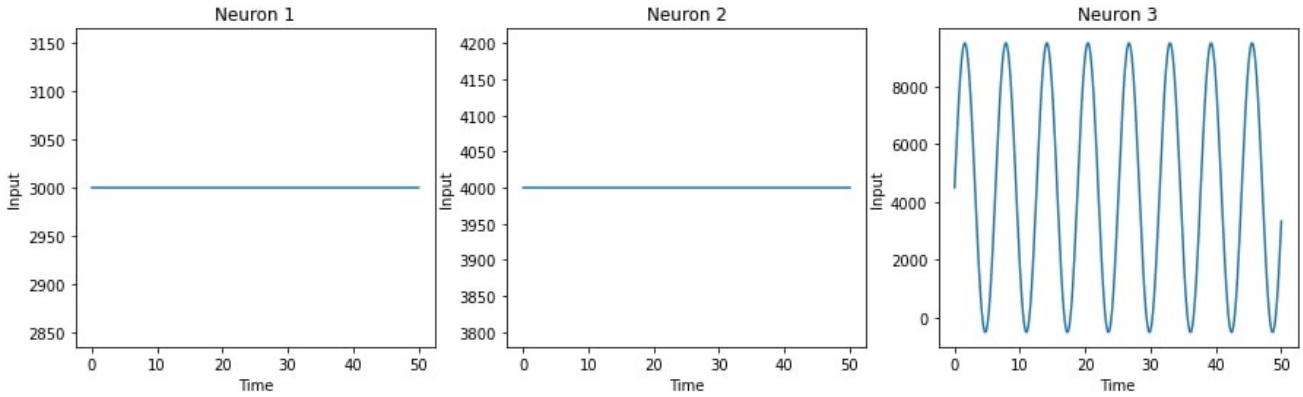
(نورون ها همه از نوع LIF هستند)

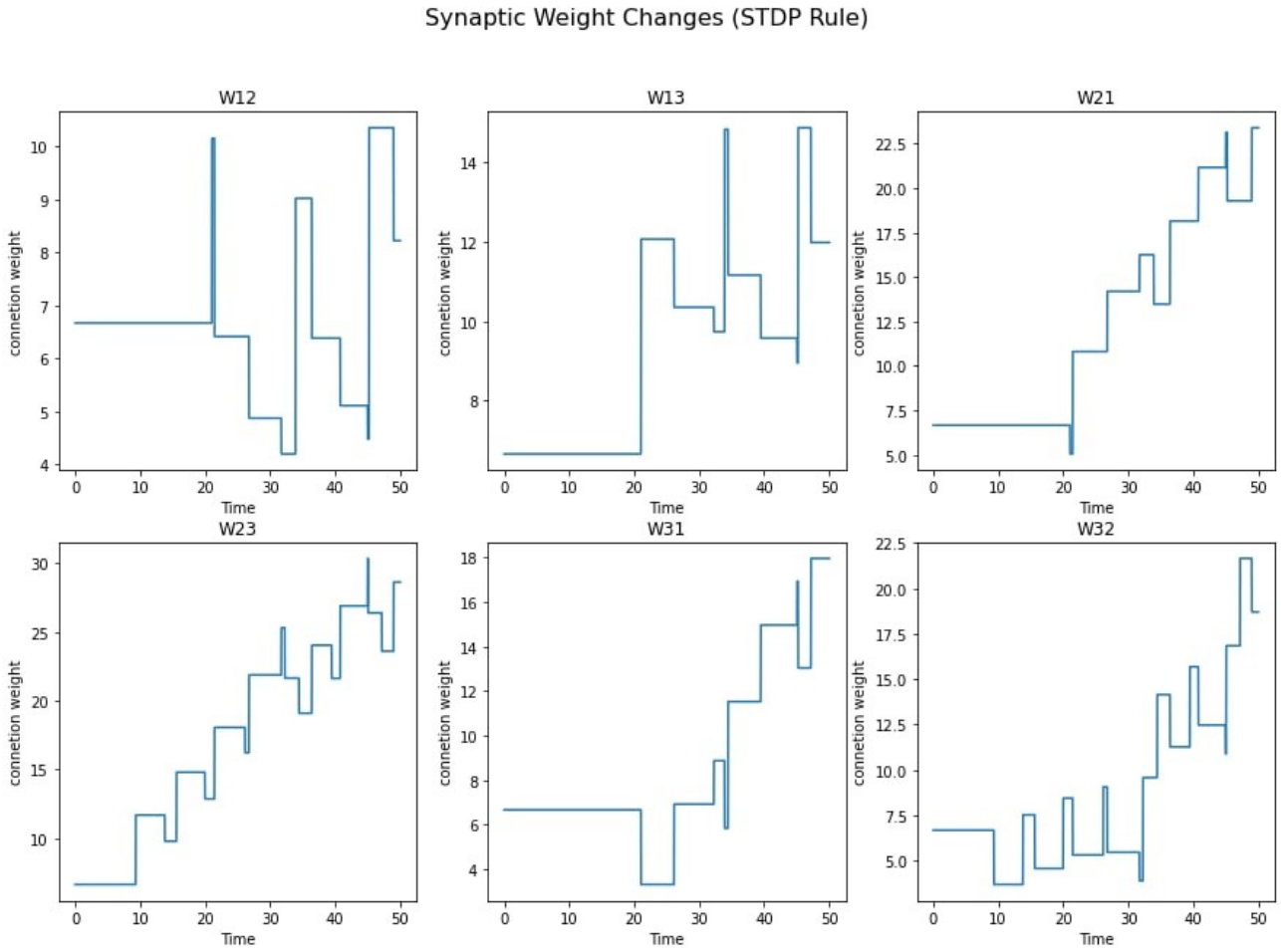
نوع اول) سه جریان ثابت و متفاوت



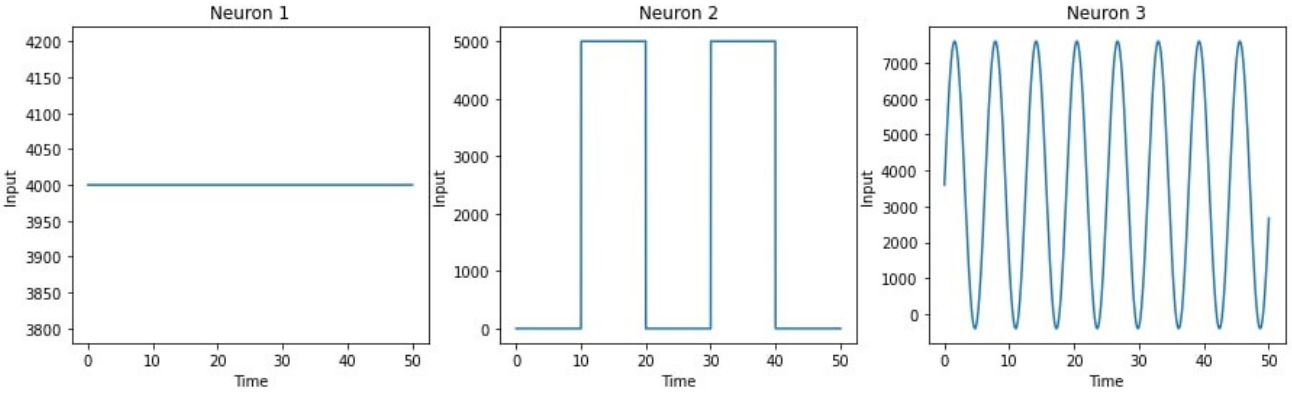


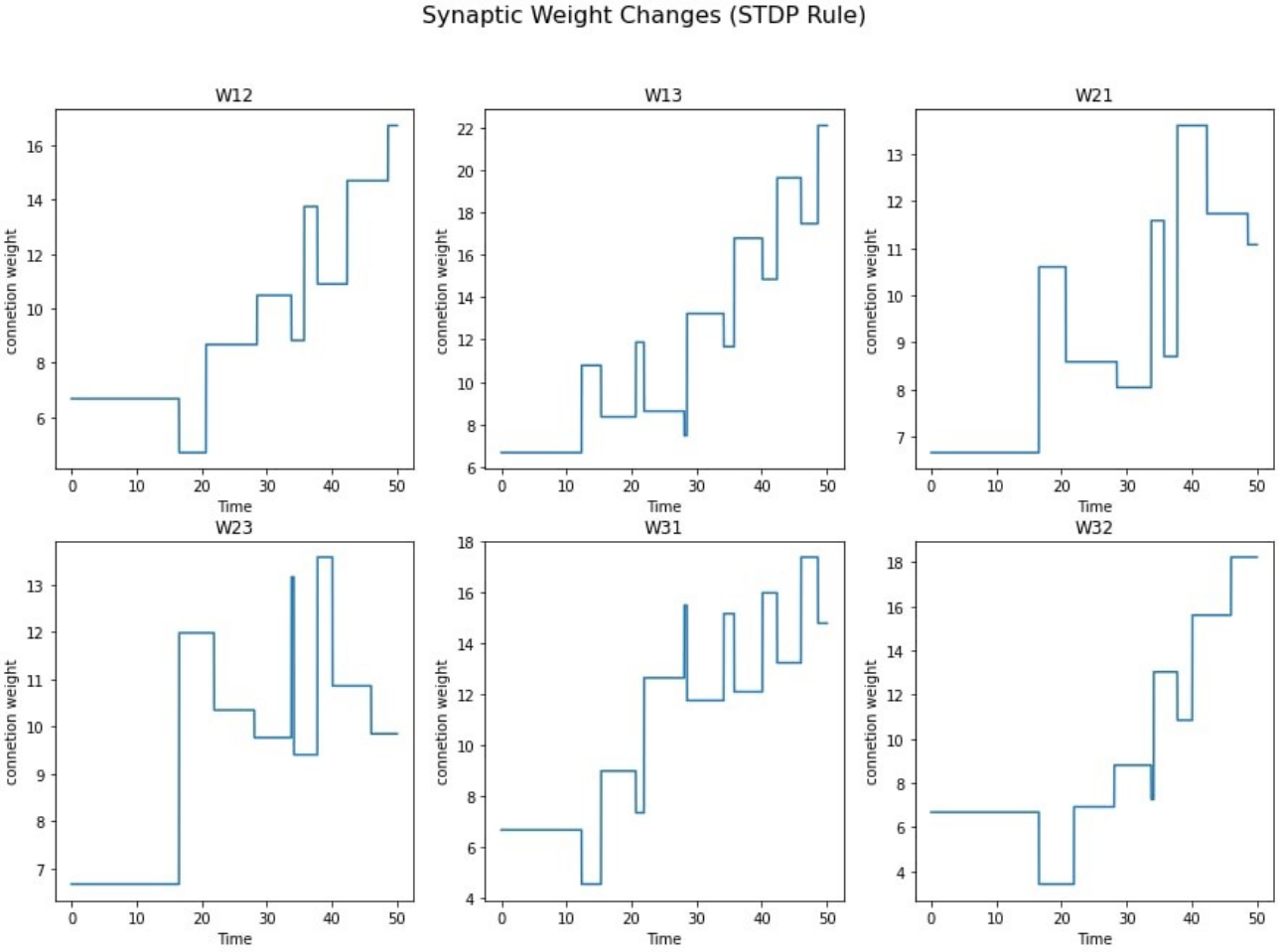
نوع دوم) دو جریان ثابت و متفاوت، یک جریان سینوسی





نوع سوم) یک جریان ثابت، یک جریان سینوسی، یک جریان پله ای ثابت





نتایج بدست آمده :

نتایج بدست آمده وابسته به Time Course Function و مقدار وزن های اولیه هستند.

وقتی نورون pre فایر میکند، مقدار وزن ها زیاد میشود و وقتی نورون post فایر میکند، مقدار وزن ها کاهش میابد.

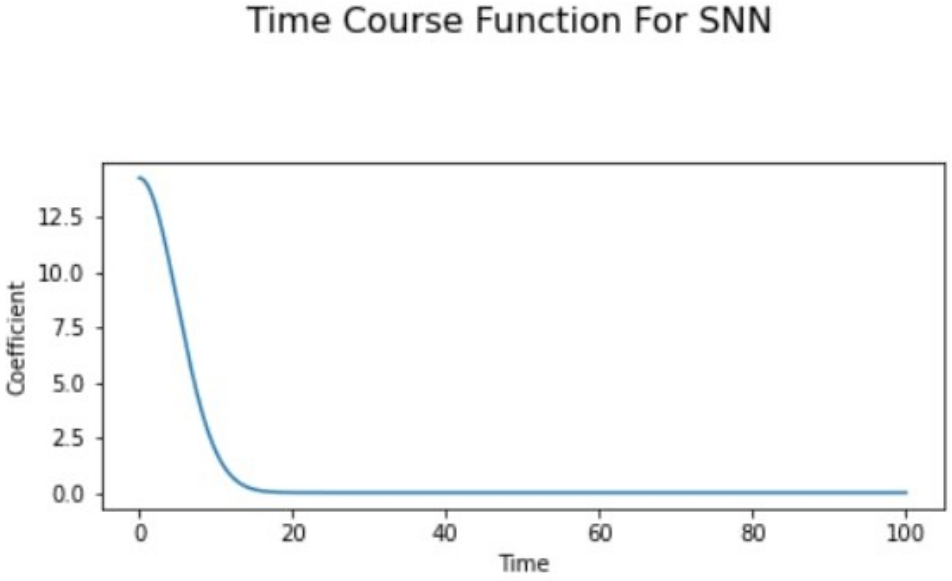
در قسمت هایی که نورون post فایر میکند مشاهده میشود که اختلاف وزن ها بشدت منفی میشود اما این الگو در همه جا رعایت نمیشود، اگر نورون pre بیشتر از نورون post فایر کند این الگو رعایت نشده و یکدیگر را خنثی میکنند.

******

بخش دوم پروژه

در این بخش قرار است به کمک یک SNN الگوریتم یادگیری تقویتی را پیاده سازی کنیم.

در ابتدا Time Course Function را تعریف کرده :



اکنون دیتای مورد نظر را از دیتاست خود خوانده :

	test	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	inpput_neuron_number	train_1	train_2	train_3	train_4	train_5	train_6	train_7	train_8	train_9	train_10
1	1.0	1.0	2.0	3.0	2.0	1.0	1.0	0.0	0.0	0.0	2.0
2	2.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	2.0
3	3.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
4	4.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0	1.0	0.0	1.0
5	5.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0	1.0	1.0	2.0
6	output_neuron_number	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0
7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
8	test	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9	inpput_neuron_number	train_1	train_2	train_3	train_4	train_5	train_6	train_7	train_8	train_9	train_10
10	1.0	1.0	0.0	0.0	2.0	1.0	1.0	3.0	0.0	1.0	2.0
11	2.0	0.0	1.0	2.0	2.0	2.0	0.0	1.0	0.0	2.0	2.0
12	3.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	2.0	0.0	2.0
13	4.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	1.0	3.0
14	5.0	0.0	0.0	0.0	0.0	0.0	3.0	3.0	2.0	1.0	3.0
15	output_neuron_number	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0

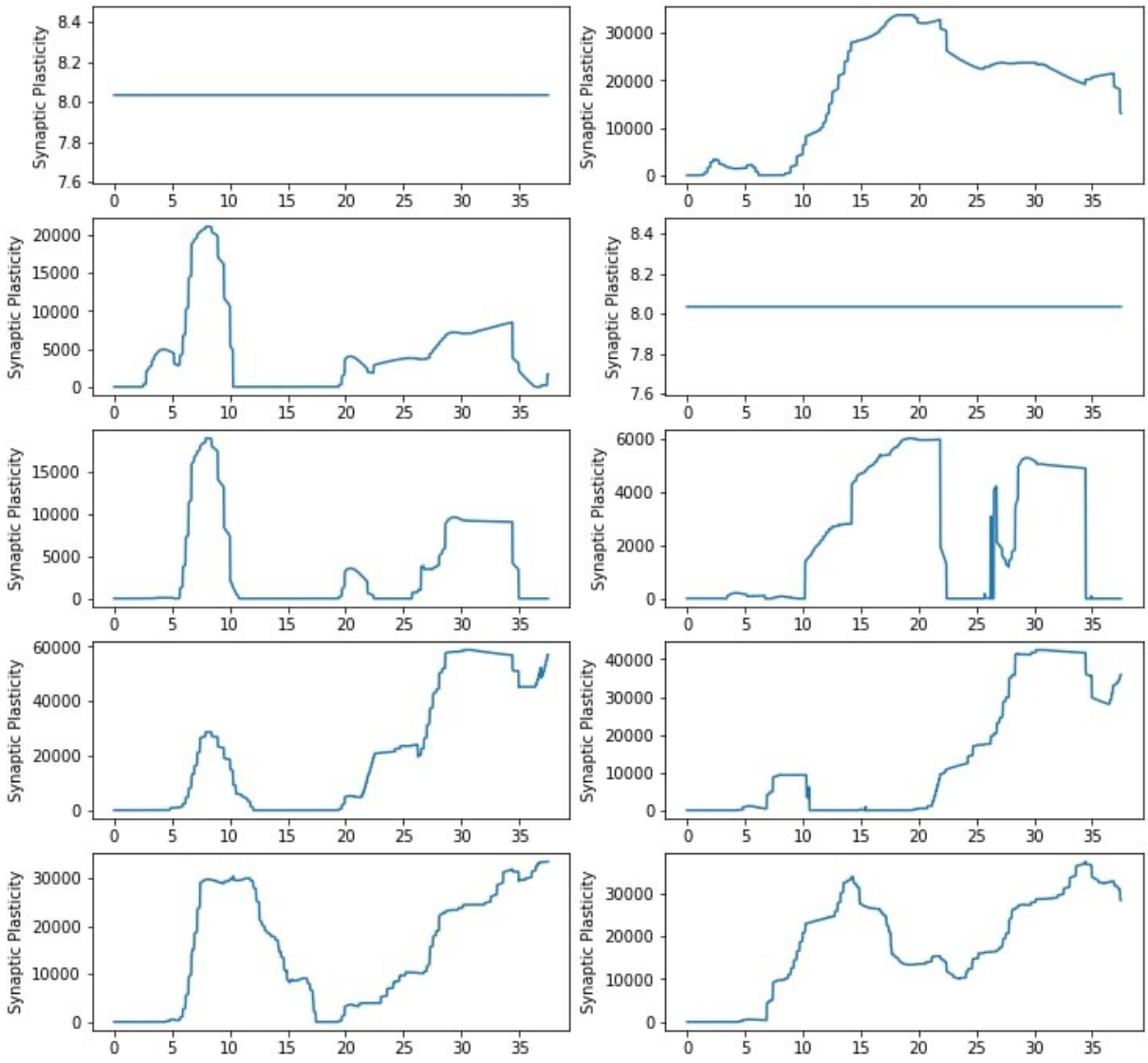
به جهت آماده سازی داده برای تزریق به الگوریتم، داده ها در 10000 ضرب شده اند.

اکنون مدل خود را آموزش داده و نتیجه را بر روی داده های Train و Test مشاهده میکنیم :

```
accuracy of SNN on test data:  70.0 %
accuracy of SNN on train data:  80.0 %
```

میتوان دید مدل تا حد خوبی آموزش دیده است.

برای دید بهتر میتوان تغییرات وزن های سیناپسی را به ازای هر ورودی نمایش داد و دید بهتری از مدل داشت.


Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity
Synaptic Plasticity