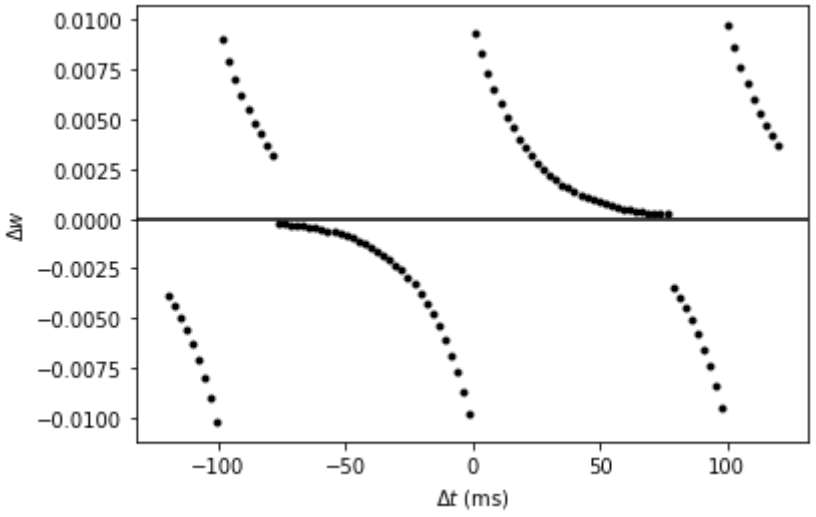
تمرین سوم علوم اعصاب محاسباتی

محمد زیاری - 97222047

1. تمرین اول راجب برقراری قانون stdp بین دو نورون است. از آنجایی که ما با NeuronGroup اگر یک نورون pre و یک نورون post در نظر بگیریم (در واقع 2 نورون پشت سر هم) مقدار زمان اسپایک دو جمعیت نورونی ما از رابطه ای بدست می آید که مقدار مخرج آن N-1 است و اگر تعداد نورون ها را یک در نظر بگیریم این مقدار 0 میشود و قاعدتا به مشکل می خوریم.

از این رو مقدار در نظر گرفته شده برای N را بیشتر در نظر میگیریم و در مقادیر مختلف نمودار وزن آن را مشاهده میکنیم.

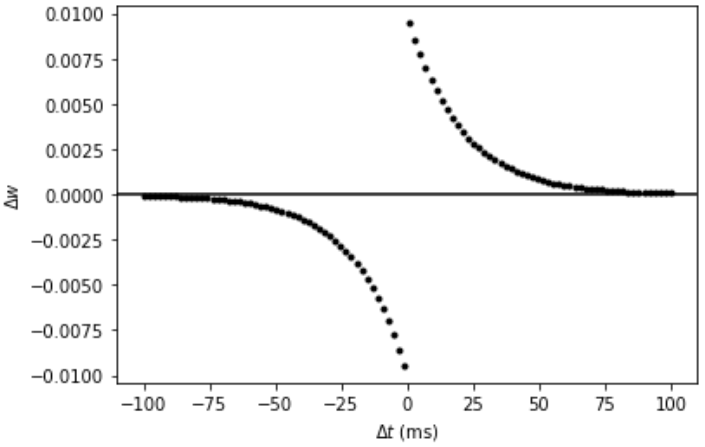
اما در رابطه با کد ابتدا متغیر های مختلفی که در معادله وجود دارند را مقدار دهی کردیم که البته در ادامه گزارش به بررسی تغییرات بر روی آنها می پردازیم. مقدار tmax باید کمتر از refractory که در نورون گروپ تعریف میشود باشد تا یک سیکل و نموداری شبیه به نمودار وزنی که برای stdp متصور میشویم داشته باشیم. برای مثال اگر میزان tmax برابر با 120 و refractory هم 100 باشد خواهیم داشت :

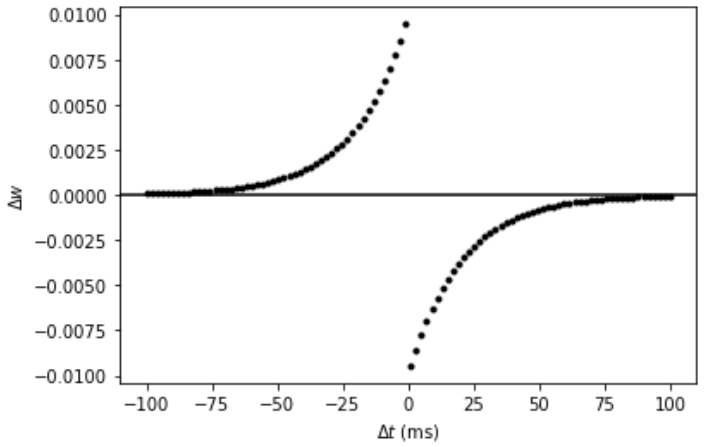


که نمودار دلخواه ما نیست. برای انطباق آن میزان tmax را 100 و refractory را 200 در نظر گرفته ایم.

سپس threshold ها را در نورون گروپ قبل از رسیدن به tspike در نظر میگیریم و tspike نورونهای pre و post را برابر با فرمولی که قبلا هم در رابطه اش بحث کردیم. سپس سیناپس را با فرمول ها ساخته ایم و پس از متصل کردن ، نمودار تغییرات وزن بر حسب تغییرات زمان را برای آن رسم کرده ایم.

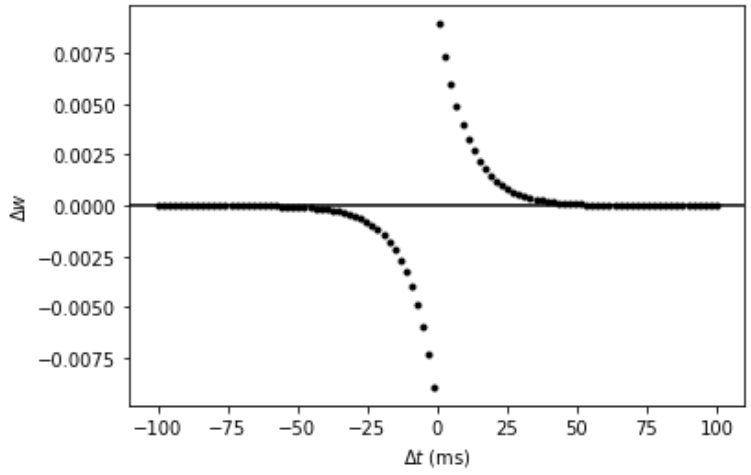
حالا نوبت آن است که راجب متغیرهایی صحبت کنیم که میخواهیم آنها را تغییر دهیم و نتایج را مقایسه و بررسی کنیم. ابتدا تاثیر a_pre و a_post را بر روی جوابمان بررسی میکنیم. میدانیم برای آنکه نموداری مشابه قانون stdp داشته باشیم طبق معادله ای که تعریف شده است ، apost باید کمتر از apre باشد (زیرا مقدار منفی آنها در معادله قرار می گیرد). حالا دو نمودار پایین بررسی هر دو حالت است در حالی که بهم بسیار نزدیک هستند. اولین نمودار در حالتی که apre=0.01 و apost=-0.01 است و دومی در حالت برعکس آنها یعنی post قبل pre اسپایک بزند:



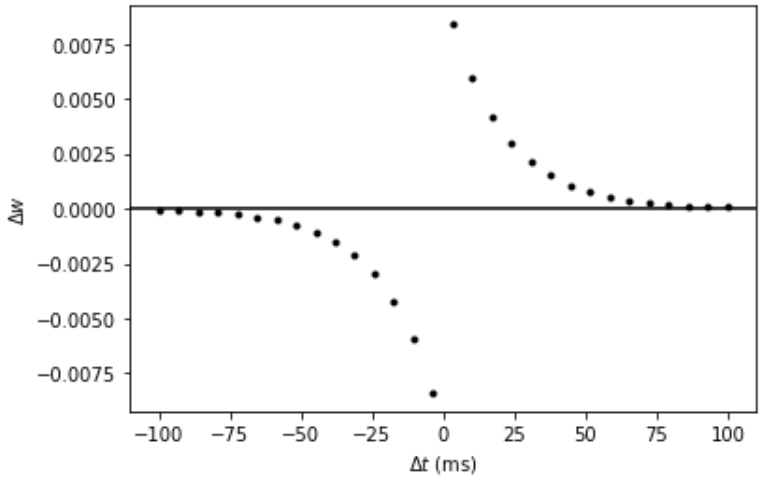


همانطور که مشاهده میکنیم با این کار نمودار ابتدایی ما درست مانند قضیه stdp شده است زیرا نورون pre قبل post اسپایک میزنند اما در نمودار دوم دقیقا نمودار برعکس شده است و نشان دهنده آن است که اسپایک زدن نورون post هیچ ارتباطی با pre ندارد.

همینطور با کم کردن taupost و taupre مشاهده میکنیم که نمودار فشرده تر میشود و قاعدتا در حالت عکسش نمودار پراکنده تر میشود. برای مثال نمودارهای بالا در حالتی بود که این پارامترها را 20ms در نظر گرفتیم. اگر آن را 10ms کنیم خواهیم داشت:

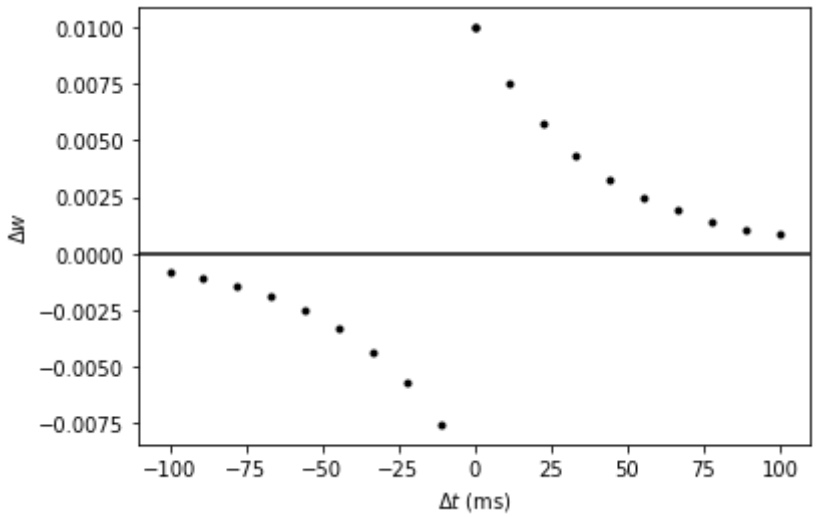


و در آخر در مورد تاثیر N اگر تعداد نورونها را کاهش دهیم قاعدتا نقطه هایی که در نمودار مشاهده میکنیم کمتر خواهد شد. لذا برای پیوستگی بیشتر میتوانیم N بزرگتر انتخاب کنیم. مثلا N در نمودار های رسم شده بالا 100 بود. حالا اگر این تعداد به 30 تا کاهش یابد خواهیم داشت:



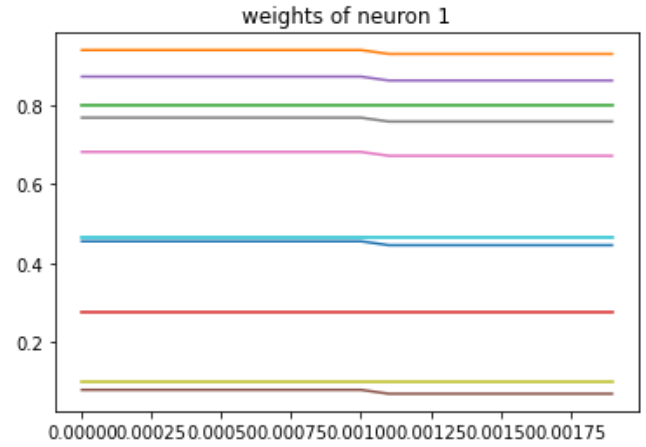
2. کاری که باید در این سوال انجام دهیم آن است که نورون گروپی با 2 نورون خروجی و 10 نورون ورودی بسازیم . نکته کار این است که اسپایک ها باید به صورت الگویی به نورون گروپ ما داده شوند.

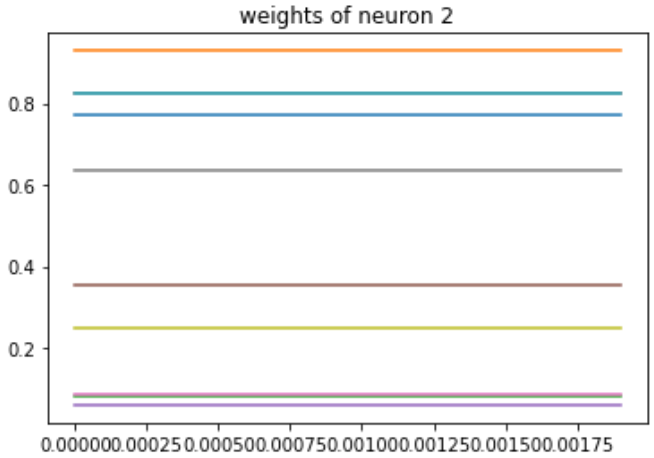
ابتدا ما نورون گروپی بدون آنکه الگویی به آن بدهیم ساختیم که مشابه سوال قبل است فقط نورونهای ورودی و خروجی را طبق سوال تغییر دادیم که نمودار وزن برحسب زمان آن را طبق پارامترهای از پیش تعیین شده در پایین مشاهده می کنیم :



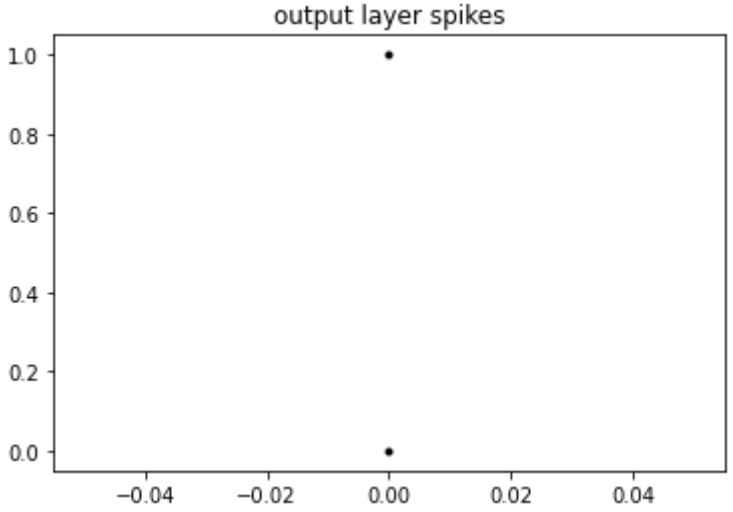
اما در حالتی که آرایه داده می شود که آن را به طور لحظه ای و در **SpikeGeneratorGroup** دادیم ، که شامل آرایه هایی از اندیس نورون هایی که اسپایک میزنند و دومین آرایه زمان اسپایکشان است. در واقع خانه هایی از آرایه که صفر بودند و اسپایک میزنند را در خانه های آن در نظر نگرفته ایم و اصلا به ورودی نداده ایم. البته مطمئنا وجود دارند زیرا تعداد نورونهای ورودی 10 است ولی باز هم تکرار میکنم آرایه بعدی تنها اندیس نورون هایی است که اسپایک میزنند.

حالا با توجه پترن اولیه که در صورت سوال هم وجود داشت ، نتایج را بررسی میکنیم. البته ذکر این نکته ضروری است که اسپایک ها به صورت لحظه ای است و فرکانسی تولید می کند که منجر به تغییر زیاد پتانسیل شود. از این رو شاید نمودارهای ما خیلی واقعی نباشد. اول نمودار وزن ها که به صورت رندم ساخته شده است و ده تای اول آنرا به نورون اول و ده تای بعدی را به نورون دوم نسبت داده ایم را مشاهده میکنیم.



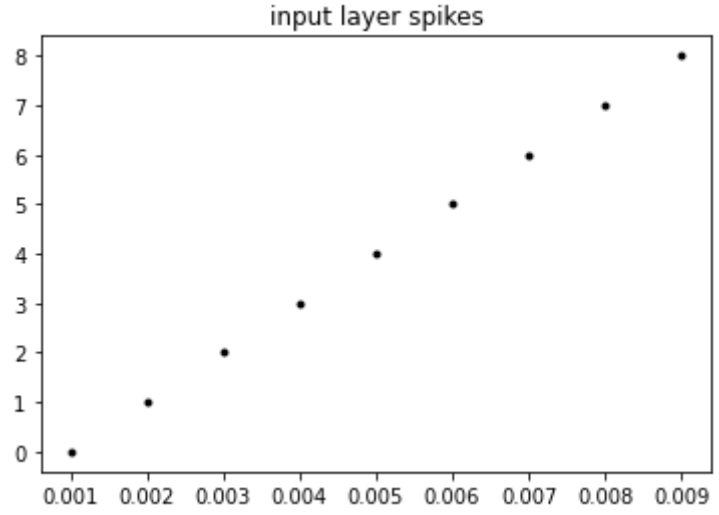


وزنها تغییر آنچنانی نکرده اند یعنی فرآیند یادگیری به خوبی انجام نشده است. دلیل واضح تر آن را میتوانیم در شکل اسپایک زدن نورونهای خروجی مشاهده کنیم:

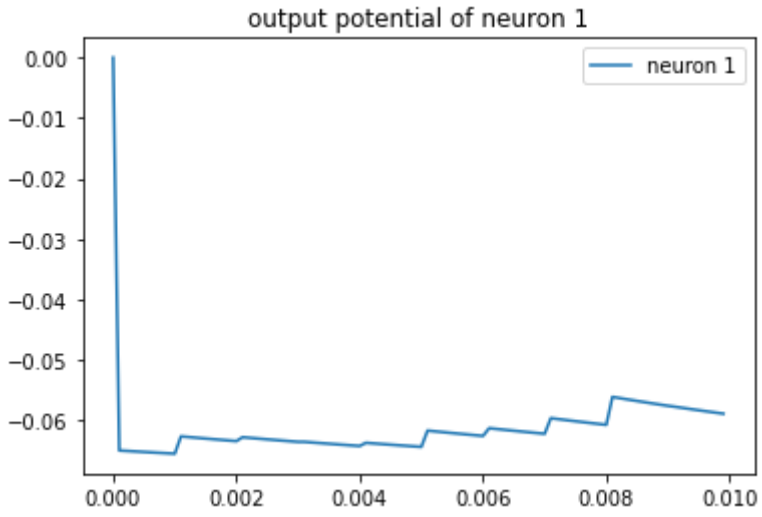


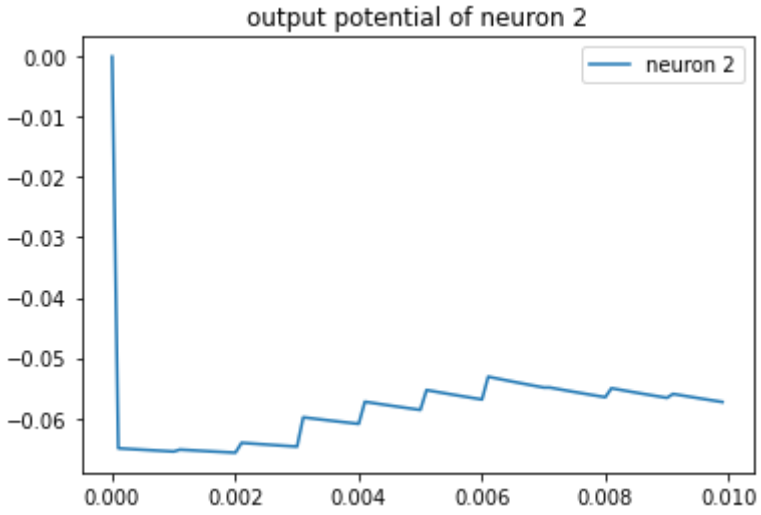
مشاهده میکنیم که اسپایک ها به صورت لحظه ای زده است و اصلا ادامه دار یا پشت سر هم نبوده که من باز هم علتش را ندادن فرکانس و تکرار آن آرایه ورودی که دادیم میدانم که در این سوال از آن استفاده نکردم.

اما برای پترن یا الگوی بعدی من اینجوری در نظر گرفتم که تمام نورونها به ترتیب با فاصله یک میلی ثانیه از هم اسپایک بزنند ، به این شکل:



خروجی که از آن دریافت کردیم اما به نظر بهتر می آید. تفاوت یادگیری در نورونهای خروجی اول و دوم محسوس است اما نسبت به الگوی قبلی قابل بررسی است که پتانسیل این دو را در شکل زیر میبینیم.





مشکل اینجاست که در هیچ کدام از دو حالت باعث نشد ما از ترشهولد عبور کنیم و این یک ms فاضله باعث decay کردن پتانسیل شد. حتی در بالا میبینیم که نزدیک به ترشهولد شدیم اما از آن عبور نکردیم که دو دلیل عمده اش از نظر خودم را در این گزارش مطرح کردم.

3. در شبکه های عصبی اسپایکی(SNN) از قاعده یادگیری STDP برای بدست آوردن ویژگی های بینایی استفاده می شود. این قاعده یک قاعده یادگیری بدون ناظر است که در حالت های مختلف از پیچیدگی میتواند به خوبی عمل کند. برای ساخت یک مدل که مشابه عملکرد بینایی در انسان عمل کند، از مدل DCNN یا همان Deep Convolutional Neural Nets استفاده کرده اند. درست است که ساختار این مدل بسیار نزدیک به عملکرد بینایی انسان است اما پردازش نورونی و مکانیزم یادگیری در قشر مغز را به خوبی شبیه سازی نمیکند . تفاوت اصلی آنها آن است که واحد های محاسباتی در DCNN ها در به توجه به فعالیت هر مرحله، به یکدیگر اعداد اعشاری را ارسال میکنند در حالی که نورون های بیولوژیکی با استفاده از فرستادن اسپایک ها به یکدیگر فعالیت میکنند.

دیگر تفاوت DCNN ها آن است که این شبکه ها برای آموزش دیدن، از الگوریتم back-propagation استفاده میکند که supervise است و در نورون های بیولوژیکی اصلا بدین صورت یادگیری صورت نمی گیرد بلکه unsupervised است. البته DCNN دقت خوبی دارد اما بسیار کند است زیرا تعداد پارامتر های آن زیاد است و عمل یادگیری بسیار کند در آن رخ میدهد برای جلوگیری از این مشکل از یادگیری بدون نظارت استفاده میکنیم. یادگیری STDP نیز دقیق است و عملکرد مغز و بینایی انسان ها را نیز به خوبی مدل میکند.

به همین دلیل امروزه از شبکه های عصبی اسپایکی بیشتر استفاده میشود، در مسائل تشخیص اشیا از SNN ها استفاده میشود. برای آنکه شبکه های عمیق تری داشته باشیم، تعدادی از این SNN ها را در کنار هم قرار میدهیم یعنی مانند DCNN ها فقط به جای شبکه های کانولوشن از شبکه های اسپایکی استفاده میکنیم. اگر چه این شبکه ها نیز باز هم مشکل دارند، زیرا برای پردازش عکس ها نیاز است که اسپایک های زیادی زده شود و همچنین زمان پردازش آن طولانی خواهد بود. علاوه بر این ، استفاده از الگوریتم یادگیری back-propagation و داشتن هر دو نوع سیناپس مثبت (تحریکی) و سیناپس های خروجی منفی (مهاری) در یک نورون از نظر زیست شناسی قابل قبول نیست.

حال در این مقاله به بررسی شبکه های عصبی اسپایکی عمیق(SDNN) و عملکرد آنها بر روی تصاویر پرداخته شده. ساختار این شبکه شامل یک لایه temporal-coding هست

و پس از آن شامل چندین لایه convolutional پشت سر هم میباشد که ویژگی ها را استخراج میکنند، همچنین لایه های pooling نیز وجود دارد.

لایه اول که temporal-coding انجام میدهد، نقش مهمی در شبکه های SNN دارد. این روش سیگنال ورودی را میگیرد آن را به چندین اسپایک ، بر حسب زمان، کدگذاری میکند. سپس بر روی آنها فیلتر DoG را اعمال میکند. این فیلتر شبیه سازی عمل ganglion cell ها در شبکیه چشم را انجام میدهد و به نوعی تضاد رنگی موجود در عکس ها را تشخیص میدهد پس از تشخیص این تضاد ها، spike میزند میتوان گفت که ترتیب اسپایک زدن ها به ترتیب تضاد های موجود در عکس است.این ترتیب رتبه‌ای برای مشخص کردن ناحیه V1(برای تشخیص لبه ها) نیز موثر است. با استفاده از DoG هم تضاد های منفی و هم مثبت را در تصاویر تشخیص میدهیم.

لایه ی بعدی layer convolution است . هر لایه convolution شامل تعدادی نورون است هر نورون با توجه به وزن سیناپسی ورودی، به یک ویژگی خاصی مربوط می شود. نورون هایی که در یک map خاص هستند، ویژگی های بصری یکسانی را تشخیص میدهند اما در مکان های متفاوت. همینطور وزن سیناپسی نورون هایی که در یک map قرار میگیرند باید یکسان باشد. هر نورون اسپایک های نورون های لایه قبل را به عنوان ورودی دریافت میکند بنابراین یک ویژگی بصری در یک لایه ترکیبی از ویژگی هایی است که در لایه های قبلی استخراج شده‌اند. نورون های موجود در هر لایه convolution از نوع non leaky integrate and fire هستند که اسپایک های ورودی را از نورون های presynaptic میگیرد و وقتی که پتانسیل آن به threshold برسد fire میکند. در هر Step زمانی پتانسیل نورون iام با استفاده از روابط زیر آپدیت میشود:

$$V_i(t) = V_i(t-1) + \sum_j W_{j,i} S_j(t-1),$$

$$V_i(t) = 0 \ \ and \ \ S_i(t) = 1, \quad if \ \ V_i(t) \geq V_{thr}.$$

در این رابطه Vi(t) نشان دهنده پتانسیل نورون کانولوشنی در زمان t است، Wi,j وزن سیناپسی بین نورون j (نورون presynaptic) و نورون i (نورون کانولوشنی) است، Sj

نشان دهنده اسپایک های نورون j است. اگر Sj(t-1) برابر با 1 باشد یعنی نورون j در زمان t-1 اسپایک زده است در غیر این صورت برابر با 0 است.

اگر Vi از میزان threshold یا Vthr فراتر شود، آنگاه نورون اسپایک میزند و Vi طبق رابطه دوم، reset میشود.

در لایه های convolution یک مکانیزم مهاری وجود دارد، این مکانیزم باعث میشود که وقتی یک نورون در یک ناحیه spike میزند، باقی نورون ها مهار میشوند و پتانسیل آنها به 0 بازمیگردد و دیگر اجازه ندارند fire کنند تا تصویر بعدی به آنها نشان داده شود. همچنین نورون ها اجازه ندارند که بیش از یک بار fire کنند. همین روند یک کدگذاری مفید است زیرا در هر مکان حداکثر یک spike داریم که نشان میدهد آیا یک ویژگی بصری خاص در آن ناحیه وجود دارد یا نه.

لایه بعدی لایه pooling است. لایه های pooling به شبکه کمک میکند تا تغییر ناپذیر شود در واقع با استفاده از روش max pooling بر روی نورون های آن لایه، ویژگی های برجسته را استخراج میکنند. برخی شواهد گفته شده که همچین عمل ماکزیمم گیری در complex cell ها در قشر بینایی انجام میشود. نورون های pooling از نوع integrate and fire هستند که ورودی آنها وزن سیناپسی است و threshold آنها 1 است بنابراین وقتی اولین اسپایک زده شود، باعث اسپایک خروجی خواهد شد زیرا تمامی آنها فعال میشوند. در لایه های pooling هیچ یادگیری رخ نمی دهد همچنین این لایه ها اطلاعات بینایی را فشرده میکنند

یادگیری چگونه انجام میشود؟

یادگیری در این مدل تنها در لایه های convolution رخ میدهد و توسط قاعده STDP انجام میشود. روش یادگیری به این صورت انجام میشود که با ترکیب ویژگی های ساده از لایه های قبل، لایه کانولوشنی سعی میکند که ویژگی های بصری را یاد گیرد. وقتی یادگیری در لایه قبلی به اتمام رسید، یادگیری در لایه بعدی آغاز خواهد شد.

وقتی مدل یک تصویر جدید را میبیند، لایه های convolution با یکدیگر رقابت میکنند و آن نورون هایی که سریعتر fire کنند، STDP بر روی آنها انجام میشود و یادگیری با آنها صورت میگیرد.

اگر بخواهیم تغییرات وزن را با STDP مشاهده کنیم، در ساده ترین حالت داریم:

$$\Delta w_{ij} = \begin{cases} a^{+} w_{ij}(1 - w_{ij}), & if \;\; t_j - t_i \leq 0, \\ a^{-} w_{ij}(1 - w_{ij}), & if \;\; t_j - t_i > 0, \end{cases}$$

i، نشان دهنده نورون postsynaptic و j، نشان دهنده نورون presynaptic است . , Ti Tj نیز زمان هایی است که این نورون ها spike میزنند. دلتا wij نیز تغییرات وزن سیناپسی را نشان میدهد. a- و a+ نیز نشان دهنده learning rate میباشند. اینکه کدام نورون قبل از دیگری spike زده باشد، در تغییرات وزن سیناپسی تاثیر گذار است. مقادیر آنها نیز بر روی قاعده learning تاثیر دارد.

اگر مقادیر آنها بزرگ باشد آنگاه حافظه یادگیری را کاهش میدهد به این معنی که فقط عکس آخر را یاد میگیرند و تصویر های قبلی را از یاد میبرند. از طرفی اگر این مقادیر کوچک باشند، سرعت فرآیند learning کاهش میابد.

اگر a- از a+ بزرگتر باشد، وزن سیناپسی به آرامی decay میکند و بدین ترتیب پس از مدتی نورون ها دیگر به میزان threshold نمیرند و fire نخواهند کرد.

بهتر است که a+ از a- بزرگتر باشد، اما اگر خیلی زیاد بزرگ باشد، نورون ها بیش از یک الگو یاد میگیرند. این حالت آخر برای انتخاب learning rate بهتر است اما مقدار آنها را نباید خیلی بزرگ یا خیلی کوچک در نظر بگیریم.

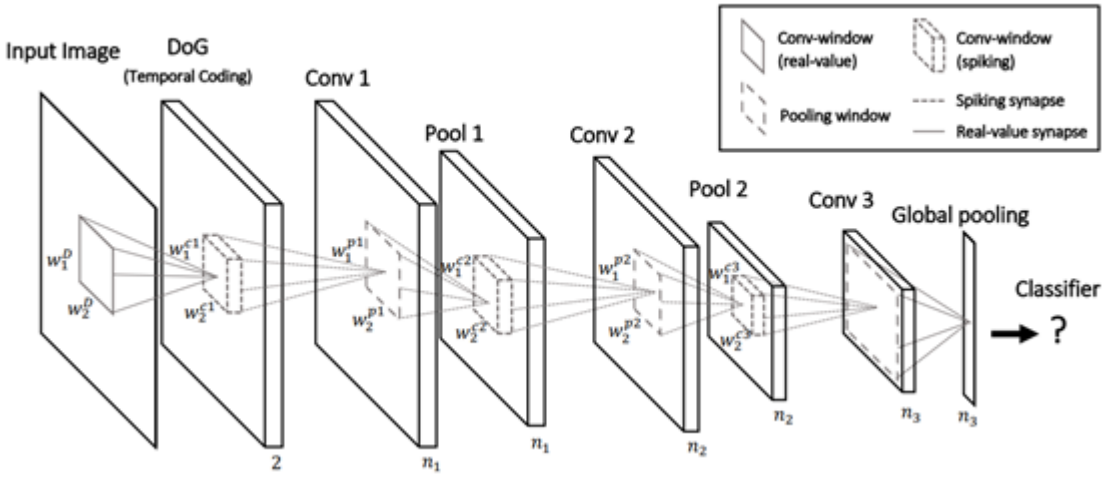
در حین فرآیند یادگیری لایه های کانولوشنی، نورون های موجود در یک ناحیه، ویژگی های یکسانی را در مکان های مختلف شناسایی میکنند. وقتی فرآیند یادگیری در یک لایه مشخص، پیشرفت میکند، نورون های آن به سمت ویژگی هایی که به طور مکرر در تصویر ورودی وجود داشت، همگرا می شوند. برای بررسی همگرایی میزان یادگیری در لایه l م از رابطه زیر استفاده میشود:

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

در این رابطه w f,i، نشان دهنده وزن سیناپسی iام ویژگی f است و n_w نشان دهنده مجموع تمامی وزن های سیناپسی برای فیچر های مستقل از هم است. اگر تمامی وزن های سیناپسی به سمت 1 یا 0 متمایل شوند، Cl نیز به سمت صفر همگرا خواهد شد و اگر این اتفاق بیفتد learning را در لایه lام متوقف خواهیم کرد.

در نهایت یک pooling سراسری بر روی هر map نورونی انجام خواهد شد تا در خروجی یک فیچر را که نشان دهنده وجود آن ویژگی در تصویر هست را به ما بدهد. سپس وقتی ویژگی ها دریافت شد، برای جداسازی آنها به یک classifier در انتها احتیاج داریم که آنها را طبقه بندی کند و نشان دهد که متعلق به کدام عکس هستند.

شکل کلی فرآیند های توضیح داده شده:



نتایج این مدل بر روی 3 دیتاست مختلف پیاده سازی شده است:

الف. Face/motorbike data set

در این دیتاست ، دو دسته تصویر چهره و موتور به شبکه داده شده و خواسته شده که با استفاده از لایه های مختلف به تشخیص این تصاویر و جداسازی آنها بپردازد. در لایه اول با استفاده از فیلتر DoG برخی از تضاد ها از عکس ها استخراج میشود، در لایه اول convolution سعی بر آن است که تصویر ها دقیق تر تشخیص داده شوند. که در این لایه، لبه های تصاویر و برخی خطوط تشخیص داده میشوند. در این لایه ها از قاعده

یادگیری STDP استفاده کردیم، بعد از هر لایه convolution یک لایه pooling میزنیم تا نورون هایی که اول اسپایک میزنند به لایه بعدی منتقل شوند، برای لایه های بعدی convolution نیز با استفاده از این قاعده ها استفاده می کنیم و ویژگی های پیچیده تر تصاویر را استخراج کنیم، سپس در لایه آخر و با استفاده از global pooling به طول دقیق تری اشیا مشخص میشوند و میتوان عکس صورت ها را از موتور ها تفکیک کرد. در تست کردن این مدل بر روی این دیتا ست به دقت 93% رسیده‌اند.

ب. ETH - 80 dataset

این دیتاست، شامل 8 دسته مختلف است : سیب ، ماشین ، گاو اسباب بازی ، فنجان ، سگ اسباب بازی ، اسب اسباب بازی ، گلابی و گوجه فرنگی . هر شی از 41 نقطه دید متفاوت ، با زاویه دید متفاوت عکس برداری شده است. این دیتاست می خواهد نشان دهد که شبکه SDNN بر روی دسته بندی 8 کلاسه نیز به خوبی عمل میکند. همانند دیتاست قبلی، شبکه را بر روی این دیتاست نیز پیاده سازی میکنیم و در نهایت از طبقه بند SVM برای جداسازی اشیای تشخیص داده شده، استفاده کرده ایم.

طبق بررسی های انجام شده، نشان داده شده که شبکه SDNN از DCNN عملکرد بهتری دارد و دقت بهتری را به ما میدهد. خطاهای مربوطه بیشتر بین تصاویری بوده که تشابه زیادی بهم داشتند و با هم اشتباه گرفته شدند.

ج. MNIST dataset

بر روی دیتاست معروف MNIST نیز این عملیات را انجام دادیم و دیدیم که نتیجه بهتر از شبکه های عصبی کانولوشنی است. بر روی این دیتاست دقت مدل به 98.4% رسیده است که دقت بسیار خوبی است.

در نهایت می توان گفت که عملکرد این مدل نسبت به مدل هایی که بدون spike هستند خیلی بهتر است زیرا این مدل ها از عملکرد نورونی در مغز پیروی میکنند و بسیار مشابه به سیستم بینایی ما هستند، پس بسیار دقت بهتری نسبت به DCNN ها دارند.

4. در مدل R-STDP از رویکرد یادگیری تقویتی استفاده شده است. در واقع از یادگیری تقویتی برای یادگیری شبکه های اسپایکی استفاده میکنیم تا دیگر به classifier در مسائل Object detection احتیاج نداشته باشیم. نورون ها در مغز ما توسط سیناپس هایی بهم متصل هستند که این اتصال میتواند ضعیف یا قوی شود. همانطور که میدانید، STDP توسط اختلاف زمانی بین نورون های pre و post عمل میکند و وزن سیناپسی را تغییر میدهد. مدل STDP برای پیدا کردن ویژگی های یکسان، عملکرد خوبی دارد اما مانند تمامی الگوریتم های unsupervised در پیدا کردن فیچرهای کمیاب و تشخیص آنها برای برخی کارهای مهم، مثلا فرایند تصمیم گیری، به مشکل برمیخورد.

به همین علت است که reward دادن و استفاده از R-STDP برای برخی از عملکرد های مغز بسیار کاربرد داد در این مدل یادگیری، شخص مورد نظر رفتار هایی که مورد reward واقع می شوند را پشت هم تکرار میکند اما رفتار هایی به منجر به punishment میشوند را سعی میکند که انجام ندهد و از آنها دوری کند.

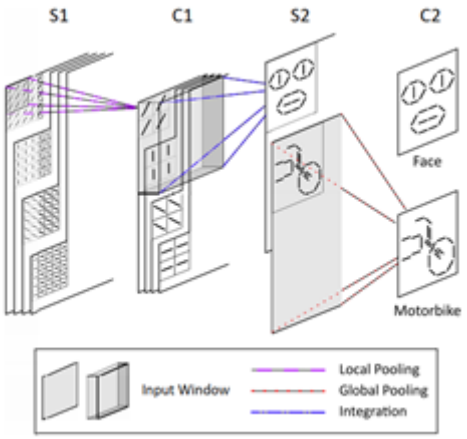
تحقیقات نشان داده است که به عنوان سیستم تشویق و پاداش در مغز، تاثیر ماده شیمیایی dopamine بسیار مهم است. تاثیر دوپامین در مغز و ترشح آن بر روی عملکرد یادگیری زیاد است. دوپامین بر روی قطبیت STDP و همچنین طول زمان انجام آن نیز تاثیرگذار است.

همین دوپامین در مغز به عنوان reward یک عمل یادگیری داده میشود و هرگاه دوپامین ترشح میشود، میزان فعالیت مغز نیز افزایش می یابد یعنی به عنوان یک reward داده شده است و برعکس کاهش میزان دوپامین باعث Punishment یا همان reward منفی خواهد بود. این روند با برعکس کردن وزن ها و پیاده کردن سیستم reward دادن بر روی STDP انجام میشود.

برای ساخت شبکه اسپایکی به جای استفاده از قاعده یادگیری STDP از قاعده R-STDP استفاده میکنیم. این قاعده همانطور که گفتیم به ما در تشخیص اشیا کمک میکند و دیگر نیاز به classifier نداریم زیرا خودش با استفاده از reward و punish دادن، می تواند دسته بندی را انجام دهد.

این شبکه دارای 4 لایه است. لایه اول تصویر ورودی را به اسپایک ها تبدیل میکند. لایه دوم عملیات Pooling بر روی اسپایک های دریافتی انجام میشود. لایه سوم شامل بخش هایی است که در آنها نورون هایی از نوع Integrate and fire وجود دارند که اطلاعات دریافتی از لایه های قبلی مانند خطوط و لبه های تصاویر را دریافت میکند، آنها را باهم ترکیب می کنند و ویژگی های پیچیده تر را استخراج میکنند. در لایه سوم یادگیری انجام میشود و برای روند یادگیری از R-STDP استفاده میکند و در لایه آخر با استفاده از سیگنال های پاداش و تنبیه ، شبکه تصمیم گیری را انجام میدهد. در لایه سوم اولین اسپایکی که زده شود، به لایه بعد منتقل خواهد شد و با استفاده از آن، تصمیم گیری انجام خواهد شد. اگر تصمیم درست باشد، reward و اگر اشتباه باشه، punish داده میشود و به همین ترتیب شبکه قوی یا ضعیف خواهد شد. همچنین برای بهتر شدن عملکرد محاسبات، هر نورون در هر سلول اجازه دارد که فقط یک بار برای هر تصویر اسپایک بزند.

لایه ها را با S1,C1,S2,C2 نام گذاری کرده اند. که یعنی دو لایه simple و دو لایه complex داریم.



لایه S1 ، لایه Simple اول، سلول های موجود در آن ، خطوط با زوایای مختلف و اندازه های متفاوت تشخیص داده میشوند. همانطور که گفته شد هدف از این لایه، تبدیل تصویر ورودی با مقیاس سیاه و سفید به اسپاک ها است. در واقع بر روی تصویر ورودی فیلتر های gabor با 4 زاویه متفاوت اعمال میشود و خطوط عکس را با 4 زاویه 0، 45،135 و 180 استخراج میشود.

لایه C1 اولین لایه complex است که بعد از لایه S1 اعمال میشود. این لایه عملکرد pooling را انجام میدهد. در این لایه اولین اسپایکی که در لایه های قبلی از تصاویر دریافت میشود، در این لایه دریافت خواهد شد. برای هر درجه چرخش، یک شبکه وجود دارد. این pooling میزان افزونگی لایه S1 را کاهش میدهد. بدین معنا که اطلاعات اضافی از لایه قبل را از بین میبرد.

لایه S2، لایه Simple دوم است. این لایه همان لایه‌ای که در آن یادگیری رخ میدهد. نورون های موجود در آن از نوع Integrate and fire هستند در این لایه ویژگی های پیچیده تر استخراج میشود. نورون های ورودی از C1 دریافت می کند و وقتی پتانسیل نورون های موجود در این لایه به threshold برسد، نورون ها اسپایک می زنند و برای اتصال سیناپسی، یادگیری را بر اساس 3 فاکتور انجام میدهد: زمان اسپایک نورون presynaptic – زمان اسپایک نورون postsynaptic – سیگنال , reward punishment پس این لایه اطلاعات را از لایه های قبلی میگیرد و با یکدیگر ترکیب می کند تا ویژگی های پیچیده تری را استخراج کند

لایه C2، لایه complex دوم است که لایه تصمیم گیری است دارای n نورون است که به هرکدام از شبکه های نورونی S2 نسبت داده شده است. اولین اسپایک دریافتی را از لایه قبل میگیرد و آن را گسترش میدهد این لایه بر اساس reward و punishment تصمیم گیری را انجام میدهد.

همانطور که گفتیم یادگیری در لایه S2 توسط R-STDP انجام میشود که طبق درستی یا نادرستی به وزن ها reward نسبت میدهد.

برای reward دادن به صورت زیر عمل میکنیم:

$$\Delta W_{ij} = \begin{cases} a_r^+ \times W_{ij} \times (1 - W_{ij}) & \text{if } t_{c1}^f(j) - t_{s2}^f(i) \leq 0, \\ a_r^- \times W_{ij} \times (1 - W_{ij}) & \text{if } t_{c1}^f(j) - t_{s2}^f(i) > 0, \\ & \text{or the } j\text{th cell is silent}, \end{cases}$$

و برای Punishment به صورت زیر عمل میکنیم:

$$\Delta W_{ij} = \begin{cases} a_p^+ \times W_{ij} \times (1 - W_{ij}) & \text{if } t_{c1}^f(j) - t_{s2}^f(i) > 0, \\ & \text{or the } j\text{th cell is silent}, \\ a_p^- \times W_{ij} \times (1 - W_{ij}) & \text{if } t_{c1}^f(j) - t_{s2}^f(i) \leq 0, \end{cases}$$

این روند اگر Punishment دریافت کند، قطبیت STDP را برعکس میکند و مانند آن است که STDP را برعکس انجام میدهیم. به این منظور که LTD را به LTP تبدیل میکند و برعکس.

در نهایت باید گفت که روش RSTDP به دلیل وجود reward دادن، عملکرد بهتری نسبت به STDP در دیتاست های مختلف دارد. و بدون نیاز به classifier میتواند مدل را ارزیابی کند