پروژه سوم علوم اعصاب محاسباتی

مهرانه مقتدائی فر

تمرین اول:

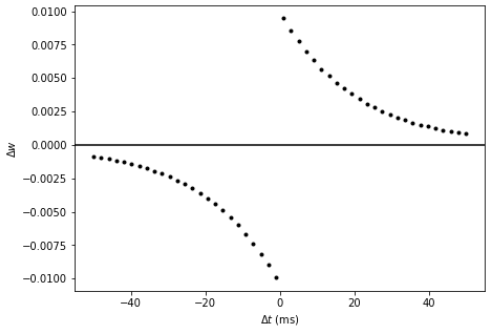
قانون یادگیری STDP وزن سیناپسی بین دو نورون post و pre را توسط معادلات زیر آپدیت میکند:

$$\Delta w_+ = A_+ . \exp(-|\Delta t| / \tau_+) \quad at\ t_{post} \quad for\ t_{pre} < t_{post}$$

$$\Delta w_- = A_- . \exp(-|\Delta t| / \tau_-) \quad at\ t_{pre} \quad for\ t_{pre} > t_{post}$$

این پیاده سازی برگفته از کتابخانه brian2 است و کد آن نیز از سایت آن آمده. توضیحاتی برای کد پیاده سازی شده میدهیم. در این پیاده سازی دقیقا از معادلات بالا استفاده کرده ایم بدین ترتیب که برای τ_{pre} و τ_{post} در ابتدا مقدار ثابت 20 ms را در نظر گرفتیم سپس میزان A را تعریف کرده ایم. ابتدا دو جمعیت نورونی، یکی برای نورون های pre و دیگری برای نورون های post ساختیم و بعد با استفاده از معادله STDP سعی کردیم که وزن بین سیناپسی را ابین دو نورون i , j برقار کنیم و کم و زیاد شدن این وزن را توسط قاعده یادگیری STDP پیاده سازی کردیم.

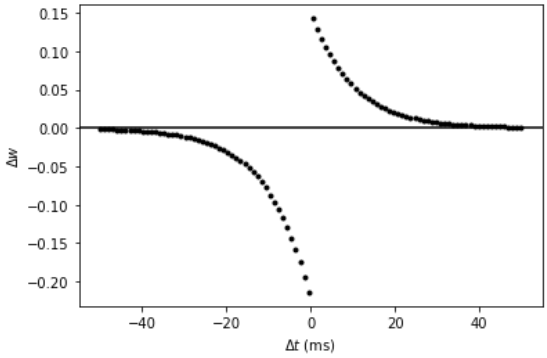
سپس نمودار تغییرات وزن را بر حسب میزان اختلاف زمان نورون های post و pre رسم کرده‌ایم.



این نمودار، نشان دهنده آن است که نورون های ما در چه زمانی اسپایک زده اند و همانطورکه میبینیم، حالت تابع نمایی دارند که در قاعده یادگیری STDP دقیقا باید به همین شکل باشد. در اینجا در هر جمعیت نورونی، ۵۰ تا نورون داشتیم.

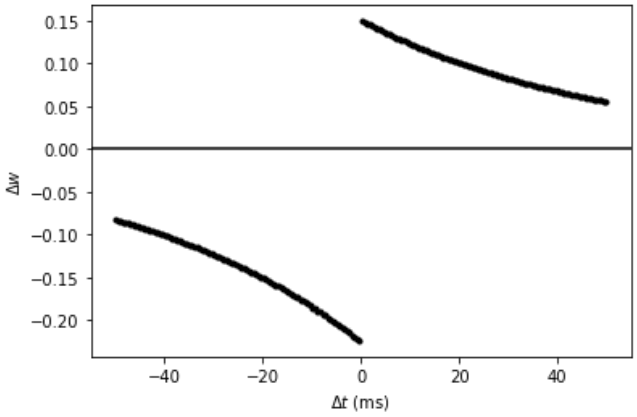
برای دیافت نتایج متفاوت، در مرحله بعد، تعداد نورون ها در هر جمعیت را به ۱۰۰ تا افزایش دادیم و همچنین میزان tau را به ۱۰ کاهش دادیم، با اینکار مخرج کسر کوچکتر میشود و بدین ترتیب نورون ها بیشتر اسپایک خواهند زد.

نمودار زیر نتایج گفته شده را به خوبی نشان میدهد:



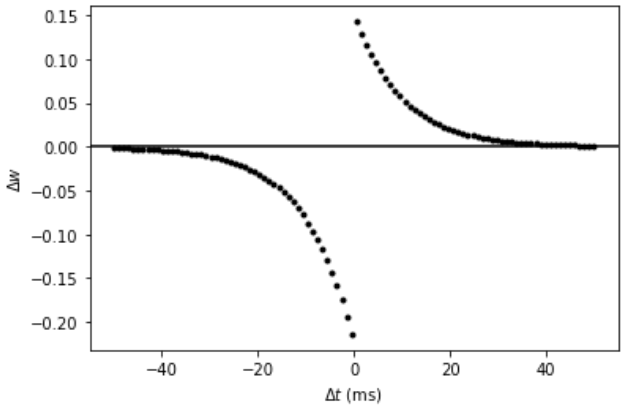
در این تصویر، همانطور که میبینید میزان اسپایک زدن و تراکم آنها بیشتر شده است.

در مرحله بعد هم تعداد نورون ها در هر جمعیت را افزایش دادیم و هم میزان tau را زیاد کردیم. نتایج کمی متفاوت تری به ما نشان داده شد:



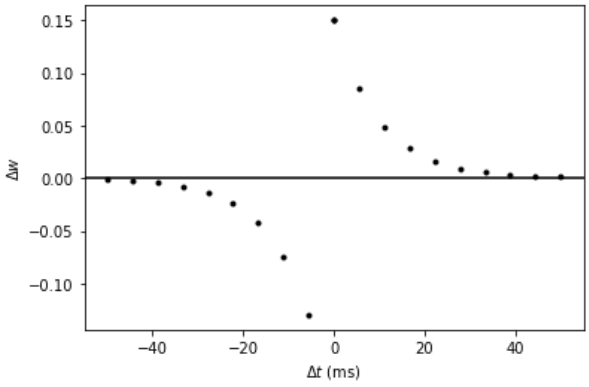
این تصویر برخلاف حالت قبلی، عملکرد STDP را به خوبی نشان نمیدهد و اصلا مشابه آن چیزی که انتظار داریم نیست.

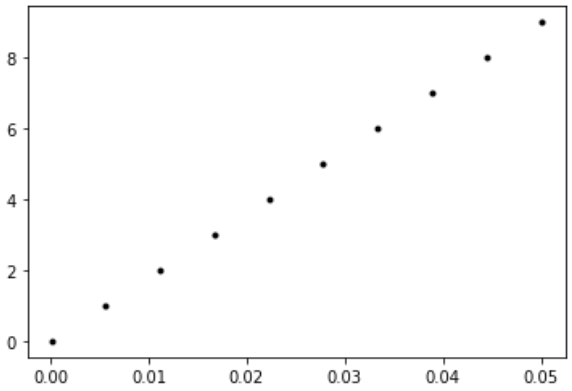
پس همانطور که گفته شد، عملکردی که ما از STDP انتظار داشتیم، در شکل دوم خیلی خوب نمایان شده است و همین بهترین حالت است و دیگر پارامتر ها را تغییر نمیدهیم.

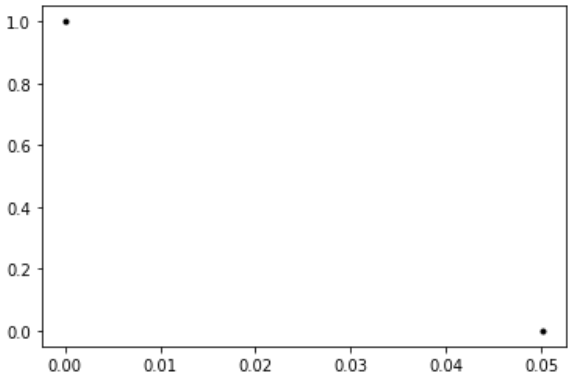


تمرین دوم:

در این سوال چیزی که از ما خواسته شده پیاده سازی یک شبکه اسپایکی با ۱۰ نورون ورودی و ۲ نورون خروجی است که بخش اول کد در همین رابطه زده شده است که متناظر با معادلاتی است که در سوال قبلی به کار بردیم و نمودار دلتا w و Δt آن در پایین آن بلاک مشخص است.بعد از آن ۲ نمودار از اسپایک های ۱۰ نورون ورودی و ۲ نورون خروجی را به ترتیب میبینیم.







اما بخش دوم صورت سوال از ما خواسته الگویی را به نورونهای ورودی بدهیم . پس از تعریف معادله ، لایه های از نورونهای گفته شده با ۲ نورون خروجی و ۱۰ نورون ورودی ساختم که fully connected هستند و پس از آن با دادن جریان I به ورودی که مشابه جریان گفته شده در سوال است ، سعی کردم مدل را با آن الگو پیاده سازی کنم .
اما متاسفانه از یک جایی به بعد به خطا خوردم و به درستی نتوانستم نمودار اسپایکهای دو نورون خروجی و روند یادگیری در آنها را نمایش دهم.

تمرین سوم:

در این مقاله به بررسی عملکرد شبکه های عصبی اسپایکی بر روی تصاویر و ساخت مدل deep از این شبکه ها پرداخته شده است.

ازشبکه های عصبی کانولوشنی عمیق (DCNN) برای بحث تشخیص اشیا(Object detection) از قبل استفاده شده است. این شبکه ها با استفاده از انتقال اعداد اعشاری از یک لایه به لایه دیگر، فعالیت های هر لایه را بررسی میکنند. این شبکه ها تا حدودی شبیه عملکرد بینایی انسان هستند اما به طور دقیق ویژگی های بیولوژیکی را بررسی نمیکنند و عملکرد دقیق را ندارند. به همین دلیل از شبکه های عصبی اسپایکی میخواهیم استفاده کنیم.

شبکه های عمیق اسپایکی همانند شبکه های DCNN هستند اما با این تفاوت که به جای شبکه های کانولوشینی از شبکه های اسپایکی استفاده میکنیم. بدین ترتیب که چندین لایه از شبکه های SNN را در کنار هم قرار میددهیم. در این شبکه ها ابتدا توسط لایه‌ای که از فیلتر DoG استفاده میکند تصاویر را عبور میدهیم سپس به چندین لایه convolutional که با استفاده از اسپایک زدن نورون ها اطلاعات را منتقل میکند میدهیم و از لایه های pooling نیز پس از هر لایه convolution استفاده میکنیم.

توضیحات این لایه هارا باهم بررسی میکنیم:

لایه اول که به آن temporal coding نیز میگویند از فیلتر های DoG استفاده میکند تا این فیلتر ها تضاد های رنگی را در تصاویر تشخیص میدهد. در واقع به این دلیل به آن temporal coding میگویند که وقتی سیگنال های ورودی را دریافت میکند، آن ها را به قطاری از spike ها، کد میکند. تضاد های موجود در تصاویر توسط فیلتر DoG و با استفاده از این اسپایک ها در این لایه بعدی نمایان میشوند. این فیلتر دو حالت on-center و off-center دارد که به ترتیب برای تضاد های منفی و مثبت است.

لایه دوم convolution هست که در این لایه یادگیری رخ میدهد. نحوه یادگیری در این لایه Unsupervised است و با استفاده ازقاعده STDP انجام میشود. هر لایه از convolution ها شامل چندین neural map است، که در هر کدام از آنها چندین نورون وجود دارد و نحوه فعال شدن این نورون ها و اسپایک زدن آنها، یادگیری را انجام میدهد.

فعالیت نورون های موجود در هر لایه با استفاده از عبارت زیر تعیین میشود:

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

در این عبارت، $V_i(t)$ نشان دهنده پتانسیل نورون کانولوشنی در زمان t است

$W_{j,i}$ وزن سیناپسی است که بین نورون j (presynaptic) و نورون i (نورون کانولوشنی postsynaptiv) است، S_j نشان دهنده اسپایک های نورون j است. اگر $S_j(t-1)$ برابر با ۱ باشد یعنی نورون j در زمان t-1 اسپایک زده است ولی اگر ۰ باشد یعنی اسپایک نزده است.

نوع نورون های موجود در هر neural map، nonleaky integrate and fire است، این نورون ها اسپایک های ورودی را از لایه های قبل میگیرند و ویژگی هایی که از لایه قبل بدست آمده، در این لایه باهم ترکیب میشوند و ویژگی های بهتری از تصویر خارج خواهد شد.

اگر این نورون ها به پتانسیل threshold برسند، در آنصورت fire میکنند. وقتی یک نورون در هر neural map اسپایک بزند، بقیه نورون ها دیگر به threshold نخواهند رسید و مهار خواهند شد. بدین ترتیب در هر neural map حداکثر یک نورون مشخص میکند که آی در این تصویر، به ویژگی مورد نظر رسیدیم یا خیر

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

این عبارت پس از آن است که نورون مورد نظر، اسپایک زده باشد و به threshold رسیده باشد. که در این حالت V آن reset خواهد شد و برابر با 0 خواهد بود و S نیز برابر با ۱ است زیرا نورون fire کرده است.

لایه بعدی که پس از هر convolution قرار میگیرد، لایه pooling است. این لایه با کدگذاری rank order و با استفاده از max pooling از هر لایه convolution آن نورونی را که سریعتر اسپایک زده انتخاب میکند و به نوعی ویژگی برجسته تر را از تصویر استخراج میکند. نورون ها در این لایه از نوع Integrate and fire هستند و threshold آنها بر روی ۱ تنظیم شده است.بدین معنا که اگر نورونی در همان ابتدا اسپایک بزند، این اسپایک به خروجی منتقل خواهد شد. در این لایه ها یک پنجره هایی وجود دارد که از اطلاعات اضافی که از دو ورودی قبلی می آید ، جلوگیری کند و مانع از ورود آنها شود و همچنین عمل فشرده سازی رخ دهد.

نحوه یادگیری همانطور که گفتیم در لایه convolution و با استفاده از قاعده STDP انحام میشود. در ساده ترین حالت برای STDP و تغییر وزن سیناپسی داریم:

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

پارامتر مهمی که انتخاب آن بسیار بر روی تغییرات وزن سیناپسی تاثیر میگذارد، a است. a درواقع نقش learning rate را دارد پس مقدار آن بسیار مهم و تعیین کننده است.

اگرمقادیر a^+ , a^- بزرگ باشد آنگاه حافظه یادگیری کاهش می‌یابد و مدل فقط تصویر آخر را به حافظه دارد و بدین صورت تاثیر تصویر های قبلی و ویژگی های آنها از لایه های قبل کاهش پیدا میکند. از طرفی اگر مقادیر a^+ , a^- کوچک باشد، سرعت فرآیند یادگیری کاهش میابد و این اصلا برای شبکه مناسب نیست. اگر a^- از a^+ بزرگتر باشد، وزن سیناپسی به آرامی کاهش پیدا میکند و این باعث میشود که نورون ها دیگر به threshold نرسند و این امر سبب از بین بردن عملکرد مدل میشود، زیرا دیگر هیچ نورون اسپایک نخواهد زد.

بهترین حالت آن است که a^+ از a^- بزرگتر باشد، اما میزان بزرگ بودن نیز باید تا حدی باشد، اگر از یک حدی بیشتر شود نورون ها بیش از یک الگو فرا میگیرند. پس بهتر است که مقدار آن خیلی بزرگ یا خیلی کوچک نباشد.

برای بررسی همگرایی میزان یادگیری در لایه Lم بدین صورت عمل میکنیم:

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

همانطور که یادگیری یک لایه خاص پیشرفت می کند ، نورون های آن به تدریج به ویژگی های بصری مختلفی که در تصاویر ورودی مکرر است ، همگرایی می کنند. C_l به سمت صفر میل میکند اگر وزن ها سیناپسی($w_{f,i}$) به سمت صفر یا یک میل کند و بدین ترتیب یادگیری را در آن لایه مورد نظر متوقف میکنیم.

در نهایت پس از این لایه ها، لایه آخر یک classifier قرار میگیرد که ویژگی های برجسته‌ای را که توسط pooling های نهایی استخراج میشود. به این نوع pooling، global max pooling میگویند که روی آخری لایه convolutional انحام میشود و برای هر فیچر، یک مقدار خروجی نهایی است که نشان دهنده وحود آن در تصویر می‌باشد. توسط این pooling ویژگی های مختلف به دسته بندی های مختلف تقسیم میشود و در طبقه بند های متفاوت قرار میدهد. معمولا در لایه آخر یک طبقه بند خطی مانند SVM قرار میدهیم که کار classification را اجرا کند.

به عنوان نتایج نهایی، این مدل را بر روی ۳ دیتاست آزمایش کرده‌اند. ۱. دیتاست face/bike ۲. دیتاست ETH-80 ۳. دیتاست MNIST

دو دیتاست اول تقریبا مشابه هم هستند. در اولی ۲ نوع تصویر(تصویر صورت انسان و موتورسیکلت) داریم و با استفاده از شبکه SDNN سعی داریم که این تصاویر را از یکدیگر جدا کنیم. اما در دومی ۸ دسته تصویر مختلف داریم که برخی از آنها شباهت هایی با یکدیگر دارند و گاها باهم اشتباه گرفته میشوند.

با اعمال مدل SDNN بر روی این دیتاست ها با استفاده از فیلتر های DoG و Gabor، ویژگی های اولیه و تضاد ها در تصاویر پدیدار میشوند و سپس با استفاده از لایه های کانولوشن اولیه، ویژگی های اولیه از تصاویر مانند خط ها و لبه ها، استخراج میشود و سپس در لایه های کانولوشن بعدی ویژگی های

یشرفته تر استخراج میشود و با ترکیب ویژگی ها از لایه های قبل، به تصویر اصلی نزدیک و نزدیکتر خواهیم شد. طبق نتایج دقت مدل SDNN در دیتاست اول بر روی داده های تست تقریبا حدود 93% بوده است و این درصد دقت از آزمایش شبکه های DCNN بر روی این دیتاست، بهتر بوده است. یعنی استفاده از شبکه های SNN به بهتر شدن عملکرد مدل ها و دریافت دقت بیشتر خیلی کمک میکند.

حتی برای دیتاست دوم نیز که ۸ کلاس مختلف بود، باز هم عملکرد این شبکه ها خیلی بهتر بوده است. درست است که در برخی موارد که تصاویر بسیار شبیه به هم بودند، شبکه اشتباه تشخیص داده، اما در هر صورت به کار گیری شبکه های اسپایکی از شبکه های کانولوشنی معمولی، برتری دارد.

دیتاست آخر نیز داده های MNIST معروف بوده است. حتی در این تصاویر نیز شبکه های SNN عملکرد خیلی بهتری داشتند و دقت شبکه تا 98.4% رسیده است.

شبکه های کانولوشنی به دلیل ساختاری که دارند، از spike ها استفاده نمیکنند، یعنی مکانیزم عملکرد آنها مانند قشر بینایی انسان نیست و این به گونه‌ای ضعف محسوب میشود. همچنین این شبکه ها چون یادگیری با نظارت دارند، باید میلیون ها پارامتر را تنظیم کنند و در عمل یادگیری آنها را آپدیت کنند، در صورتی که در شبکه های اسپایکی اینگونه نیست و سرعت یادگیری و دقت آن نیز به همین سبب بالاتر میرود. پس استفاده از شبکه های اسپایکی برای تشخیص اشیا، نتایج بسیار خوبی داشته است و بهتر از شبکه های CNN بوده است.

تمرین چهارم:

RSTDP یا Reward Modulated STDP درواقع یادگیری تقویتی است که از reinforcement learning استفاده میکند. تفاوت اینگونه یادگیری با STDP در reward دادن است. این reward دادن توسط تشرح ماده‌ای به نام dopamine در مغز اعمال میشود. اگر میزان دوپامین افزایش یابد باعث تقویت یادگیری STDP میشود و اگر مقدار آن از یک میزانی کمتر شود به نوعی punishment رخ میدهد. میزان ترشح دوپامین بر روی ۲ چیز در STDP تاثیر می‌گذارد. ۱. طول زمانی فعالیت را ممکن است کم یا زیاد کند(STDP window) و ۲. ممکن است قطبیت آن را به طور کل تغییر دهد. یعنی درجایی که LTD رخ میداد LTP رخ دهد و برعکس.

برای آنکه این تاثیر را بر روی مدل STDP نشان دهیم، داریم:

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

$$\frac{ds}{dt} = cd$$

$$\tau = t_{post} - t_{pre}$$

در این رابطه وزن سیناپسی را با s نشان میدهیم و c نشان دهنده آن است که سیناپس اخیرا فعال بوده است یا خیر. d هم نشان دهنده میزان دوپامین است که به عنوان یک ماده خارجی به سیناپس وارد میشود. اگر d>0 باشد یعنی دوپامین افزایش پیدا کرده است و اگر d<0 باشد به این معناست که میزان دوپامین کاهش یافته. از آنجایی که در معادله دوم d در c ضرب میشود، درواقع به قطبیت STDP مربوط میشود. اگر مثبت باشد STDP به حالت اصلی خود اجرا میشود و اگر منفی باشد، برعکس خواهد شد.

اگر نورون اخیرا اسپایک زده باشد، c زیاد میشود و به دلیل وجود – این میزان به تدریج به سمت صفر میل میکند و decay میکند. همچنین جمله STDP را هرگاه نورون pre یا post اسپایک بزنند، آپدیت خواهد شد .

همچنین خود d و میزان دوپامین نیز با معادله زیر کنترل میشود و روند decay کردن دوپامین را نشان میدهد.

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

با استفاده از RSTDP و reward دادن یا punish کردن شبکه، میتوانیم عملکرد نورون ها را کنترل کنیم. در STDP شبکه بر اساس شباهت ها یادگیری را انجام میدهد، در صورتی که در RSTDP میتواند بر اساس تفاوت ها تصمیم گیری کند.

برای مدل سازی میتوانیم، شبکه‌ای با ۴ لایه را در نظر بگیریم، که شامل ۲ لایه simple و ۲ لایه complex است. مانند شبکه SNN که از STDP برای یادگیری استفاده کرده بود، اکنون همان شبکه اسپایکی را مدل سازی میکنیم با این تفاوت که به جای STDP برای یادگیری از RSTDP استفاده کردیم.

شبکه جدید با استفاده از RSDTP یادگیری را انجام میدهد و دیگر نیازی به classifier ندارد زیرا با عمل reward یا punishment تصمیم گیری را اعمال میکند.

این شبکه شامل ۴ لایه است:

لایه اول : S1

در این لایه با دریافت عکس ورودی، با استفاده از فیلتر gabor، خطوط با زوایای متفاوت را از عکس استخراج میکنیم. این ویژگی ها به اسپایک ها تبدیل میشود و نورون های موجود در هر ناحیه اسپایک میزنند. این لایه اولین لایه simple است . در این لایه هر نورونی که سریعتر اسپایک بزند که لایه بعدی منتقل خواهد شد.

لایه دوم: C1

این لایه اولین لایه complex است که به عنوان pooling layerعمل میکند و هر اسپایکی که زودتر بیاید را در هر ناحیه دریافت میکند و به عنوان ویژگی برجسته انتخاب میکند این عملیات باعث کاهش نورون ها در لایه S1 میشود زیرا هرچه نورون اضافی که لازم نیست، در آن لایه باشد از بین میبرد و مهم ترین آنها را جدا میکند.

لایه سوم: S2

این لایه دومین لایه simple است که ویژگی هارا از لایه های قبلی گرفته و آنها را باهم ترکیب میکند تا در نهایت به جز خطوط و زوایا، بتواند اشکال پیچیده تر را انتخاب کند. در این لایه learning رخ میدهد. بدین ترتیب که با اسپایک های نورون های pre و نورون post و همچنین سیگنال های reward و punishment ، آپدیت رخ میدهد. در این لایه با استفاده از قاعده R-STDP یادگیری رخ میدهد و تصمیم گیری انجام خواهد شد.

لایه چهارم: C2

در این لایه pooling نهایی رخ میدهد. این Pooling باعث میشود که یکی از اسپایک ها در نهایت انتخاب شوند و تصمیم گیری نهایی رخ دهد. اگر این تصمیم درست باشد با استفاده از reward، تقویت میشود و نتیجه نهایی را مشخص میکند و در کلاس های مختلف قرار میدهد. اگر تصمیم نادرست باشد، punishment انجام میشود تا دیگر شبکه جلوتر نرود و متوقف شود.

بدین ترتیب مدل SNN را با استفاده از قاعده یادگیری RSTDP پیاده کردیم، این مدل با توجه به شواهد بهتر از STDP عمل میکند و این نشان میدهد که یادگیری تقویتی خیلی به عملکرد شبکه ها کمک میکند.