



گزارش پژوهه سوم علوم اعصاب محاسباتی

امیرحسین انتظاری

۱۴۰۳ اردیبهشت ۱۶

فهرست مطالب

۱	پیاده سازی کدگذاری ها	۱.۰
۱	کدگذاری به روش Time-to-first-spike	۱.۱.۰
۴	کدگذاری اعداد	۲.۱.۰
۴	کدگذاری به کمک توزیع پواسون	۳.۱.۰
۸	یادگیری بدون ناظر	۲.۰
۸	Pair-based STDP	۱.۲.۰
۹	آزمایش ها	۲.۲.۰
۱۷	اضافه کردن دو نورون غیر فعال	۳.۲.۰
۱۸	بررسی مدل با فعالیت کمینه برای لایه ها	۴.۲.۰
۲۰	قانون یادگیری تقویتی	۳.۰
۲۰	آزمایش ها	۱.۳.۰
۲۸	اضافه کردن دو نورون غیر فعال	۲.۳.۰
۳۰	بررسی مدل با فعالیت کمینه برای لایه ها	۳.۳.۰

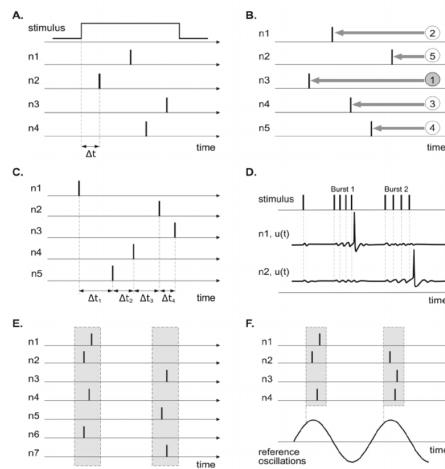
چکیده

هدف از این پژوهه، پیاده سازی کدگذاری کردن ورودی در شبکه های عصبی ضربه ای، یادگیری بدون ناظر و یادگیری تقویتی است. است. در این پژوهه از میاحشی که در پژوهه های قبلی یاد گرفته ام، (مانند مدل های نورونی، سیناپس و...)، استفاده می کنیم تا یک شبکه عصبی ضربه ای دو لایه را تشکیل دهیم. ابتدا روش های مختلف کدگذاری، از جمله کدگذاری Time-to-first-spike، کدگذاری اعداد به کمک توزیع نرمال و کدگذاری به روش پواسون را بررسی می کنیم. بعد از کد کردن محرك ها، نوبت به ورودی دادن آن ها به شبکه می شود. در ادامه، شبکه ای مشکل از دو لایه را پیاده سازی می کنیم و قانون های یادگیری مختلف، از جمله قانون یادگیری انعطاف پذیری وابسته به زمان ضربه (*STDP*) و مدل تقویتی آن یعنی *RSTDP* را روی مدل اعمال می کنیم و پارامتر های آن را در شرایط مختلف آزمایش و تحلیل می کنیم.

۱۰۰ پیاده سازی کدگذاری ها

در این بخش می خواهیم نحوه کد کردن اطلاعات (مانند یک تصویر) در شبکه های عصبی ضربه ای را پیاده سازی و تحلیل کنیم. در حوزه علوم اعصاب محاسباتی، روش های کدگذاری برای ترجمه محرك های دنیای واقعی به سیگنال های عصبی که می توانند توسط مدل های محاسباتی پردازش شوند، مهم هستند. برای اینکار، روش های مختلفی وجود دارد، مانند روش time-to-first-spike^۱، کدگذاری ترتیبی^۲، کدگذاری براساس تاخیر^۳، کدگذاری براساس همزمانی ضربه ها^۴، کدگذاری اعداد، کدگذاری به روش پواسون^۵ و روش های دیگر بسیار که کاربرد های مختلفی دارند. (شکل ۱) در این پژوهش، ما تمکرمان را روی سه روش کدگذاری زیر میگذاریم و آن ها پیاده سازی و تحلیل میکنیم:

- روش کدگذاری time-to-first-spike
- روش کدگذاری مقادیر عددی
- روش کدگذاری به کمک توزیع پواسون



شکل ۱: روش های مختلف کدگذاری. (A) زمان برای اولین ضربه؛ (B) کدگذاری رتبه یا کدگذاری مرتبه ضربه. (C) کدگذاری تأخیر بر اساس زمان دقیق افزایش ها. (D) کدگذاری انفجاری روزنامنی. (E) کدگذاری توسط همزمانی. (F) کدگذاری فازی

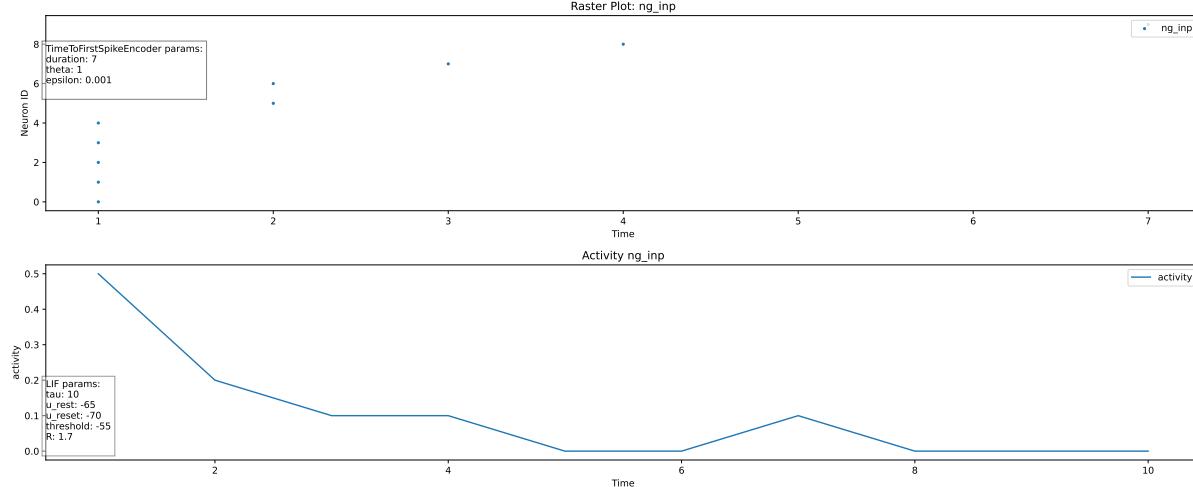
۱۱۰ کدگذاری به روش Time-to-first-spike

کدگذاری Time-to-First-Spike (TFS) روشی است که در آن از زمان اولین اسپایک پس از شروع محرك برای انتقال شدت محرك استفاده می شود. هر چه زمان تا اولین ضربه کوتاه تر باشد، شدت محرك درک شده بیشتر می شود. این روش کدگذاری برای پردازش سریع اطلاعات بسیار مفید است.

برای کدگذاری Time-to-First-Spike (TFS) باید شدت یک محرك را به زمان بندی اولین ضربه عصبی تبدیل کنیم. برای اینکار، ابتدا نیاز است با نرم افزاری داده های ورود حاصل شود که همه مقادیر بین یک آستانه حداقل و حداکثر نگاشت می شوند و کمی نیز تنظیم می شوند تا از مقادیر شدید جلوگیری شود. فرآیند کدگذاری از یک آستانه استفاده می کند که به مرور توسط یکتابع نمایی کاهشی بر اساس مدت زمان کدگذاری و یک پارامتر θ کاهش می یابد. این کار به این معنی است که محرك های شدیدتر منجر به ضربه های زودتر می شود. این رمزگذار زمان اولین ضربه را برای هر محرك ثبت می کند و مقدار محرك پس از ضربه را برای جلوگیری از کدگذاری مجدد تنظیم می کند. نتیجه یک ماتریس باینری است که زمان های افزایش نسبت به شروع محرك را نشان می دهد و به طور موثر اولین پاسخ عصبی قابل توجه را به شدت های مختلف محرك ثبت می کند. به عنوان مثال، یک الگوی ورودی مانند آرایه $[100, 70, 30, 10, 0]$ را با $\theta = 1$ کدگذاری کنیم، نمودار raster و نمودارفعالیت به صورت شکل ۲ خواهد شد.

Rank-order encoding^۱
Latency encoding^۲
Coding by synchrony^۳
Poisson^۴

Time to first spike encoder

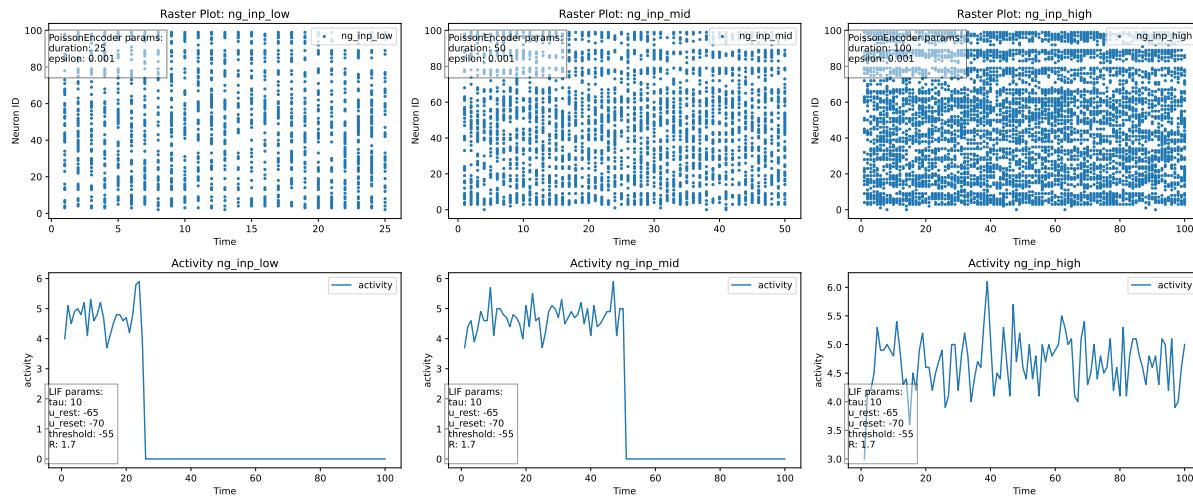


شکل ۲: کدگذاری به روش *TTFS*. همانطور که در شکل ملاحظه می شود، با انتخاب کردن $\theta = 1$ مقادیر بالای 10^0 در یک لحظه ضربه زده و مقادیر کمتر در لحظه های بعدی ضربه می زند. همچنین هر چه مقادیر کمتر می شوند، فاصله بین ضربه ها نیز کمتر می شود.

تغییر در θ

دلیل آنکه مقادیر بالا در یک زمان ضربه زده اند این است که مقدار θ نسبتاً زیاد انتخاب شده است. در شکل ۳ ملاحظه میکنیم با انتخاب $\theta = 0.01$ فواصل بین ضربه ها با محک شدیدتر نیز بیشتر می شود. به طور کلی میتوان گفت که کاهش θ میتواند باعث تفکیک بیشتر بین مقادیر ورودی شود.^۳

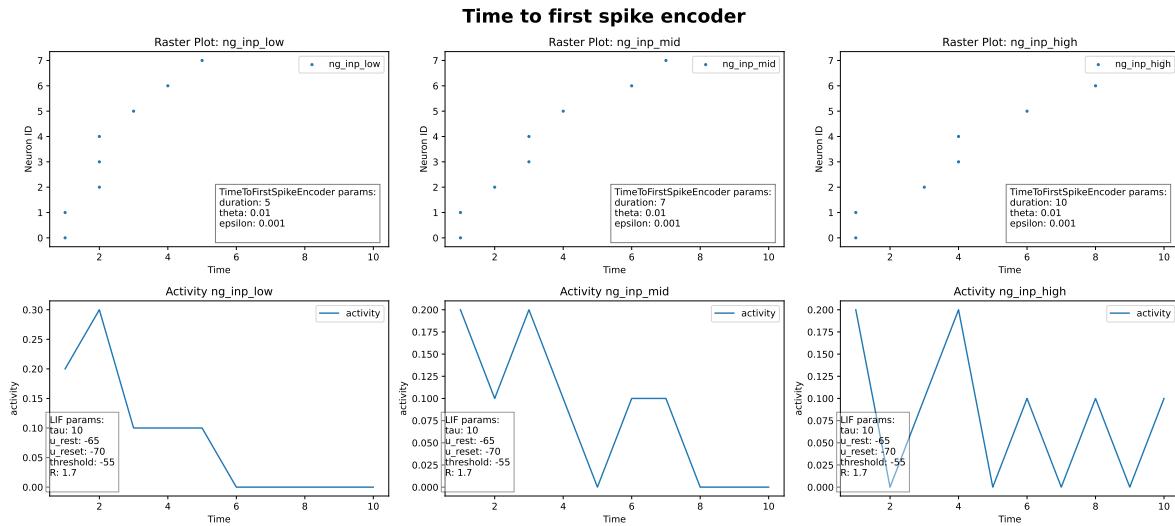
Time to first spike encoder



شکل ۳: کدگذاری به روش *TTFS*. همانطور که در شکل ملاحظه می شود، کاهش پارامتر θ میتواند باعث بیشتر شدن حساسیت جمعیت به مقادیر مختلف ورودی شود.

تغییر در مدت زمان ورودی

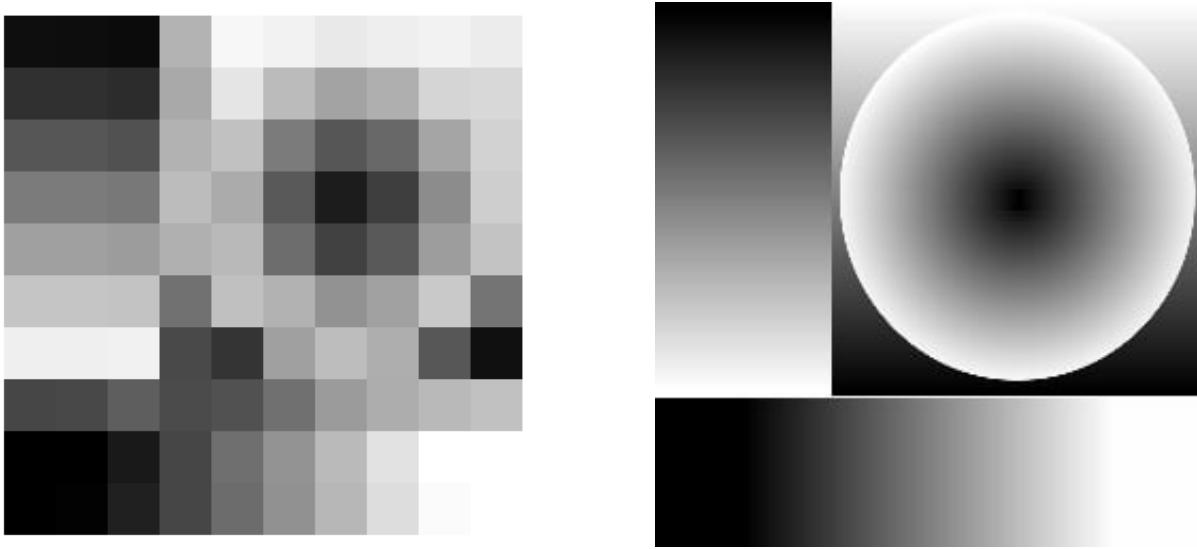
پارامتر دیگری که در خروجی کدگذاری تاثیر دارد، مدت زمانی است که ورودی به شبکه داده می شود. این پارامتر تعیین میکند که چقدر طول میکشد تا ورود به نورون ها داده می شود. از این رو، کاهش این پارامتر باعث می شود که الگوی ضربه زدن کدگذاری شده در زمان کمتری به شبکه داده شود و از این رو فواصل بین ضربه ها می تواند کاهش یابد یا حتی در یک زمان ضربه زده شود. از آنجا که در روش کدگذاری *TTFS* فاصله تا اولین نورونی که ضربه می زند مهم است، انتخاب نادرست این پارامتر میتواند باعث شود تا ورودی به خوبی کدگذاری نشود. افزایش این مدت زمان نیز باعث می شود تا فواصل بین ضربه ها بیشتر شود. (شکل ۴)



شکل ۴: کدگذاری به روش *TTFS*. همانطور که ملاحظه می شود، از راست به چپ، با کاهش مدت زمان ورودی، فاصله بین ضربه ها کمتر شده و زمان ضربه زدن نورون ها فشرده تر می شود. مثلاً نورون شماره ۳ و ۴ در یک زمان ضربه می زنند.

کدگذاری تصویر

حال یک قدم فراتر گذاشت، و سعی میکنیم یک تصویر را با روش *TTFS* کدگذاری کنیم. برای انتخاب تصویر، از مخزن دانشگاه واترلو^۵ استفاده میکنیم. این تصاویر همگی سیاه و سفید هستند. از این رو میتوانیم کل تصویر را به صورت یک آرایه درنظر بگیریم، به طوری که سطر های تصویر را پشت سر هم ردیف میکنیم. یک تصویر مانند شکل ۵ در نظر میگیریم. حال یک آرایه به طول $m \times n$ داریم که مقادیر آن بین 0 و 255 هستند. از آنجاکه ابعاد تصاویر مخزن، حدود 500×500 هستند، تعداد نورون های مورد نیاز برای کد کردن این ورودی سیار زیاد خواهد بود و از هدف این پروژه خارج می شود. از این رو ابتدا ابعاد تصاویر را دستکاری کرده و به 10×10 تغییر می دهیم. اکنون کافی است مانند بخش قبل ، این آرایه را به عنوان ورودی کدگذاری کرده و به شبکه بدheim.

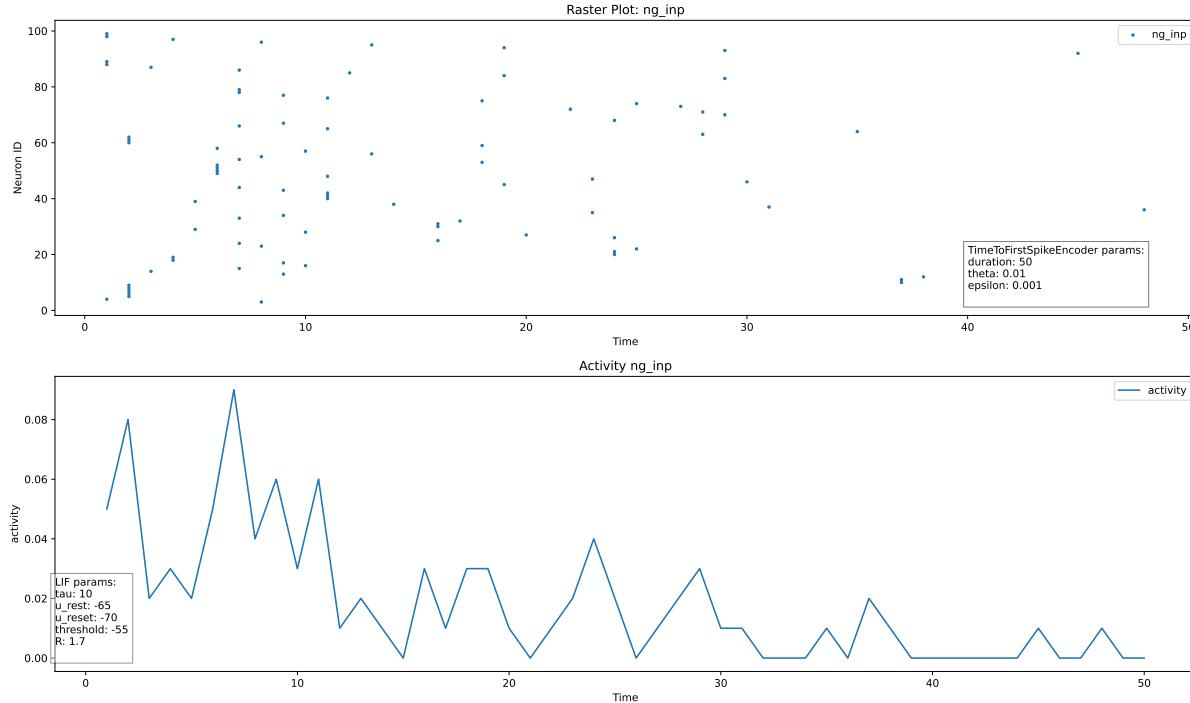


شکل ۵: یک تصویر به عنوان محرک

حال اگر این تصویر را پس از به عنوان ورودی به شبکه بدheim نمودار ضربه های کدگذاری شده این تصویر به صورت شکل ۶ خواهد بود. تغییر در مدت زمان ورودی یا θ نتیجه ای مشابه با تغییراتی که در هنگام کدگذاری آرایه اعداد داشتیم خواهد داشت.

⁵مخزن تصاویر دانشگاه واترلو

Time to first spike encoder



شکل ۶: کدگذاری تصویر به روش *TTFS*

۲.۱۰ کدگذاری اعداد

در این بخش به بررسی یک روش برای کدگذاری اعداد می‌پردازیم. این روش با استفاده از تابع توزیع نرمال کدگذاری می‌کند. کدگذاری بدین صورت است که برای کد کردن یک عدد، چندین توزیع نرمال مختلف بین یک بازه از اعداد را در نظر گرفته، و مقدار عدد ورودی را به هر یک از این توزیع‌ها داده و مشاهده می‌کنیم که این عدد هر یک از این توزیع‌ها را در کجا قطع می‌کند. به طور به خصوص فرض کنید می‌خواهیم یک عدد بین ۱ تا ۱۰، مثلاً ۷.۴ را کدگذاری کنیم. حال به ازای هر یک از اعداد صحیح در این بازه یک نورون و یک تابع توزیع نرمال با میانگین آن عدد در نظر می‌گیریم. حال مقدار هر یک از این توزیع‌ها در نقطه ۷.۴ را محاسبه می‌کنیم. هر توزیعی که مقدار بیشتری داشت، بدان معنا است که زودتر ضربه می‌زند. (شکل ۷ را نگاه کنید).

زمان ضربه زدن نورون‌ها به روش‌های مختلفی می‌تواند انجام شود. مثلاً تابع توزیع با میانگین ۱، ۲، ۳، ۸، ۹، ۱۰ برای عدد ۷.۴ مقدار بسیار کم و نزدیک به همی دارند، و در نتیجه همگی در آخرین لحظات پنجه زمانی^۶ ضربه خواهند زد. از این رو، باید انتخاب کنیم که آیا نیاز به ضربه زدن همه نورون‌های بازه داریم یا ضربه زدن مقادیر نزدیک تر کافی است. اینکار توسط انتخاب یک پارامتر مانند ϵ انجام می‌گیرد که تعیین می‌کند مقادیر توزیع‌ها در x از چه مقداری باید بیشتر باشند. همچنین برای مقادیر بزرگتر، میتوانیم یک بازه بزرگتر انتخاب کنیم، یا اعداد را به همین بازه $0 \dots 10$ مان نگاشت کنیم. همه این روش‌ها بستگی به استفاده ما دارد. نکته دیگری که در رابطه با این روش کدگذاری وجود دارد، این است که این روش برای کدگذاری داده‌های زیاد ممکن است مناسب نباشد، چرا که ما به ازای کد کردن هر عدد، نیاز به تعدادی نورون داریم. (به عنوان مثال، برای کد کردن تنها 10^0 عدد، نیاز به $100 * 100 = 10000$ نورون داریم!).

۳.۱۰ کدگذاری به کمک توزیع پواسون

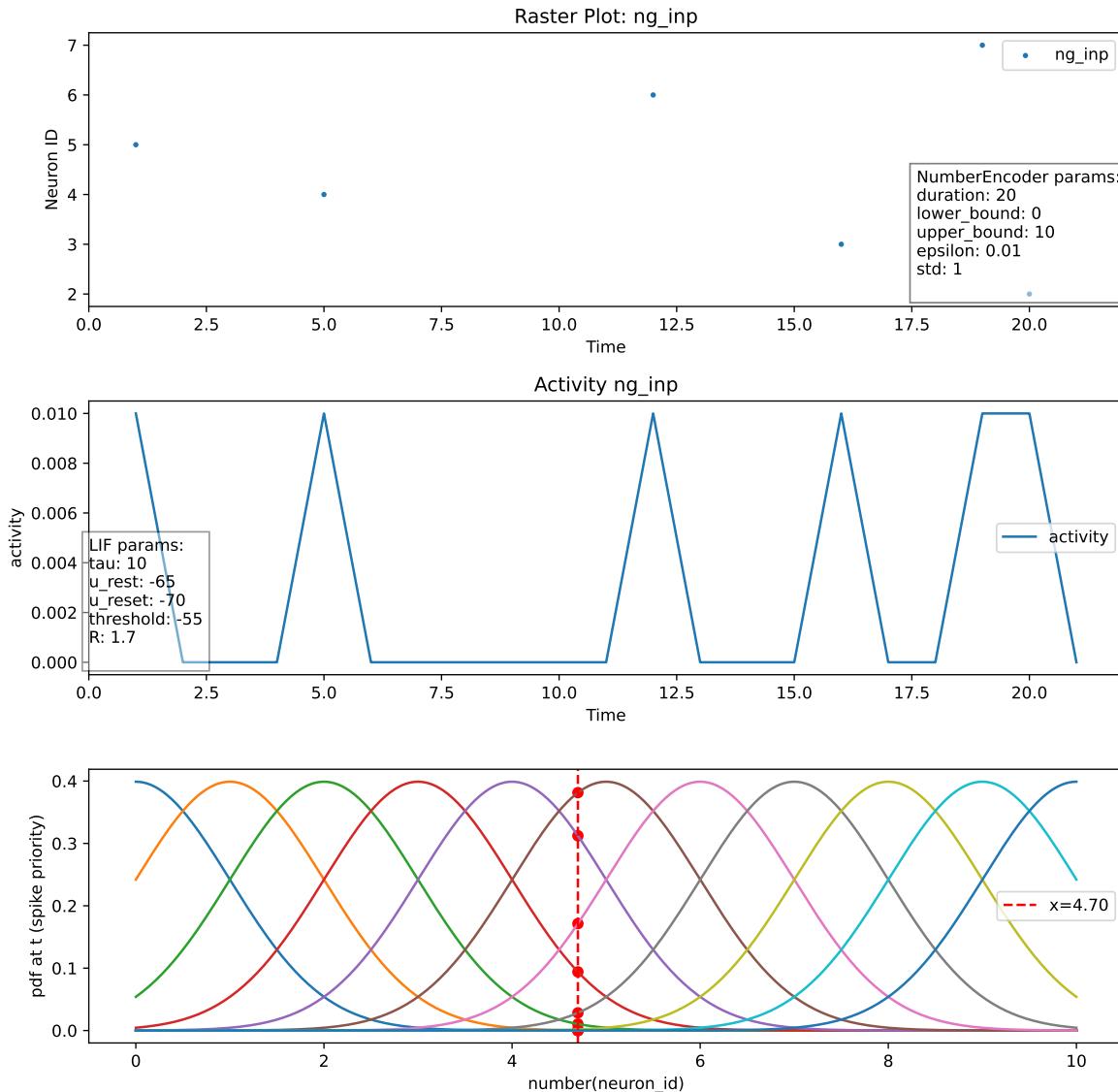
قبل از اینکه بررسی کنیم کدگذاری به روش پواسون چگونه انجام می‌شود، بباید تعریف توزیع پواسون را یکبار با یکدیگر مرور کنیم: توزیع پواسون: در آمار و احتمال توزیع پواسون یک توزیع احتمالی گسسته است که احتمال اینکه یک حادثه به تعداد مشخصی در فاصله زمانی یا مکانی ثابتی رخ دهد را شرح می‌دهد؛ به شرط اینکه این حادث با نرخ میانگین مشخصی و مستقل از زمان آخرین حادثه رخ دهنده. (توزیع پواسون همچنین برای تعدادی از حوادث در فاصله‌های مشخص دیگری مثل مسافت، مساحت یا حجم استفاده شود)[۴]

یک متغیر تصادفی گسسته X دارای توزیع پواسون است، با پارامتر $0 < \lambda$ اگر تابع جرم احتمالی به صورت زیر باشد:

$$f(k, \lambda) = P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \quad (1)$$

time window^۶

Number Encoding



شکل ۷: کدگذاری اعداد. همانطور که در شکل مشاهده می شود، اولین توزیعی که $x = 4.7$ را قطع میکند، توزیع مربوط به نورون ۵ است که در واقعیت نیز عدد ۵ نزدیک ترین عدد صحیح به ۷.۴ است. توزیع های بعدی که $x = 4.7$ را قطع کرده اند به ترتیب مربوط به توزیع با میانگین ۴، توزیع با میانگین ۶، توزیع با میانگین ۳، توزیع با میانگین ۷ و توزیع با میانگین ۳ هستند. بقیه توزیع ها بدلیل داشتن مقدار کم در $x = 4.7$ و نزدیک بودن به هم، با توجه به انتخاب ϵ با مقدار کم، ضربه ای نزدیک نداشتند.

توزیع پواسون را می توان برای سیستم هایی با تعداد زیادی رویداد ممکن که هر کدام نادر هستند اعمال کرد. تعداد چنین رویدادهایی که در یک بازه زمانی ثابت رخ می دهنند، در شرایط مناسب، یک عدد تصادفی با توزیع پواسون است. معادله را می توان طوری بازنویسی کرد اگر به جای میانگین تعداد رویدادهای λ ، نرخ متوسط r به ما داده شود. که در آن وقایع رخ می دهد. از این رو $r = \lambda t$ و:

$$P(X = k) = \frac{e^{-rt}(rt)^k}{k!} \quad (2)$$

ما میدانیم نورون ها ضربه هایی از خود تولید می کنند که این ضربه ها را می توان به عنوان رویدادهای گسسته مدل کرد. به عبارت دیگر زمان هر ضربه را می توان به عنوان یک رویداد تصادفی مشاهده کرد.

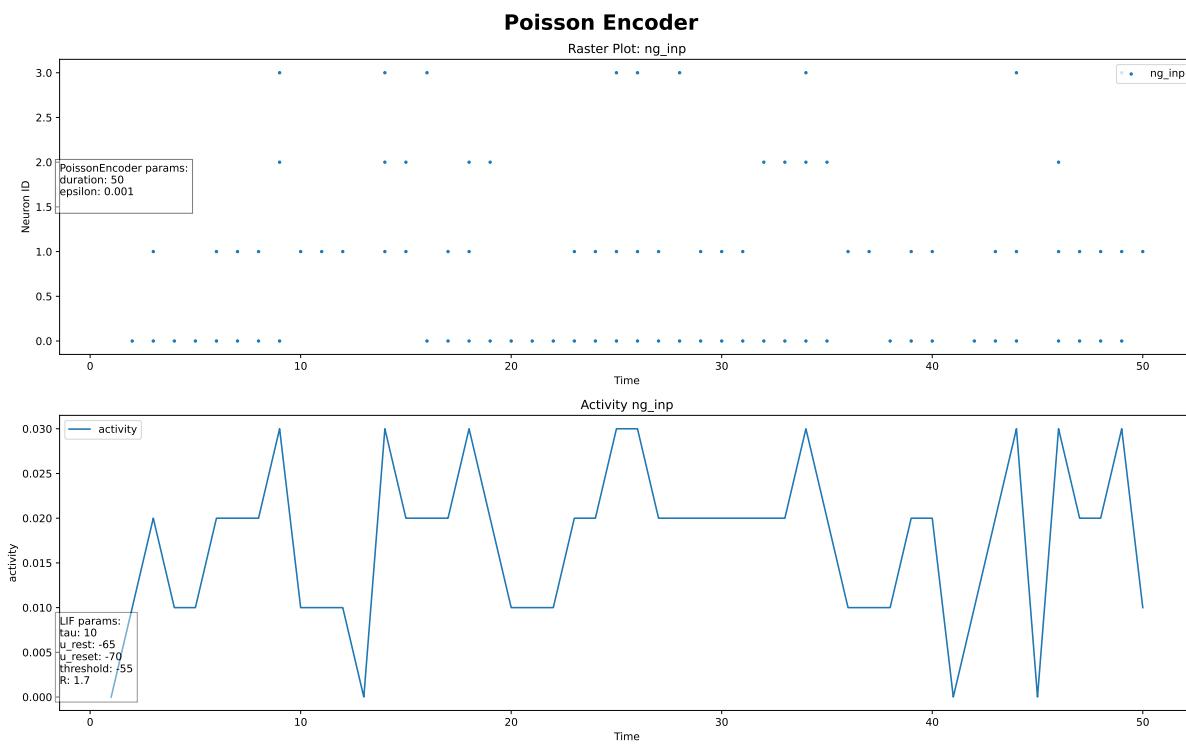
با استفاده از توزیع پواسون، می‌توانیم احتمال P مشاهده دقیق k ضربه در بازه زمانی ثابت t را به صورت زیر مدل کنیم:[۴]

$$P(X = k) = \frac{e^{-\lambda t} (\lambda t)^k}{k!}, \quad (3)$$

که در آن،

- X تعداد ضربه‌ها است.
- عدد اویلر است e .
- تعداد ضربه‌هایی است که ما به طور خاص در بازه t به آنها علاقه مندیم.
- تعداد ضربه‌های مورد انتظار در بازه t است.

حال که نحوه عملکرد و پیاده سازی این کدگذاری را فهمیدیم، می‌توانیم به سراغ آزمایش کردن کدگذاری ورودی‌ها برویم. به طور مثال، همانطور مشاهده می‌شود، به ازای ورودی [100, 70, 30, 10, 0] ضربه‌های کدگذاری شده به صورت شکل ۸ خواهد بود.



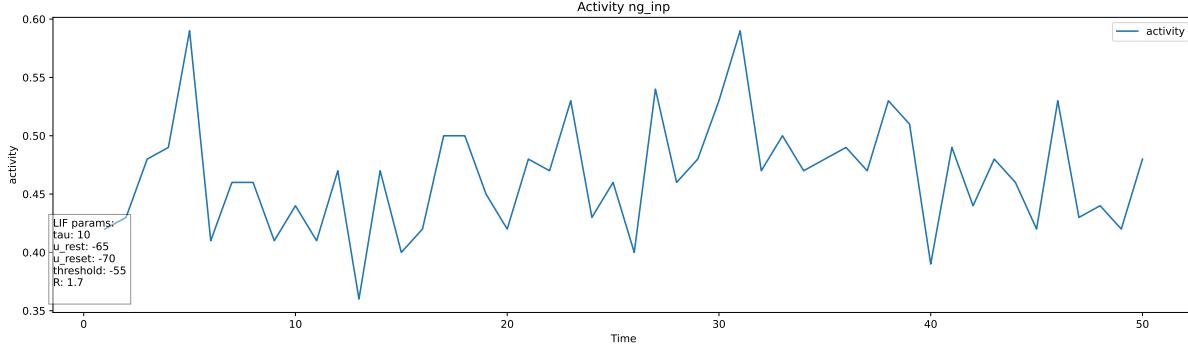
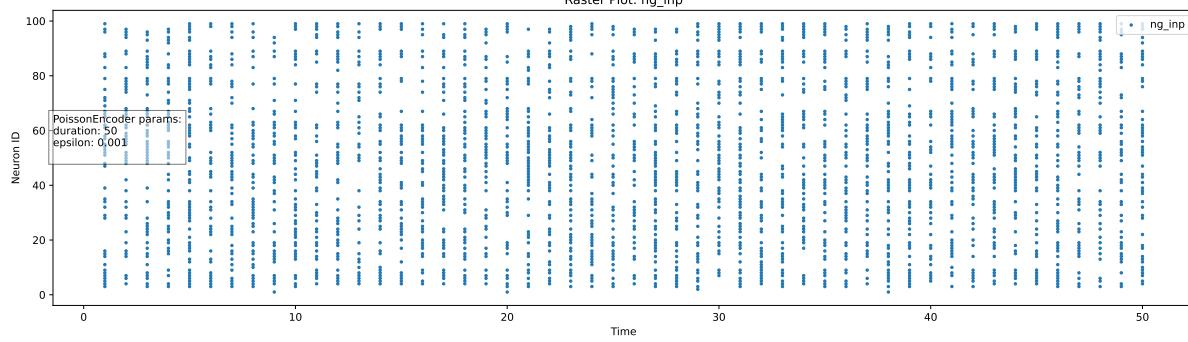
شکل ۸: کدگذاری به کمک توزیع پواسون. همانطور که ملاحظه می‌شود، نورون شماره صفر که بیشترین مقدار، یعنی ۱۰۰ را دارد، بیشترین ضربه را در پنجه زمانی ورودی زده است. نورون‌های دیگر نیز به نسبت اندازه خود نرخ ضربه داشته‌اند.

کدگذاری تصویر

کدگذاری تصویر به روش پواسون یکی از مواردی است که در ادامه پژوهه نیز با آن سر و کار داریم. از این رو می‌خواهیم یک دید کلی نسبت به کدگذاری تصویر به کمک توزیع پواسون داشته باشیم. برای اینکار دوباره از مخزن دانشگاه واترلو استفاده می‌کنیم و تصویر ۵ استفاده می‌کنیم. ضربه‌های کدگذاری شده به صورت شکل ۹ خواهد بود.

Poisson Encoder

Raster Plot: ng_inp



شکل ۹: کدگذاری تصویر به کمک توزیع پواسون. شکل بالا مربوط به کدگذاری یک تصویر 10×10 است. تصاویر ابتدا به صورت یک آرایه در می‌آیند و سپس کدگذاری می‌شوند. در شکل بالا هر نورون مسئول کد کردن یک پیکسل است.

۲۰۰ یادگیری بدون ناظر

در مطالعه علوم اعصاب محاسباتی، درک مکانیسم‌های پلاستیسیته سیناپسی^۷ - چگونگی تقویت یا تضعیف ارتباطات بین نورون‌ها در طول زمان، که از آن به عنوان یادگیری نیز یاد می‌شود - بسیار مهم است. یکی از مدل‌های اصلی برای شبیه‌سازی یادگیری در مغز، قانون یادگیری انعطاف‌پذیری وابسته به زمان ضربه (STDP)^۸) است. این مدل به عنوان یک چارچوب ساده ولی قوی برای بررسی چگونگی تأثیر زمان ضربه‌ها در ارتباطات عصبی بر ارتباطات سیناپسی، در نتیجه یادگیری و حافظه در شبکه‌های عصبی مغز قرار می‌گیرد. STDP یک فرآیند بیولوژیکی است که قدرت اتصالات بین نورون‌های مغز را تنظیم می‌کند. این فرآیند، نقاط قوت اتصال را بر اساس زمان‌بندی نسبی خروجی یک نورون خاص و پتانسیل‌های عمل ورودی (یا ضربه) تنظیم می‌کند.

در فرآیند STDP، اگر یک ضربه ورودی به یک نورون، به طور متوسط، بلافاصله قبل از ضربه خروجی آن نورون رخ دهد، آن ورودی خاص تا حدودی قوی‌تر می‌شود. اگر یک ضربه ورودی، به طور متوسط، بلافاصله پس از یک ضربه خروجی رخ دهد، آن ورودی خاص تا حدودی ضعیفتر می‌شود، بنابراین معنای «یادگیری انعطاف‌پذیری وابسته به زمان ضربه» برای ما روشن‌تر می‌شود. بنابراین، ورودی‌هایی که ممکن است علت برانگیختگی نورون پس سیناپسی باشند، احتمالاً در آینده، کمک می‌کنند، در حالی که ورودی‌هایی که علت ضربه پس سیناپسی نیستند، کمتر در آینده کمک می‌کنند. این فرآیند تا زمانی ادامه می‌یابد که زیرمجموعه‌ای از مجموعه اولیه اتصالات باقی می‌ماند، در حالی که تأثیر همه اتصالات دیگر به ۰ کاهش می‌یابد. [۵] از آنجایی که یک نورون زمانی که بسیاری از ورودی‌هاییش در یک دوره کوتاه اتفاق می‌افتد یک ضربه خروجی تولید می‌کند، زیرمجموعه ورودی‌هایی که باقی می‌مانند عبارتند از آنهایی که تمایل به همبستگی در زمان داشتند. علاوه بر این، از آنجایی که ورودی‌هایی که قبل از خروجی تقویت می‌شوند، ورودی‌هایی که اولین نشانه‌های همبستگی را ارائه می‌دهند، در نهایت به ورودی‌هایی نورون تبدیل می‌شوند. STDP خود به چندین مدل میتواند پیاده سازی شود که از بین این مدل‌ها، مدل Pair-based STDP with local variables بررسی اختیار شده است.

Pair-based STDP ۱۲۰

در مدل STDP مبتنی بر جفت، تنظیم وزن‌های سیناپسی با لحظه‌های دقیق وقوع ضربه در نورون‌های پیش سیناپسی و پس سیناپسی انجام می‌شود. این مدل از دو متغیر محلی - اثر^۹ ضربه پیش سیناپسی و اثر ضربه پس سیناپسی - برای ثبت تاریخچه ضربه عصبی استفاده می‌کند که بر تغییرات سیناپسی تأثیر می‌گذارد.

• اثر ضربه پیش سیناپسی x : این اثر در طول زمان با یک ثابت کاهشی، کاهش می‌یابد اما با هر ضربه پیش سیناپسی دوباره افزایش می‌یابد و به طور مؤثر زمان این ضربه‌ها را مشخص می‌کند. نحوه بروزرسانی این مقدار توسط فرمول زیر می‌تواند محاسبه شود:

$$\frac{dx_j}{dt} = -\frac{x_j}{\tau_+} + \sum_f \delta(t - t_j^f) \quad (4)$$

• اثر ضربه پس سیناپسی y : مشابه اثر پیش سیناپسی، این اثر نیز در طول زمان تحلیل می‌رود، اما با ضربه پس سیناپسی افزایش می‌یابد. نحوه بروزرسانی این مقدار توسط فرمول زیر می‌تواند محاسبه شود:

$$\frac{dy_i}{dt} = -\frac{y_i}{\tau_-} + \sum_f \delta(t - t_i^f) \quad (5)$$

این اثر‌ها (ردپاها) بسیار مهم هستند زیرا مبنای برای محاسبه تغییرات سیناپسی بلافاصله پس از ضربه زدن هستند و باعث می‌شود که مدل به زمان ضربه پاسخ دهد.

سپس نوبت به تغییرات وزن‌ها میرسد. این تغییرات بر اساس مقادیر فعلی اثر‌های گفته شده در هر زمان که یک ضربه زده می‌شود محاسبه می‌شود. این روش نه تنها با مشاهدات بیولوژیکی هماهنگ است، بلکه امکان شبیه‌سازی دقیق فرآیندهای یادگیری و تشکیل حافظه در شبکه را نیز فراهم می‌کند. نحوه بروزرسانی وزن‌ها نیز توسط فرمول زیر امکان پذیر است:

$$\frac{dw_{ij}}{dt} = -A_-(w_{ij})y_i(t) \sum_f \delta(t - t_j^f) + A_+(w_{ij})x_j(t) \sum_f \delta(t - t_i^f) \quad (6)$$

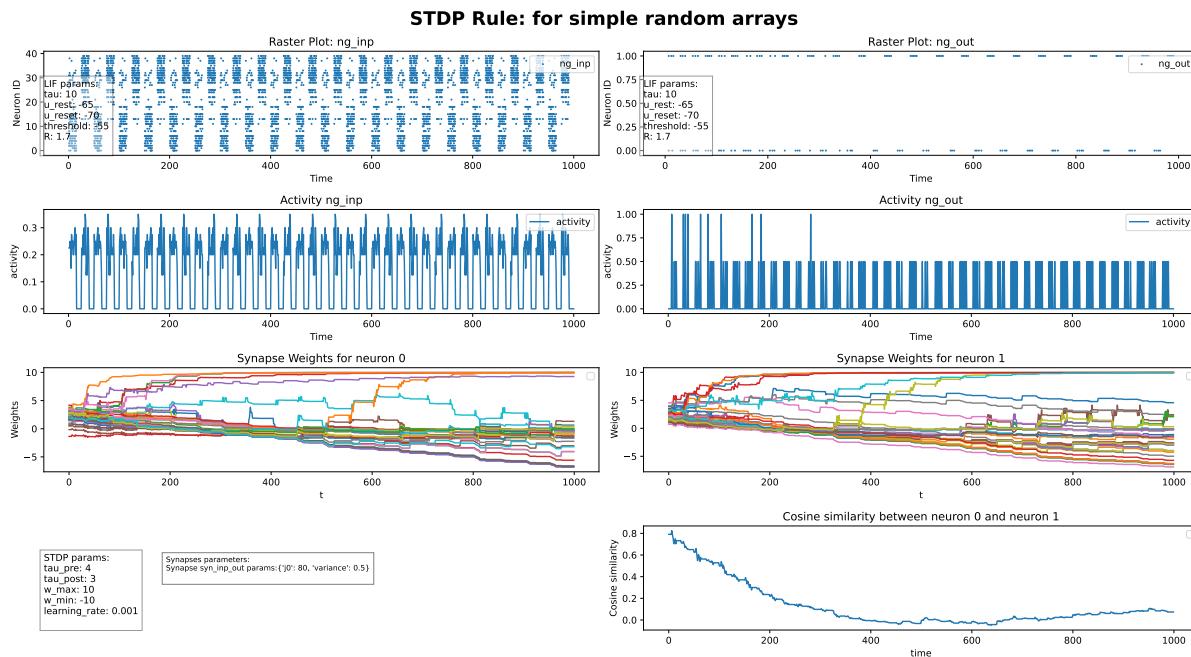
قبل از رفتن به آزمایش کردن مدل، یک نکته دیگر باقی می‌ماند و آن این است که برای A_- ، A_+ میتواند توابعی^{۱۰} برای کنترل کردن محدوده وزن‌ها استفاده شود.

⁷ synaptic plasticity
⁸ Plasticity Spike-Timing-Dependent
⁹ trace
¹⁰ Soft/Hard bound

۲.۲۰ آزمایش ها

شباهت کسینویسی: دقت شود که نمودار شباهت کسینویسی در تمام شکل ها آمده است.

حال در این قسمت، به سراغ آزمایش مدل *STDP* مان میرویم و نمودار های خواسته شده در موارد الف تا و را بررسی میکنیم. ابتدا دو الگوی به صورت آرایه را به مدل میدهیم. برای اینکار میتوانیم یک آرایه تصادفی از اعداد ۱ تا ۱۰ و یک آرایه تصادفی دیگر که مقادیر آن دوبرابر آرایه اول است را به عنوان ورودی به جمعیت ورودی بدهیم. از این رو، مطابق شکل ۱۰ مشاهده میکنیم که با تغییر وزن های سیناپسی، هر نورون یکی از الگو ها را یادگرفته است. به طوری که وقتی الگوی یاد گرفته شده را میبینند نرخ ضربه زدن آن بیشتر می شود.



شکل ۱۰: قانون یادگیری *STDP*. همانطور که از شکل نیز مشخص است، وزن های سیناپسی طوری تنظیم می شوند که هر نورون، یکی از الگو ها را یاد میگیرد. همچنین طبق نمودار شباهت کسینویسی، به مرور زمان، شباهت بین وزن های دونورون خروجی نیز کاهش می یابد.

تغییر در اندازه ورودی

هر چند این یادگیری به دلیل بدون ناظر بودن شکله و همچنین ساده بودن آن، از عهده یادگیری الگو های پیچیده تر ممکن است برニاید. به طور مثال، با زیاد کردن طول آرایه، ویژگی های مورد نیاز برای تشخیص دادن نیز افزایش یافته پیداست، یادگیری ممکن است به خوبی انجام نشود.

به عنوان مثال، در شکل ۱۱ یادگیری به خوبی انجام شده است، ولی نسبت به شکل ۱۰ دیرتر یادگیری شکل گرفته است.

حال مطابق خواسته پیروز، ورودی ها را به گونه ای به شبکه می کنیم که نورون های ورودی در دریافت محرك همپوشانی داشته باشند. یعنی بعضی نورون ها، مسئول گرفتن هر دو ورودی باشند. (مطابق شکل ۱۲)

ابتدا با همپوشانی کم شروع می کنیم. مطابق شکل ۱۳ دریافت می شود که همپوشانی نورون ها در دریافت ورودی نمی تواند مانع یادگرفتن مدل شود هر چند که سرعت آن را کاهش می دهد.

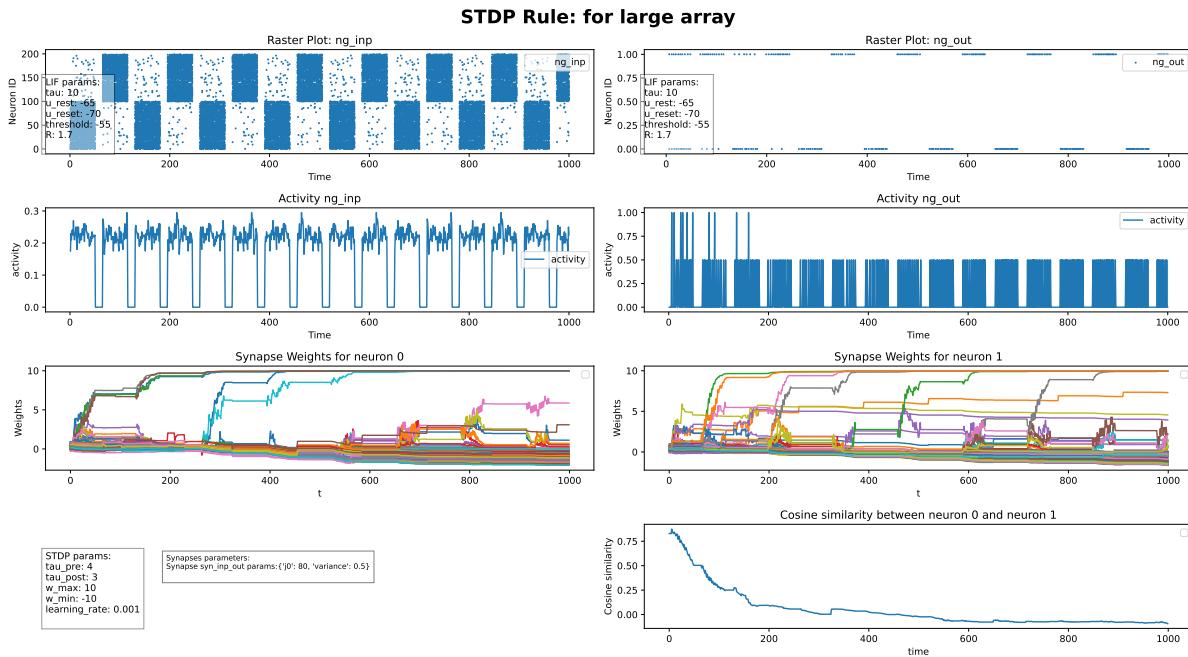
حال این همپوشانی را حداکثر میکنیم به طوری که تمام نورون های ورودی مسئول دریافت هر دو الگو باشند. انتظار داریم که افزایش این همپوشانی، منجر به اختلال در یادگیری شود یا یادگیری را به تعویق بیندازد. شکل ۱۴ این موضوع را تایید میکند.

ممکن است این موضوع که مسلمان می تواند الگو های نظر آرایه های بالا را تشخیص دهد، کنگاکوی ما را برانگیزد تا بخواهیم الگو های پیچیده را نیز تشخیص دهیم. برای اینکار، ابتدا از الگو های ساده تری مانند تصاویر ۱۱آ و ۱۱ب استفاده میکنیم تا قدرت مدل را بسنجدیم و سپس به سراغ الگو های پیچیده تر می رویم.

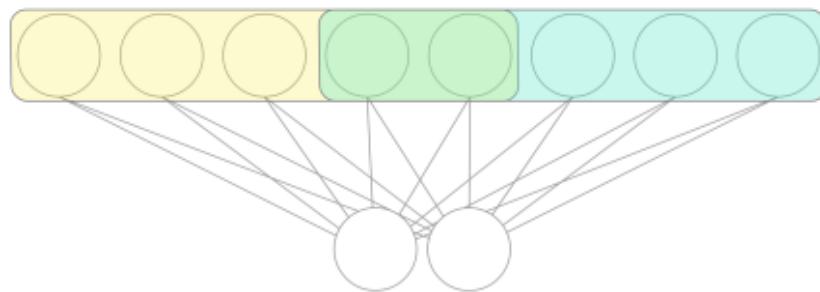
مطابق شکل ۱۶ ملاحظه میکنیم که مدل به خوبی از عهده تشخیص الگو های ساده ۱۱آ و ۱۱ب برآمد و میتواند آن ها را تشخیص دهد.

حال یک قدم فراتر رفته و یک تصویر بزرگتر را به عنوان ورودی به مدل می دهیم. ابعاد این تصاویر بزرگتر بوده (10×10) و پیچیده تر هستند. (تصاویر ۱۷آ و ۱۷ب)

هرچند انتظار زیادی از مدل *STDP* ساده برای تشخیص تصاویر نداریم، ولی مطابق شکل ۱۸ به نظر می رسد که مدل پس از تکرار های بیشتری نسبت به حالت های قبل، می تواند از عهده تشخیص این الگو ها نیز برآید!



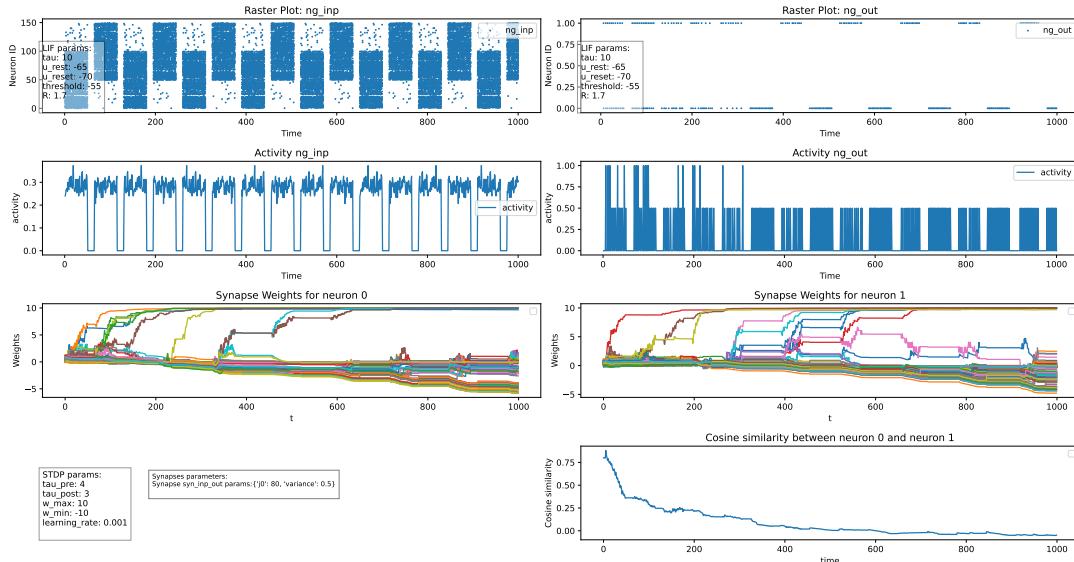
شکل ۱۱: قانون یادگیری *STDP* برای ورودی بزرگتر. مطابق شکل ملاحظه میکنیم علی رغم اینکه مدل توانسته تا تکرار ۱۰۰۰ ام شبیه سازی به خوبی الگوهای را یاد بگیرد، ولی نسبت به شکل ۱۰ که ورودی کوچکتر بود این اتفاق دیرتر افتاده است. هرچند با تغییر پارامترهایی مانند پنجه زمانی یا نرخ یادگیری میتوان این یادگیری را سریع‌تر انجام داد که این موضوع را در آدامه بررسی خواهیم کرد.



شکل ۱۲: نورون‌هایی که با رنگ سبز مشخص شده‌اند، هر دو الگو را به عنوان ورودی دریافت میکنند.

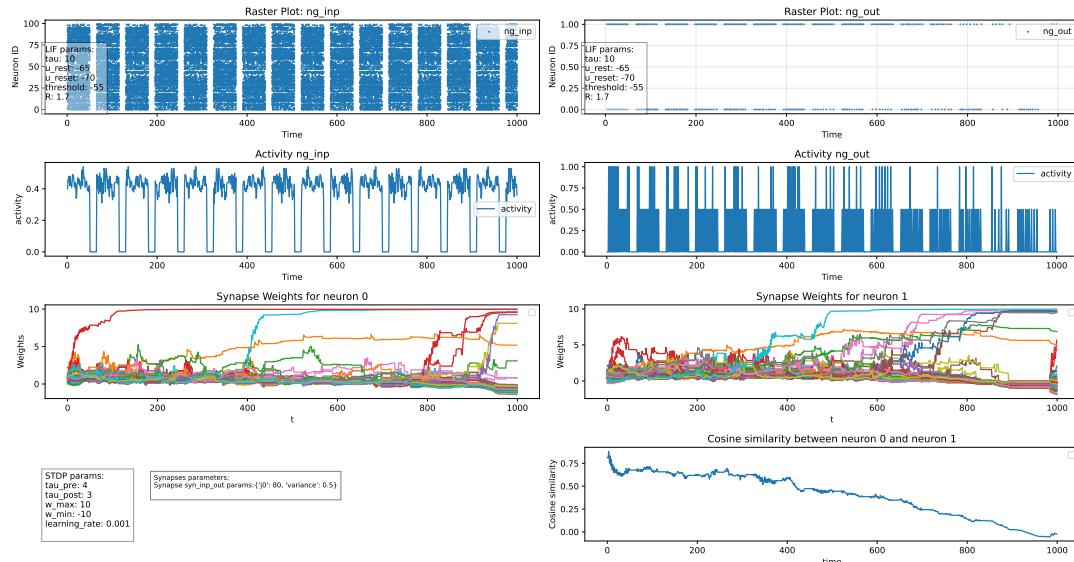
مشابه کاری که برای آرایه‌ها کردیم، اینجا نیز همپوشانی نورون‌های ورودی را برای دریافت محرک بیشتر میکنیم تا تاثیر آن برایمان روشن تر شود. (شکل ۱۹)

STDP Rule: for large array with overlap



شکل ۱۳: همپوشانی نسبی نورون های لایه اول برای دریافت ورودی. مطابق شکل ملاحظه میکنیم که همپوشانی بین نورون های ورودی باعث می شود یادگیری در زمان کمی دیرتری اتفاق بیفتد ولی مانع آن نمی شود.

STDP Rule: for large array with high overlap



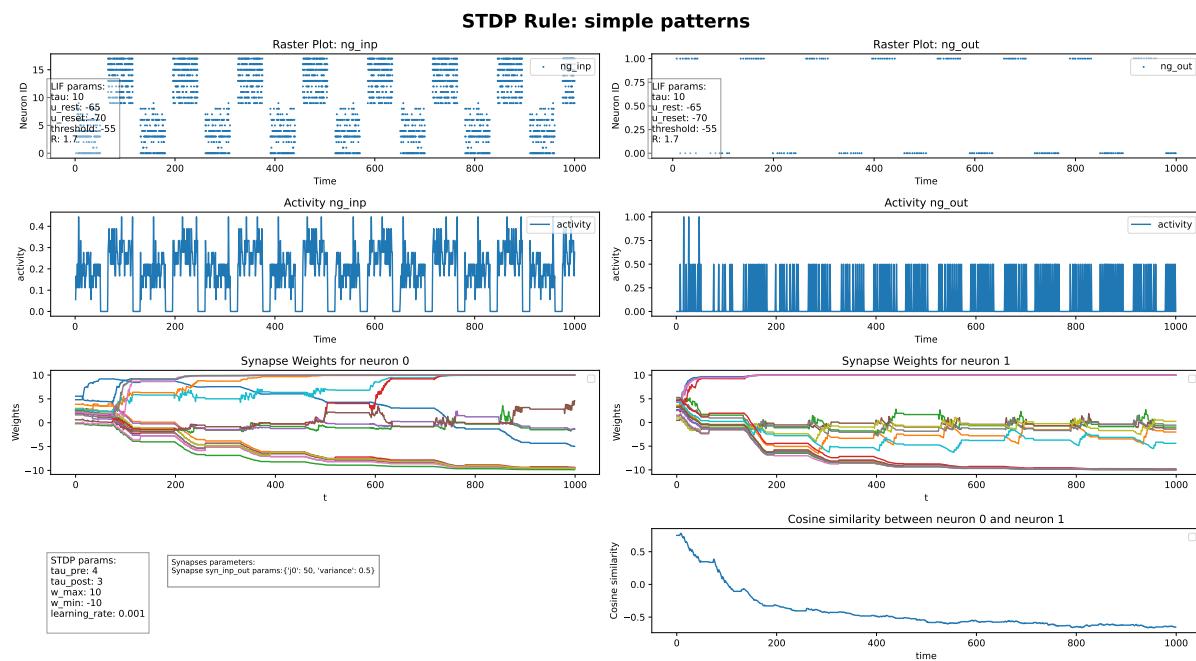
شکل ۱۴: همپوشانی کامل نورون های لایه اول برای دریافت ورودی. مطابق شکل ملاحظه میکنیم که همپوشانی کامل بین نورون های ورودی باعث می شود یادگیری مختلف شود و مثلا در شکل بالا فقط یکی از نورون ها یکی از الگو ها را یاد بگیرد. این به دلیل ساده بودن مدل STDP است. به طوری که در بخش سوم مشاهده میکنیم که مدل RSTDP نسبت به این همپوشانی پایدار است.



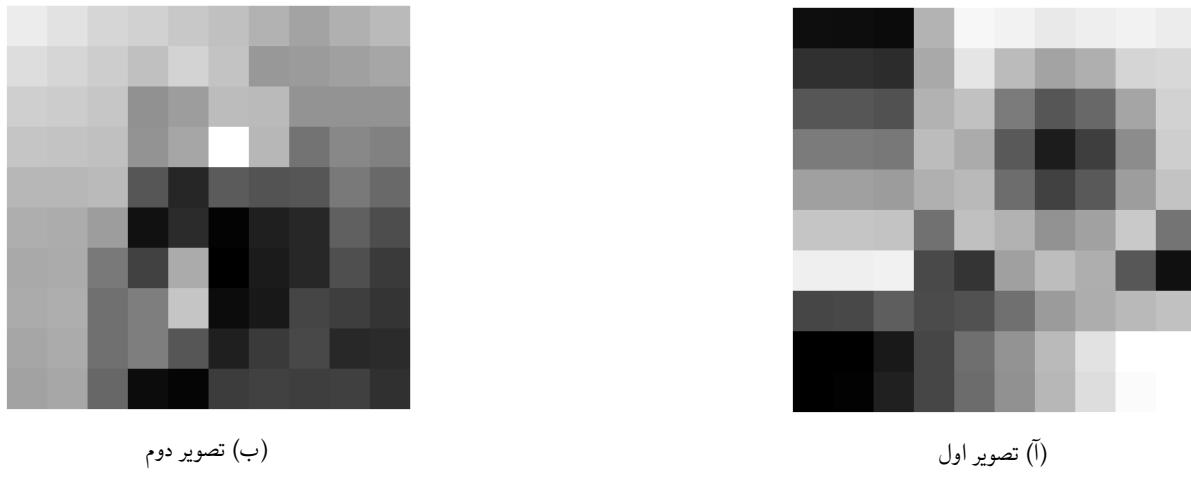
(ب) الگوی دوم

(ا) الگوی اول

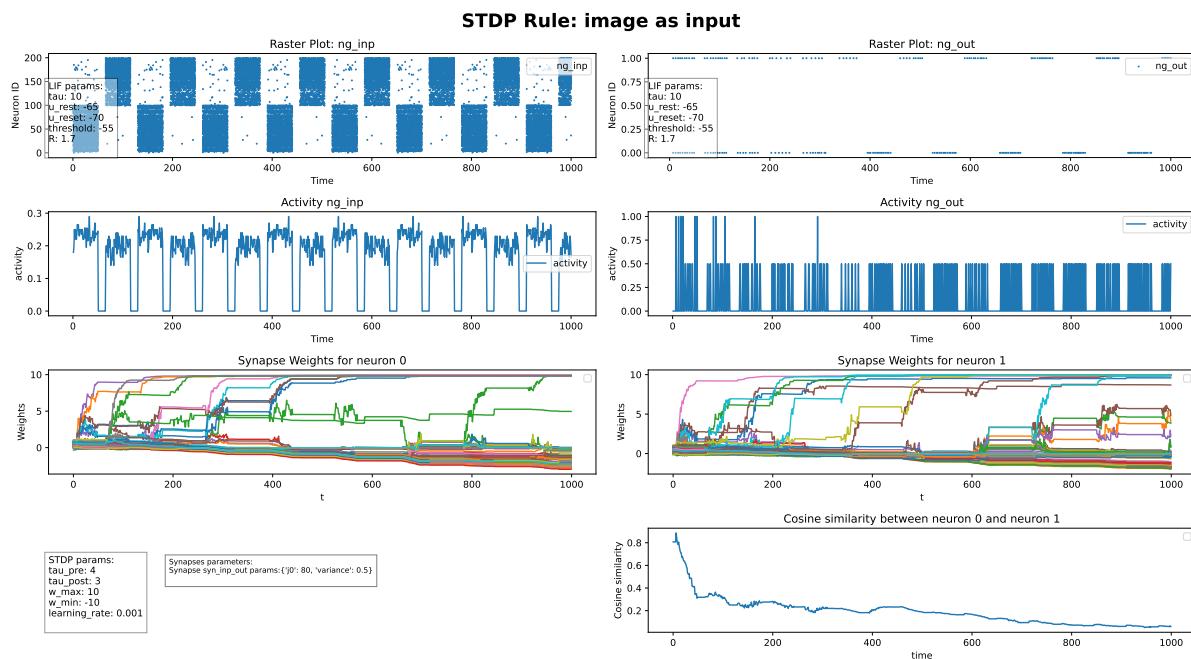
شکل ۱۵: دو تصویر به عنوان محرک



شکل ۱۶: قانون یادگیری STDP برای ورودی تصویر ساده. مشاهده میکنیم که مدل توانسته به خوبی پ. الگوی تصویری ساده را از یکدیگر تشخیص دهد.

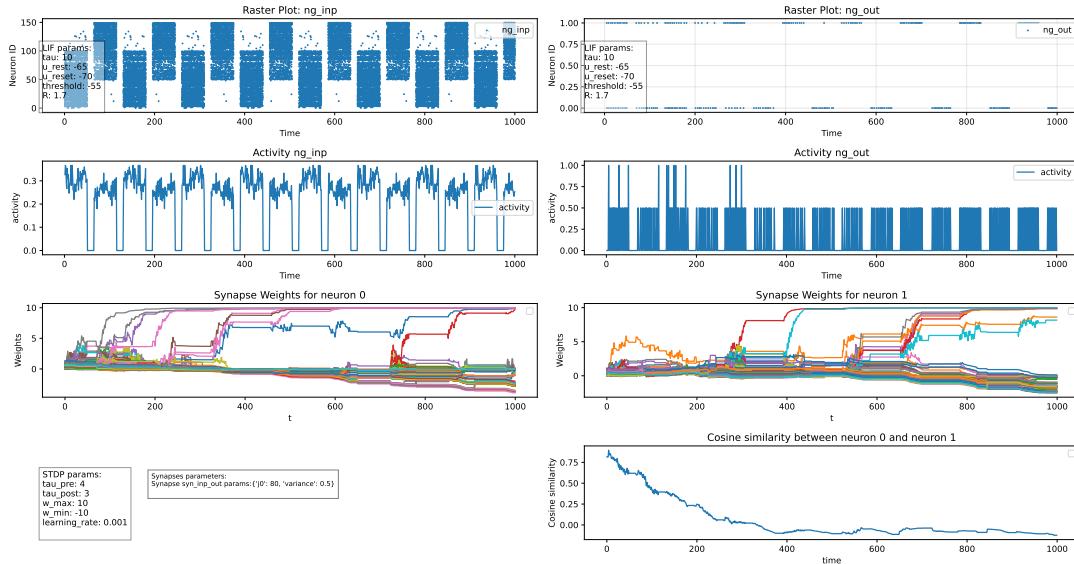


شکل ۱۷: یک تصویر به عنوان محرک



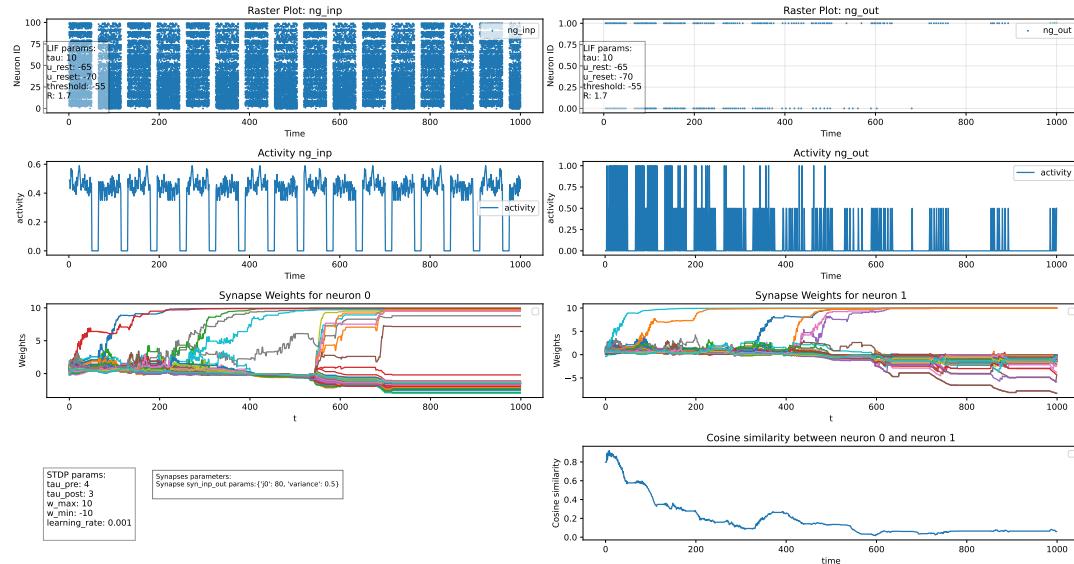
شکل ۱۸: قانون یادگیری STDP برای ورودی تصویر. مطابق شکل دریافت می‌شود که مدل توانسته است دو ورودی تصویر را نیز از یکدیگر تشخیص دهد.

STDP Rule: images with overlap



(ا) ورودی با همپوشانی نسبی.

STDP Rule: images with complete overlap



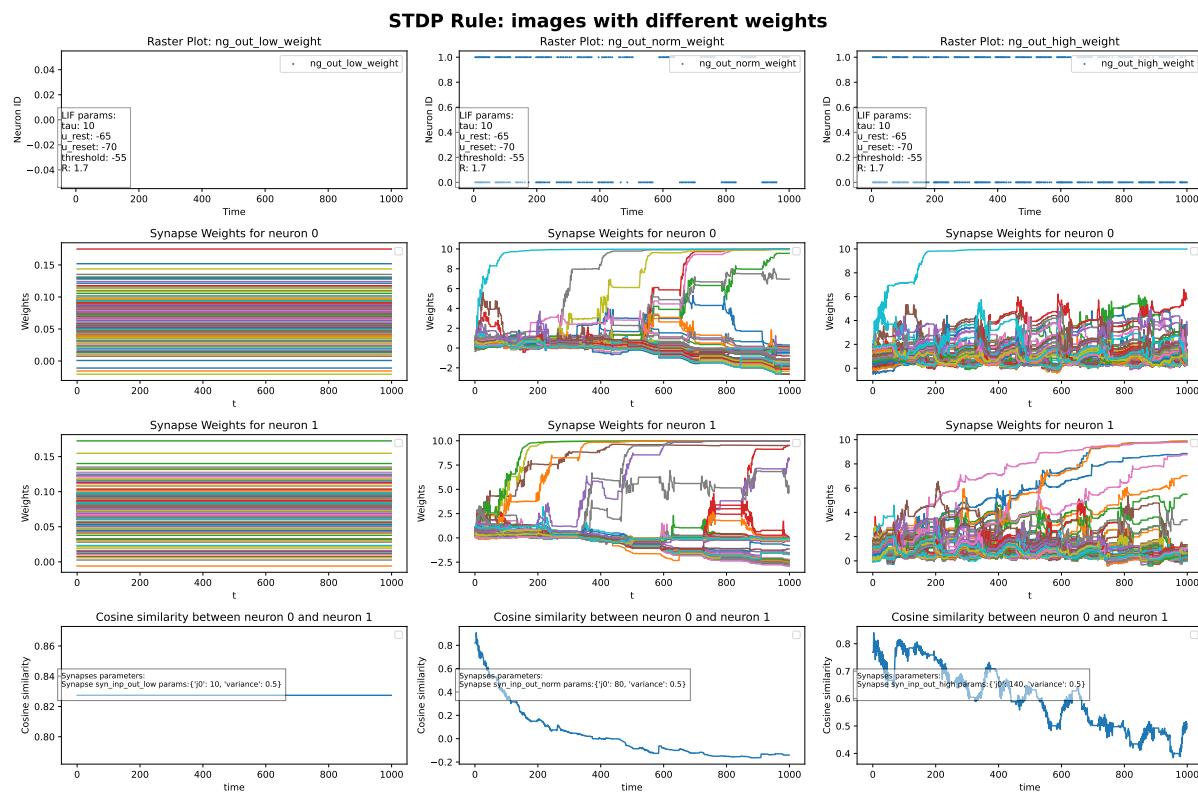
(ب) ورودی با همپوشانی کامل.

شکل ۱۹: مشاهده میکنیم که افزایش همپوشانی همانند آرایه ها باعث می شود یادگیری نتواند به طور صحیح انجام شود.

تغییر در وزن های اولیه

در شبکه های عصبی مصنوعی، انتخاب وزن های اولیه در شکل دهنده مسیر یادگیری و عملکرد کلی شبکه بسیار مهم است. این پارامترهای اولیه نقطه شروع فرآیند بهینه سازی را تعیین می کنند و بر سرعت و تاثیرگذاری شبکه می توانند به یک راه حل خوب همگرا شوند. وزن های اولیه بد انتخاب شده می توانند منجر به همگرای کند، گیر کردن در مینیمم محلی یا ناتوانی در یادگیری کلی شود. بر عکس، وزن های اولیه به خوبی انتخاب شده می توانند همگرای سریع تر را تسهیل کرده و شبکه را قادر می سازند تا به حداقل های عمیق تر و مؤثر تر در فضای راه حل دست یابد. در مورد شبکه های عصبی ضربه ای نیز، انتخاب وزن های اولیه یک عامل حیاتی است که می تواند تأثیر قابل توجهی بر تاثیر الگوریتم های یادگیری داشته باشد. این تنظیم وزن های اولیه زمینه را برای چگونگی سازگاری و عملکرد شبکه در طول فرآیند یادگیری فراهم می کند.

ما در این آزمایش، وزن های اولیه مدل را با سه مقدار کم، نرمال و زیاد آزمایش و سپس نتایج را تحلیل می کنیم. مطابق شکل ۲۰ ملاحظه می کنیم که انتخاب وزن های خیلی کم یا خیلی زیاد هر دو میتوانند فرآیند یادگیری را مختلف کنند. به طور کلی، وزن های سیناپسی بیشتر، نرخ ضربه زدن نورون ها را بیشتر می کنند و این وزن ها باید به گونه ای انتخاب شوند که نه خیلی کم باشد که از ضربه زدن جلوگیری شود و نه خیلی زیاد باشد که نورون ها به طور مداوم ضربه بزنند.



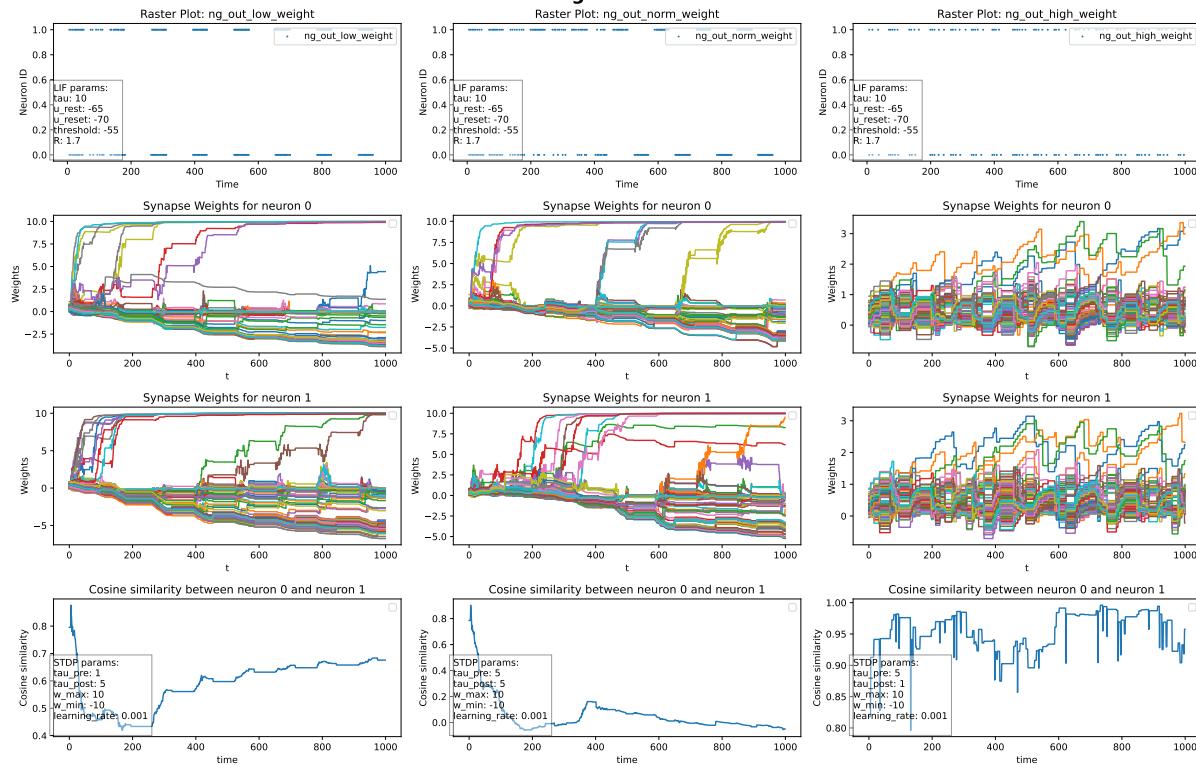
شکل ۲۰: قانون یادگیری $STDP$ با مقادیر مختلف وزن های اولیه. مطابق شکل مشاهده می کنیم که انتخاب وزن های خیلی کم ممکن است باعث شود کلا ضربه ای در جمعیت زده نشود و در نتیجه وزن ها نیز ثابت مانده و درکل فعالیتی دیده نشود. همچنین انتخاب وزن های خیلی زیاد نیز باعث می شود که نرخ ضربه زدن به شدت بالا رفته و تاثیر یادگیری نیز از بین بود و نورون ها فقط هنگام دریافت ورودی ضربه بزنند.

پارامتر های τ_{post} و τ_{pre}

حال نوبت به آزمایش دو پارامتر τ_{pre} و τ_{post} می رسد. به طور کلی این دو پارامتر، میزان ردیابی باقی مانده ضربه های پیش سیناپسی یا پس سیناپسی را تنظیم می کنند. علاوه بر مقدار خود این پارامتر ها، نسبت این دو پارامتر به یکدیگر نیز میتواند در یادگیری مدل موثر باشد. من در آزمایش هایی که داشتم، به طور کلی این نتیجه را گرفتم که این دو پارامتر باید تقریباً با هم برابر باشند ولی میزان پارامتر τ_{pre} باید کمی بیشتر باشد. در این بخش آزمایش را صرفا برای نسبت های مختلف این دو پارامتر به یکدیگر بررسی می کنیم.

همانطور که در شکل ۲۱ نیز ملاحظه می کنیم، خیلی کمتر بودن یا بیشتر نسبت τ_{post} به τ_{pre} می تواند باعث شود که مدل به خوبی نتواند الگو های ورودی را یاد بگیرد.

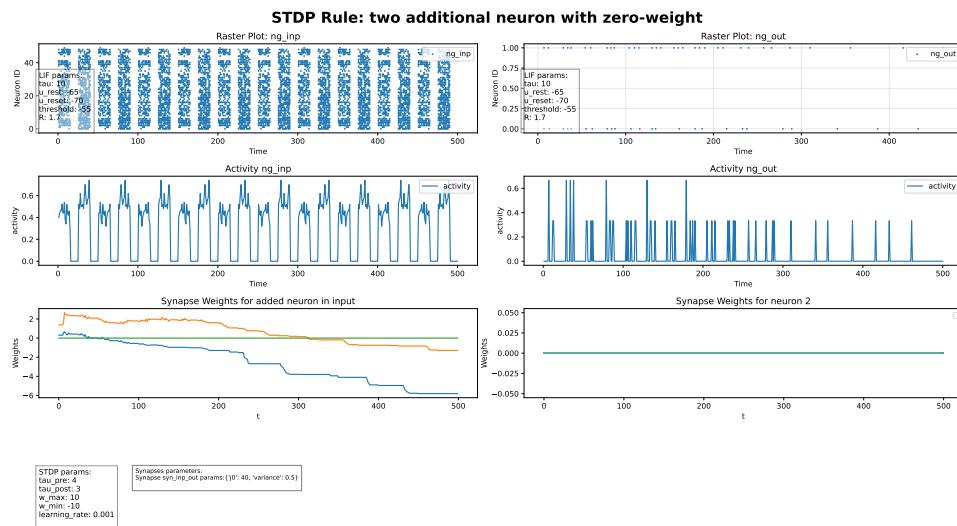
STDP Rule: images with different tau



شکل ۲۱: قانون یادگیری STDP با نسبت های مختلف پارامتر های τ_{pre} و τ_{post} . مطابق شکل بالا دریافت می شود که کمتر بودن نسبت پارامتر τ_{post} به τ_{pre} میتواند باعث شود که تاثیر ضربه های نورون های پس سیناپسی به سرعت از بین برود و فقط ضربه های نورون های پیش سیناپسی روى وزن ها اثر بگذارند. در این حالت واضح است که یادگیری نمی تواند به خوبی انجام شود و مدل ها همزمان روی هر دو الگو ضربه می زندند. در حالت دیگر که نسبت τ_{pre} به τ_{post} بیشتر می شود، باعث می شود تاثیر ضربه های نورون های پیش سیناپسی تا مدت زیادی بماند طوری که حتی پس از ورودی الگوی جدید نیز ممکن است این ردپا ها حضور داشته باشند. در این حالت نیز واضح است که یادگیری به خوبی انجام نخواهد شد. به طور کلی این نتیجه را گرفتم که این دو پارامتر باید تقریباً با هم برابر باشند ولی میزان پارامتر τ_{pre} باید کمی بیشتر باشد.

۳.۲۰ اضافه کردن دو نورون غیر فعال

در این قسمت به هر یک از دو لایه ورودی و خروجی یک نورون اضافه میکنیم که در طول آموزش ضربه‌ای نزنند و سپس وزن های متصل به این دو نورون را تحلیل میکنیم. از آنجاکه توضیحات زیادی در این باره در فایل پژوهه داده نشده است، ما فرض را بر این میگیریم که ضربه نزدن نورون خروجی، با صفر کردن وزن های متصل به آن اتفاق بیفتد. روش دیگر برای اینکار، بالا بردن آستانه پتانسیل عمل آن نورون است. من هردو روش را آزمایش میکنم و نتایج را می آورم. برای روش اول، که جلوگیری از ضربه زدن نورون توسط صفر کردن وزن سیناپسی اتفاق میافتد، مطابق شکل ۲۲ ملاحظه میکنیم که اضافه کردن یک نورون به لایه ورودی و یک نورون به لایه خروجی که ضربه ای نزند، باعث می شود که یادگیری به طور کامل با اختلال رو به رو شود و عمل یادگیری اتفاق نیافتد.

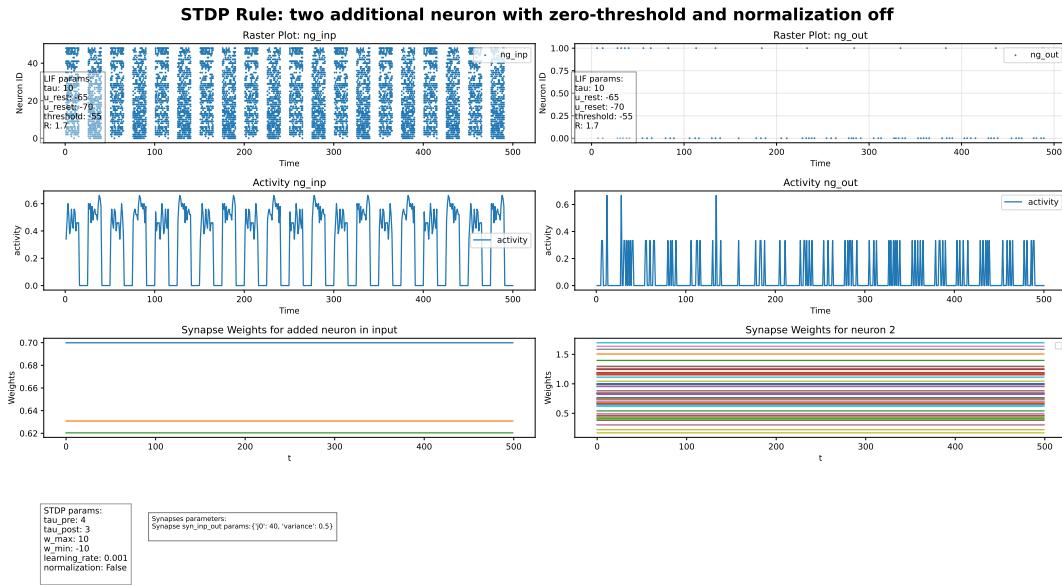


(آ) جلوگیری از ضربه زدن نورون های اضافه شده از طریق صفر کردن وزن ها. مشاهده میکنیم که اضافه کردن نورون به ۲ لایه که ضربه ای نمیزنند، میتواند در یادگیری مدل اختلال ایجاد کند به طوری که دیگر یادگیری انجام نشود. همچنین در نمودار وزن ها ملاحظه میکنیم که یکی از وزن های متصل به نورون اضافه شده در خروجی تغییر نداشته اند.



(ب) جلوگیری از ضربه زدن نورون های اضافه شده از طریق افزایش آستانه پتانسیل عمل نورون اضافه شده. مشاهده میکنیم، مانند شکل قبل اضافه کردن نورون به ۲ لایه، میتواند در یادگیری مدل اختلال ایجاد کند به طوری که علاوه بر یادگیری انجام نمیشود. همچنین در نمودار وزن ها ملاحظه میکنیم که یکی از وزن های متصل به نورون اضافه شده در ورودی و همچنین تمام وزن های نورون اضافه شده در خروجی تغییر نداشته اند. تنها تفاوت با شکل قبل این است که وزن های نورون اضافه شده در خروجی صفر نیستند و همان مقادیر اولیه را دارند.

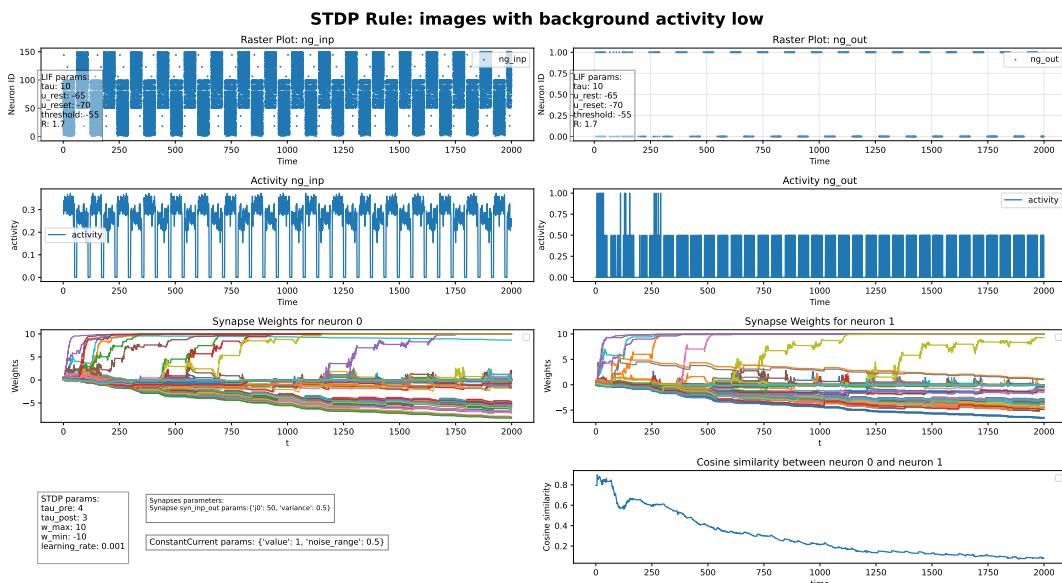
حال اگر وزن ها را نرمال سازی نکنیم، نمودار ها به صورت شکل ۲۳ تغییر خواهد کرد و یادگیری به طور کامل مختل خواهد شد.



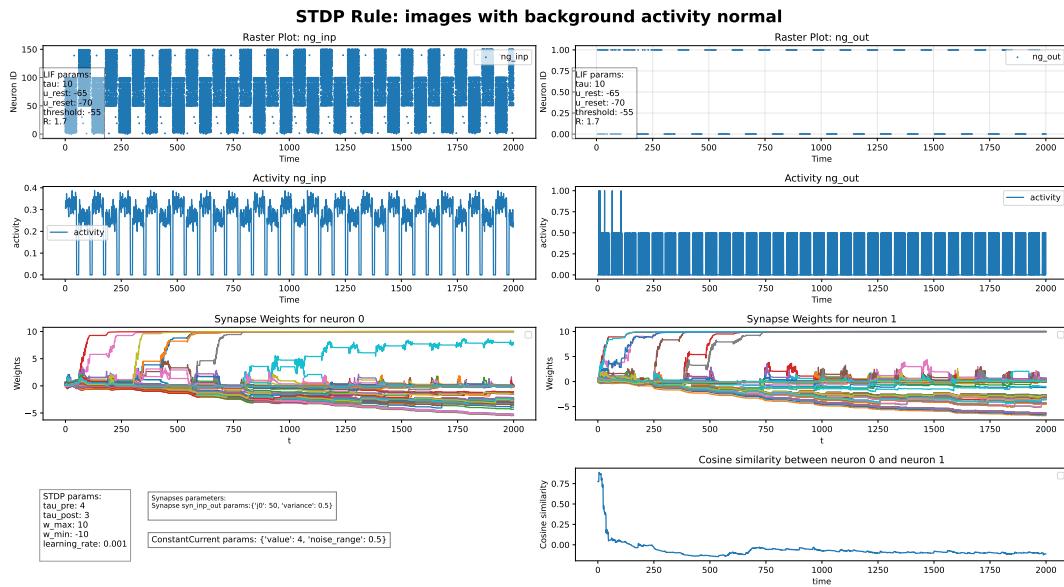
شکل ۲۳: جلوگیری از ضربه زدن نورون های اضافه شده از طریق افزایش آستانه پتانسیل عمل نورون اضافه شده و نرمال سازی خاموش. مشاهده میکنیم، همانند شکل های ۲۲آ و ۲۲ب با خاموش کردن نرمال سازی، یادگیری مدل به طور کامل مختلف می شود. همچنین تغییر دیگری که ملاحظه می شود، تغییر در وزن های دیگر متصل به نورون جدید در ورودی است به طوری که وزن ها تا پایان شبیه سازی ثابت مانده اند.

۴.۲.۰۵ بررسی مدل با فعالیت کمینه برای لایه ها

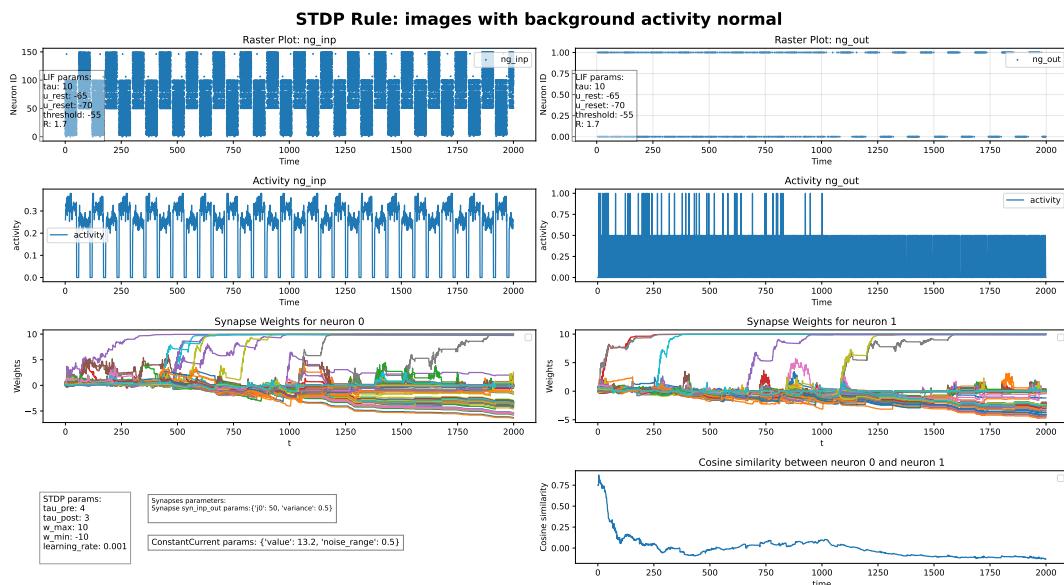
در آخرین آزمایش مربوط به این قسمت، به بررسی رفتار مدل هنگامی که یک فعالیت کمینه در دو لایه وجود دارد را بررسی میکنیم. برای اینکار، آزمایش را با سه مقدار متفاوت فعالیت زمینه انجام میدهیم. همچنین حالت پایه آزمایش را، الگوی تصاویر با هم پوشانی نسبی در نظر میگیریم: همانطور که از شکل ۲۴ دریافت می شود، به طور کلی افروزن یک جریان ثابت نویزی با، مانع یادگیری مدل نمی شود.



شکل ۲۴: مدل با فعالیت زمینه کم.



شکل ۲۵: مدل با فعالیت زمینه عادی.



شکل ۲۶: مدل با فعالیت زمینه زیاد. مشاهده میکنیم که تغییر در میزان فعالیت زمینه، مشکلی در یادگیری مدل در زمان طولانی ایجاد نمیکند. داشتن فعالیت زمینه در اندازه عادی (شکل ۲۵) یعنی اندازه‌ای که نورون نیاز دارد تا برانگیخته شود، مشکلی برای فرایند یادگیری ایجاد نمیکند. اما زیاد از حد بودن فعالیت کمیه میتواند یادگیری را مدت زیادی به تأخیر بیندازد. در مقایسه با حالت قبل نیز میتوان گفت، که افزودن فعالیت کمیه دربرابر افزودن نورون بدون ضربه، کمتر یادگیری را مختلف میکند.

۳۰۰ قانون یادگیری تقویتی

انعطاف‌پذیری وابسته به زمان ضربه تعديل شده با پاداش ($R - STDP$)^{۱۱} به عنوان یک مدل پیچیده برای درک چگونگی تکامل ارتباطات سیناپسی بر اساس زمان بندی ضربه های عصبی و تأثیر سیگنال های پاداش عمل می‌کند. این مدل، که شامل تعديل کننده های عصبی مانند دوپامین است، قدرت سیناپسی را برای ترویج رفتارهایی که منجر به پاداش می‌شود، تنظیم می‌کند. در اینجا، پیاده‌سازی $R - STDP$ را بررسی می‌کنیم که این پویایی‌ها را در یک چارچوب محاسباتی ادغام می‌کند و مطالعه مکانیسم‌های یادگیری را که زیربنای فرآیندهای یادگیری مبتنی بر پاداش است، تسهیل می‌کند.

مدل $R - STDP$ که در اینجا پیاده‌سازی کرده‌است، مکانیسم کلاسیک $STDP$ را با ترکیب اثرات دوپامین، که به عنوان یک سیگنال بازخورد برای تقویت یا مجازات اعمال عمل می‌کند، گسترش می‌دهد. این مدل برای رسیدگی به پیچیدگی‌های انعطاف‌پذیری سیناپسی تحت تأثیر زمان ضربه‌ها و حضور سیگنال‌های پاداش، نوشته شده است.

- **ردیابی‌های سیناپسی و دوپامین:** این مدل از اثر (ردیبا) های پیش سیناپسی و پس سیناپسی استفاده می‌کند که در طول زمان با ثابت τ_{pre} و τ_{post} کاهش می‌یابند. این ردیبا ها زمان ضربه‌ها را ثبت می‌کنند و به عنوان یک متغیر اصلی برای محاسبه تغییرات سیناپسی عمل می‌کنند.

- **بهروزرسانی وزن‌ها با دوپامین:** بهروزرسانی‌های اثربخشی سیناپسی به تعامل بین آثار ضربه و سطوح دوپامین بستگی دارد، که به معنای این است که رفتارها منجر به افزایش ضربه شده است یا خیر. این مدل به صورت پویا وزن‌ها را بر اساس اینکه آیا رفتارها با نتایج مثبت (دوپامین مثبت) یا نتایج منفی (دوپامین منفی) مرتبط است، تنظیم می‌کند.

- **پاسخ‌های متفاوت دوپامین:** این مدل بین سیگنال‌های دوپامین مثبت و منفی، تمایز قائل می‌شود که وزن‌های سیناپسی را بر این اساس تعديل می‌کنند. این تنظیم دوگانه به مدل اجازه می‌دهد تا نقاط قوت سیناپسی را بر اساس نتایج خاص اقدامات تنظیم کند.

پاداش دادن به وزن‌ها نیز میتواند با روش‌های مختلفی انجام‌پذیرد. به عنوان مثال، ورودی نوع A به شبکه ورودی داده می‌شود و انتظار داریم نورون خروجی شماره ۱ آن را تشخیص دهد. این تشخیص دادن را می‌توان به دو صورت در نظر گرفت. یک روش آن که نورونی که نورونی که زودتر ضربه می‌زند، این تشخیص را انجام داده است، روش دیگر آنکه نورونی که در پنجه زمانی ورودی دادن بیشترین نرخ ضربه را داشته است، آن را تشخیص داده است. بسته به اینکه کدام روش انتخاب شود، به وزن‌های هر یک از نورون‌ها میتوانیم دوپامین ها را اعمال کنیم. من بدليل آنکه ورودی‌ها نیز به صورت نرخ ضربه زدن کدگذاری شده بودند، از روش دوم برای تشخیص دادن نیز استفاده کردم.

۱۰۳۰ آزمایش‌ها

شباهت کسینوسی: دقت شود که نمودار شباهت کسینوسی در تمام شکل‌ها آمده است.
حال به سرعت آزمایش کردن مدلمان با ورودی و پارامترهای مختلف می‌رویم. اولین آزمایشی که مطابق معمول انجام میدهیم، ورودی دادن اندازه‌های مختلف ورودی است.

تأثیر اندازه

اولین ورودی، ۲ الگوی ورودی از جنس یک آرایه از اعداد است. همانطور که در شکل ۲۷ نیز مشاهده می‌شود، مدل به خوبی توانسته است الگوهای ورودی را یاد بگیرد. ورودی در این مدل، شامل دو آرایه تصادفی است که مقادیر یکی، دو برابر دیگری است. مقادیر حال دو الگوی ۲۸A و ۲۸B را به عنوان ورودی به نورون‌ها ورودی داده شده که اینکه نورونی که آیا شبکه از عهده یادگرفتن دو الگوی ساده بر می‌آید یا خیر. همانطور که از شکل ۲۹ نیز بر می‌آید، نورون‌ها در یادگیری الگوهای ذکر شده موفق بوده‌اند.

حال ورودی‌ها را به طوری که در صورت پروژه گفته شده بود به شبکه میدهیم، یعنی بعضی نورون‌ها، هر دو الگو را به عنوان ورودی دریافت میکنند. نورون‌های سیز در شکل (۳۰)

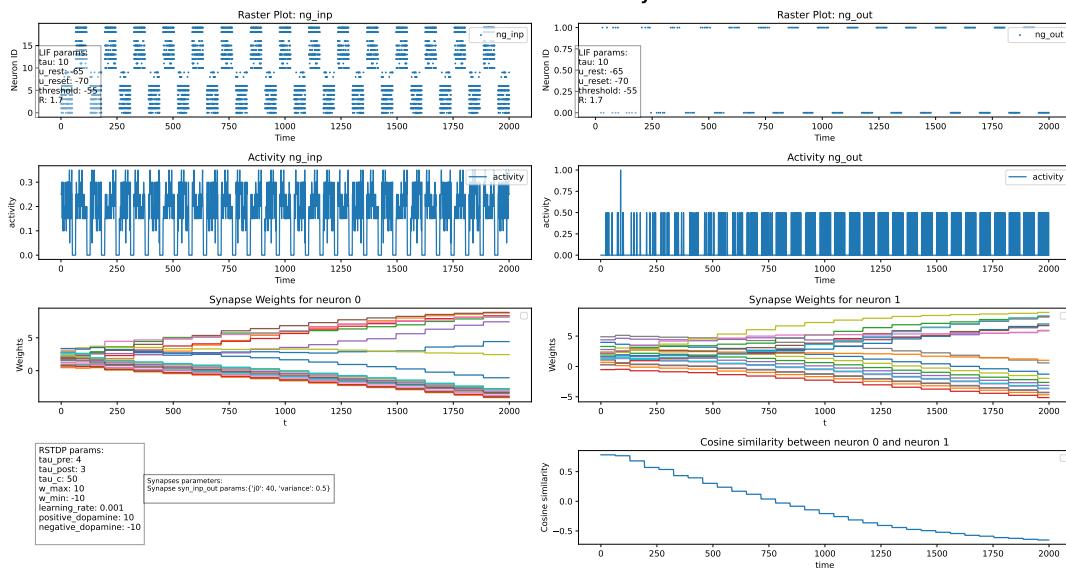
مطابق شکل ۳۱ مشاهده میکنیم که هم پوشانی نورون‌های ورودی در دریافت محرک، باعث می‌شود که شبکه کمی دیرتر الگوهای را یاد بگیرند. حتی اگر میزان همپوشانی را به حداقل برسانیم (به طوری که همه محرک‌ها به همه نورون‌ها داده شود) باز هم شبکه قادر به تشخیص الگوهای خواهد بود. (شکل ۳۲)

حال در نهایت یک قدم رفتار گذاشته، و دو تصویر با اندازه‌های 10×10 را به عنوان ورودی به مدل میدهیم. (تصاویر ۱۳۳ و ۱۳۴)

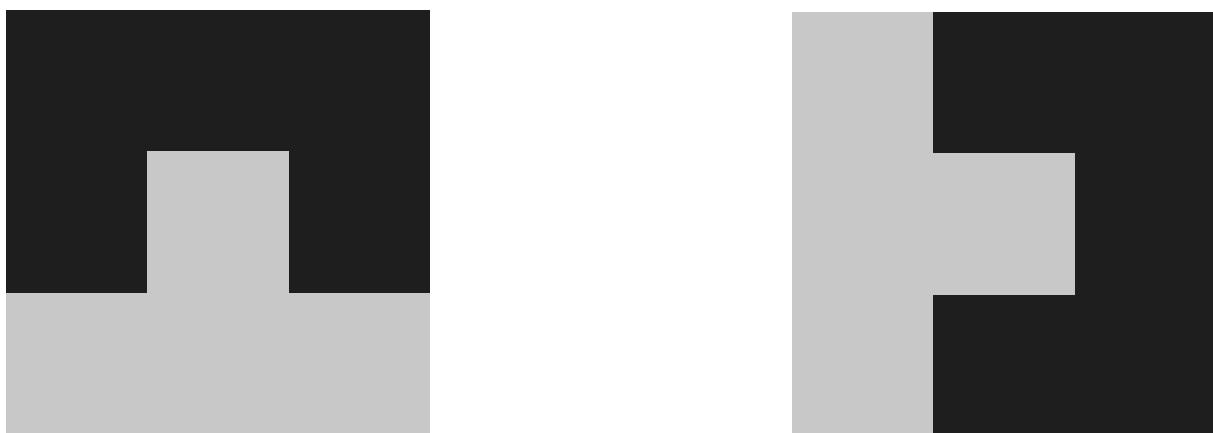
ممکن است انتظار داشته باشیم با زیاد شدن ابعاد ورودی، تشخیص الگوهای نیز سخت تر شود، اما شکل ۳۴ این فرض را رد میکند و مشاهده میکنیم که یادگیری دو الگو به خوبی انجام شده است.

مدل نسبت به هم پوشانی نورون‌ها برای گرفتن ورودی نیز پایدار است و همچنان الگو را یاد می‌گیرد. (شکل ۳۵) حتی با بیشتر کردن هم پوشانی نیز باز هم مدل می‌تواند تصاویر را یاد بگیرد. (شکل ۳۶) هر چند نسبت به حالت‌های قبلی این یادگیری دیرتر اتفاق می‌افتد، اما با تنظیم پارامترهایی مانند پنجه زمانی یا دوپامین که در ادامه بررسی می‌شوند می‌توان این یادگیری را سریع تر کرد.

RSTDP Rule: small-array

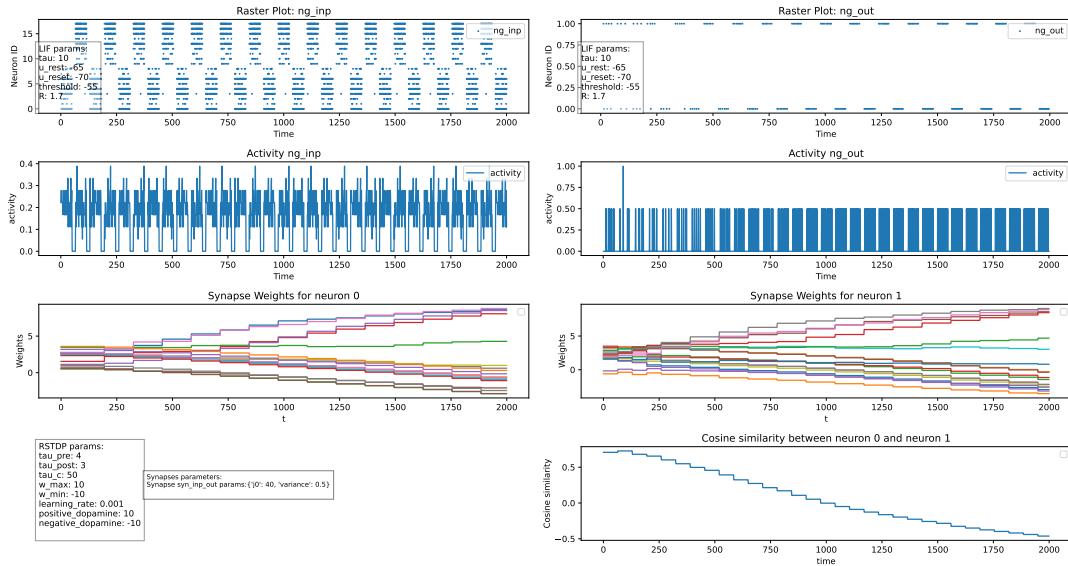


شکل ۲۷: همانطور که در شکل مشاهده میکنیم، مدل پس از حدود ۵۰۰ مرحله توانسته است الگوهای ورودی را یاد بگیرد. با گذشتן مراحل بیشتر نیز، نورون‌ها ورودی‌ها را بیشتر یاد گرفته به طوری که فعالیت کلی آن‌ها در طول بازه ورودی خود، بیشتر می‌شود. همانطور که هر نورون در لایه خروجی یاد می‌گیرد که به ویژگی‌های خاص مرتبط با یک کلاس خاص به شدت پاسخ دهد، بردارهای وزن آن‌ها از هم جدا می‌شوند. این واگرایی منجر به کاهش شباهت کسینوسی می‌شود، زیرا بردارهای وزنی با تطبیق خود برای تشخیص ویژگی‌های متمایز کلاس‌های مختلف، موازی‌تر می‌شوند.

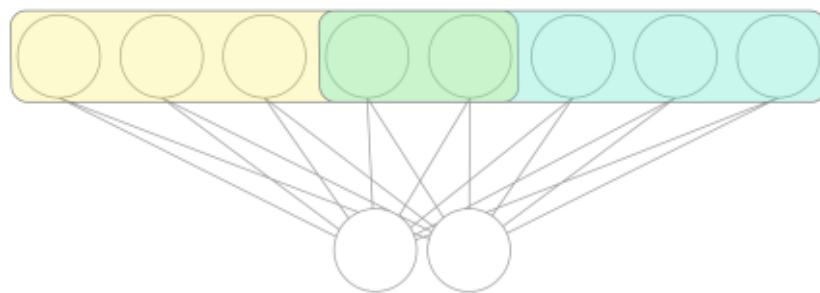


شکل ۲۸: دو تصویر به عنوان محرک

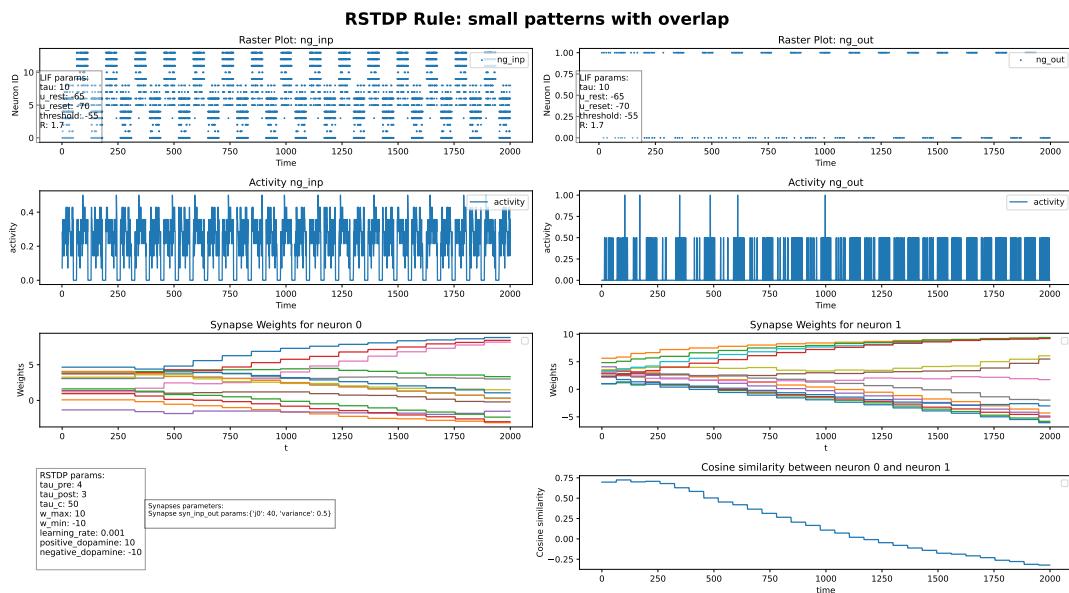
RSTDP Rule: small-patterns



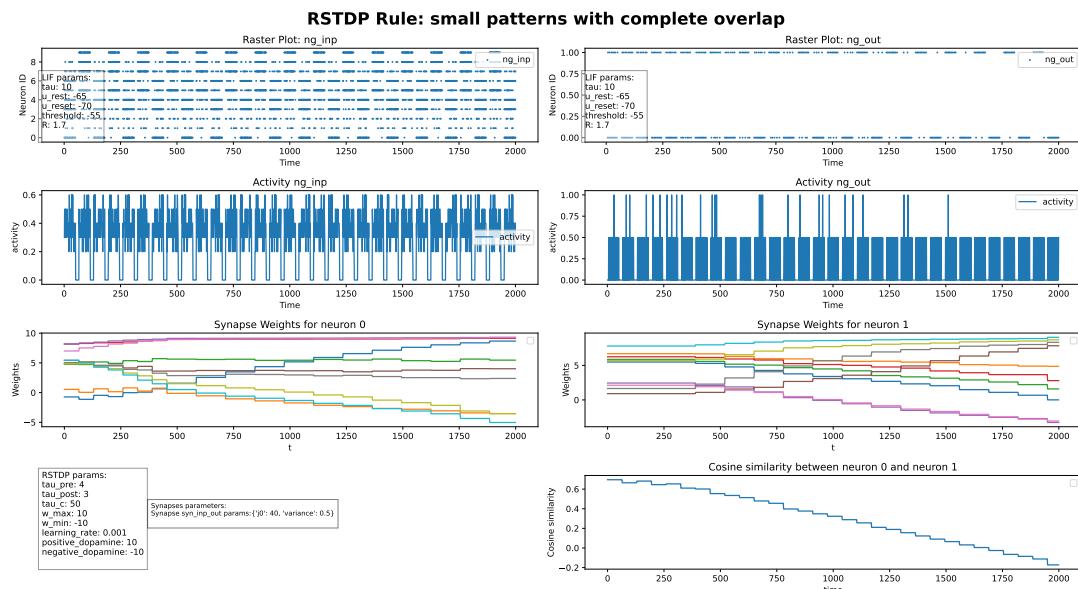
شکل ۲۹: قانون یادگیری RSTDP برای دو الگوی ساده. مطابق نمودار بالا دریافت می شود که نورون ها با استفاده از قانون یادگیری RSTDP توانسته اند الگو های آآ۲۸ و ۲۸ ب را به خوبی یاد بگیرند. به طوری که این یادگیری تا ۵۰۰ مرحله یا به عبارتی با ۳ بار ورودی دادن هر الگو، انجام شده است که نکته قابل توجهی است. تا مرحله ۱۵۰۰ ام شبیه سازی نیز این یادگیری عمیق تر شده و نورون ها فقط در زمانی که الگوی خود را میبینند ضربه می زنند.



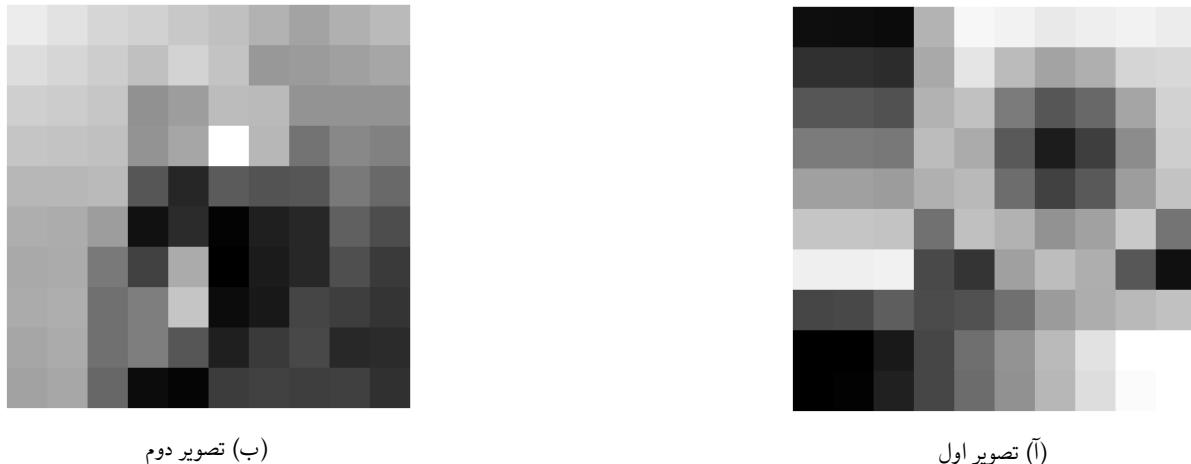
شکل ۳۰: نورون هایی که با رنگ سبز مشخص شده اند، هر دو الگو را به عنوان ورودی دریافت میکنند.



شکل ۳۱: هم پوشانی نورون ها در دریافت ورودی. مطابق نمودار های بالا، اختصاص بعضی نورون ها به دریافت دو نوع الگو، باعث می شود که شبکه کمی دیرتر الگو ها را یاد بگیرد. به طوری که در مقایسه با شکل ۲۹ یادگیری در ۳۰۰ مرحله بعد انجام شده است. هر چند هر دو مدل در نهایت الگو های ورودی را فراگرفته اند.



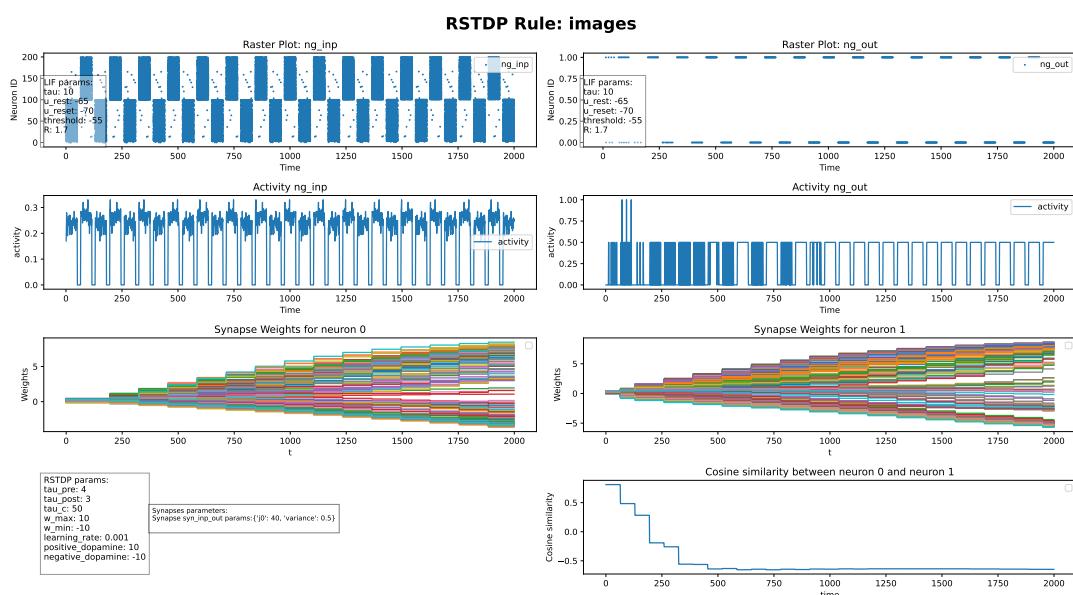
شکل ۳۲: هم پوشانی جداکننده نورون ها در دریافت ورودی. مشاهده میکنیم که حتی با هم پوشانی کامل نیز، شبکه قادر به یادگیری الگو ها می باشد، هر چند بعضی اوقات نورون ها هنگامی که ورودی دیگر را میبینند نیز ضربه میزنند که اینکار با تنظیم پارامتر ها قابل بهبود است و در بخش بعدی به آن می پردازیم.



شکل ۳۳: یک تصویر به عنوان محرک

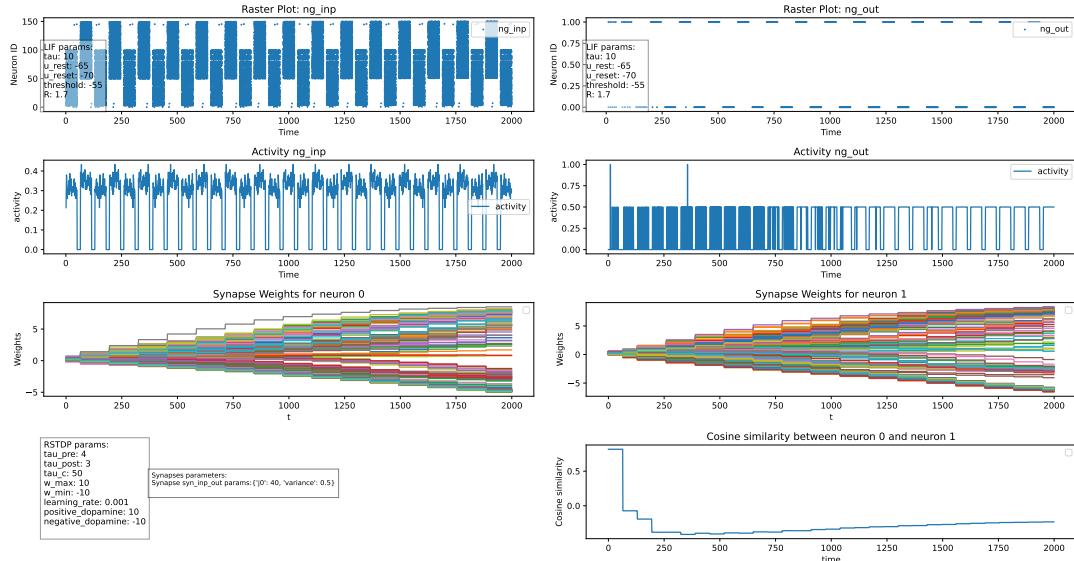
(b) تصویر دوم

(l) تصویر اول



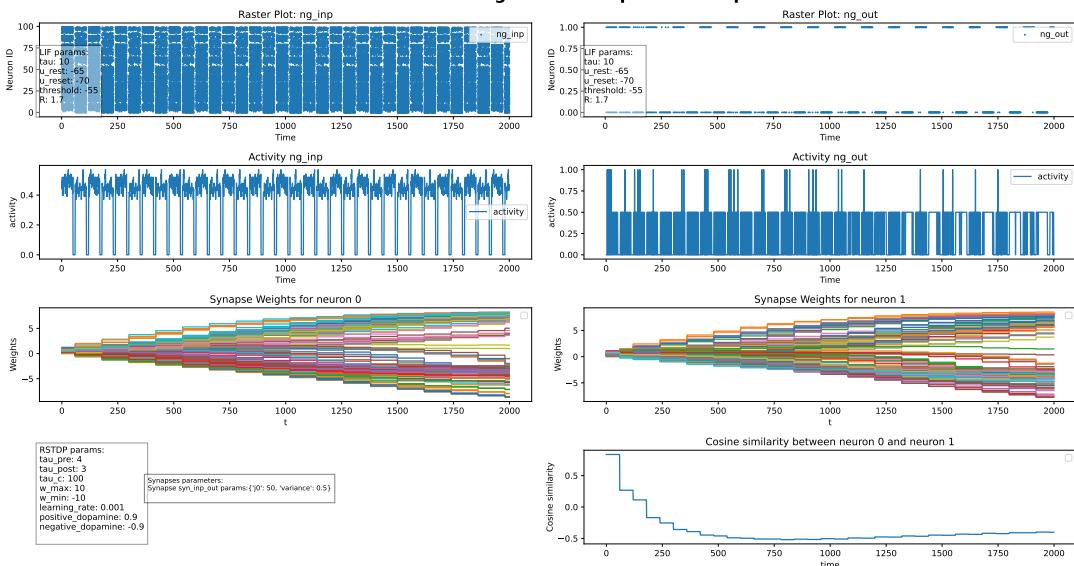
شکل ۳۴: یادگیری الگوی دو تصویر توسط قانون یادگیری *RSTDP*. مطابق شکل بالا مشاهده میکنیم که با وجود ابعاد بالای الگوی ورودی نیز، یادگیری هنوز به خوبی انجام می شود. نکته قابل توجه دیگری که از این شکل برداشت می شود نیز این است که پس از حدود 150° مرحله از شبیه سازی، نمودار فعالیت نورون ها ثابت تر می شود، بدان معنا که بین تکرار های 100° و 150° فعالیت نورون ها در بعضی لحظه ها ۵۰% و در بعضی لحظه ها کمتر است اما بعد از تکرار 150° ، هر گاه که ورودی مربوط به یک الگو داده می شود، در تمام آن پنجره زمانی، نورون های مربوط ضربه میزنند و پیوسته فعالیت کامل دارند.

RSTDP Rule: images with overlap



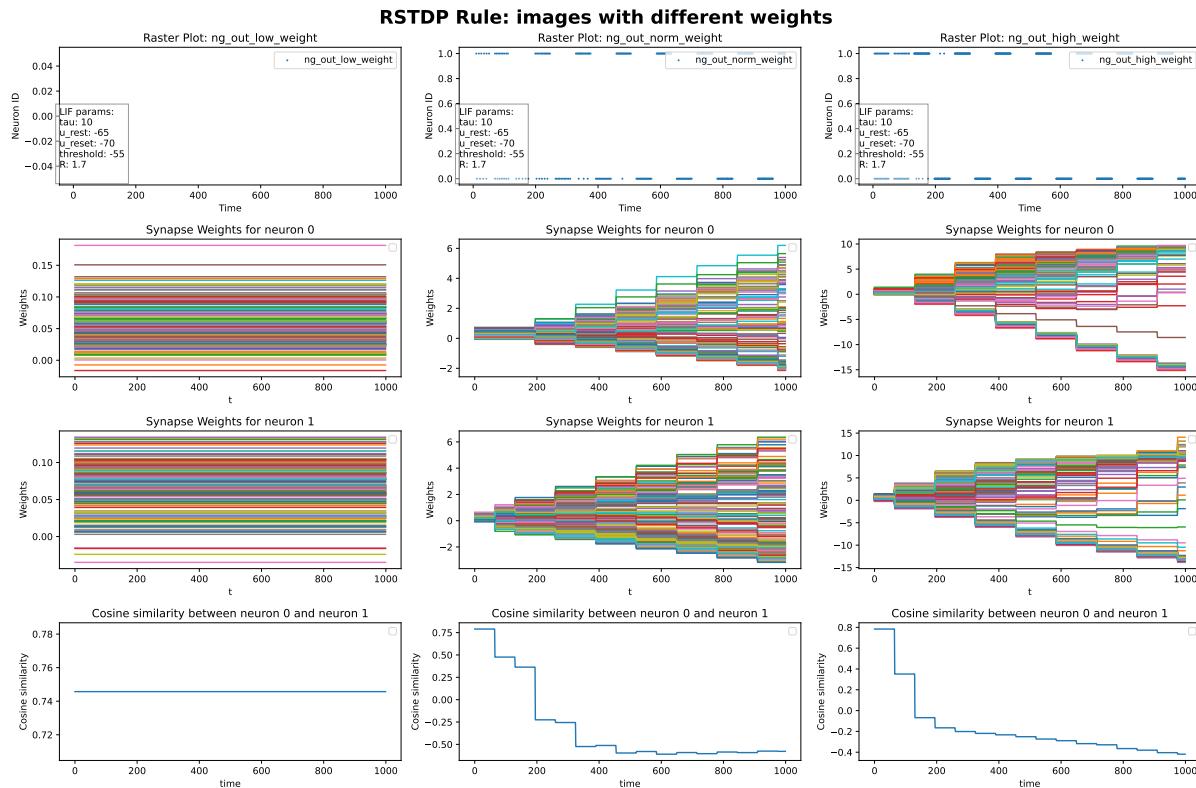
شکل ۳۵: یادگیری الگوی دو تصویر توسط قانون یادگیری *RSTDP* با هم پوشانی نورون‌ها. مطابق شکل ملاحظه می‌کنیم که هم پوشانی نورون‌ها در دریافت ورودی نیز نمیتواند از یادگیری الگو‌ها توسط مدل جلوگیری کند. هرچند این اتفاق کمی دیرتر می‌افتد که به این دلیل است بعضی نورون‌ها باید هر دوی الگو‌ها را یاد بگیرند. (تجویه کنید که اکنون جمعیت ورودی شامل ۱۵۰ نورون است در حالی که الگو‌ها 10° پیکسل دارند و از آنجا که بیشترین میزان فعالیت نورون‌ها بعد از یادگیری ۵۰٪ است، ۷۵ نورون باید 10° پیکسل را فراگیرند. به عبارتی دیگر، بعضی نورون‌ها هر دو الگو را یاد می‌گیرند.)

RSTDP Rule: images with complete overlap



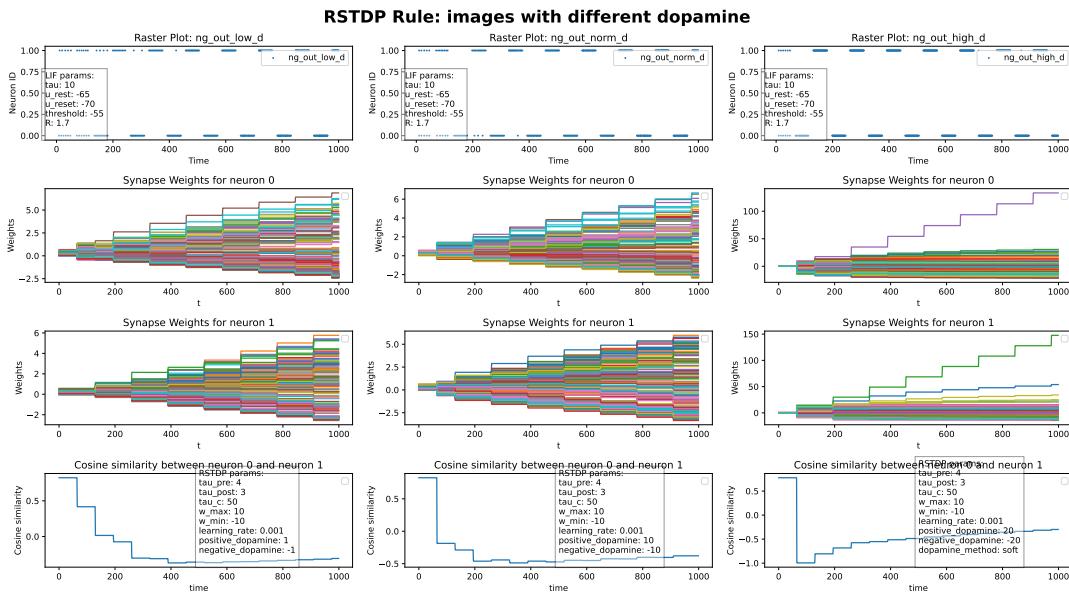
شکل ۳۶: یادگیری الگوی دو تصویر توسط قانون یادگیری *RSTDP* با هم پوشانی کامل نورون‌ها. همانطور که در نمودار بالا نیز ملاحظه می‌شود، حتی با هم پوشانی کامل نورون‌ها نیز دو الگو از یکیگر تشخیص داده می‌شوند. نکته ایکه وجود دارد این است که این یادگیری نسبت با حالت‌های با هم پوشانی کمتر، دیرتر اتفاق می‌افتد.

در گذشته در شبکه های عصبی کلاسیک، وزن های اولیه ممکن بود تاثیر زیادی در یادگیری شیکه بگذارند. در این بخش ما یادگیری مدلمان را روی دو الگوی ورودی، برای وزن های متفاوت آزمایش میکنیم. بدیلیل زیاد بودن نمودارها، فقط نمودار های متفاوت و مهم را در این قسمت می آوریم. مطابق شکل ۳۷ مشاهده میکنیم که مقادیر وزن های اولیه می توانند در روند یادگیری مدل تاثیر بگذارد.



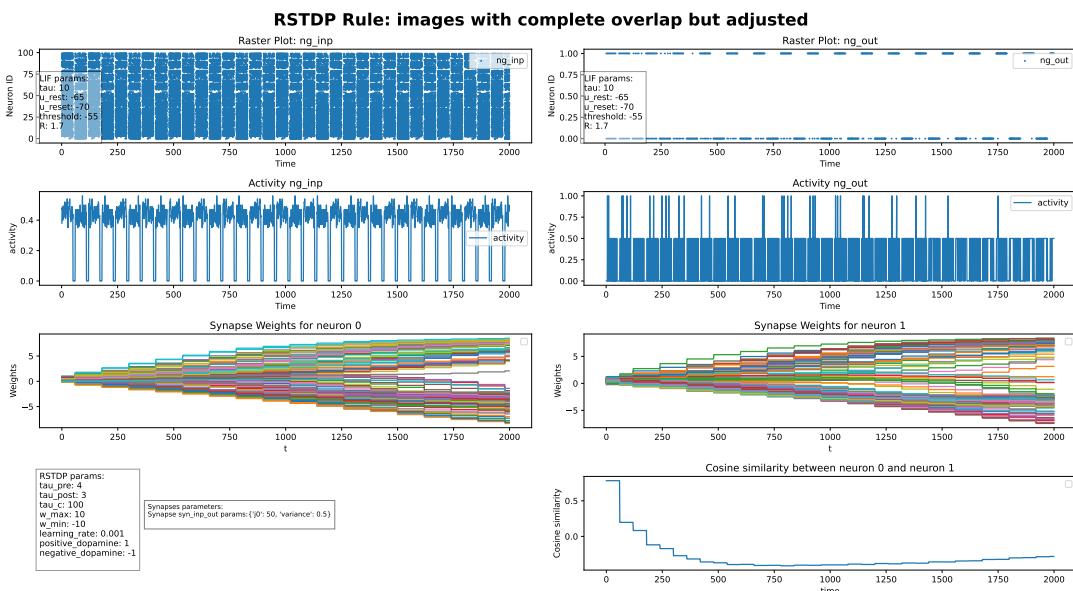
شکل ۳۷: تاثیر وزن های اولیه بر یادگیری با قانون $RSTDP$. مطابق نمودار بالا مشاهده میکنیم که وزن های اولیه میتوانند بر یادگیری اولیه مدل تاثیر بگذارد. به طور که انتخاب وزن های کم باعث می شود که در طول شبیه سازی نورون ها ضربه ای نزنند و یادگیری اتفاق نیوفتد، و انتخاب وزن های اولیه بزرگ نیز ممکن است باعث شود که نرخ ضربه زدن نورون ها بیش از حد شود و یادگیری به درستی انجام نشود.

میتوان گفت دوپامین مهم ترین تفاوت $RSTDP$ با $STDP$ است. از این رو انتظار داریم که تغییر در این پارامتر بتواند رفتار مدل را نیز تغییر دهد. مطابق شکل ۳۸ مشاهده میکنیم که تغییر در میزان دوپامین، میتواند وزن های سیناپسی و سرعت یادگیری را تغییر دهد. هر چند در هر سه مدل، در نهایت یادگیری انجام شده است.



شکل ۳۸: تاثیر میزان دوپامین بر یادگیری با قانون $RSTDP$. مطابق نمودار بالا مشاهده میکنیم که تغییر در میزان دوپامین می تواند سرعت یادگیری را تغییر دهد. به عنوان مثال، افزایش دوپامین توانسته است به ترتیب از چ به راست یادگیری را در مراحل زودتری تمام کند. نکته دیگر این است که افزایش دوپامین، باعث شده است که بعضی وزن های سیناپسی نیز از حد خود فراتر رفته و بسیار بزرگ شوند.

حال که تاثیر انواع پارامتر های موثر در مدل را بررسی کردیم میتوانیم یک مدل خوب برای یادگیری دو الگوی متفاوت (تصاویر ۳۳ و ۳۴) را به عنوان ورودی به مدل بدهیم تا به خوبی آن ها را یاد بگیرد. این مدل در شکل ۳۹ آمده است.

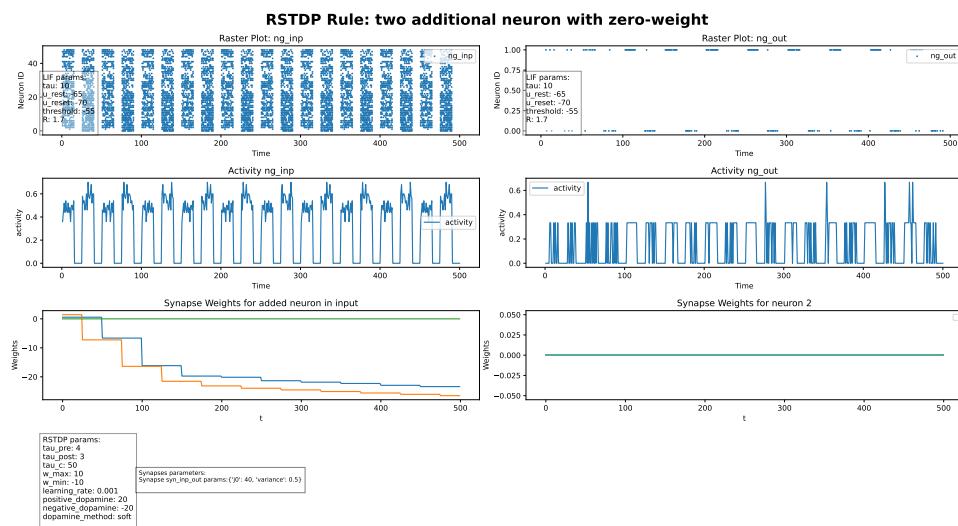


شکل ۳۹: یادگیری صحیح دو الگوی تصویر توسط قانون یادگیری $RSTDP$. همانطور که ملاحظه می شود، پس از حدود ۵۰۰ مرحله شبیه سازی مدل به خوبی الگو ها را یادگرفته است.

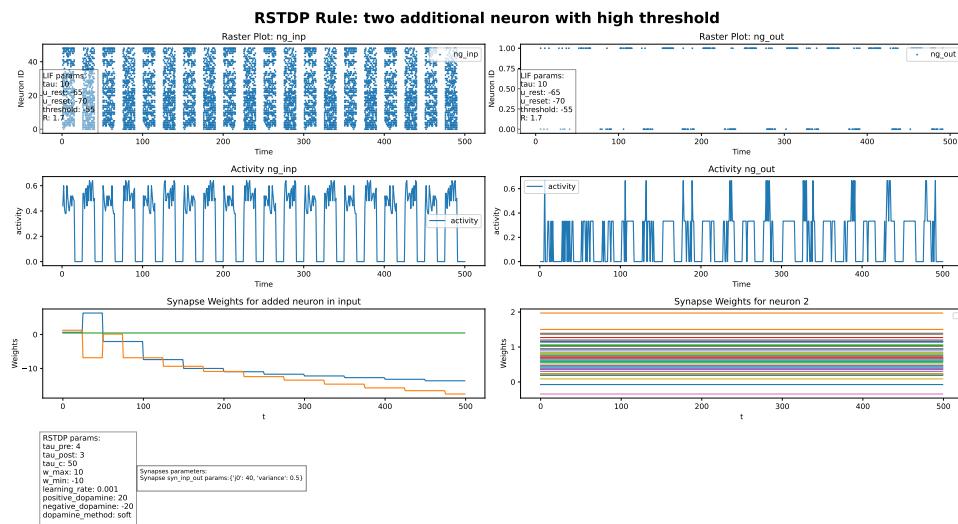
۲.۳.۰ اضافه کردن دو نورون غیر فعال

در این قسمت به هر یک از دو لایه ورودی و خروجی یک نورون اضافه میکنیم که در طول آموزش ضربهای نزنند و سپس وزن های متصل به این دو نورون را تحلیل میکنیم. از آنجاکه توضیحات زیادی در این باره در فایل پژوهه داده نشده است، ما فرض را بر این میگیریم که ضربه نزدن نورون خروجی، با صفر کردن وزن های متصل به آن اتفاق بیوفتد. روش دیگر برای اینکار، بالا بردن آستانه پتانسیل عمل آن نورون است. من هردو روش را آزمایش میکنم و نتایج را می آورم. برای روش اول، که جلوگیری از ضربه زدن نورون توسط صفر کردن وزن سیناپسی اتفاق می افتد، مطابق شکل ۴۰ آنالوگه میکنیم که اضافه کردن یک نورون به لایه ورودی و یک نورون به لایه خروجی که ضربه ای نزند، باعث می شود که یادگیری کمی با اختلال مواجه شود، هر چند هنوز دو نورون قبلی خروجی قادر به تشخیص الگو ها هستند، ولی فعالیت آن ها هنگام تشخیص دادن الگو های کمی کاهش یافته است.

حال جلوگیری از ضربه زدن نورون های اضافه شده را با استفاده از افزایش آستانه پتانسیل عمل انجام میدهیم. مجدداً مانند حالت قبل ملاحظه میکنیم که یادگیری با اختلال مواجه شده است، هر چند هنوز الگو ها تشخیص داده می شوند. (شکل ۴۰ ب)

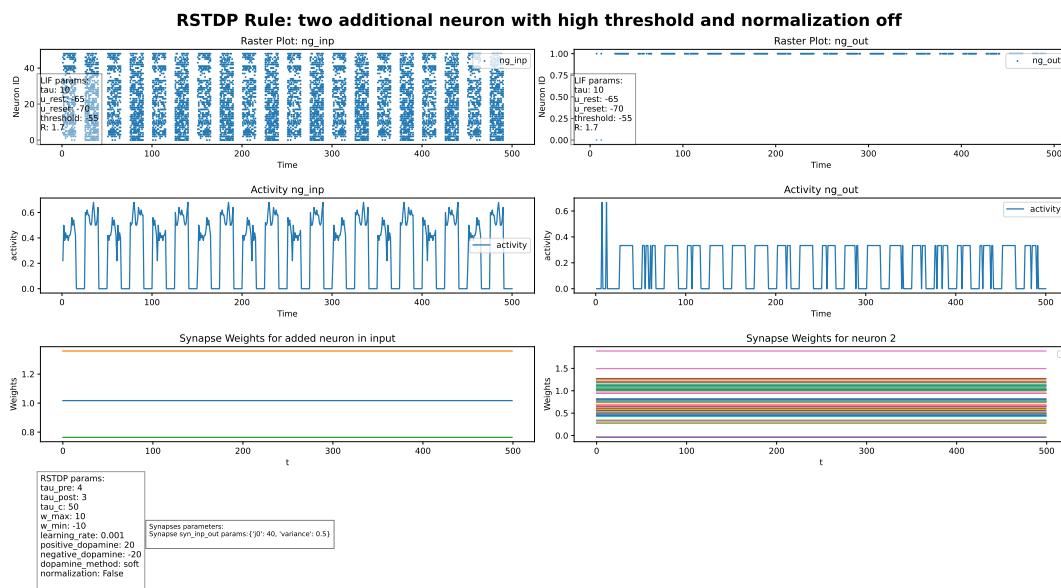


(آ) جلوگیری از ضربه زدن نورون های اضافه شده از طریق صفر کردن وزن ها. مشاهده میکنیم که اضافه کردن نورون به ۲ لایه که ضربه ای نمیزنند، میتواند در یادگیری مدل اختلال ایجاد کند. هر چند هنوز مدل می تواند تا حدی الگوها را تشخیص دهد. همچنین در نمودار وزن های ملاحظه میکنیم تمام وزن های نورون اضافه شده در خروجی تغییری نداشته اند.



(ب) جلوگیری از ضربه زدن نورون های اضافه شده از طریق افزایش آستانه پتانسیل عمل نورون اضافه شده. مشاهده میکنیم، مانند شکل قبل اضافه کردن نورون به ۲ لایه، میتواند در یادگیری مدل اختلال ایجاد کند. هر چند هنوز مدل می تواند تا حدی الگوها را تشخیص دهد. همچنین در نمودار وزن های ملاحظه میکنیم که یکی از وزن های متعلق به نورون اضافه شده در ورودی و همچنین تمام وزن های نورون اضافه شده در خروجی تغییری نداشته اند. تنها تفاوت با شکل قبل این است که وزن های نورون اضافه شده در خروجی صفر نیستند و همان مقادیر اولیه را دارند.

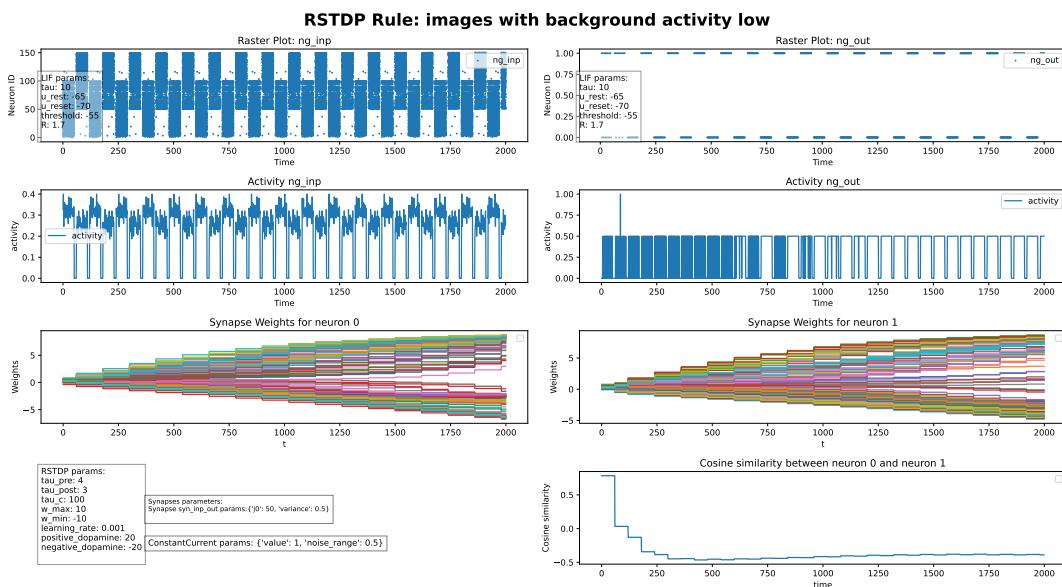
حال اگر وزن ها را نرمال سازی نکنیم، نمودار ها به صورت شکل ۴۱ تغییر خواهد کرد و یادگیری به طور کامل مختل خواهد شد.



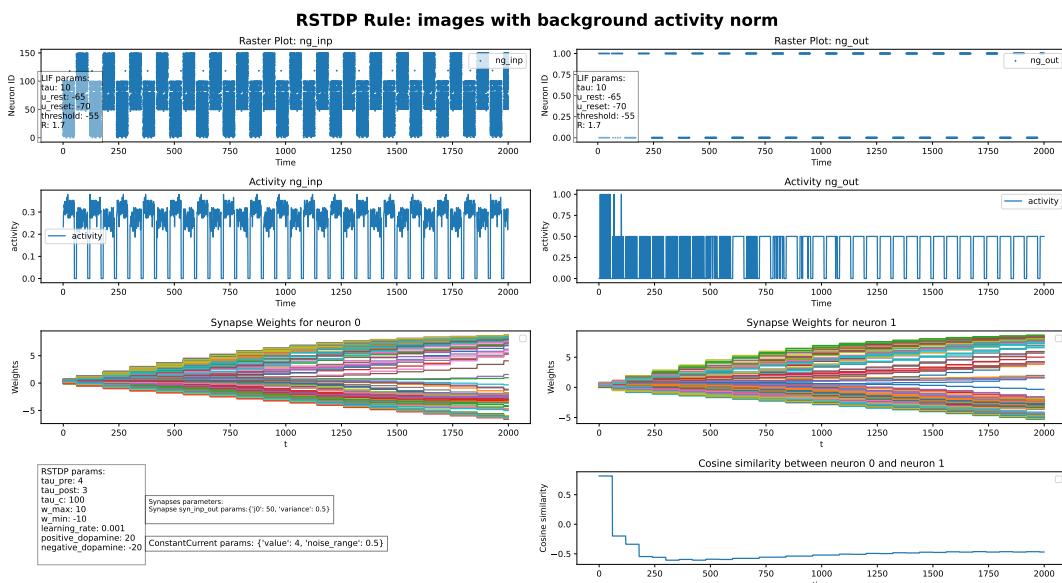
شکل ۴۱: جلوگیری از ضربه زدن نورون های اضافه شده از طریق افزایش آستانه پتانسیل عمل نورون اضافه شده و نرمال سازی خاموش. مشاهده میکنیم، برخلاف شکل های ۴۰آ و ۴۰ب با خاموش کردن نرمال سازی، یادگیری مدل به طور کامل مختل می شود.

۳.۳.۰ بررسی مدل با فعالیت کمینه برای لایه ها

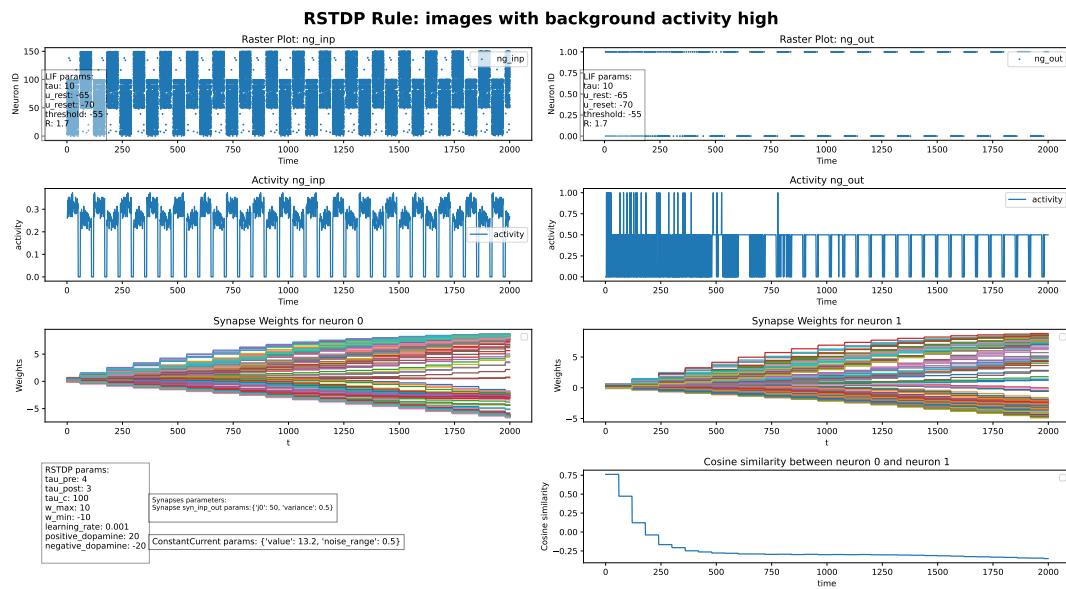
در آخرین آزمایش مربوط به این قسمت، به بررسی رفتار مدل هنگامی که یک فعالیت کمینه در دو لایه وجود دارد را بررسی میکنیم. برای اینکار، آزمایش را با سه مقدار متفاوت فعالیت زمینه انجام میدهیم. همچنین حالت پایه آزمایش را، الگوی تصاویر با هم پوشانی نسبی در نظر میگیریم. همانطور که از شکل ۴۲ دریافت می شود، به طور کلی افزودن یک جریان ثابت نویزی با، مانع یادگیری مدل نمی شود. حتی با انتخاب معقولی از فعالیت زمینه، ممکن است یادگیری مدل بهتر نیز بشود، (شکل ۴۳)



شکل ۴۲: مدل با فعالیت زمینه کم.



شکل ۴۳: مدل با فعالیت زمینه عادی.



شکل ۴۴: مدل با فعالیت زمینه زیاد. مشاهده میکنیم که تغییر در میزان فعالیت زمینه، مشکلی در یادگیری مدل در زمان طولانی ایجاد نمیکند. هرچند که داشتن فعالیت زمینه در اندازه عادی (شکل ۴۳) میتواند حتی باعث بهتر شدن یادگیری نیز بشود، اما زیاد از حد بودن فعالیت کمینه میتواند یادگیری را کمی به تأخیر بیندازد. در مقایسه با حالت قبل نیز میتوان گفت، که افزودن فعالیت کمینه دربرابر افزودن نوروون بدون ضربه، کمتر یادگیری را مختل میکند.

کتابنامه

- [۱] Computational Neuroscience Course, School of computer science, University of Tehran
- [۲] PymoNNtorchPytorch-adapted version of PymoNNto
- [۳] Neuronal Dynamics, Wulfram Gerstner, Werner M. Kistler, Richard Naud and Liam Paninski
- [۴] Poisson Distribution. Wikipedia [Link]
- [۵] Spike-timing-dependent plasticity. Wikipedia [Link]