Assignment 3

Project 1

Description

Implement The STDP method using 2 neurons, one Pre synaptic and one Post synaptic neuron

Experiments

In this project I've implemented the STDP rule with the following approach :
Every time one neuron spikes I used a window of 500ms and each Spike I find on the other neuron's spike chart, I'll apply the STDP with the pair founded.
Also for when I'm about to charge the Post Synaptic neuron with the Pre Synaptic Spike I've used 2 methods, one I'll give a burst of a high amount of Input Current for a short period (only the next time stamp of the Post Synaptic neuron's Input will change) and on the second one I gave a more continues but less amount of Input Current as charge for a higher duration (meaning the effect of each spike received lasts for some amount of time stamps (5500ms).

Exp.	*I*	*Fire Method*	*A+ and A-*	*Spike Detection*
1	Constant Slope	Continues	50, -40	5500ms Window
2	Sin wave	Burst	50, -40	Last Spike
3	Sin wave	Continues	50, -40	Last Spike
4	Zero Slope 1	Continues	50, -40	5500ms Window
5	Zero Slope 2	Continues	50, -40	5500ms Window
6	Zero Slope 3	Continues	50, -40	5500ms Window

Table 1: Parameters of each experiments

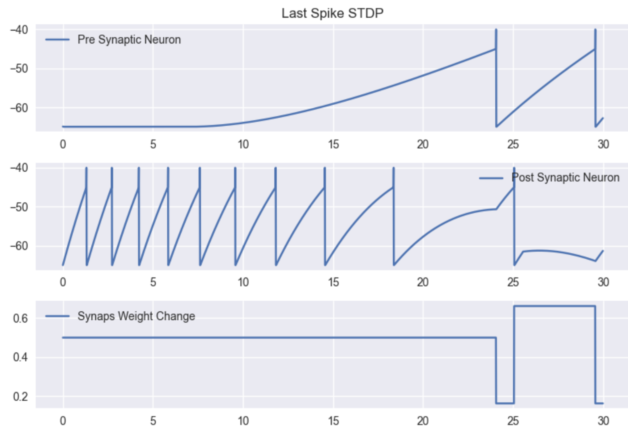
Result and Discussion

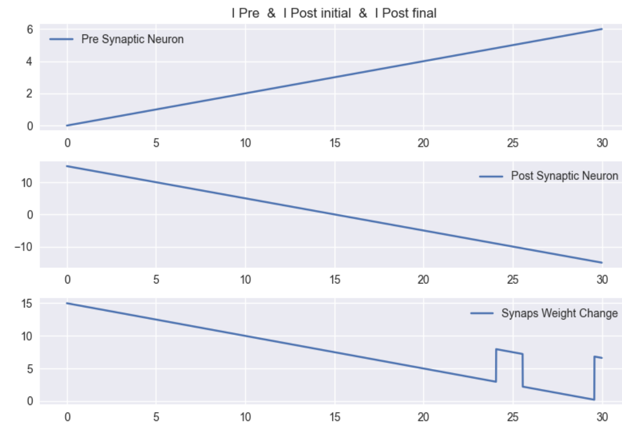
In the first experiment we'll see how the STDP reacts to constant progression of Input in the Pre Synaptic neuron :

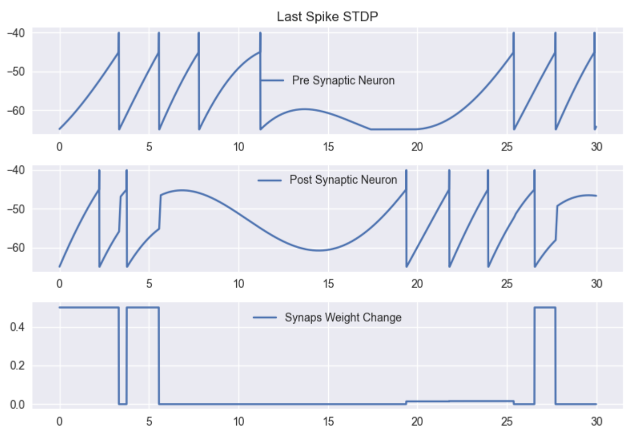
The left column shows the change of U in each neuron and Synaptic Weight History, and the right column shows the I applied to each neuron (the bottom most plot is the Input Current of Post synaptic neuron after the simulation when this neuron had received spiked, you can see the difference between the firing methods in this plot and why I used Continuous method for the rest of the experiments cause it made more sense comparing to the Burst method.

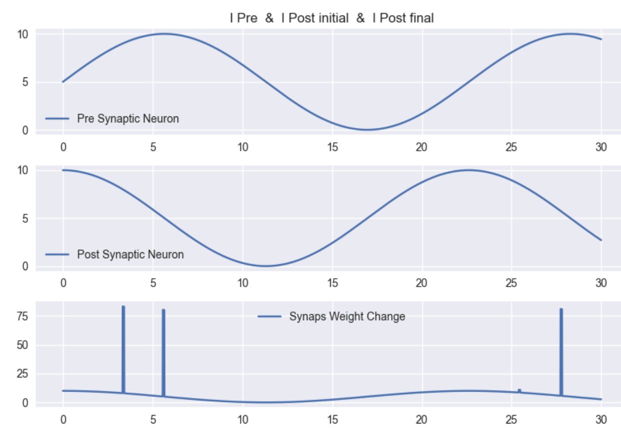
Each row represents the corresponding experiment

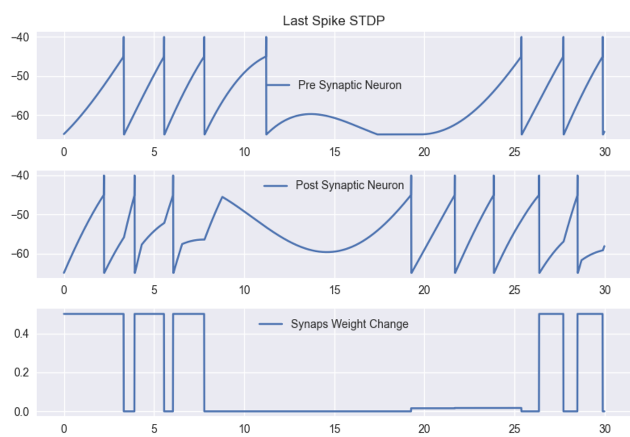
(And don't mind the legends on the right column)

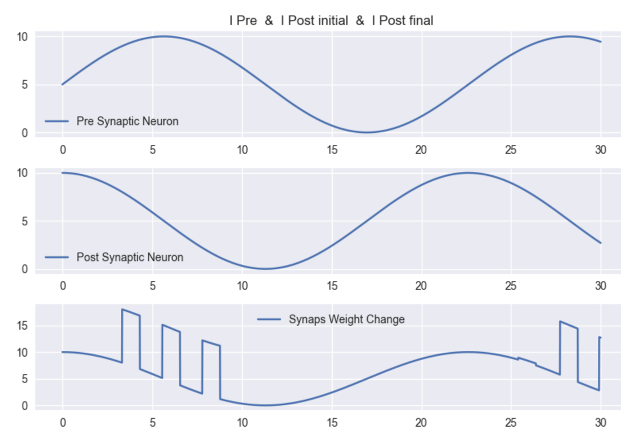

Last Spike STDP
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

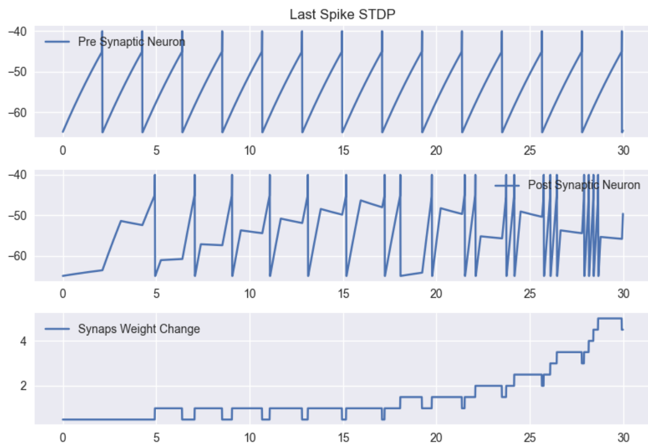

I Pre & I Post initial & I Post final
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

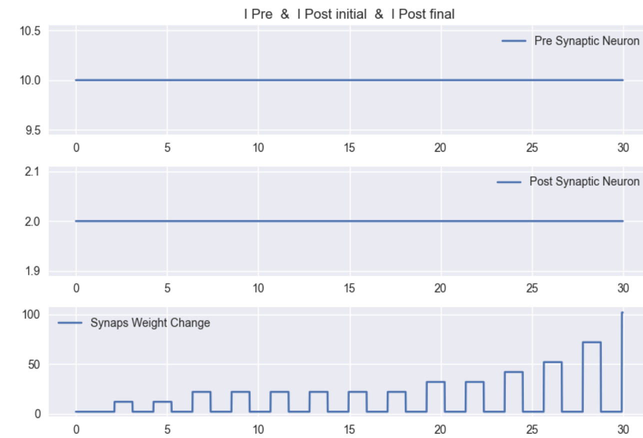

Last Spike STDP
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

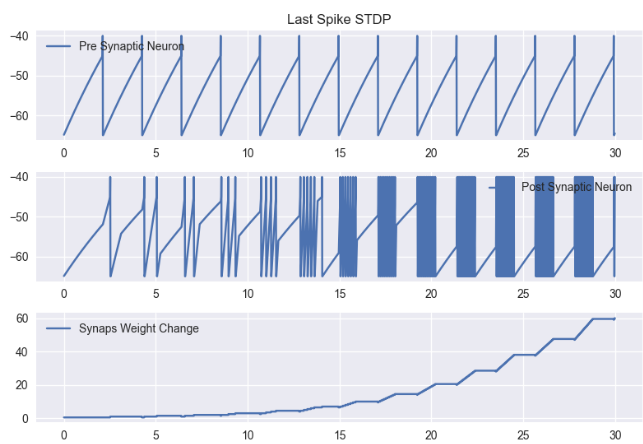

I Pre & I Post initial & I Post final
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

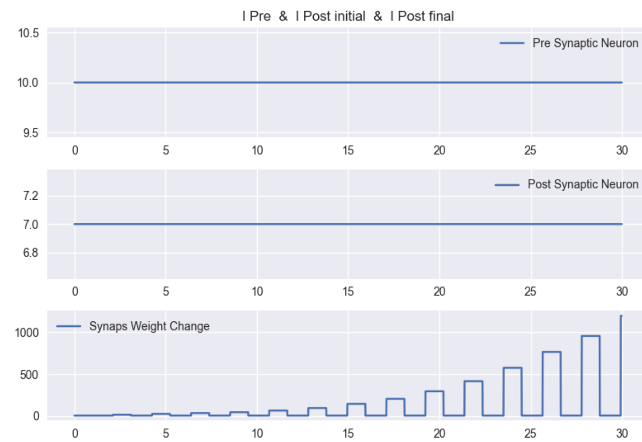

Last Spike STDP
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

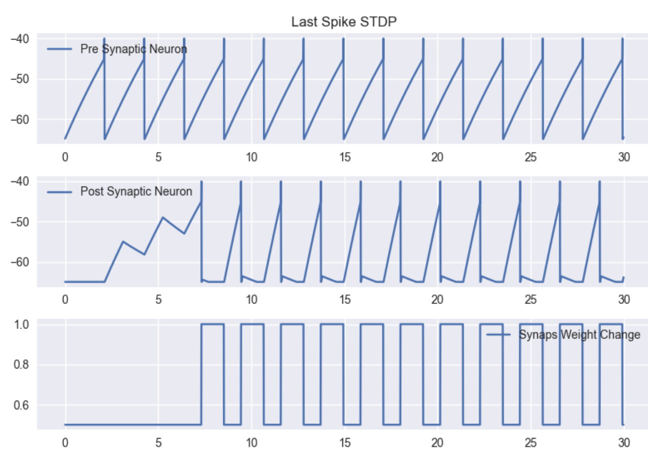

I Pre & I Post initial & I Post final
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

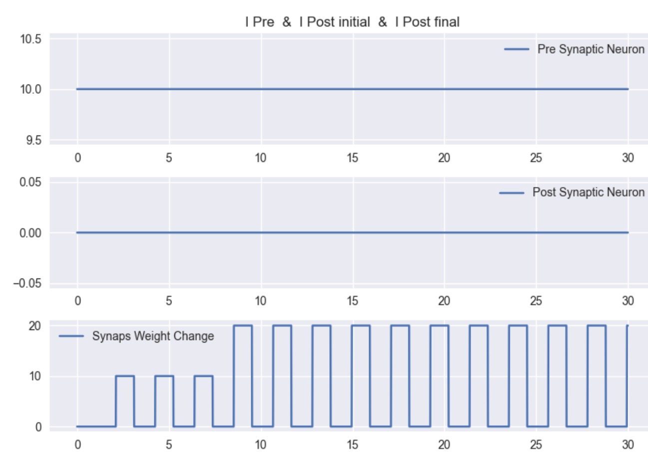

Last Spike STDP
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change


I Pre & I Post initial & I Post final
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change


Last Spike STDP
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

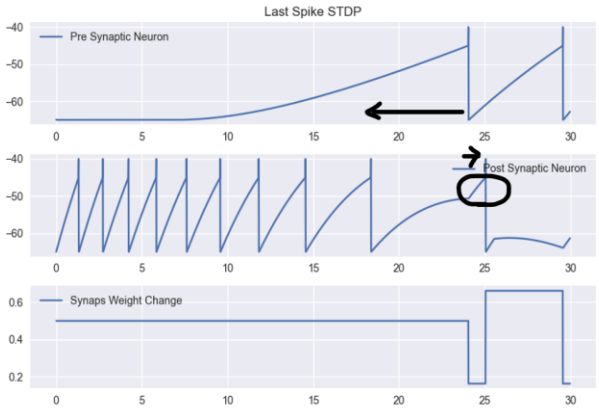

I Pre & I Post initial & I Post final
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change


Last Spike STDP
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

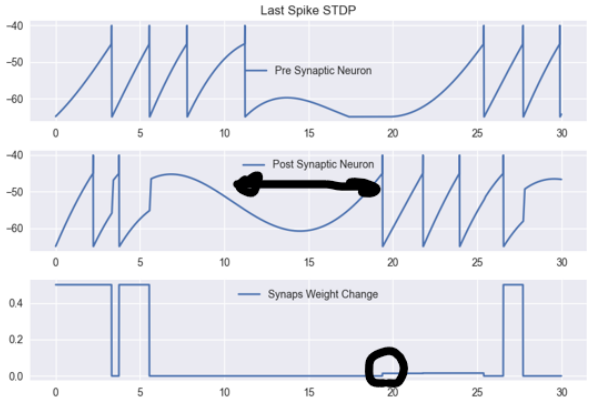

I Pre & I Post initial & I Post final
Pre Synaptic Neuron
Post Synaptic Neuron
Synaps Weight Change

for the A+ and A- parameters I've used the fixed value of 50 and -40 mostly because other variations of this combination would have yielded either very dramatic results or not much learning to take place (lowering these variable resulted on our weight becoming pale enough to vanish the Input change of post synaptic neuron due to the firing of pre synaptic one, and increasing it would have make the post synaptic neuron to enter a period of dramatic spiking after each spike of pre synaptic neuron something like what we saw on experiment 5.
So the range selected for A+ and A- was around 30 – 100 and 50 was suitable choice.)

You can see below (experiment 1), that as happened with most of the experiments the presynaptic spike was the direct cause of spiking in the post synaptic neuron and so cause to a sudden LDP of our synapse weight



Also as seen in experiments 4, 5 and 6, in order to have a balanced synapse its Needed to use a lower Input Current for the post Synaptic neuron as if we don't The weight would escalate in according to the constant progression of previous LTPs causing more and more LTP to take place with each step



Also with looking at experiment 2 and 3 we can observe the difference (slight but visible) of using a window to detect pairs to apply STDP to, and using the last spike to detect the pair. As the STDP applied in the 18th second of the simulation couldn't be detected by a window of (5000ms) but still got used as a learning pair in this method. The whole purpose of using a last spike method as a pair detection for experiment 2 and 3 was to demonstrate this. The rest use 5000ms window.

Project 2

Description

Implement a SNN using the STDP method using 10 input neurons, and 2 output neurons, apply 2 different connectivity patterns and compare the results

Experiments

For this part I have used three different connectivity pattern, each for specific use. You're seeing the parameters used in each experiment in the table below.
Some use the same Input current gave to all the inputs, and some use different inputs for each neuron in the input layer. (3 and 4 use random Input changes and 5 uses a queue like Input meaning that each neuron gets activated for a specific duration right after the previous neuron stops receiving input, it will make more sense when seeing the plots.

Exp.	*I Distribution*	*Connectivity Pattern*
1	Same	[0, 1, None, 0, 1, None, 0, 1, None, None]
2	Same	[1, 1, 0, 0, 1, 1, 0, 0, 1, 1]
3	Different (Random)	[1, 1, 0, 0, 1, 1, 0, 0, 1, 1]
4	Different (Random)	[1, 1, 0, 0, 1, 1, 0, 0, 1, 1]
5	Different (Queue)	[1, 1, 0, 0, None, None, 1, 1, 0, 0]

Table 2: Parameters of each experiments

Result and Discussion

As we will see in the pictures after each time an STDP rule Applies a LTP (long term potentiation) we'll have a stronger synaptic weight so the next time would each input neuron spikes, the output neurons will excite more and more easily, by itself it demonstrates a kind of learning in the network for when a specific combination of the neurons on the input layer spike together (on of the ones that the network has been trained one or one similar to the train set), the specified output neuron follows by spiking. But for this cause we'll need to apply more than 1 epoch (the results shown are only one epoch of training for each experiment) to actually teach the network (have enough of synaptic strengths) to react to input combinations right after.

The results shown below do speak for them selves but I've tried to explain the notable properties of each below the pictures

Each row represents the corresponding experiment and of each row the left column represents the potential change of 1^{th} output neuron and the right column showing the 2^{th} output neuron

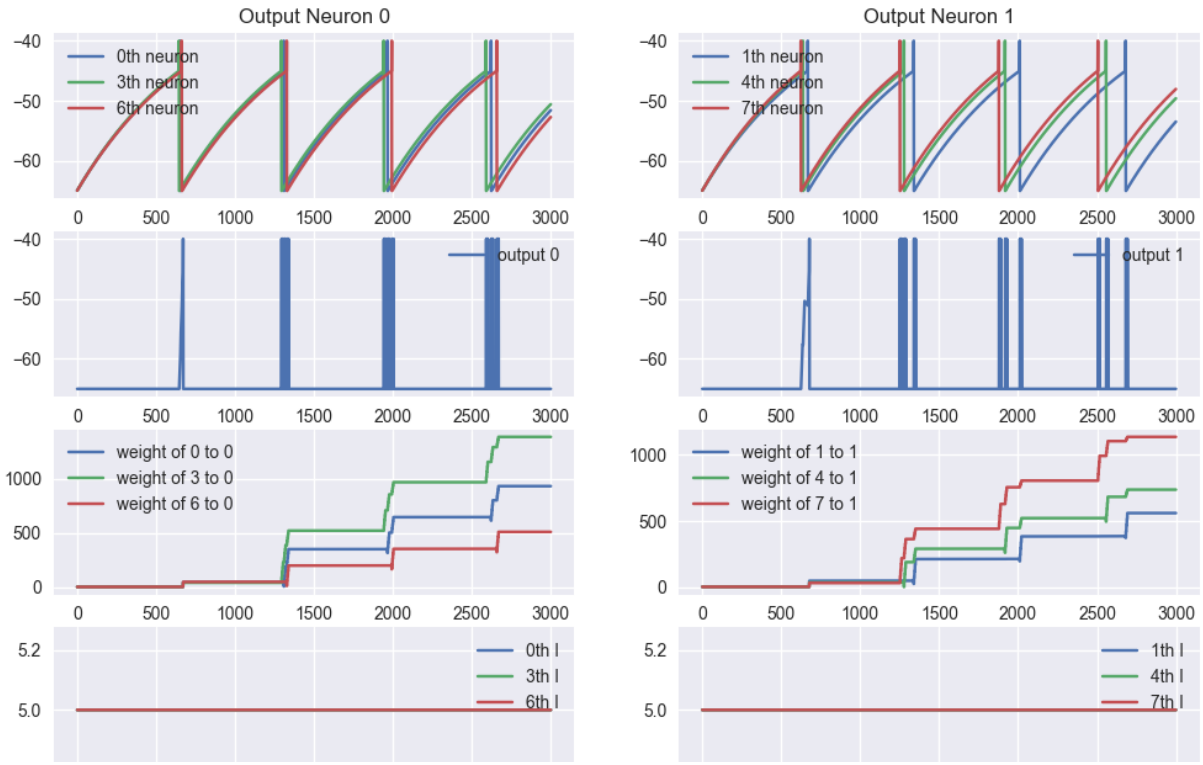


Figure 1 : Experiment 1

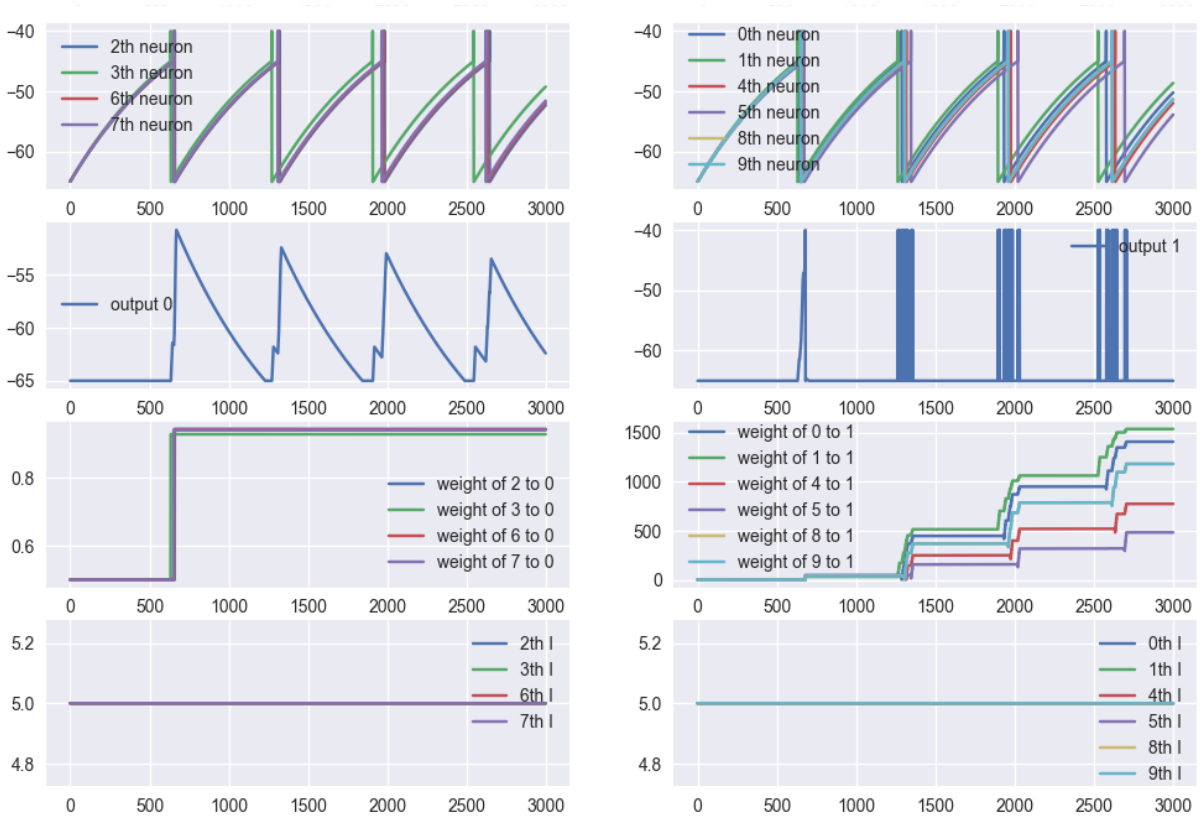


Figure 2 : Experiment 2

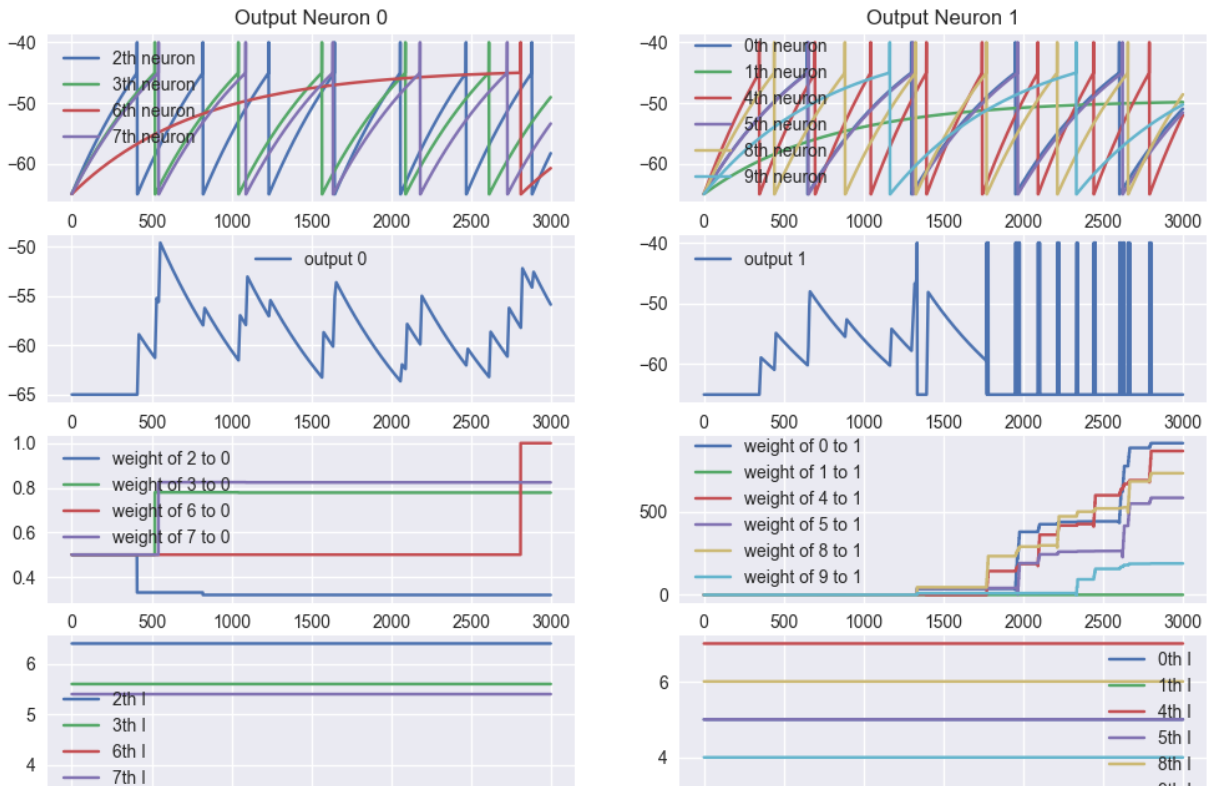


Figure 3 : Experiment 3

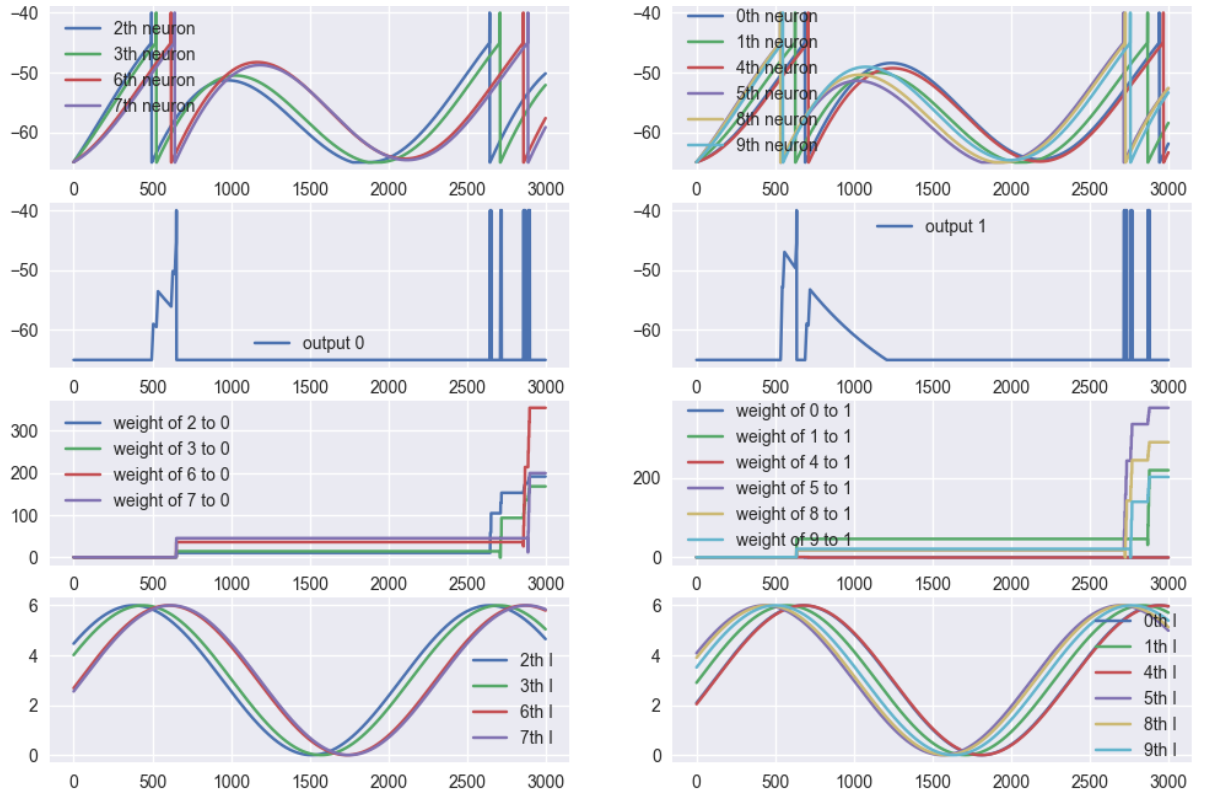


Figure 4 : Experiment 4

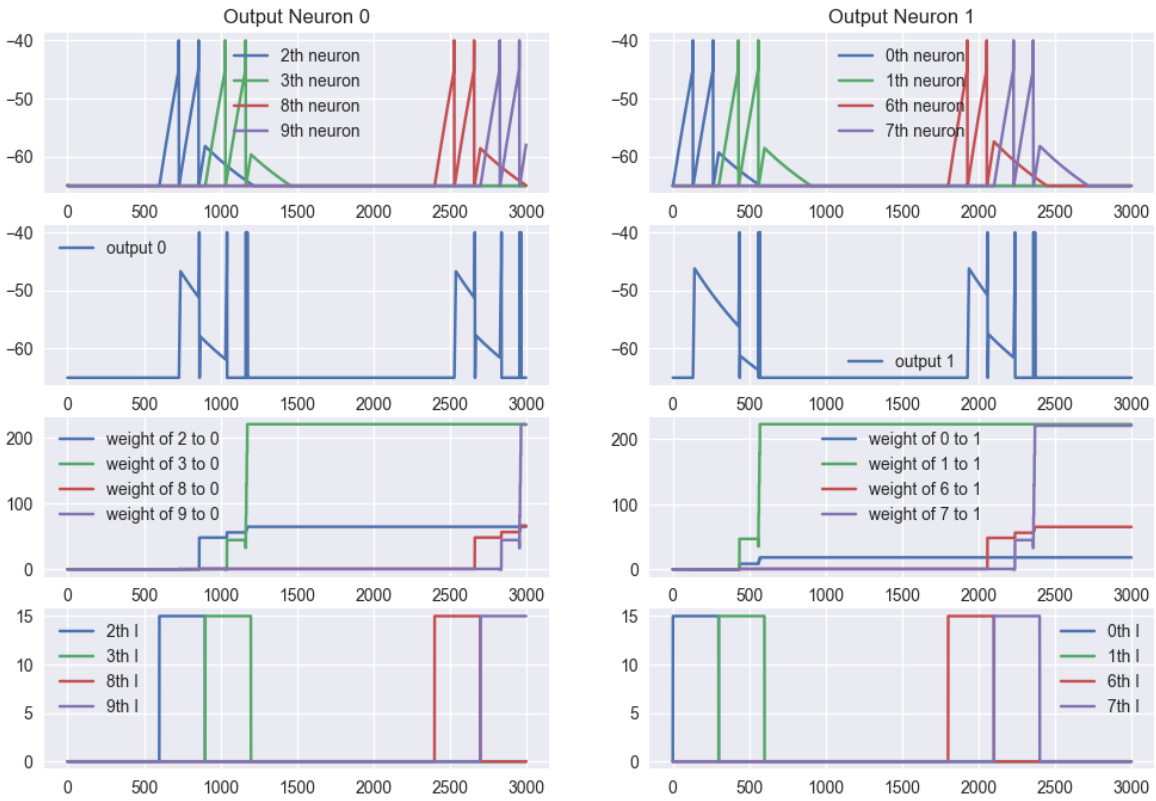


Figure 5 : Experiment 5

1st Experiment : for this and the rest of the experiments I added a random noise to the resistance of each neuron created (around 5 percent added or subtracted from the Base number which was 10 so that we can actually distinguish the neurons from Each other. In this experiment we see the potentiation of the synapses in the 3th row Of figure 1

2nd Experiment : in contrast to the previous figure here we see due to the low number Of connections to the 1th (index 0) output neuron, this neuron couldn't even start the Learning process as it never spiked so that we could have a pair to apply STDP on But the 2nd (index 1) output neuron received enough inputs to trigger it's first spike And then from that point the learning could be begin and LTPs could be achieved

3th Experiment : same as the previous experiment but with random inputs which Yielded the same results

4th Experiment : it's notable to know that I had to use a really low number for A- Comparing to A+ because the LTD would be dominating and reduce the weight to Almost zero (because we have much more Spikes on the input layer, and whenever An Input neuron spikes this will yield to LTD and whenever the output neurons Spike It will lead to a LTP

5th Experiment : here I have used a queued input to the neurons, the idea behind this Was to see if could lead to the point where each neuron could excite the specified

Output neuron alone, of course by applying the above experiment on higher numbers
Of epochs (5 or so probably) we could achieve a point where each neuron of the input
Layer can excite the output layer which it corresponds to (one of the applications
It could have in real life is that the network will learn to answer for seeing an object
From only one perspective (assuming that by seeing different perspectives we mean
The combination of input neurons activating for example seeing a toy car from above
Excites neuron 1 of input layer, seeing it from sides excite neuron 2 and 3 of input
Layer and so on … and the neuron 1 of output layer corresponding to a toy car)
With this approach we could react correctly to a toy car by having only one input
Layer neuron active at a time)

Project 3

Description

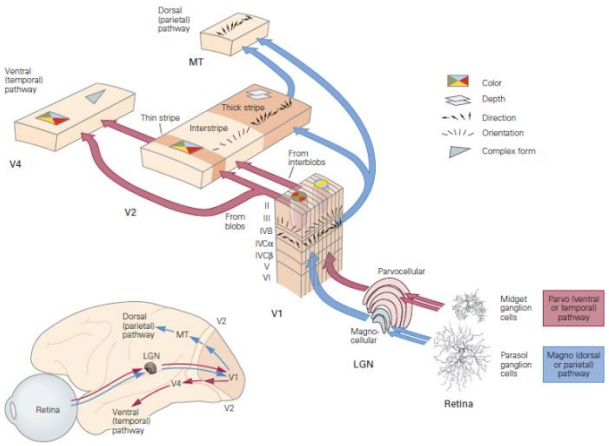
کاربرد شبکه های عصبی اسپایکی عمیق درون مسئله بینایی کامپیوتری

Summary

این مقاله یک ساختار عمیق شبکه عصبی اسپایکی را معرفی میکند که از 3 لایه کانولوشن (پیچیده) برای استخراج ویژگی ها استفاده میکند. لایه اول خطوط و زوایا را بیرون کشیده. لایه دوم با استفاده از خطوط و زوایای استخراج شده اشکال ساده ای را استخراج کرده و لایه سوم اشکال پیچیده تری را استخراج میکند. همچنین در ابتدای این شبکه از یک لایه DoG نیز برای شبیه سازی کار سلول های گنگلیون و استخراج تضاد در پنجره های کوچک تصویر استفاده میشود (قبل از لایه اول)
نوع یادگیری در این مدل Unsupervised است و مشابه سیستم مغزی در آن عمل شده
برای قانون یادگیری این مدل از قانون STDP استفاده میکند
برای توضیح دقیق تر این مدل ابتدا مروری بر مباحث مربوط به بیولوژی مغز میکنیم

Detailed Explanation

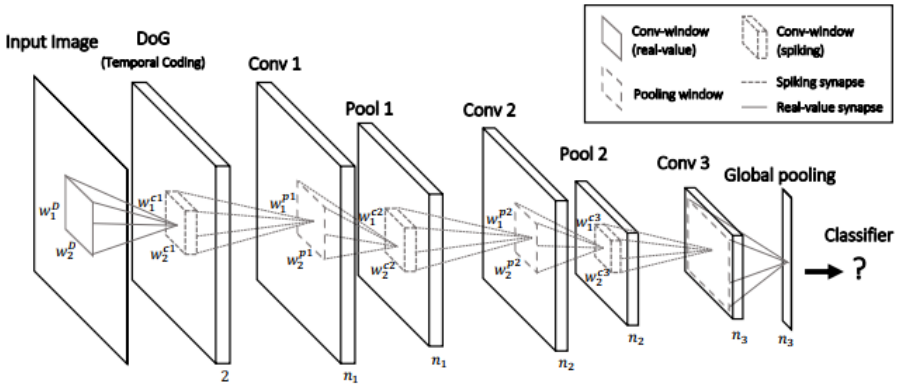
این مدل سازی متشکل از چند لایه کتنولوشنی که هر کدام مختصی شبیه سازی بخش خاصی از سیر حرکتی سیگنال های بینایی در کورتکس بینایی مغز است هستند.
روند انتقال سیگنال های بینایی در مغز در تصویر زیر آمده است



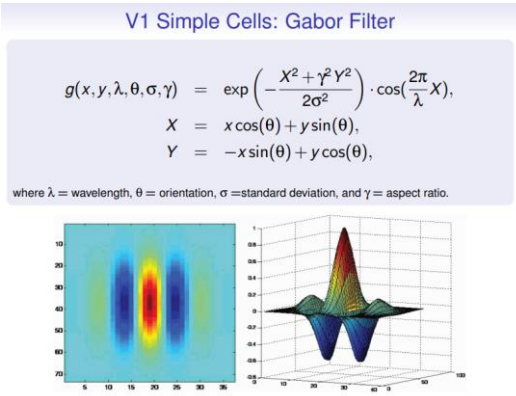
در این عکس میبینیم که ورودی بینایی که توسط گیرنده های قرنیه به LGN و سپس به V1 منتقل میشوند، در این لایه خطوط و زوایای مشخصی از آن ها استخراج میشود. سپس این اطلاعات به V2 (بخش محدودی هم به V4 و MT منتقل میشود) و در هر بخش اطلاعات پیچیده تری استخراج خواهند شد که خود این فرایند از اطلاعات لایه قبلی استفاده میکند.
سپس پس از پردازش اطلاعات در ناحیه V4 اطلاعات وارد ناحیه IT میشود که این بخش پردازشات در پیچیده ترین سطح خود انجام شده و اشکل و اشیاء پیچیده ای مانند دوچرخه و صورت ها استخراج میشوند

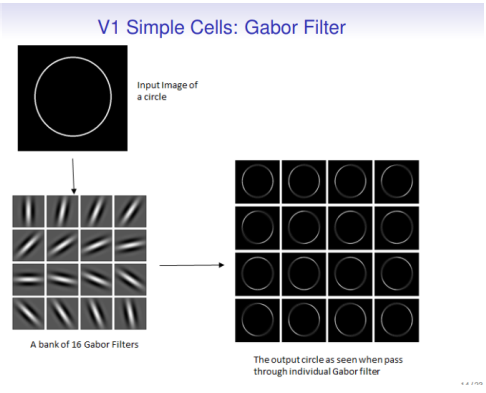
این بخش ها توسط مجموعه ای از 3 لایه کانولوشنی ساخته میشوند.

این ساختار به شکل زیر عمل میکند :



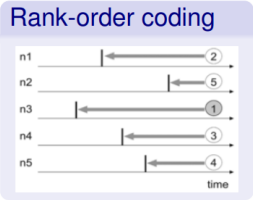
ناحیه V1 با کمک سلول های خاصی که در 18 زاویه مشخص قرار گیری کرده اند زوایا را استخراج میکند که ما میتوانیم با کمک یک فیلتر به اسم فیلتر گابور همین کار را انجام دهیم
عکس زیر فرمول استفاده شده در این فیلتر را نشان میدهد





در عکس سمت راست میتوانم فیلتر های گابور مربوط به هر زاویه را که بروی مرز یک دایره اعمال شده اند ببینیم.
نکته اینجاست که در این لایه انتخاب خروجی (و همچنین در لایه های دیگر کانولوشن) با استراتژی winner takes all انجام میشود به این معنی که اولین اسپایک در هر فیلتر به عنوان خروجی این لایه مشخص میشود و تماما یادگیری از طریق همین نورون بروی کل لایه انجام میشود

در لایه DoG (که صرفا کار آن شبیه سازی کار مربوط به سلول های گنگلیون در لایه قرنیه چشم است که باعث میشود بتوان تضاد های رنگی را در پنجره هایی با سایز های کوچک بیرون کشید تا در ادامه در V1 از آن ها استفاده شود) از انکودیکی به اسم Rank-Order Coding استفاده میکند که بطور خلاصه درباره این نوع روش کد کردن میتوان گفت :



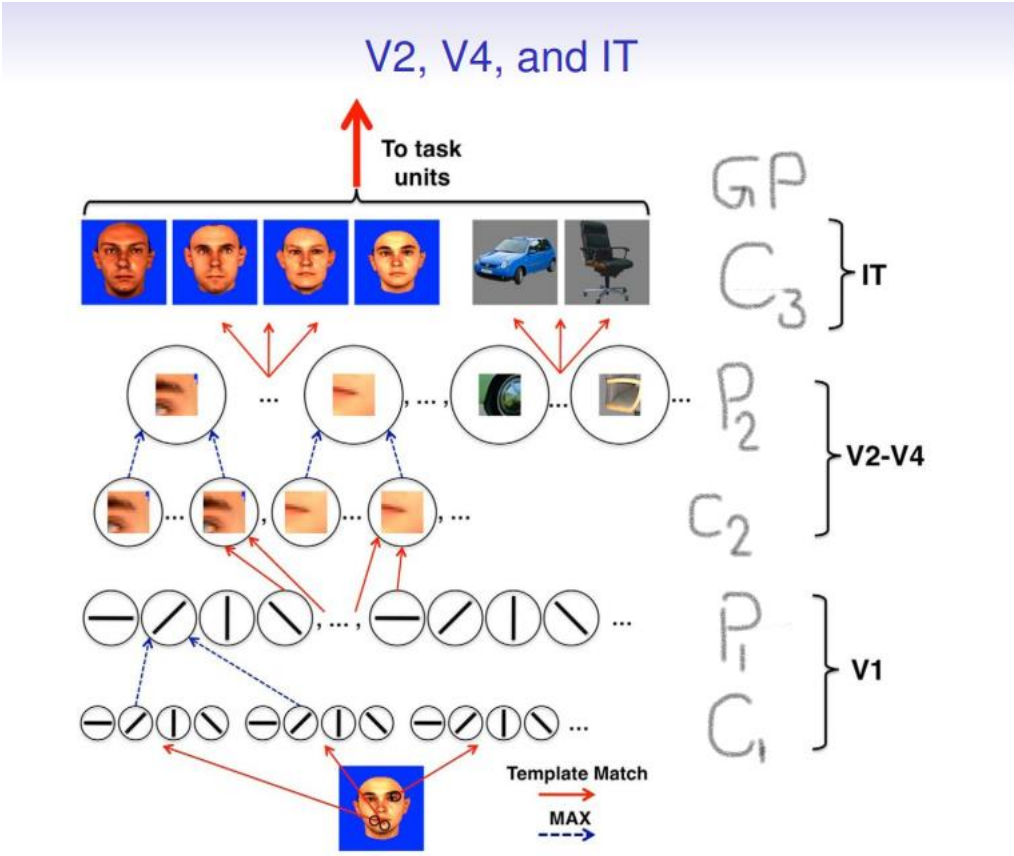
در واقع این نوع کد کردن فقط به ترتیب اسپایک ها در یک جمعیت (اینجا یک لایه) اهمیت میدهد و دلیل رخداد استراتژی Winner Takes All هم همین است

این ناحیه در مدل سازی مقاله در لایه Convolution اول به انجام میرسد و سپس با استفاده از یک لایه Pooling که که در واقع یادگیری ای در آن انجام نمیشود، اطلاعات چکیده شده و به لایه بعدی فرستاده میشوند

Conv2 : در این لایه پردازشات مربوط به نواحی V2 و V4 انجام میشود که کار استخراج وجه ها و اشکال را دارند روند انجام شده در این لایه همانند لایه قبل انجام میشود تنها ویژگی های پیچیده تری را استخراج میکنند

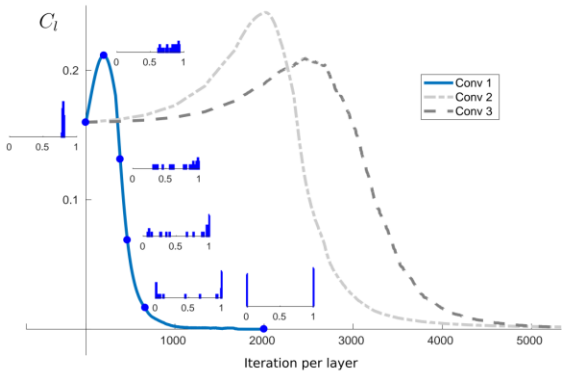
Conv3 : در این لایه پردازشات مربوط به بخش IT انجام میشود. در نهایت در این بخش ویژگی های سطح بالایی مانند صورت ها یا ماشین یا صندلی استخراج میشوند و نورون مربوط به فعال شدن نشان میدهد چه تصویری تشخیص داده شده تفاوت دیگر این لایه با لایه های قبل در این است که در این لایه پولینگ اعمال شده بصورت جامع بروی تمام فیلتر اعمال میشود که خود یعنی تعداد فیلتر های ورودی به تعداد محدودی فیلتر خروجی تبدیل میشوند که در اصل تعداد اشیاء مورد نظر ماست.

خلاصه ای از اتفاقات درون این چند لایه را در عکس زیر میبینیم :



در نهایت یک لایه Classifier برای تصمیم گیری درباره شکل تشخیص داده شده استفاده میشود (این لایه را بر خلاف عدم وجود آن در مغز باید به مدل اضافه کرد تا بتوانیم تصمیمات گرفته شده را برچسب گذاری کرد)

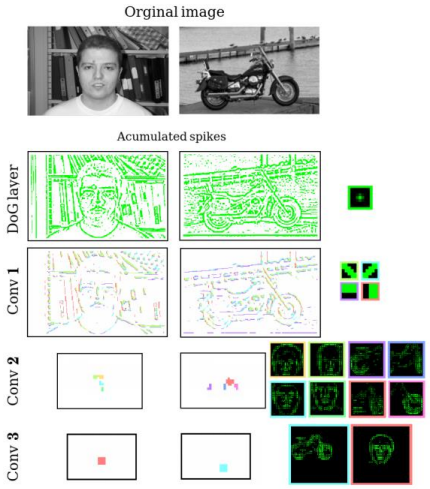
برای روند های توضیح داده شده در بالا از یک ورژن کمی تغییر یافته ای از STDP استفاده میشود که به شکل زیر برای آن عمل میکنیم :
موقع روند یادگیری نورون های درگیر در این یادگیری به طور خودکار به سمت ویژگی هایی کشیده میشوند که در تصاویر ورودی



یادگیری لایه l ام با فرمول زیر انجام میشود (Wf,i) وزن نورون i ام از لایه ویژگی f ام است و Nw تعداد تمام سناپس ها است

$$C_l = \sum_f \sum_i w_{f,i}(1 - w_{f,i})/n_w,$$

همانطور که از فرمول پیداست زمانی که تمام وزن ها به سمت 1 یا 0 میل کنند یادگیری متوقف میشود (به طور شهودی میتوان نتیجه گرفت این کار باث میشود تصمیم گیری ها به سمت قطعی تر شدن و به تبع آن به سمت دقت های بالاتر (لزوما نه اما عموما بله) پیش رود

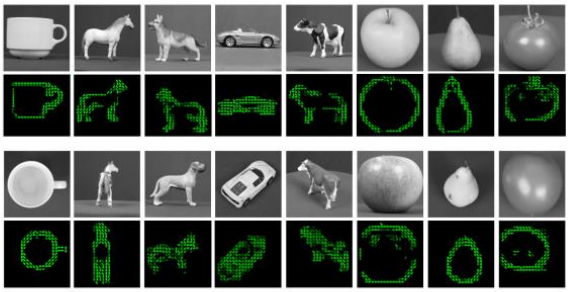


در تصویر روبرو روند بالا را که روی یک عکس اعمال شده میبینیم که در هر لایه تعدادی فیلتر نماینده هر ویژگی خواهند بود (تعداد فیلتر ها را ما مشخص میکنیم بر اساس نیازمان)
در در نهایت در لایه آخر نورونی در فیلتر مربوط به موتور میتواند وجود موتور را تشخیص دهد که سپس در پولینگ جامعه بعد از آن و سپس Classifier موتور انتخاب میشود (همینطور برای عکس صورت)

نتایج این مدل بروی دو دیتا ست را در زیر میبنییم که درصد های خوبی را نمایش میدهد (ردیف های آخر)

Experimental Results (ETH-80)

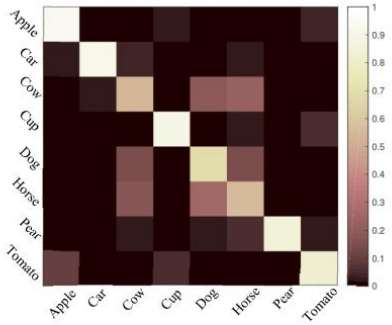
The preferred feature of an activated neuron in the third convolutional layer on ETH-80 dataset:



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:



Experimental Results (MNIST dataset)

MNIST

10 classes

~10000 training & ~1000 testing samples per class

Gabor-like features learned by first convolutional layer

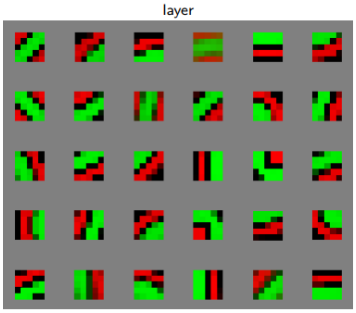


Table: Recognition accuracies of the proposed DSNN and some other SNNs over the MNIST dataset.

Architecture	Neural coding	Learning-type	Learning-rule	Accuracy (%)
Dendritic neurons (Hussain et al., 2014)	Rate-based	Supervised	Morphology learning	90.3
Convolutional SNN (Zhao et al., 2015)	Spike-based	Supervised	Tempotron rule	91.3
Two layer network (Querlioz et al. 2013)	Spike-based	Unsupervised	STDP	93.5
Spiking RBM (O'Connor et al. 2015)	Rate-based	Supervised	Contrastive divergence	94.1
Two layer network (Diehl & Cook, 2015)	Spike-based	Unsupervised	STDP	95.0
Convolutional SNN (Diehl et al., 2015)	Rate-based	Supervised	Back-propagation	**99.1**
Proposed DSNN	Spike-based	Unsupervised	STDP	98.4

Project 4

Description

توضیح Reward Modulated STDP

Summary

بطور خلاصه میتوان گفت این روش از مکانیزم تنبیه و پاداشی برای تقویت سرعت و دقت یادگیری به نسبت STDP استفاده میکند که در مغز این عامل توسط ترشح ماده ای بنام دوپامین انجام میشود

Detailed Explanation

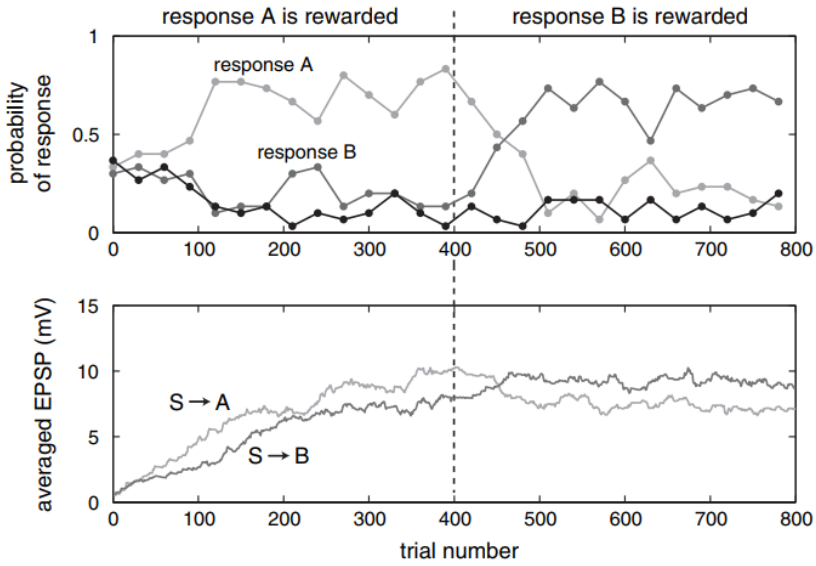
این روش عامل شرطی شدن نورون به ورودی های خاص است
با اضافه کردن یک ترم جدید که مستقیما مرتبط با ترشح دوپامین است در این مدل میتوان خروجی های در هر صورت تقویت کننده در زمان ترشح دوپامین داشت. به این معنی است که تغییر عامل ترشح کننده دوپامین میتواند به کلی عوامل یادگیری را برعکس کند که خواهیم دید.

این روش را میتوان به شکل زیر فرموله کرد

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

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)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 اتفاقی 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

در این روش از غلظت ترشح شده دوپامین برای عاملی به منظور تشخیص مقدار و جهت (LDT یا LTP) یادگیری استفاده میشود.
با این توصیف یعنی دوپامین میتواند به طور کلی رفتار دو نورون را جابجا کند



یعنی همانطور هم که در عکس بالا میبینیم میتوان با تغییر عامل ترشح کننده دوپامین مفاد یادگیری را بطور کامل برعکس کرد همچنین همانور که در عکس زیر دیده میشود میتوان با استفاده از دوپامین به یادگیری هایی دامن زد که پاداش یک عمل با تاخیر نسبتا زیادی از تصمیم گرفته شده مربوطه بیاید. اینکار با وجود متغیر دیگری به اسم c(t) انجام میشود که مشخص میکند زمانی که بین یادگیری و تصمیم وقفه بیافتد تا چه حد میتواند طولانی شود (نرخ نشت c هر چه کمتر باشد یادگیری امکان وقفه های بلند تر را خواهد داشت