Soroush Heidary 96222031

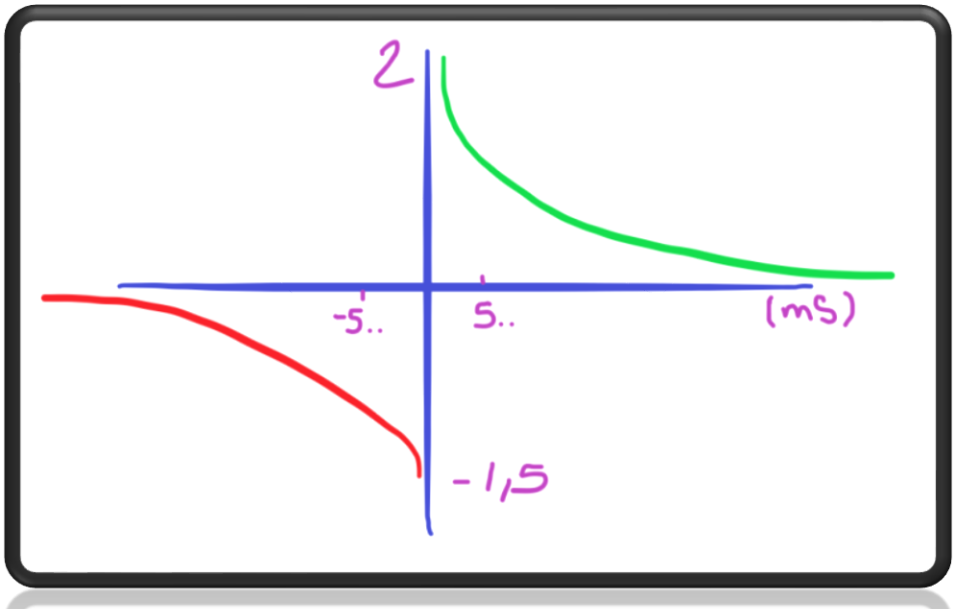
Exercise 3

Part 1 : STDP Rule Implementation

In this section we have 2 neurons (Instances Of Ex1 with some further modifications), whenever one neuron spikes (which by the way the two neurons have different Input Currents) the, The STDP Learning Rule gets applied to modify the strength between the two neurons (Pre synaptic and Post synaptic)

The Internal Input Current generated by Spikes only travels from presynaptic neuron to the postsynaptic one, meaning the postsynaptic neuron's spike wouldn't apply any discharge on the presynaptic one.

The STDP Rule which I have used is similar to this picture

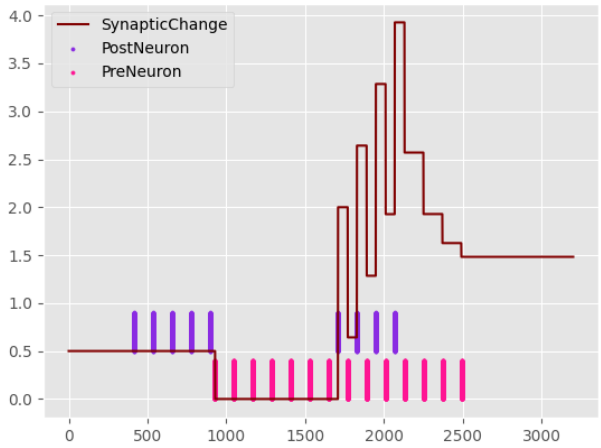


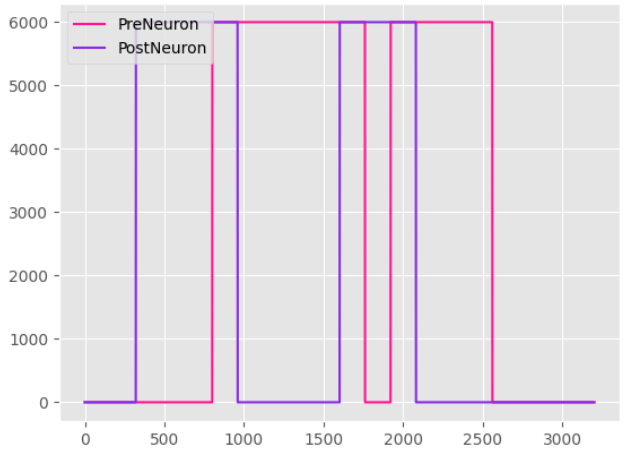
The values seems is tuned to demonstrate an STDP with higher clearance and is not what you'd see in a real neurons as 500ms interval wouldn't really cause any learning in brain (we would need less interval time for learning such as 10-20ms)

Two approaches were used for this purpose :

1- The first variation uses only the last spike as the leaning window.

   Using this approach, we'll expect less learning speed as the number of STDPs getting applied less than variation 2, here's the results :





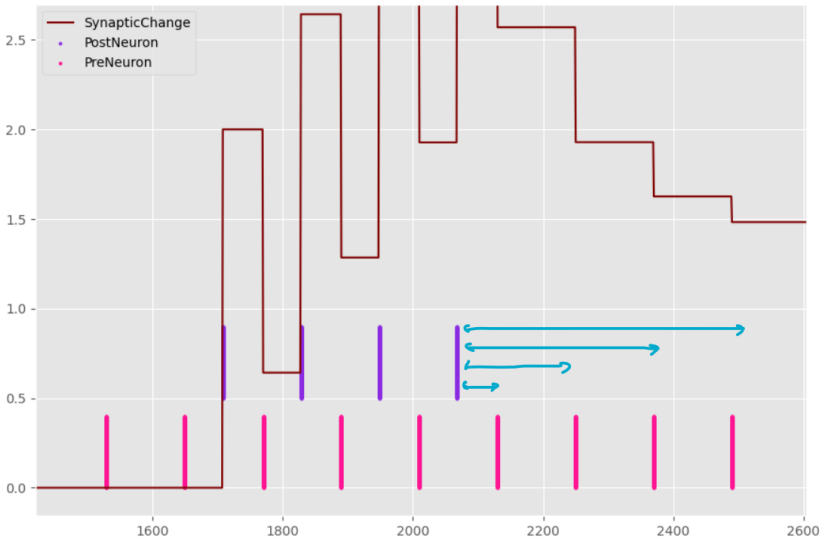
The right picture shows the Input Current of each neuron.

In the left one we're seeing the STDP rule applied after each spike (synaptic strength change right after each spike)

We see the zoomed version of the above picture here :

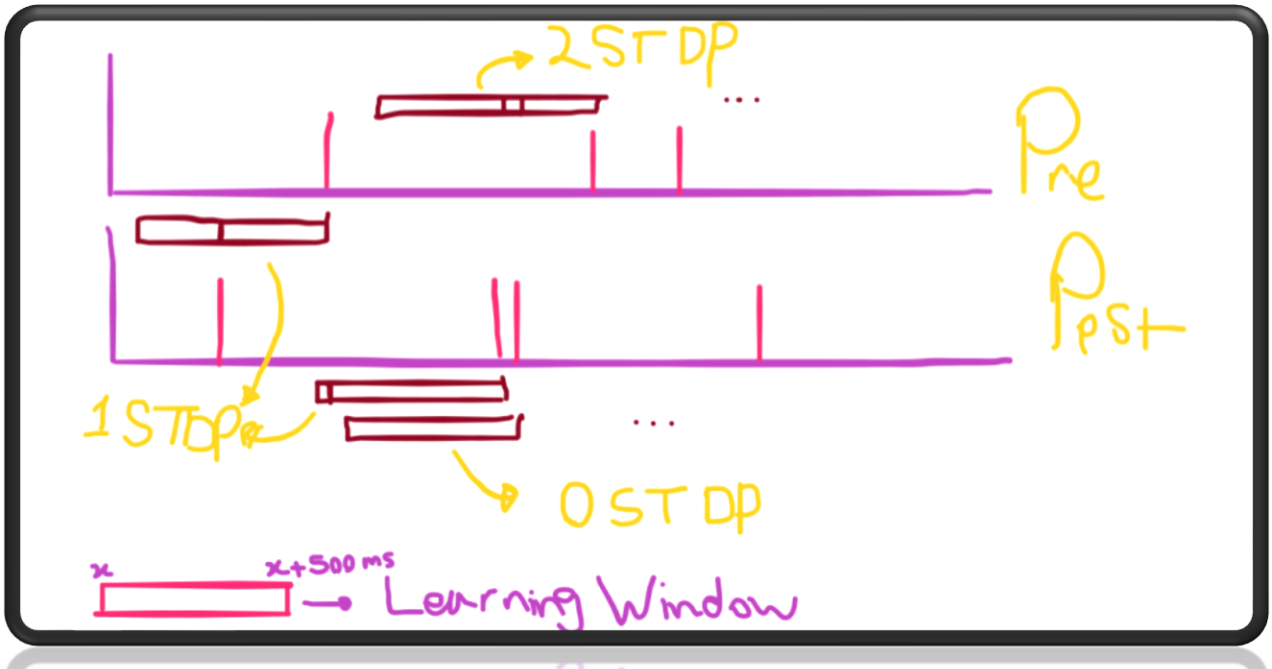
We'll see with the blue arrows, as the interval between last post spike and the proceeding pre spikes progresses the LTD lessens

(spikes are shown with a simple line, the y values doesn't have any meaning)



Variation 2 :

This time we'll use a window of spikes for the learning purpose, meaning each time one neuron spikes the STDP will be applied to a chunk of spikes in the other neuron's spike chart using a specified window like the following picture :



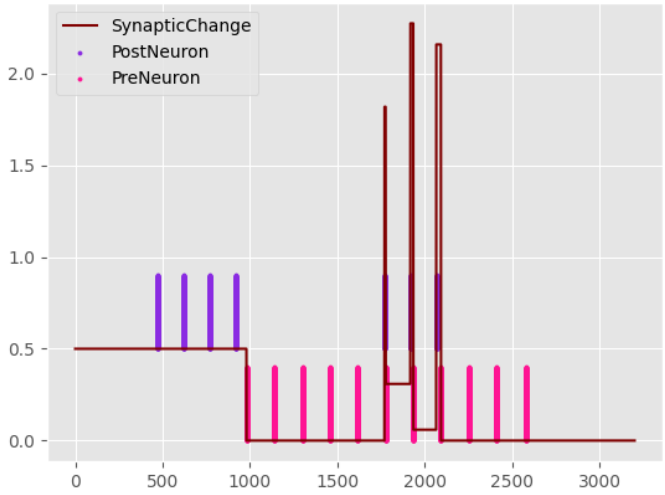
With this variation we'll have the possibility to have both 0 and more than 1 learning with each spike which we didn't have with variation 1

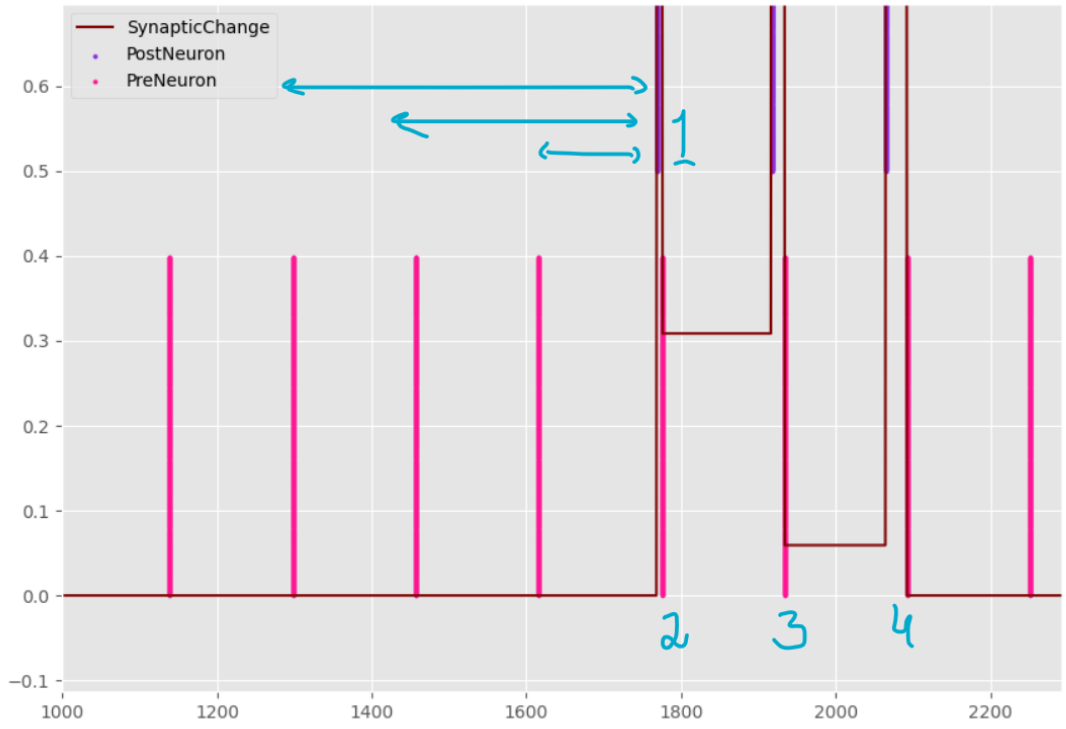
Also more exaggerated learning is expected :

Here we see some examples :

As you see the leaning have become very drastic comparing to variation 1, again this is not what we'd see in an actual neuron

Here's a zoomed version :





If we want to make this simulation more realistic we would need a smoother tuning (lower parameters leads to less learning)

```
def LearnByLastSpike(self, t1, t2) :
    if(t1 > t2) :
        deltaW = min(self.Aplus * math.exp(-abs(t1 - t2) * 0.03125 * .8), 2)
    elif(t1 < t2) :
        deltaW = max(-self.Aminus * math.exp(-abs(t2 - t1) * 0.03125 * .8), -1.5)
    else :
        deltaW = 0
```

The min and max are meant to limit the change to 2 and -1.5, these are one of the parameters which can be changed,

The 0.8 is used to scale the x axis and the Aplus and Aminus are meant to scale the y axis of our plot, using these parameters we can modify our STDP plot which we saw earlier. Using lower amounts of these variables makes the learning smoother, and more real.

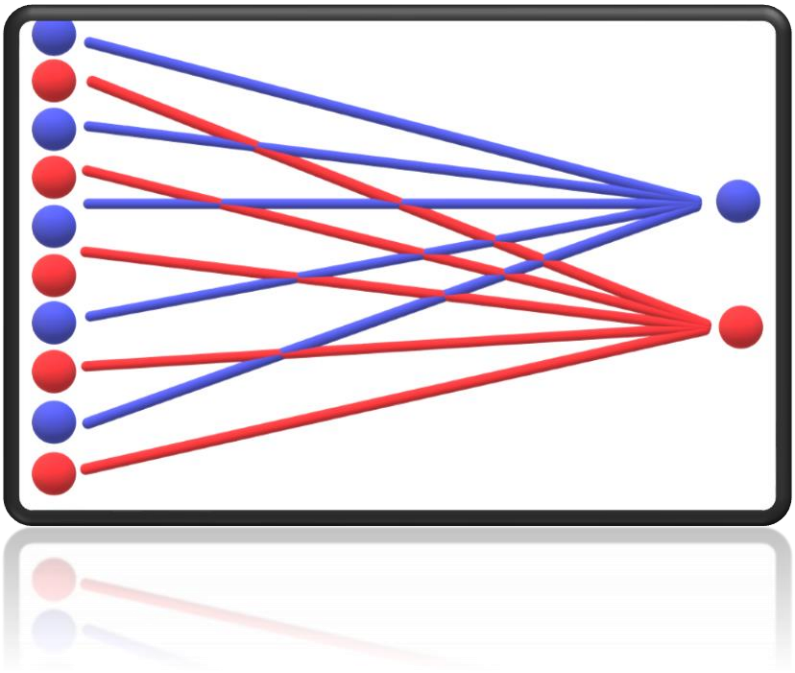
Part 2 : SNN

In this section we'll use the STDP Learning rule applied to a simple neural population with 1 input layer and 1 output layer.

We'll use 2 different connectivity schemes for this section

- [1, 2, 1, ,2 ,1 ,2, 1, 2, 1, 2]
- [1, 1, 1, 1, 0, 2, ,2 ,2 ,2, 0]

This simple network also called a Spiking Neural Network works like normal ANNs but here we use STDP as a learning rule instead of backpropagation as a way of modifying weights (learning) and the neurons are more complicatedly implemented (using an AELIF neuron model instead of just simple block of synapses which ANNs use)



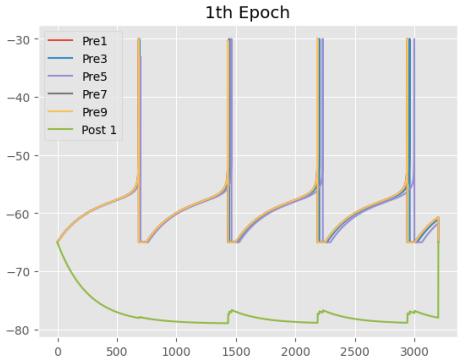
First Scheme :

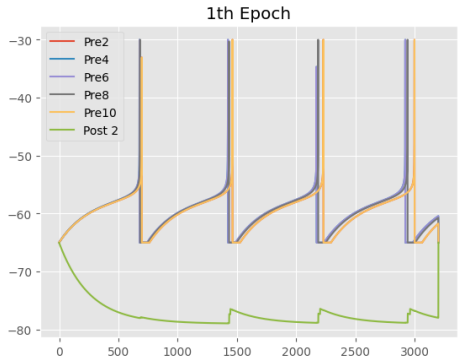
In this connectivity pattern we use the same input current for all the input neurons (the output neurons doesn't receive any external Is)

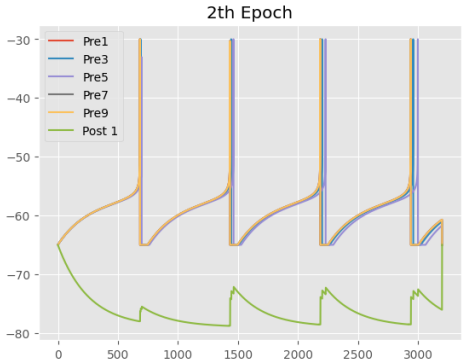
Which is just a simple constant I(s) (with a little slope so that we can actually cause a spike in output layer so we can start learning)

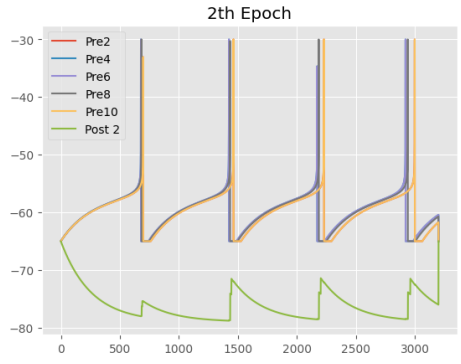
At the end of epoch 5 the post neuron(green one) we'll have enough presynaptic strengths to trigger spikes whenever all the preceding neurons spike together (blue neurons are shown in the left column and the red neurons, right column).

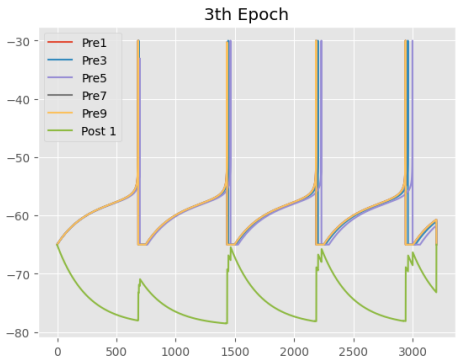
It takes 5 epochs in this SNN to teach blue output neuron to spike upon receiving the appropriate input pattern (in this case all the 5 blue inputs spiking together) there is one problem with this scheme which we'll be solved in scheme 2.

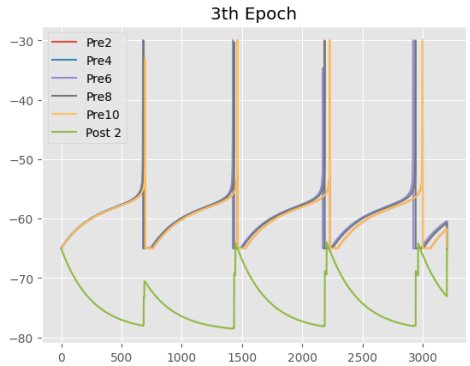

1th Epoch
Pre1
Pre3
Pre5
Pre7
Pre9
Post 1

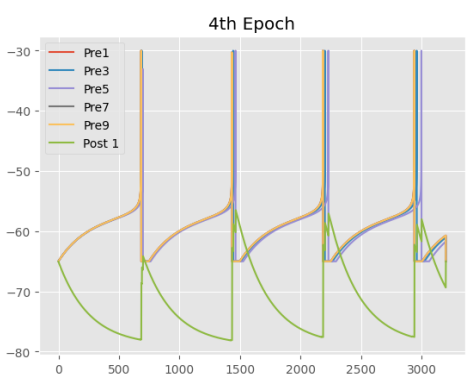

1th Epoch
Pre2
Pre4
Pre6
Pre8
Pre10
Post 2

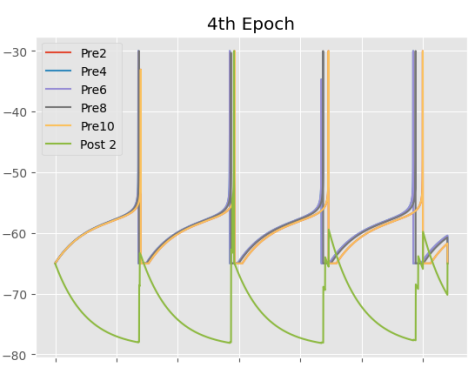

2th Epoch
Pre1
Pre3
Pre5
Pre7
Pre9
Post 1

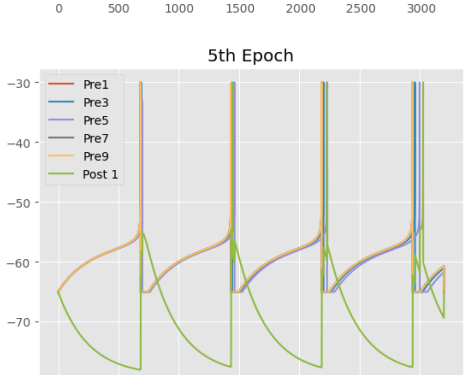

2th Epoch
Pre2
Pre4
Pre6
Pre8
Pre10
Post 2

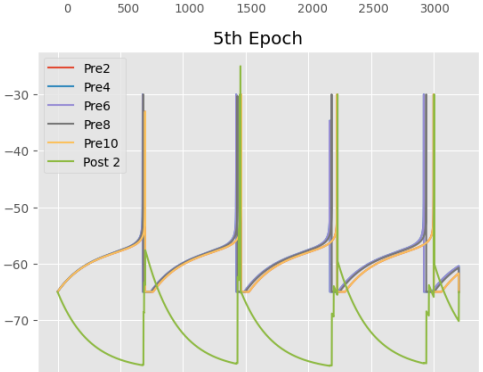

3th Epoch
Pre1
Pre3
Pre5
Pre7
Pre9
Post 1


3th Epoch
Pre2
Pre4
Pre6
Pre8
Pre10
Post 2


4th Epoch
Pre1
Pre3
Pre5
Pre7
Pre9
Post 1


4th Epoch
Pre2
Pre4
Pre6
Pre8
Pre10
Post 2


5th Epoch
Pre1
Pre3
Pre5
Pre7
Pre9
Post 1


5th Epoch
Pre2
Pre4
Pre6
Pre8
Pre10
Post 2

Scheme 2 :

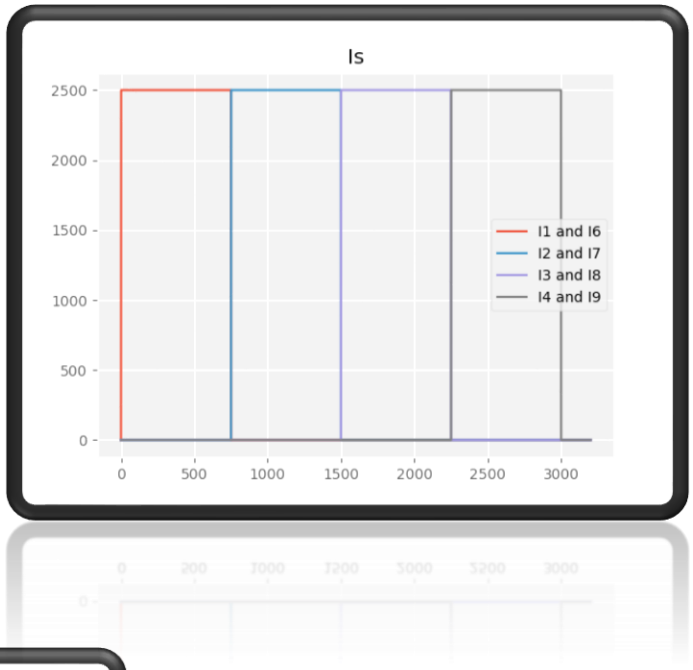
This scheme uses another connectivity pattern, also we'll give each neuron a constant (which increases by the neuron index) but only on a specific interval

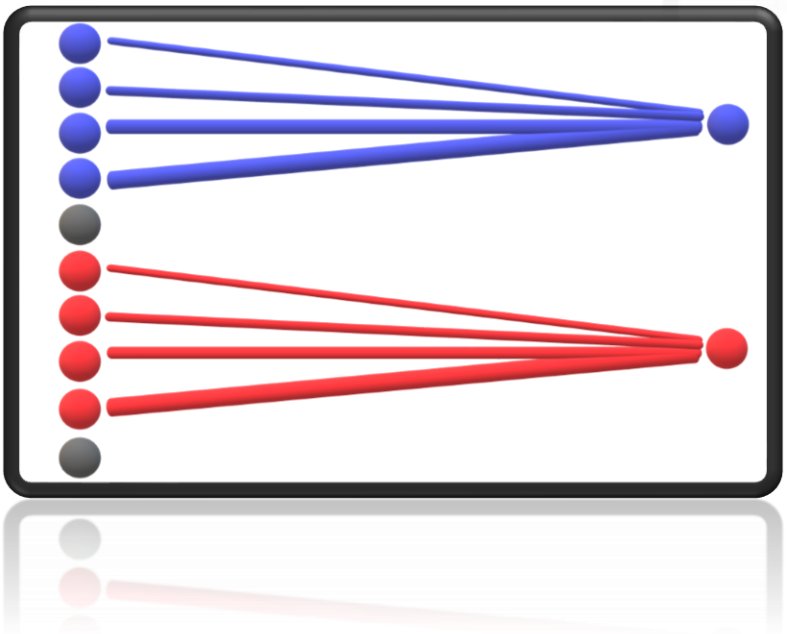
This helps us have a different mechanism for the first output neuron's spike to start the learning process (the forth and the nineth neuron will trigger the first spike of the first epoch, and then the learning starts to take place

This picture shows the Input Current of the Input Neurons :

This time each neuron in input layer have a different initial synaptic strength. We'll see that two neurons get inputs only on a specific interval

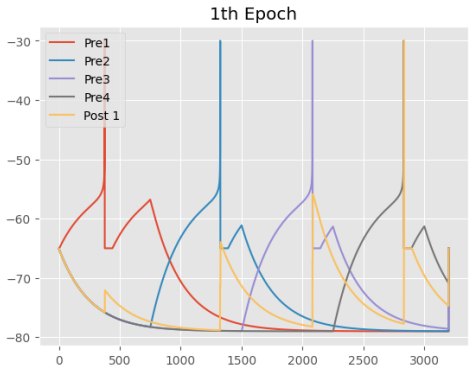
The 5th and 10th neuron doesn't receive any inputs.

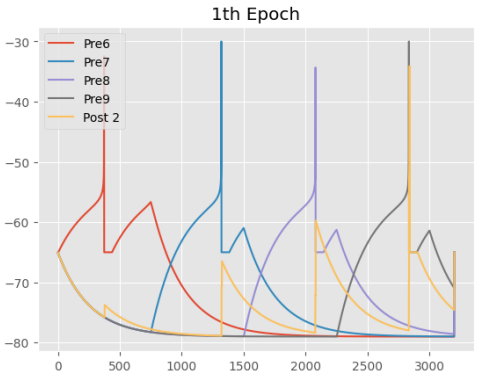




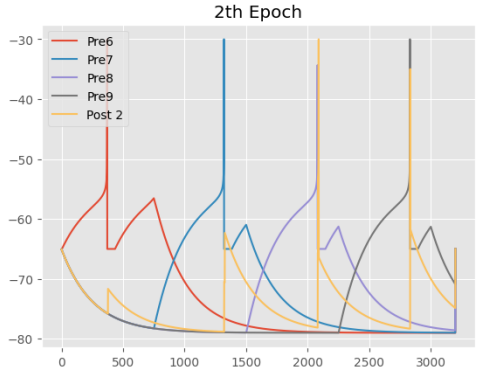
And by what we see below each neuron's initial strength increases by the index of the neuron (causing the last neuron to start the learning process at epoch 1 so we can apply the STDP

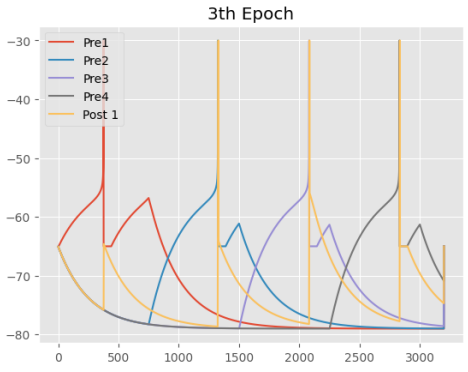
(this solves the problem of scheme 1 when we didn't have any initial spike on the output layer to start the learning)

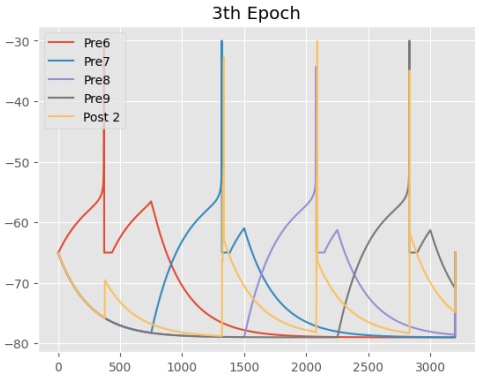

1th Epoch
Pre1
Pre2
Pre3
Pre4
Post 1

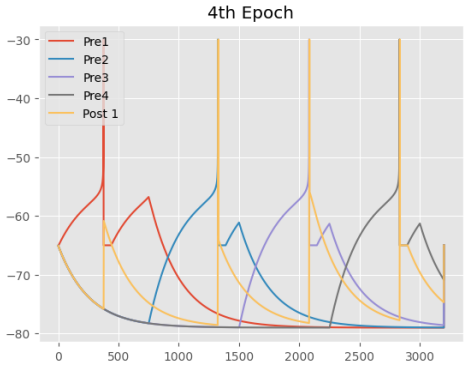

1th Epoch
Pre6
Pre7
Pre8
Pre9
Post 2

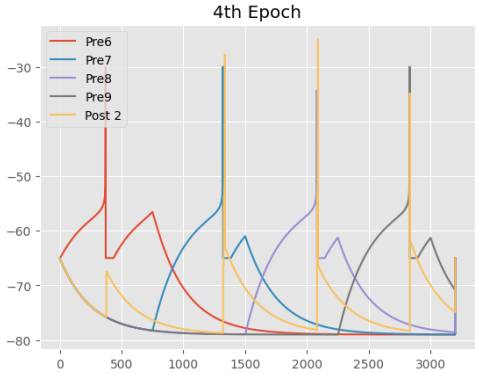

2th Epoch
Pre1
Pre2
Pre3
Pre4
Post 1

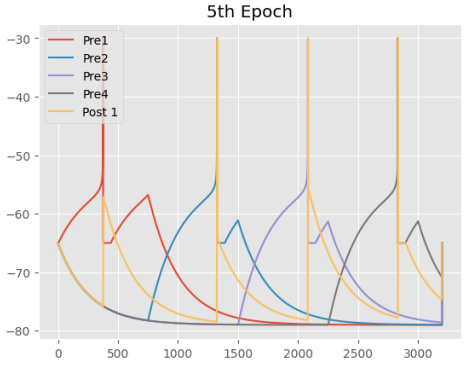

2th Epoch
Pre6
Pre7
Pre8
Pre9
Post 2

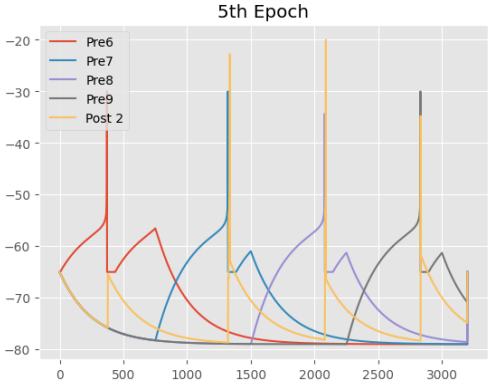

3th Epoch
Pre1
Pre2
Pre3
Pre4
Post 1


3th Epoch
Pre6
Pre7
Pre8
Pre9
Post 2


4th Epoch
Pre1
Pre2
Pre3
Pre4
Post 1

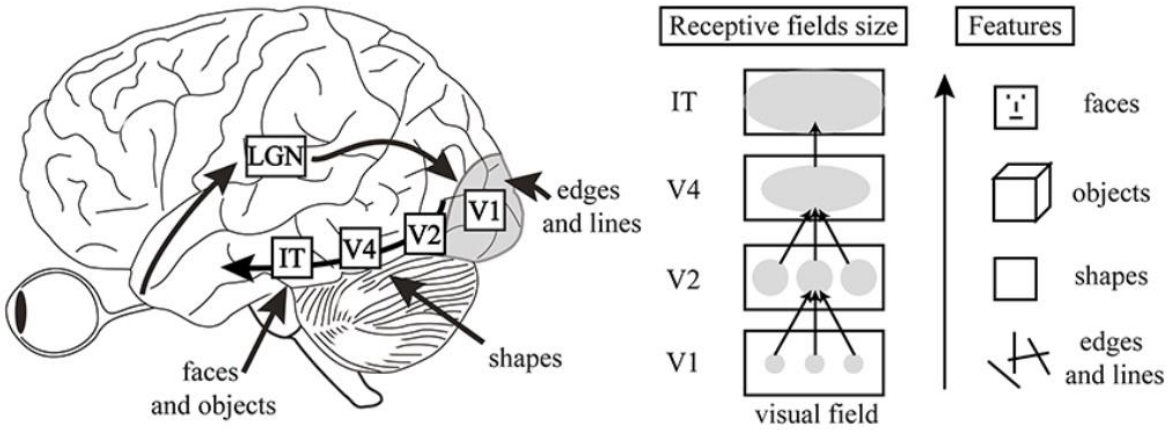

4th Epoch
Pre6
Pre7
Pre8
Pre9
Post 2


5th Epoch
Pre1
Pre2
Pre3
Pre4
Post 1


5th Epoch
Pre6
Pre7
Pre8
Pre9
Post 2

Part 3 : STDP In Computer Vision

تا آنجایی که بنده متوجه شدم در این مقاله از ساختار زیر که مدل ساده شده ای از پردازش تصویر انجام شده توسط مغز است استفاده میشود بطور خلاصه در شکل میبینیم در هر بخش چه ویژگی هایی استخراج میشود



با استخراج خطوط و زوایای آن ها مغز قادر خواهد بود در بخش های بعدی طی آموزش هایی که به طور Unsupervised در آن اتفاق میافتد اشکال و سپس در لایه بعدی شکل های سه بعدی کلی تر و بعد از آن الگو های پیچیده و جزی تر مانند صورت را استخراج کند.

چند جنس مختلف خروجی از لایه V1 خارج میشود که بطور کلی میتوان خروجی های مربوط به رنگ و خروجی های مربوط به خطوط و زوایای آن ها را نام برد با ارسال این خروجی به V2 عمق به پیچیدگی پردازشات اضافه میشود و در این مرحله خروجی ها به دو بخش Temporal و Parietal منتقل میشوند.

همانطور که مشاهده میکنیم مسیر طی شده تا حد خوبی شبیه شبکه های عصبی Convolutional است که برای تشخیص تعداد زیادی اشکال مختلف آموزش دیده شده باشد که به تبع نیاز به تعداد بسیار زیادی فیلتر برای Feature Extraction خواهد بود که عموما در اینگونه شبکه ها دیده نمشود و هر شبکه مختص تعداد خاصی از اشیاء بکار برده میشود.

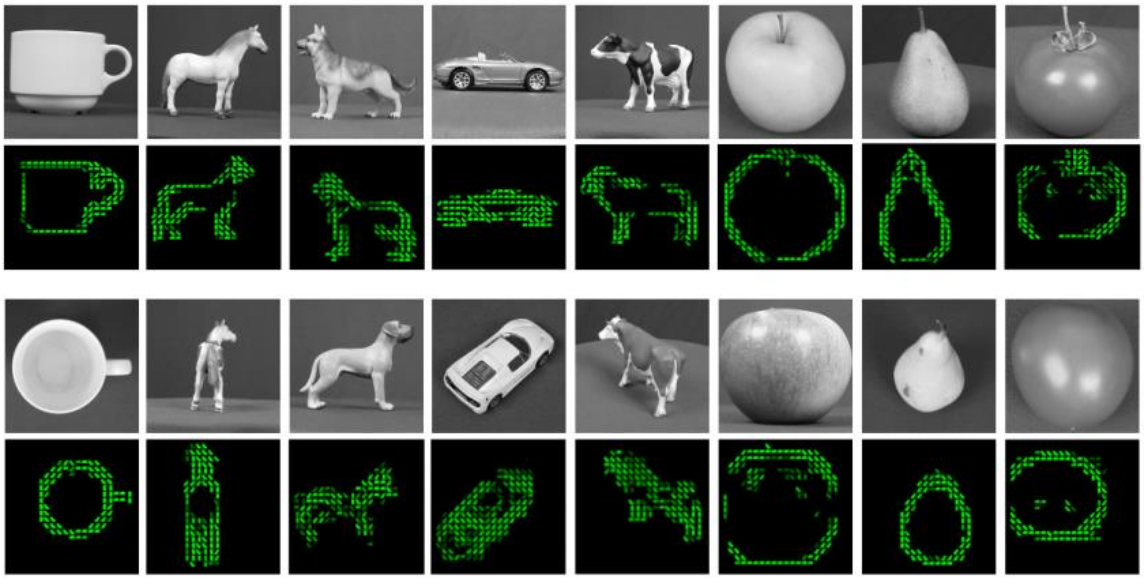
قبل به ورودی به توضیحات مربوط به هر لایه خالی از لطف نخواهد بود که کمی درباره ایده پشت این مدل صحبت کنیم.

شبکه های عمیق کانولوشنی در مقالات عملکرد خیلی خوبی در حیطه بینایی از خودشون نشون دادند. و حتی قابلیت تشخیص یک شی از زوایای مختلف رو هم تا حدود خوبی تونستن پیاده سازی کنن که البته با Unsupervised کردن اونها بنظر شخصی من حتی میشه این تحمل زوایا و حالات مختلف یک شکل رو قوی تر هم کرد (هم از طرفی برای یادگیری Unsupervised میشه دیتا های برای آموزش داشت و هم اینکه تمرکز یادگیری بیشتر روی ویژگی های مختص هر شی خواهد رفت)

منتها مشکلی که با شبکه های کانولوشنی عادی بوجود میاد دور بودن بیش از حد اونها از ایده اولیه پشتشون (شبیه سازی لایه های کورتکس بینایی مغز) هستش که هم به دلیل Supervised شدن این یادگیری بر خلاف یادگیری در مغز و هم به دلیل تبدیل کردن نورون های اسپایکی به اعداد اعشاری صرف و هم به دلیل نوع اعمال یادگیری که از backpropagation استفاده شده اما در مغز یادگیری ها بطور STDP (و دقیق تر به طور RSTDP انجام میشه)

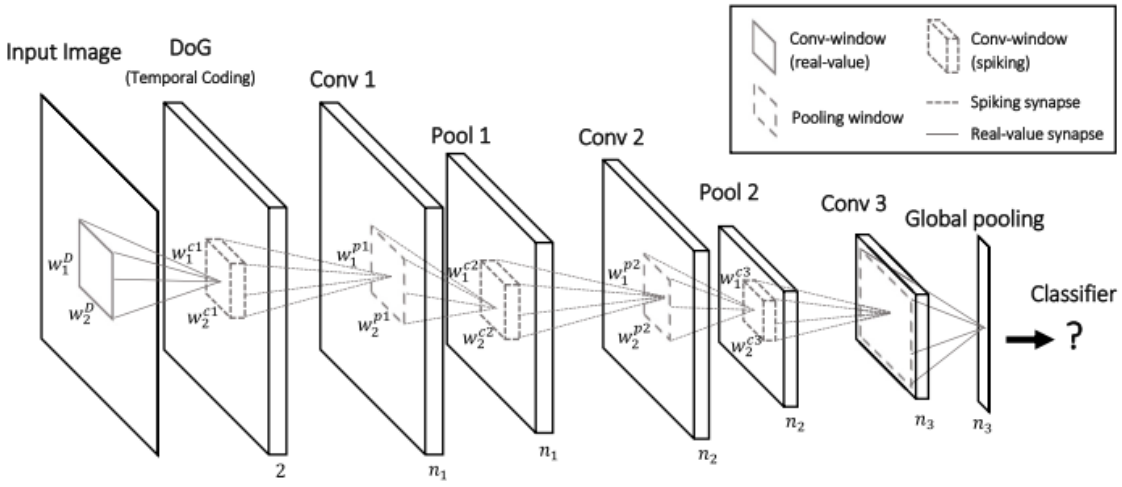
به دلیل هایی که بالا گفته شد یادگیری در شبکه های عمیق عادی میتونه سرعت کمی پیدا کنه (بر عکس مغز که میشه با عاملی پاداشی یا تنبیهی مثل دوپامین سرعت خوبی برای یادگیری و حتی یادگیری بلند مدت تری مثل LSTM ها داشت) و همچنین نیاز بسیاری به دیتا های لیبل خورده داشت که همیشه در دسترس ما نیست

توی تصویر زیر تعدادی از داده های تست این شبکه رو میبینیم (که در ادامه با جزییات بیشتری قابل فهم تر میشه) که ایده تشخیص حالات و زوایای مختلف اشیا رو خیلی خوب نمایش میده



میبینیم برای شکلی مثل ماشین اسباب بازی یا فنجون از هر زاویه خطوط و اشکال استخراجی خیلی از هم متفاوت میشن اما بخاطر کمتر بودن محدودیت توی جمع اوری دیتا های لیبل نخورده و ماهیت Unsupervised بودن این مدل (که منجر به اجبار در پیدا کردن الگو های کلیدی تری میشه) عملکرد بهتری نسبت به شبکه های کانولوشنی عادی ارائه میده

همچنین چیزی که باید در نظر داشت اینه که (در ادامه توضیحات بیشتری خواهیم داشت) توی این ساختار نوعی تاخیر در ارسال هر اسپایک وجود داره که مربوط به تاخیر در ارسال سیگنال های درون مغز به دلیل (واضحا محدودیت های فیزیکی 😊) میشه. همچنین روند یادگیری پیوسته تر از روند یادگیری در شبکه های عادی کانولوشنی هستش به این معنی که ورودی ها با بازه های زمانی کوتاهی وارد میشن و منتظر نمیشن تا ورودی قبلی کامل پردازش بشه و یادگیری براشون انجام شه

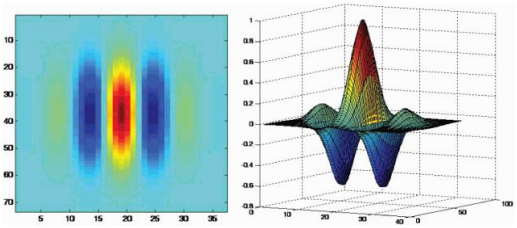


لایه اول (DoG) : این لایه برای شبیه سازی دقیق تر از عملکرد مغزی اضافه شده و در اصل کارش تشخیص تضاد نوری (شدت رنگ هر پیکسل) بین خودش و اطرافشه. مشابه همین کار رو سلول های گنگلیون و دوقطبی توی لایه قرنیه انجام میدن و اطلاعاتی که به مغز میفرستن به نوعی خطوط مرزی تصویر وروی به چشم هستش که به شکل چند دسته (سیاه سفید که توسط سلول های استوانه ای و رنگی که توسط سلول های مخروطی استخراج میشن) فرستاده میشن

برای شبیه سازی این بخش از نوعی فیلتر به اسم فیلتر گابور استفاده میشه که فرمولشو اینجا میبینیم

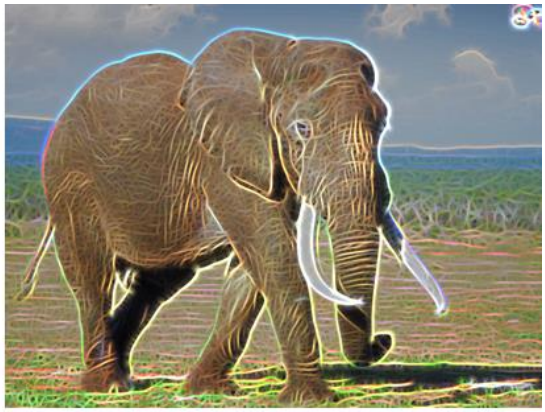
$$g(x,y,\lambda,\theta,\sigma,\gamma) = \exp\left(-\frac{X^2+\gamma^2Y^2}{2\sigma^2}\right)\cdot\cos\left(\frac{2\pi}{\lambda}X\right),$$
$$X = x\cos(\theta)+y\sin(\theta),$$
$$Y = -x\sin(\theta)+y\cos(\theta),$$

where λ = wavelength, θ = orientation, σ =standard deviation, and γ = aspect ratio.



و خروجی که از این فیلتر بیرون میاد رو در این عکس مشاهده میکنیم :

a



b

البته شکل بالا که در سمت راست میبینیم ترکیب شده تعدادی فیلتر اعمال شده بصورت جداگانه (برای رنگ های مختلف) هستش که البته فقط وجود تضاد در اطراف برای تشخیص مرز ها در این مرحله مهم هست

اینکار به ما اجازه میده در مرحله های بعدی که قراره زوایای مختلف خطوط بیرون کشیده بشه بتونیم با دقت بهتری این زوایا رو بیرون بکشیم

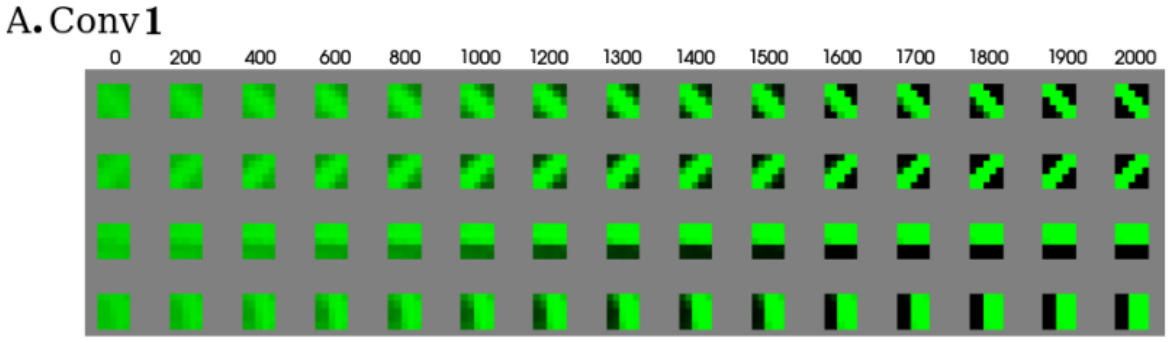
این لایه برای ارسال سیگنال ها از یک نوع کد گذاری به اسم Rank-Order استفاده میکنه که توضیح این نوع کد کردن دیگه از حوصله این گزارش خارجه. همچنین تعداد فیلتر ها درون هر لایه از ابتدا مشخص میشود

لایه کانولوشن اول : اونطور که من متوجه شدم در اصل این لایه همون کار V1 رو انجام میده. به عبارتی تضاد بیرون کشده شده از لایه قبلیش رو تبدیل به خطوطی به زوایای مختلف میکنه و در پنجره های کوچی از تصویر این خطوط رو بیرون میکشه سپس در لایه بعدی نوعی پولینگ محلی روشون اعمال میشه که ابعاد این خطوط رو میشه در اینجا به اندازه های بزرگتری داشت و استخراج کرد

نکته اینجاست که وقتی بطور پیوسته در حال دریافت و پردازش اصلاعات هستیم (مانند مغز) اطلاعات ورودی به لایه کانولوشن اول به شکل زیر خروج پیدا میکنه

در اصل اینجا به نوعی میشه گفت مقدار فدای تاخیر میشه. یعنی اولین پنجره ای که نورون مربوط بهش اسپایک بزنه به عنوان خروجی تمام این لایه به پولینگ مرحله بعد میره (اصطلاحا میگن Winner Takes All یعنی وقتی اولین اسپایک در این لایه کانولوشنی رخ بده تمام اعتبار خروجی از این لایه رو به خودش اختصاص میده و رون لرنینگ رو فراخوانی میکنه STDP) توجه هم داریم که یادگیری ها در لایه های پولینگ انجام نمیشن

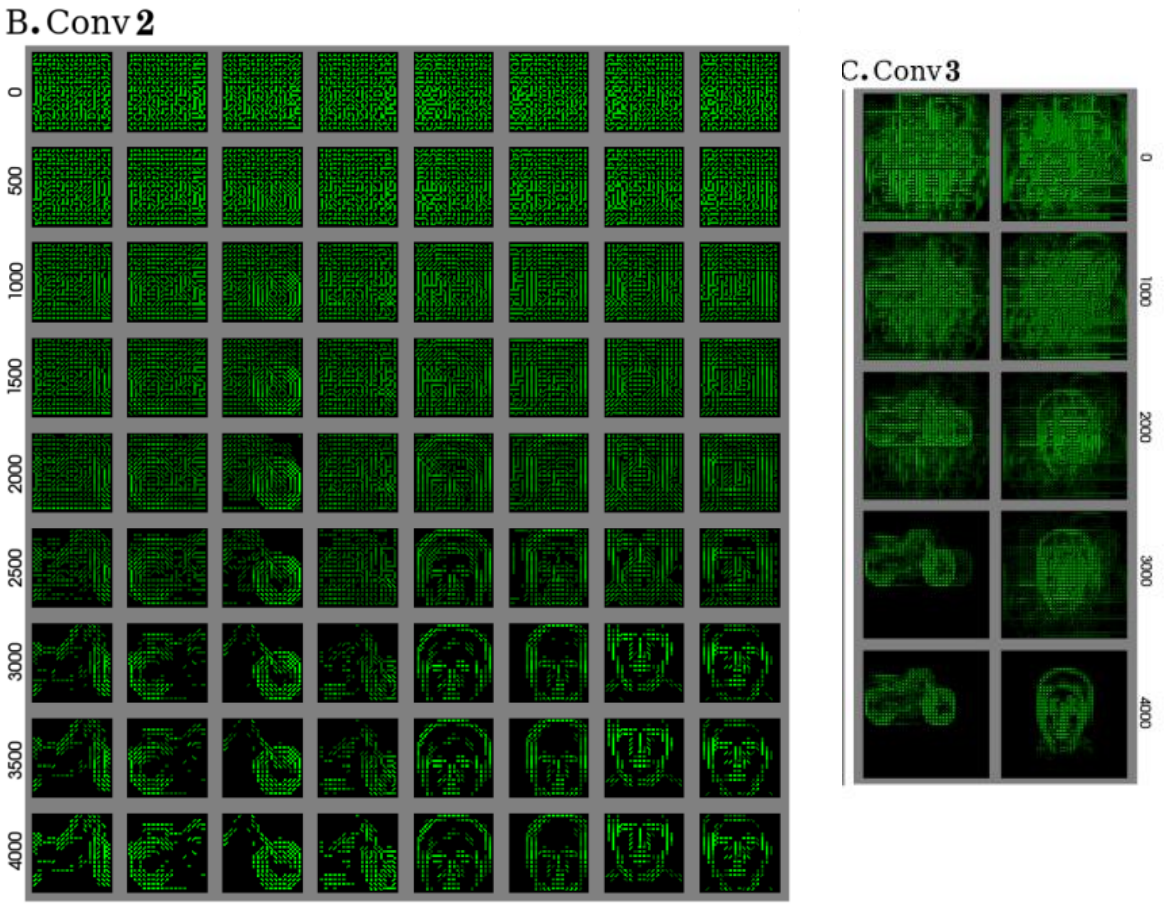
خروجی نمونه ای از این لایه رو در شکل زیر میبینم که چهار فیلتر از آن بیرون کشیده شده



لایه کانولوشن دوم : در این لایه به نوعی کار هر دو بخش V2 و V4 انجام میشه (اگر اشتباه متوجه نشده باشم) یعنی استخراج اشکال و سپس اشیاء در این لایه انجام میشه. بطور کلی کاری که همه کانولوشن ها انجام میدن اینه که اطلاعات چکیده شده کانولوشن قبلی که توسط پولینگ لایه قبل چکیده میشه رو با ترکیب بندی های خاصی در نظر میگرن تا بتونن اشکال و اشیاء رو بیرون بکشن. در اینجا هم همین اتفاق میفته. یعنی با استفاده از خروجی لایه قبل (که مربوط به اولین اسپایک زده شده که اون هم مربوط به غالب ترین زاویه موجود در V1 باشه) اشکال و اشیاء خاصی رو یادمیگیرن (در اصل منظور از یادگرفتن اشکال خاص اینه که الگو های زمانی مختلفی رو یادمیگیرن که با رسیدن هر ورودی مشخص میشه)

باز هم مثل لایه قبل اولین پنجره ای که اسپایک بزنه (اولین (که میشه به نوعی به غالب ترین یا شبیه ترین تشبیهش کرد) شکلی که پیدا شه) موجب اسپایک زدن نورون مربوطه میشه و مثل قبل طی روند Winner Takes All یادگیری رو فراخوانی میکنه که روی تمام پنجره ها اعمال میشه و سپس به لایه پولینگ منتقل میشه تا چکیده اون ها به مانولوشن بعدی بره

خروجی نمونه ای از این لایه رو میتونیم تو عکس زیر ببینیم که 8 فیلتر از آن بیرون گشده شده:



لایه سوم کانولوشن : این لایه در اصل همون کار لایه IT رو انجام میده. روند انجام شده در اینجا دقیقا شبیه به روند لایه کانولوشن دوم هستش و خیلی توضیحات اضافه دیگه ای نداره. عکس مربوط به این لایه رو در سمت راست بالا میبینیم که تنها دو فیلتر استخراج شده از اون وجود داره که در اصل اشکال یا همان اشیاء استخراج شده نهایی ما هستن

منتها فرق این لایه با کانولوشن قبلی آن در این است که در اینجا پولینگ بعد از یادگیری به شکل غیر محلی (یعنی تمامیت دار) در تمام پیکسل ها انجام میشه که خروجی آن یکی از اشکال پیچیده مورد نظر ما مثل موتور یا صورت انسان است

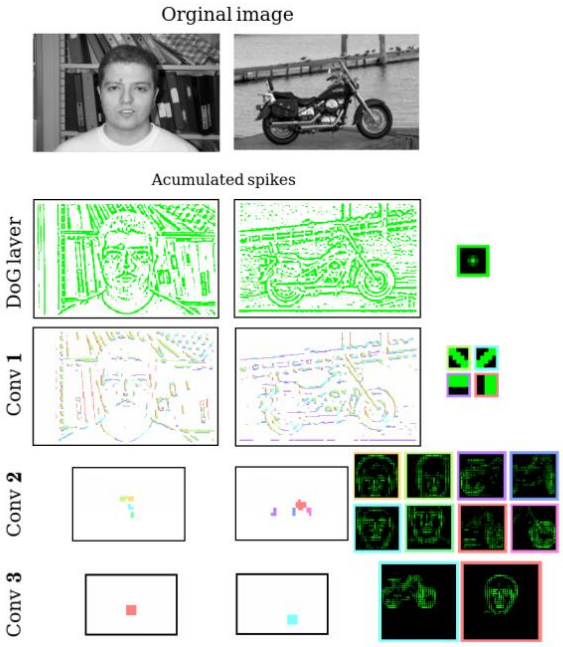
تا اینجای کار با اسپایک زدن هر نورون در لایه کانولوشن سوم میدونیم چه شکلی دیده شده و الگوی غالب دید ما به چه شکلی نزدیک تر بوده (که منجر به تاخیر بیشتر در اسپایک زدن فیلتر مربوطه شده)

اما برای اینکه بتوانیم تصمیم گیری کنیم و لیبلی رو به شکل تشخیص داده شده انتصاب بدیم مجبوریم لایه ای رو که یک کلاس بندی کننده است (Classifier) به انتهای شبکمون اضافه کنیم که لیبیل مورد نظر رو به شکل تشخیص داده شده

اطلاق کنه (این Classifier نوعی ماهیت Supervise از خودش داره که در واقع هم درون مغز این لایه اخر رو نداریم و تشخیص و Decision Making بشکلی که هنوز ماهیت واقعی آن مشخص نشده انجام میشه.

(توضیحات زیر نظر شخصی من هست و هیچ بنیه رسمی علمی ای نداره !)

به حدس شخصی خودم باید اینطوری باشه که در مغز تصمیم گیری بسیار نسبی تر از تصمیم گیری که ما در این شبکه داریم انجام میشه. شاید بشه به نوعی گفت اصلا تصمیم گیری ای به این شکل انجام نمیشه که فلان شکل رو ماشین بنامه بله بطور نسبی شکلی که میبینه رو به نسبت محیط اطرافش به یک مفهوم آشنا مثل ماشین اطلاق میکنه. دلیل اینکه اینطور فکر میکنم این هستش که در واقعیت بر خلاف این مدل که ما از ابتدا تعداد فیلتر های هر لایه رو در مدل بر اساس نیازمون مشخص میکنیم، توی مغز تعداد فیلتر ها مشخص نمیشه و شاید در اصل تعداد خاصی هم نداره بلکه همه چیز رو بطور نسبی یاد میگیره. شاید ایده احمقانه ای باشه اما بنظرم فقط دو آپشن در مغز وجود خواهد داشت : شیء x و غیر شیء x، که بطور نسبی همه اشکال دیده شده تو این دسته بندی تصمیم گیری میشن



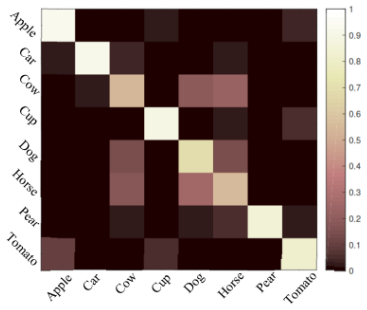
در عکس روبرو خلاصه ای از هر مرحله رو میشه دید که وقتی یک فیلتر در یک پنجره کوچک ناحیه بینایی موجب اسپایک زدنی میشه همون خروجی به لایه بعد فرستاده میشه و یادگیری توسط فقط همین نورون برای همه نورون ها رخ خواهد داد.

همچنین در عکس زیر هم نتایج این مدل رو روی یک دیتا ست متشکل از چند شیء مختلف میبینیم که درصد های خیلی خوبی به نسب بقیه مدل ها ارائه داده

Comparison of recognition accuracies:

Method	Accuracy (%)
HMAX	69.0
Shallow SNN	81.1
Pre-trained AlexNet	79.5
Fine-tuned AlexNet	94.2
Supervised DCNN	81.9
Unsupervised DCA	80.7
Proposed DSNN	82.8

Confusion matrix:



Part 4 : RSTDP

به طور خیلی خلاصه میتوان گفت این روش عامل شرطی شدن نورون به ورودی های خاص است (که در ادامه توضیحات بیشتری در این باره میبینیم)

با اضافه کردن یک ترم جدید که مستقیما مرتبط با ترشح دوپامین است در این مدل میتوان خروجی های در هر صورت تقویت کننده در زمان ترشح دوپامین داشت. به این معنی است که تغییر عامل ترشح کننده دوپامین میتواند به کلی عوامل یادگیری را برعکس کند که خواهیم دید

این روش را میتوان به شکل روبرو فرموله کرد

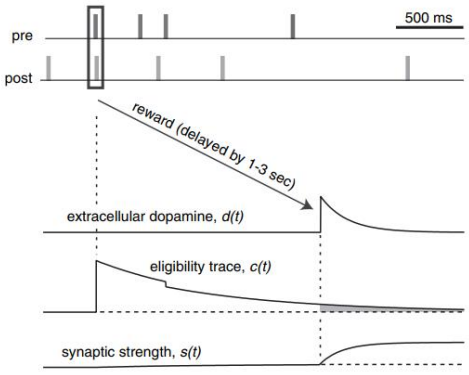
$$\frac{dc}{dt} = -\frac{c}{\tau_c} + STDP(\tau)\delta(t - t_{pre/post}),$$
$$\frac{ds}{dt} = cd,$$
$$\frac{dd}{dt} = -\frac{d}{\tau_d} + DA(t),$$

در این روش تغییر وزن (s) وابسته به دو متغیر (c : یک آنزیم مهم برای رخداد روند یادگیری و d : عامل مربوط به میزان دوپامین ترشح شده)

هر دو عامل c و d درون خود Leaky Term دارند که باعث میشود در صورت نبود محرک مداوم مقدار آن ها کاهش پیدا کند

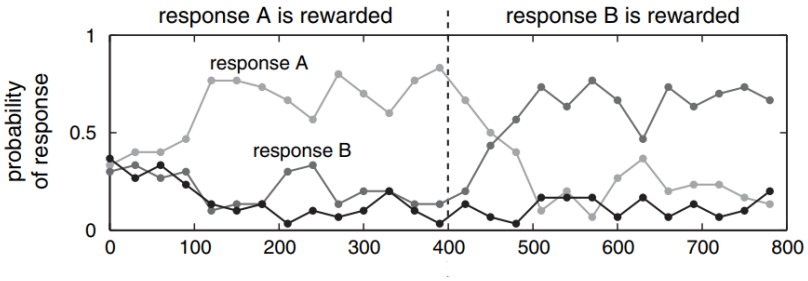
اشاره به این نکته ضروریست که دریافت پاداش توسط مغز ممکن است فرایندی زمانبر باشد که روند یادگیری را سخت میکند. به هر حال میتوان c و d را بطوری تنظیم کرد (به تعیین نرخ Leak کمی برای c که تا چند ثانیه 0 شدنش طول بکشد اینگونه مغز چند ثانیه فرصت خواهد داشت تا با دریافت پاداش توسط یک تصمیم گرفته شده یادگیری را اعمال کند)

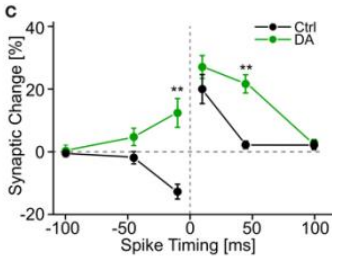
در عکس زیر روند توضیح داده شده را مشاهده میکنیم. و میبینیم که پاداش مربوط به یک تصمیم 1.3 ثانیه بعد ترشح شده اما هنوز یادگیری به دلیل انتخاب نرخ Leak کم برای c قابل انجام بوده است



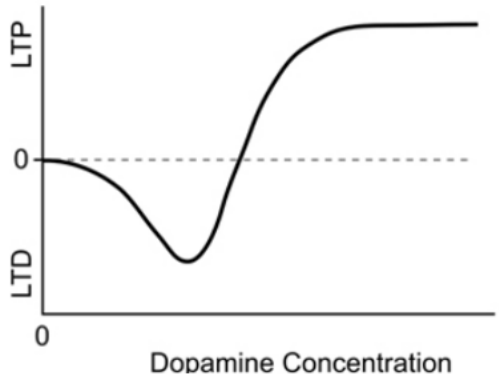
همچنین در شکل زیر میبینیم که بر خلاف STDP در این مدل میتوانیم با تغییر عامل ترشح دوپامین (پاداش) میتوان بکلی رفتار پیشین را تغییر داد

همچنین در سمت چپ تفاوت STDP و RSTDP را میبینیم که با وجود عامل پاداش RSTDP همیشه نتیجه مثبت خواهد داد. ورای ترتیب اسپایکی





تفاوت های فاحشی با STDP در این روش وجود دارد که در توضیحات بالا به طور غیر متمرکز به آن ها اشاره شد اما برای نوعی جمع بندی از این تفاوت ها میتوان گفت یادگیری مستقل از ترتیب اسپایک هاست و مستقیما وابسته به میزان دوپامین ترشح شده است که دقیق تر آن را در این شکل میبینیم. همچنین برای تصمیم گیری روش دقیق تریست و همچنین به نسبت STDP (حالات ساده تر آن) یادگیری توسط تشابه اتفاق اسپایک ها رخ نخواهد داد.



همچنین یادگیری در این روش تعاملی تر است. یعنی با محیط تعاملی بیشتر از STDP برقرار است. البته STDP به نسبت RSTDP یادگیری لحظه ای تری را ارائه میدهد که ممکن است در RSTDP پاداش آنقدری دیر برسد که یادگیری دقیقی رخ ندهد (که البته در خیلی از مسائل مهم زندگی اینگونه است :/)