گزارش پروژه چهارم علوم اعصاب محاسباتی

امیرحسین انتظاری

۲۰ خرداد ۱۴۰۳

فهرست مطالب

۱ ساز و کار های درون یک لایه نورونی ۱
۱.۱ مقدمه ۱
۱.۱.۱ مجموعه داده مورد استفاده ۱
۲.۱.۱ یک مدل ساده ۱
۲.۱ افزودن ساز و کار مهار جانبی (Lateral Inhibition) ۳
۱.۲.۱ آزمایش مدل با اشتراک متفاوت الگو ها ۵
۲.۲.۱ نتایج ۵
۳.۱ k ـ برنده همه چیز را می‌گیرند(K-Winners-Take-All) ۷
۱.۳.۱ آزمایش مدل با اشتراک متفاوت الگو ها ۸

۲ هم‌ایستایی یا هومئوستازی (Homeostasis) ۱۲
۱.۲ ساز و کار هم‌ایستایی ۱۲
۲.۲ ساز و کار هم‌ایستایی براساس فعالیت ۱۲
۱.۲.۲ افزایش تعداد الگو ها ۱۲
۲.۲.۲ نتایج ۱۲
۳.۲ ساز و کار هومئوستازی براساس ولتاژ (امتیازی) ۱۶
۴.۲ (امتیازی) آزمایش با نسبت های مختلف الگو ها و لایه خروجی ۱۸

چکیده

هدف از این پروژه، آشنایی با ساز و کار های موجود بین نورون های یک لایه و تاثیر آن ها بر فرآیند یادگیری است. در این پروژه قصد داریم سه ساز و کار مهار جانبی [۱]، برنده–همه–چیز–را–می‌گیرد [۲] و هم‌ایستایی [۳] را بررسی کنیم. در اکثر آزمایش های این پروژه، از قانون یادگیری انعطاف‌پذیری وابسته به زمان ضربه ($STDP$[۴]) استفاده می‌کنیم. همچنین بررسی می‌کنیم انواع الگو ها از نظر تعداد و پیچیدگی آن ها چه تاثیری بر فرایند یادگیری می‌گذارد. در ادامه پارامتر های مختلف و تاثیر آن ها را نیز آزمایش خواهیم کرد و خواهیم دید که اضافه کردن هر یک از این ساز و کار ها به پروژه چه تاثیری در فرایند یادگیری مدل خواهد گذاشت.

---

۱ Lateral inhibition
۲ k-winners-take-all
۳ Homeostasis
۴ Spike-Timing-Dependent Plasticity

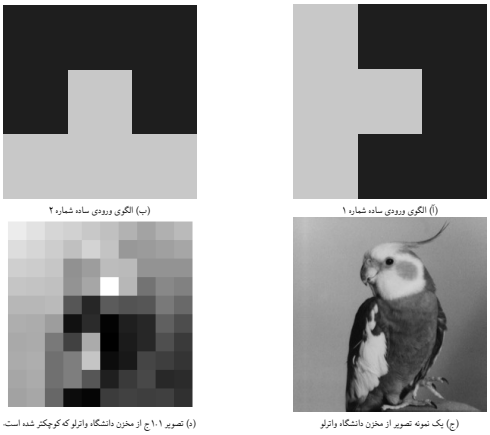
بخش ۱

ساز و کار های درون یک لایه نورونی

۱.۱ مقدمه

۱.۱.۱ مجموعه داده مورد استفاده

در این پروژه، برای الگو های ساده، از الگوی ۱.۱آ و ۱.۱ب استفاده می‌کنیم و برای الگو های پیچیده تر از از مخزن دانشگاه واترلو [۱] با نسبت اندازه های مختلف 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. کدگذاری مورد استفاده در این پروژه نیز، کد گذاری پواسون می‌باشد.



(أ) الگوی ورودی ساده شماره ۱

(ب) الگوی ورودی ساده شماره ۲

(ج) یک نمونه تصویر از مخزن دانشگاه واترلو

(د) تصویر ۱.۱ج از مخزن دانشگاه واترلو که کوچکتر شده است.

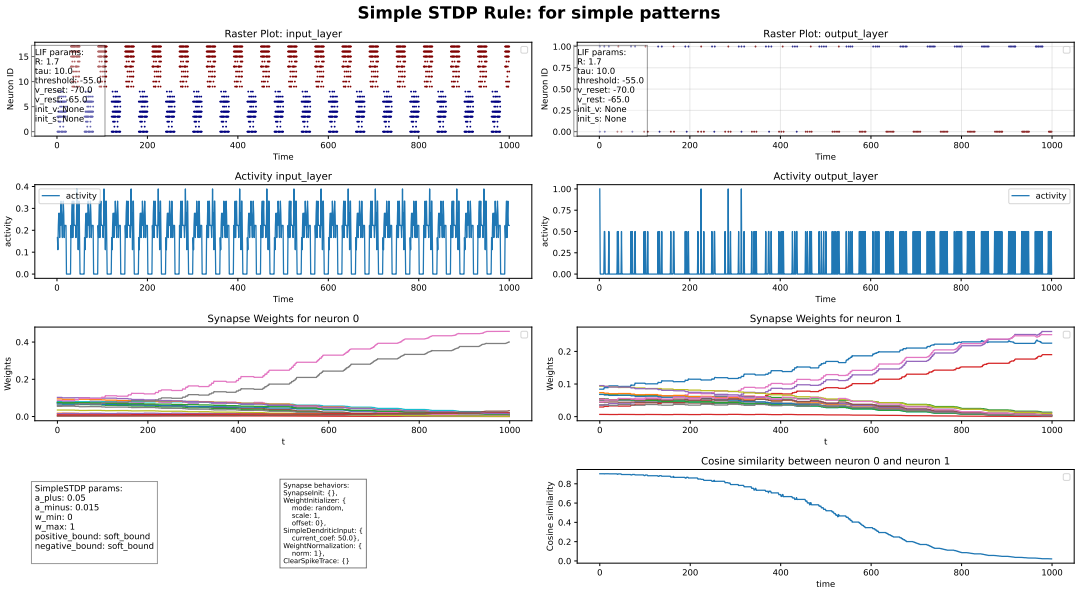
شکل ۱.۱: مجموعه داده مورد استفاده

۲.۱.۱ یک مدل ساده

در این پروژه می‌خواهیم با ساز و کار های موجود بین نورون های یک لایه آشنا شویم و تاثیر آن ها را بر فرآیند یادگیری بررسی کنیم. برای اینکار لازم است ابتدا، یک آزمایش ساده برای حالتی که هیچ یک از این ساز و کار ها درون لایه ها وجود ندارد انجام دهیم تا نتایج را بهتر باآن مقایسه کنیم. همانطور که در فایل پروژه نیز آمده است، در شبکه مورد نظر، تنها یک لایه ورودی و یک لایه خروجی در نظر میگیریم. سپس یک سیناپس با اتصال کامل و وزن های اولیه تصادفی بین این دو لایه ایجاد می‌کنیم. همچنین از قانون یادگیری انعطاف‌پذیری وابسته به زمان ضربه ($STDP$[۲]) برای آموزش شبکه استفاده می‌کنیم. از آنجا که این آزمایش به عنوان یک مرجع برای مقایسه در نظر گرفته می‌شود، رفتار دندریت ها را نیز یک رفتار ساده در نظر میگیریم. در نهایت، شبیه سازی را برای ۱۰۰۰ تکرار انجام می‌دهیم. همانطور که پروژه قبل نیز ملاحظه کردیم، قانون یادگیری $STDP$ به تنهایی نمی‌تواند در یادگیری الگو ها به خوبی $RSTDP$ عمل کند. هر چند که این قانون، مطابق شکل ۲.۱ توانسته است الگو های ورودی را به خوبی یاد بگیرد، اما این یادگیری پایدار نیست و به ازای تکرار ها مختلف ممکن است به طرق متفاوت عمل کند. (شکل ۳.۱)

[۱] مخزن تصاویر دانشگاه واترلو

[۲] Spike-Timing-Dependent Plasticity

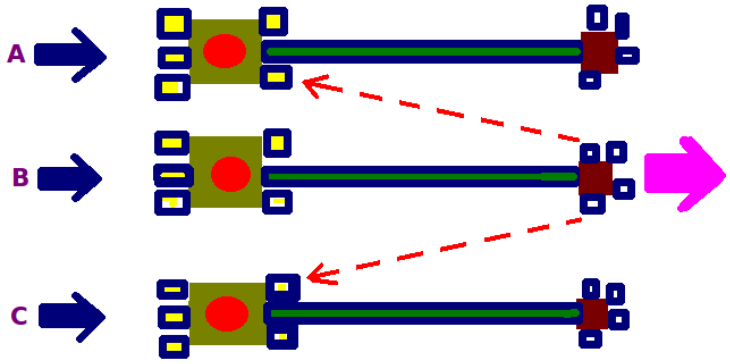


شکل ۲.۱: **یک شبکه ساده با قانون یادگیری $STDP$ ساده و بدون ساز و کار اضافه.** برای درک بهتر و تحلیل ساز و کار های درون یک لایه، بهتر است یک شبکه ساده در نظر گرفته و رفتار آن را با شبکه هایی که ساز و کار به آن ها اضافه می‌شود مقایسه کنم. همانطور که در پروژه قبل دیدیم، قانون یادگیری $STDP$ می‌تواند در یادگیری الگو های ساده، تا حد قابل قبولی عمل کند. هر چند این عملکرد ممکن است پایدار نباشد.(شکل ۳.۱)

شکل ۳.۱: **مقایسه نتایج شبیه سازی های مختلف شبکه تنها با قانون یادگیری $STDP$.** همانطور که در شکل بالا نیز مشاهده می‌کنیم، اجرای شبیه سازی یک شبکه یکسان که تنها قانون یادگیری $STDP$ را دارد، می‌تواند منجر به نتایج متفاوت شود. در شکل بالا، در شبیه سازی اول، شبکه نتوانسته است به خوبی الگو ها را یاد بگیرد، اما شبیه سازی های دوم و سوم عملکرد خوبی داشته‌اند.

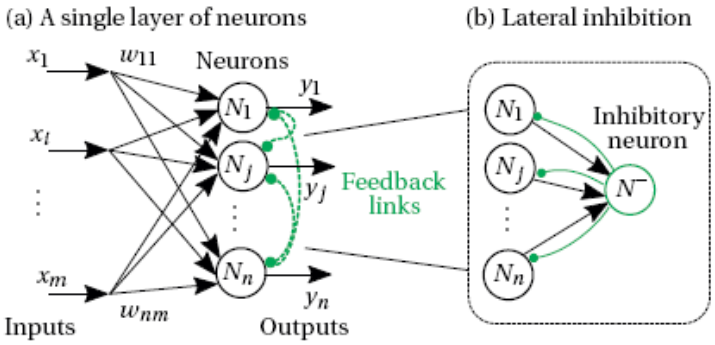
۲.۱ افزودن ساز و کار مهار جانبی (Lateral Inhibition)

در علوم اعصاب، مهار یا بازداری جانبی ظرفیت یک نورون برانگیخته برای کاهش فعالیت همسایگان خود است. مهار جانبی گسترش پتانسیل های عمل را از نورون های برانگیخته به نورون های همسایه در جهت جانبی غیرفعال می کند. این عمل یک کنتراست در تحریک ایجاد می کند که باعث افزایش ادراک حسی می شود و به آن تضاد جانبی نیز گفته می شود و عمدتاً در فرآیندهای بینایی، بلکه در پردازش لمسی، شنوایی و حتی بویایی نیز رخ می دهد. [۴]



شکل ۴.۱: محرکی که بر هر سه نورون تأثیر می‌گذارد، اما بر B قوی‌ترین یا اول تأثیر می‌گذارد، اگر B سیگنال‌های جانبی را به همسایگان A و C ارسال کند تا ضربه نزنند، در نتیجه آنها را مهار می‌کند. مهار جانبی در بینایی برای $sharp$ کردن سیگنال‌های مغز (فلش صورتی) استفاده می‌شود.

برای شبیه سازی ساز و کار مهار جانبی، ممکن است بتوان به چندین روش عمل کرد. از آنجا که می‌خواهیم یک نورون برانگیخته همسایگان خود را مهار کند، می‌توان اینکار را از طریق ایجاد یک جریان منفی از سمت این نورون به نورون های همسایه یا پایین آوردن اختلاف پتانسیل آن ها این کار را انجام داد. روشی که درون این پروژه استفاده شده است، همان روشی است که در کتابخانه $CoNeX$ نیز استفاده شده است. شمای کلی این روش، در شکل ۵.۱ که از [۵] برداشته شده، آمده است.

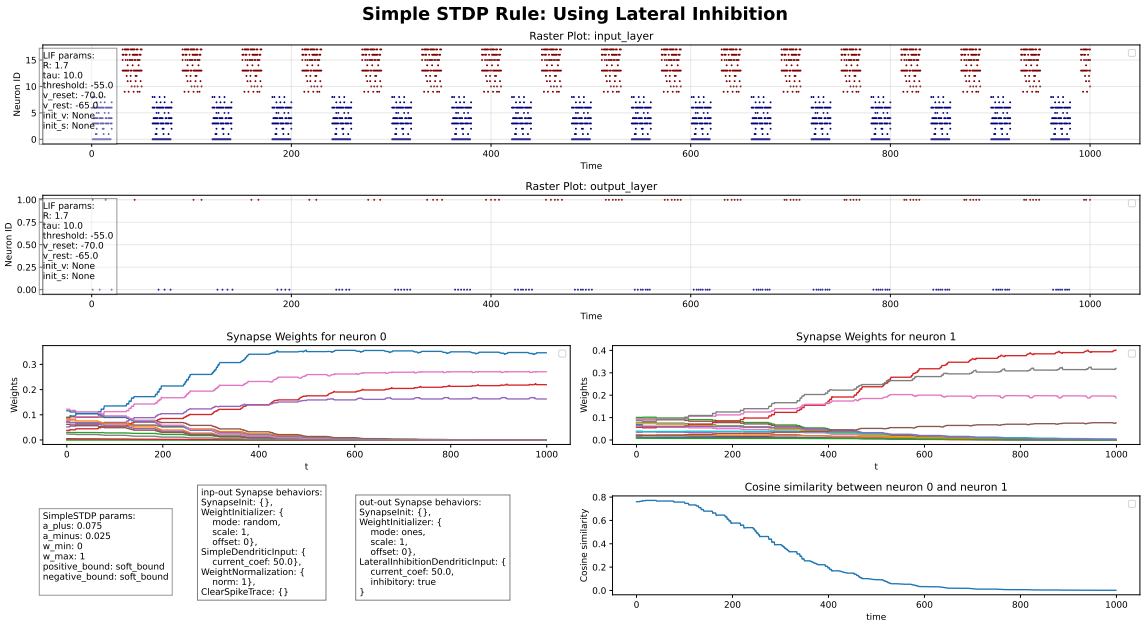


شکل ۵.۱: **یک شبکه SNN تک لایه با پیوندهای بازخورد** (الف) نورون های تحریکی $N1...Nn$ با پیوندهای بازخوردی در ارتباط هستند که فعالیت های واگرا را در نورون ها تقویت می کند. (ب) پیوندهای بازخورد اغلب ساز و کار مهار جانبی را اجرا می کنند، که در آن یک نورون بازدارنده اضافی N- نورون های تحریکی را مهار می کند. (رجوع کنید به [۵])

حال که با نحوه ساز و کار مهار جانبی آشنا شدیم، آن را به مدلمان افزوده و آزمایش می‌کنیم. برای اینکار لازم است که یک سیناپس جدید، از لایه خروجی به خودش تشکیل و دهیم، و به آن رفتار مهار جانبی را اضافه کنیم.

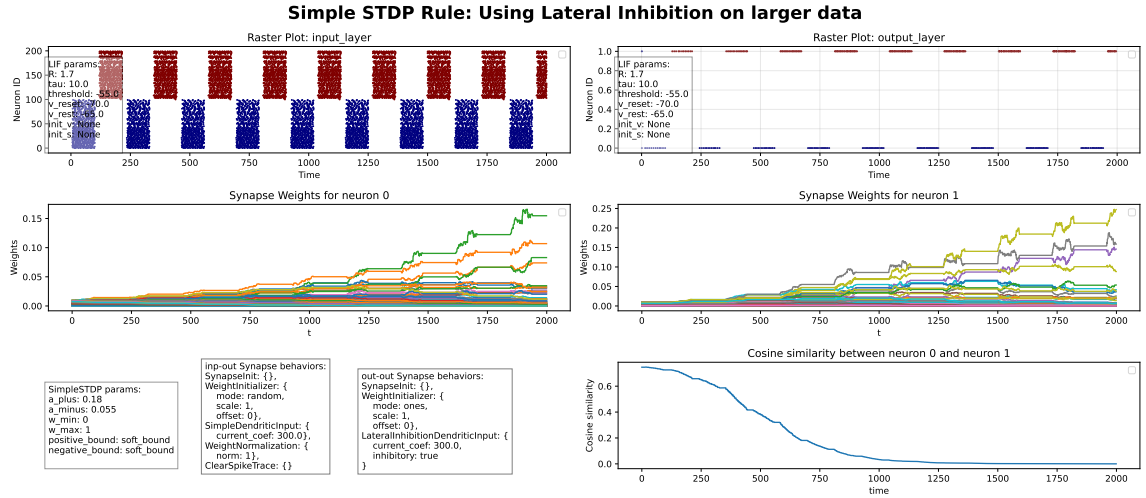
همانطور که در شکل ۶.۱ نیز ملاحظه می‌کنیم افزودن ساز و کار مهار جانبی به لایه خروجی باعث می‌شود فرایند یادگیری مدل بهبود یابد و نه تنها در مراحل زودتری، بتواند الگو ها را تشخیص دهد، بلکه با تکرار شبیه سازی، با احتمال بیشتری دقت پایداری را حفظ کند. همچنین

همانطور که ملاحظه می‌کنیم نمودار شباهت کسینوسی آن به سمت صفر میل می‌کند.



شکل ۶.۱: **افزودن ساز و کار مهار جانبی به لایه خروجی.** همانطور که در شکل بالا مشاهده می‌کنیم، افزودن ساز و کار مهار جانبی به لایه خروجی، باعث بهبود فرایند یادگیری مدل می‌شود. به طوری که مدل هم در مراحل زودتری الگو ها را تشخیص می‌دهد، و هم با تکرار آزمایش، با احتمال بیشتری نسبت به حالتی که این ساز و کار وجود ندارد پایداری خود را حفظ می‌کند. نکته مهم دیگری که در این شکل مشاهده می‌شود، نمودار شباهت کسینوسی بین وزن های لایه خروجی است. این نمودار در حالتی که این ساز و کار وجود نداشت، پس از اینکه تا حدی کم می‌شد، نزدیک صفر نوسان میکرد، اما با افزودن ساز و کار مهار جانبی، شباهت کسینوسی بعد از حدود ۲۵۰ تکرار به مقدار صفر میل می‌کند.

تکرار آزمایش با الگوی پیچیده تر نیز (مجموعه داده دانشگاه واترلو) نتایج مشابهی به همراه خواهد داشت و مدل توانایی تشخیص الگو ها را همچنان دارد.(شکل ۷.۱)



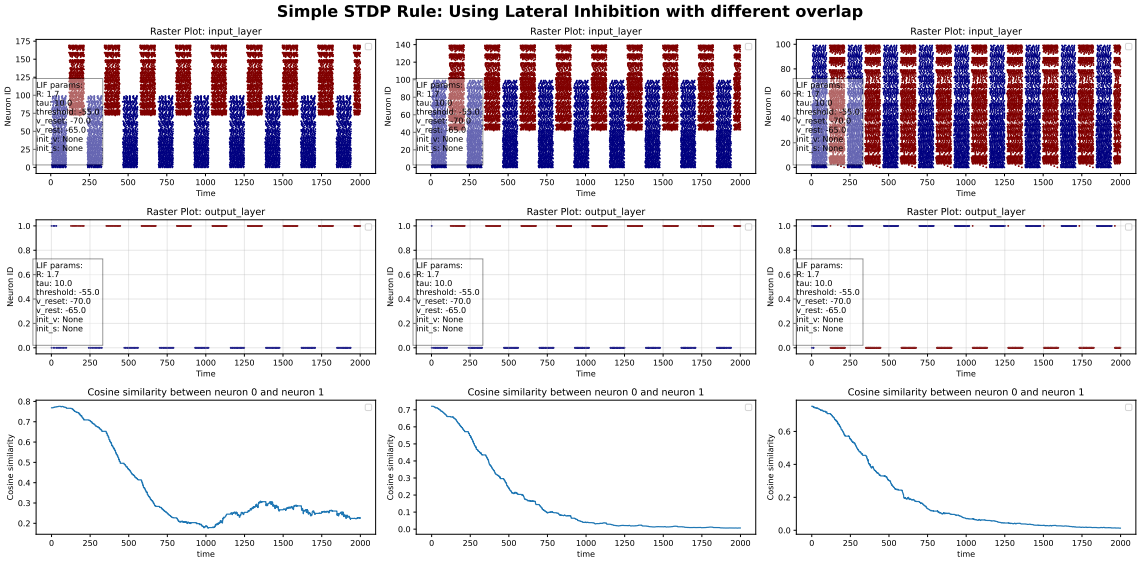
شکل ۷.۱: **افزودن ساز و کار مهار جانبی به لایه خروجی: الگو های های پیچیده تر.** مشاهده می‌کنیم حتی با شبیه سازی آزمایش با الگو های پیچیده تر، یعنی تصاویر مجموعه داده دانشگاه واترلو نیز، مدل توانایی خود در تشخیص دادن را از دست نداده است.

نتیجه گیری

افزودن ساز و کار مهار جانبی به لایه خروجی، باعث می‌شود توانایی مدل در تشخیص الگو ها بیشتر شود و نورون ها به الگو های متفاوت حساس تر شوند. در ساز و کار مهار جانبی، برانگیخته شدن یک نورون، باعث می‌شود همسایه های خود را مهار کند. از این رو، افزودن این ساز و کار به لایه خروجی، باعث می‌شود هنگامی که یک نورون به یک الگو حساس می‌شود، نورون دیگر را مهار کرده و در نتیجه، تشخیص الگو ها مقداری بهتر شود. هرچند این ساز و کار به تنهایی برای یادگیری انواع مدل ها کافی نیست.

۱.۲.۱ آزمایش مدل با اشتراک متفاوت الگو ها

در این قسمت، آزمایش را با میزان اشتراک متفاوت الگو ها تکرار می‌کنیم. برای اینکار، سه مقدار متفاوت 20%، 50% و 100% را بین دو الگو در نظر گرفته و برای هر کدام شبیه سازی را انجام می‌دهیم. مطابق شکل ۸.۱ مشاهده می‌کنیم که با افزودن میزان اشتراک بین الگو های ورودی نیز همچنان مدل توانایی تشخیص دادن الگو ها را دارد. هر چند در مدلی که میزان اشتراک به حداکثر خود رسیده است، مشاهده می‌کنیم که مدل دیرتر الگو ها را تشخیص می‌دهد و همچنین نمودار شباهت کسینوسی آن نیز دیر به سمت صفر میل می‌کند.

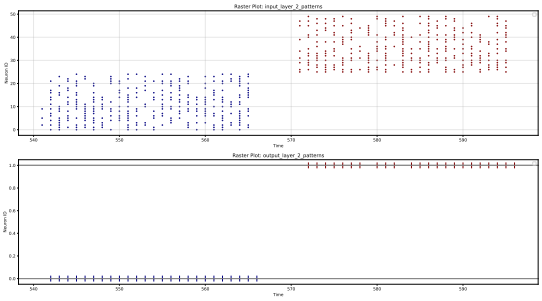


شکل ۸.۱: **ساز و کار مهار جانبی در لایه خروجی: آزمایش میزان اشتراک متفاوت الگوها** مشاهده می‌کنیم حتی با شبیه سازی آزمایش با میزان اشتراک متفاوت الگو هانیز مدل توانایی خود در تشخیص دادن را از دست نداده است. هرچند در شکل سمت راست که مدل حداکثر اشتراک را در الگو ها دارد، تشخیص دادن در مراحل دیرتری اتفاق می‌افتد، اما در نهایت موفق به تشخیص الگو ها می‌شود. علاوه بر آن، نمودار شباهت کسینوسی نیز دیر به سمت صفر میل می‌کند. در این مدل، بقیه پارامتر ها همانند شکل ۷.۱ می‌باشد.

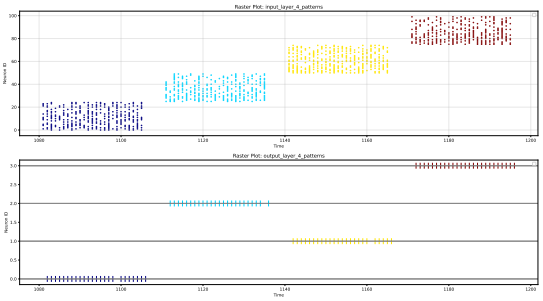
۲.۲.۱ نتایج

در این قسمت، مدل را با اندازه الگو های مختلف شبیه سازی می‌کنیم و بررسی می‌کنیم که مدل برای چه تعداد ورودی می‌تواند آن ها را تشخیص دهد. مطابق شکل ۹.۱ مشاهده می‌کنیم که مدل توانسته است تا حدود ۶ الگوی ورودی را به خوبی فرا بگیرد. هر چند ممکن است گاهی با اجرای دوباره نتیجه مطلوب گرفته نشود. اما با بیشتر شدن این تعداد تا ۸ عدد، در اکثر مواقع مدل موفق به یادگیری برخی از این الگو ها نمی‌شود و فقط تعدادی را یاد می‌گیرد.

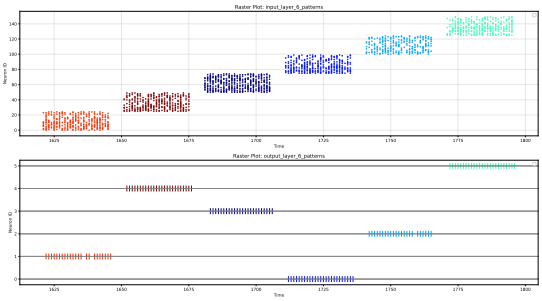
[۲] برای تشخیص عملکرد مدل، دقت کنید در هر بازه‌ای که یک نورون برای الگویی ضربه زده است، نورون های دیگر ضربه‌ای نزده‌اند همچنین هر نورون، در زمان ورودی دادن فقط برای یک الگو ضربه زده است.



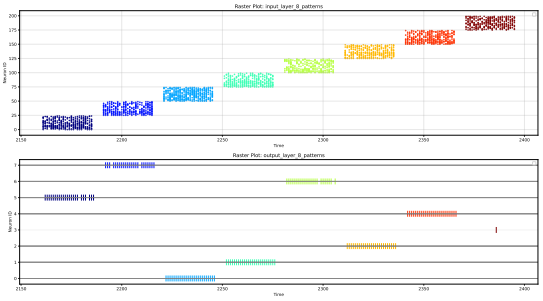
(آ) ۲ الگو



(ب) ۴ الگو



(ج) ۶ الگو

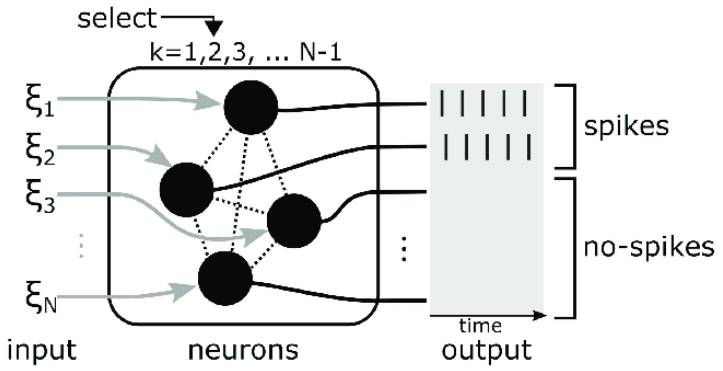


(د) ۸ الگو

شکل ۹.۱: **آزمایش مدل با تعداد الگو های متفاوت.**در این آزمایش یک مدل ترین شده روی الگو ها را، به اندازه یک بار ورودی دادن همه الگو ها شبیه سازی می‌کنیم. (تعداد باری که هر الگو را مدل دیده است در هر آزمایش برابر است.) مطابق شکل مشاهده می‌کنیم که مدل توانسته است تا حدود ۶ الگو را به خوبی تشخیص دهد. هر چند این تشخیص ممکن است گاهی در تکرار شبیه سازی موفق نشود. اما هنگامی که مدل را با ۸ الگو آزمایش می‌کنیم مشاهده می‌کنیم که بعضی از الگو ها تشخیص داده نشده اند و نورون های خروجی به کلی ضربه نزده اند. در ادامه خواهیم دید برای جلوگیری از این مشکل (که بعضی نورون ها در زمان ورودی گرفتن کل الگو ها ضربه‌ای نزنند.) چه ساز و کار هایی به کمک ما خواهند آمد.

۳.۱ k ـ برنده همه چیز را می‌گیرند(K-Winners-Take-All)

ساز و کار K-winners-take-all (KWTA) در شبکه‌های عصبی ضربه‌ای (یا حتی مصنوعی) شکلی از یادگیری رقابتی است که به تعیین اینکه کدام نورون‌ها در یک شبکه به شدت در پاسخ به ورودی‌های خاص فعال می‌شوند کمک می‌کند. در زمینه شبکه‌های عصبی ضربه‌ای، که در آن سعی می‌کنیم از نحوه عملکرد نورون‌های واقعی الهام بگیریم، ساز و کار KWTA با اجازه دادن به نورون‌ها برای رقابت بر اساس پتانسیل‌های غشایی یا سرعت ضربه زدن عمل می‌کند. می‌دانیم هر نورون در یک شبکه جریان را دریافت می کند و این جریان ورودی بر پتانسیل آنها تأثیر می گذارد. نورون هایی که به آستانه خاصی می رسند ضربه می‌زنند. از این رو در پیاده سازی KWTA، از بین تمام نورون‌های یک لایه یا گروه خاص، فقط k نورونی که پتانسیل‌شان بالاتر است (یا آن‌هایی که به شدت افزایش یافته اند) مجاز به فعال ماندن یا ”برنده شدن“ هستند. بقیه سرکوب می شوند یا فعالیت آنها کاهش می یابد، از این رو اصطلاح ”برنده ها همه چیز را می گیرند“ برای آن به کار برده می‌شود. ساز و کار KWTA معمولاً از طریق روش‌های مختلفی مانند بازخورد مهاری، که در آن نورون‌ها فعالیت نورون‌های دیگر را در مجاورت خود مهار می‌کنند، یا از طریق طراحی شبکه که خود معماری تضمین می‌کند که فقط k نورون برتر می‌توانند در هر نقطه فعال باقی بمانند، انجام می‌شود. زمان. این رویکرد را می توان به عنوان شکلی از رقابت بین نورون ها دید که یک ویژگی اساسی سیستم های عصبی بیولوژیکی است و برای کارهایی مانند تشخیص الگو و فرآیندهای تصمیم گیری در هر دو سیستم طبیعی و مصنوعی بسیار مهم است. خلاصه این مفهوم در شکل ۱۰.۱ آمده است.



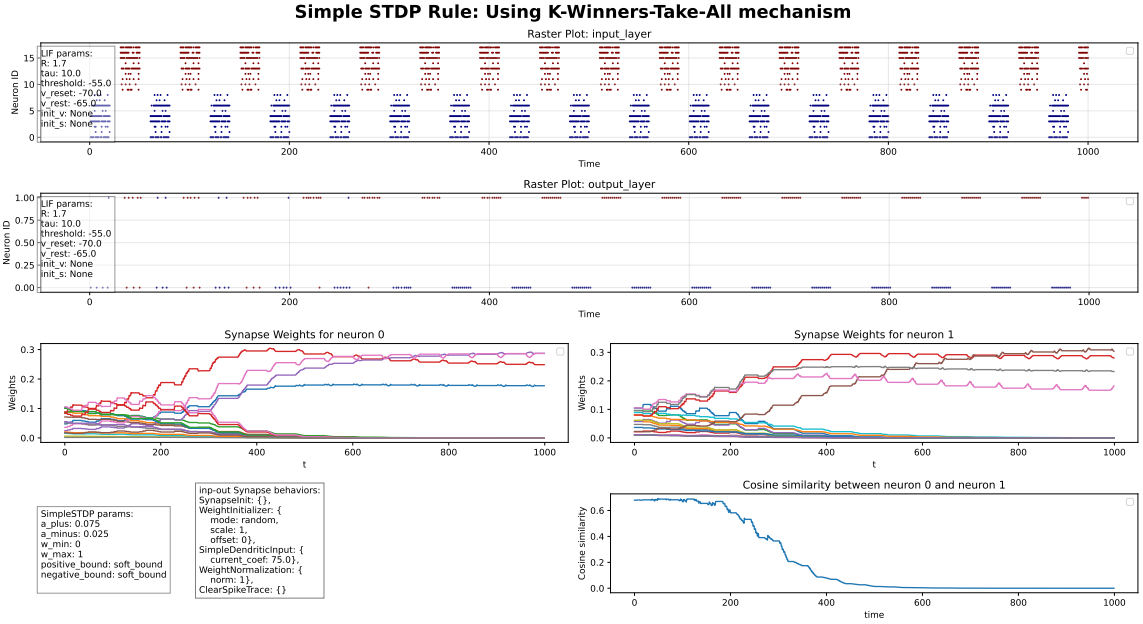
شکل ۱۰.۱: **ساز و کار « k ـ برنده همه چیز را می‌گیرند».** در هر مرحله از شبیه سازی، فقط آن نورون‌هایی که بزرگترین ورودی را دریافت می‌کنند، ضربه می‌زنند، در حالی که ضربه های همه نورون‌های دیگر به صورت پویا سرکوب می‌شوند.

حال می‌خواهیم این ساز و کار را به شبکه‌مان اضافه کنیم. ابتدا، حالتی را در نظر می‌گیریم که در آن شبکه، از یک لایه ورودی و خروجی و یک سیناپس از لایه ورودی به لایه خروجی استفاده می‌کند. همچنین مشابه قسمت قبل، از قانون یادگیری $STDP$ برای آموزش مدل استفاده می‌کنیم. همانطور که در شکل ۱۱.۱ نیز ملاحظه می‌کنیم، افزودن ساز و کار k-winners-take-all به لایه خروجی باعث می‌شود در همان مراحل اولیه، مدل الگو ها را یاد بگیرد. هر چند در این حالت نیز ممکن است در برخی شبیه سازی ها مدل نتواند الگو ها ها را به خوبی یاد بگیرد، اما قرار گرفتن این ساز و کار در کنار ساز و کار های دیگر که بررسی کردیم یا بررسی خواهیم کرد، توانایی مدل را افزایش خواهد داد. همانطور که از شکل برمی‌آید، اضافه کردن این ساز و کار باعث شده است تمایز وزن های بین نورون ها را شاهد باشیم و همچنین شباهت کسینوسی نسبت به حالت بدون ساز و کار یا با ساز و کار مهار جانبی، زودتر به صفر میل کند.

مقایسه ولتاژ در این پاراگراف می‌خواهیم دو ساز و کار مهار جانبی و برنده–همه–چیز–را–می‌گیرد را از نظر تاثیر روی ولتاژ مقایسه کنیم. طبق شکل ۱۲.۱ مشاهده می‌کنیم که نحوه تاثیر این دو ساز وکار بر روی ولتاژ متفاوت است بدین صورت که ولتاژ نورون های سرکوب شده در ساز و کار مهار جانبی، توسط جریان کاهش می‌یابد (یک جریان منفی به نورون ها وارد می‌شود) در حالی که در ساز و کار برنده–همه–چیز–را–می‌گیرد این اتفاق به طور مستقیم و ناگهانی در پیاده سازی رخ می‌دهد. دانستن این نکته ممکن است در ساز و کار دیگری که در ادامه آزمایش می‌کنیم، (هم‌ایستایی مبتنی بر ولتاژ) مفید باشد.

حال به لایه خروجی، ساز و کار مهار جانبی که در بخش قبل انجام دادیم را نیز اضافه می‌کنیم تا رفتار مدل را در حضور هر دو ساز و کار بررسی کنیم. مطابق شکل ۱۳.۱ مشاهده می‌کنیم که با داشتن هر دو ساز و کار k-winners-take-all و مهار جانبی، مدل به خوبی توانسته است دو الگو را تشخیص دهد. مطابق شکل، دریافت می‌شود که در مراحل اولیه، مدل در یادگیری الگو ها دچار مشکل بوده است با این حال، توانسته است پس از مراحلی، الگو ها را یاد بگیرد. در حالی که هنگامی که این شرایط برای مدل های دیگر پیش می‌آید، مدل حتی پس از طی کردن مراحل بیشتر شبیه سازی نیز موفق به یادگیری الگو ها نمی‌شود.

حال مدل را برای الگو های پیچیده تر نیز آزمایش می‌کنیم. مطابق شکل ۱۴.۱ مشاهده می‌کنیم که مدل از عهده تشخیص الگو های پیچیده‌تر نیز برمی‌آید. مجددا در این حالت نیز جداسازی وزن ها به خوبی انجام می‌شود و شباهت کسینوسی نیز به صفر میل می‌کند. همچنین مدل پایدار است و با تکرار آزمایش نیز می‌تواند الگو ها را تشخیص دهد.



شکل ۱۱.۱: **اضافه کردن ساز و کار k-winners-take-all به تنهایی در لایه خروجی یک شبکه ساده.** مشاهده می‌کنیم که شبکه توانسته است به خوبی الگو های ورودی را یاد بگیرد. نسبت به حالتی که ساز و کاری وجود نداشت یا حالتی که ساز و کار مهار جانبی داشتیم، یاد گرفتن الگو ها زودتر اتفاق افتاده و همچنین شباهت کسینوسی زودتر به سمت صفر میل کرده است.

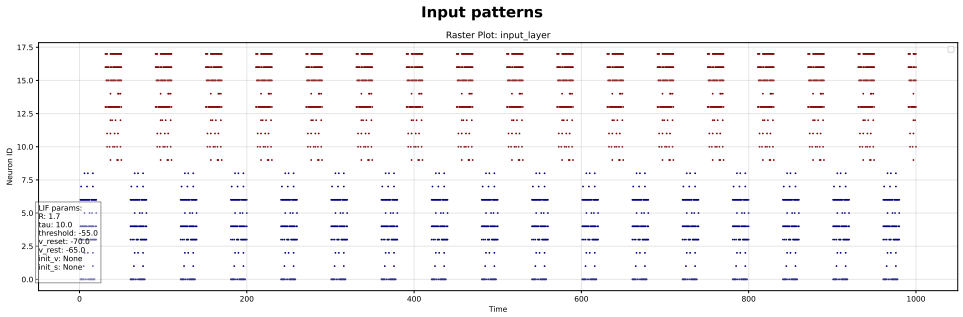
۱.۳.۱ آزمایش مدل با اشتراک متفاوت الگو ها

حال مشابه قسمت قبل، آزمایش های بالا را برای الگو ها با میزان اشتراک متفاوت انجام می‌دهیم. انتظار داریم که عملکرد شبکه‌ای که هر دو ساز و کار را دارد، بهترین عملکرد را داشته باشد. شکل ۱۵.۱ این موضوع را تایید می‌کند.

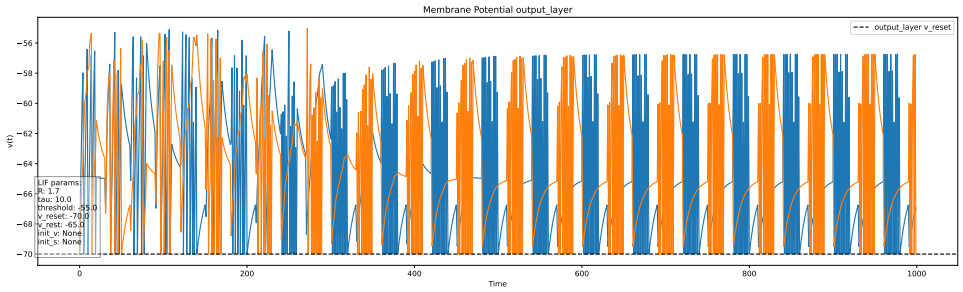
نتایج

مطابق آزمایش هایی که انجام دادیم، مشاهده کردیم که اضافه کردن ساز و کار k-winners-take-all به یک شبکه ساده، می‌تواند همانند ساز و کار مهار جانبی، قدرت مدل را در تشخیص الگو ها بهبود ببخشد. هر چند تاثیر آن اندکی بهتر از ساز و کار مهار جانبی است چرا که پایداری مدل را نسبت به تکرار آزمایش افزایش می‌داد. اما در کل تاثیری که هر دوی این ساز و کار ها روی رفتار مدل به تنهایی می‌گذارند یکسان است. ما در مدل هایمان به این علاقه داریم که هر نورون فقط هنگامی که ورودی مخصوص خود را می‌بیند ضربه بزند و در پنجره زمانی که ورودی دیگر داده می‌شود ضربه‌ای نزند. هر دوی این ساز و کار ها به نحوی اینکار را سهولت می‌بخشند. یکی با دادن جریان منفی به نورون دیگر، و دیگری با سرکوب کردن نورون دیگر.

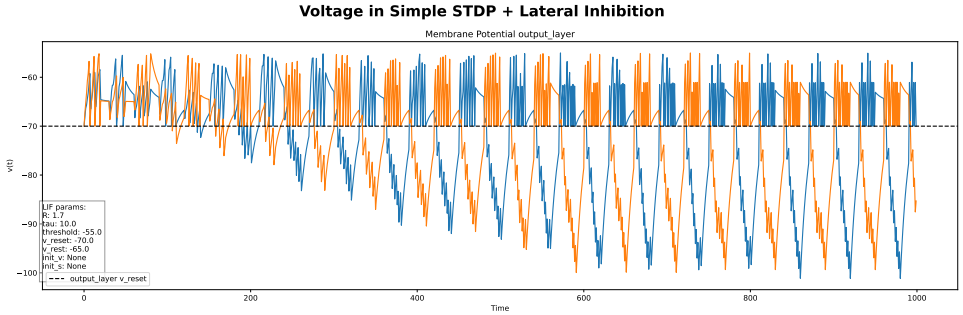
در نهایت مشاهده کردیم که هر دوی این ساز و کار ها به تنهایی، تا حدودی نسبت به افزایش میزان اشتراک مقاوم هستند. هر چند اشتراک حداکثری ممکن است پایداری آن ها را کاهش دهد. بهترین عملکرد را هم زمانی شاهد بودیم که هر دوی این ساز و کار ها به لایه خروجی اضافه شدند و مدل نسبت به تکرار آزمایش یا افزایش اشتراک الگو ها پایداری بیشتری نشان می‌داد، این مورد دقیق تر در آزمایش ۵.۲ آمده است. در بخش بعدی، ساز و کار دیگری به شبکه‌مان اضافه می‌کنیم.



(آ) الگو های ورودی.

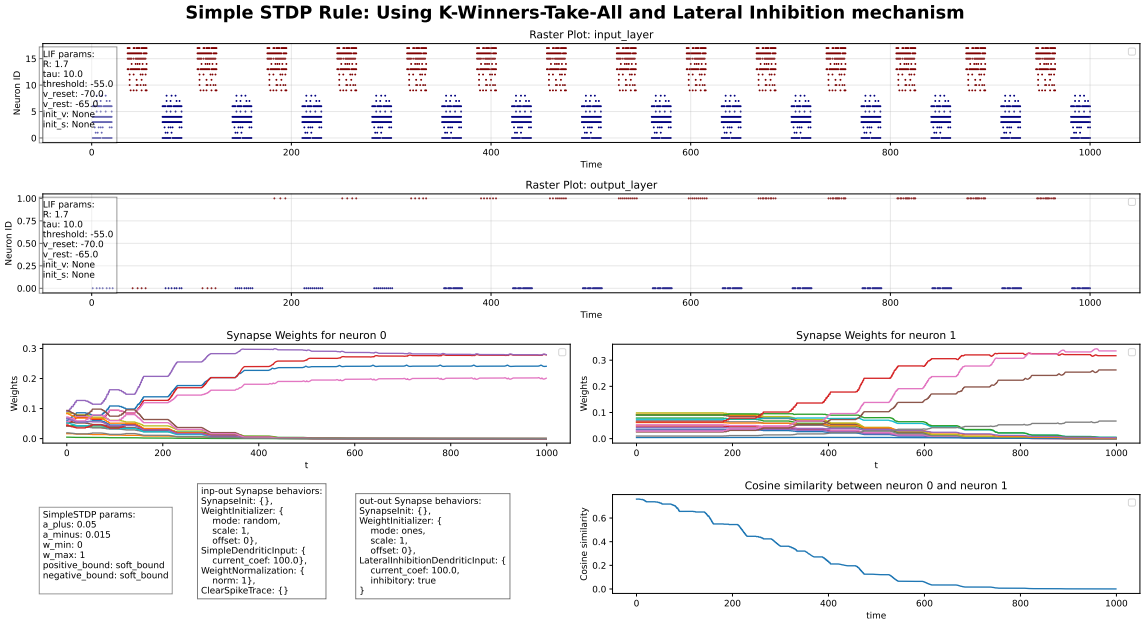


(ب) شبکه ساده به همراه ساز و کار مهار جانبی: نمودار ولتاژ دو نورون لایه خروجی.

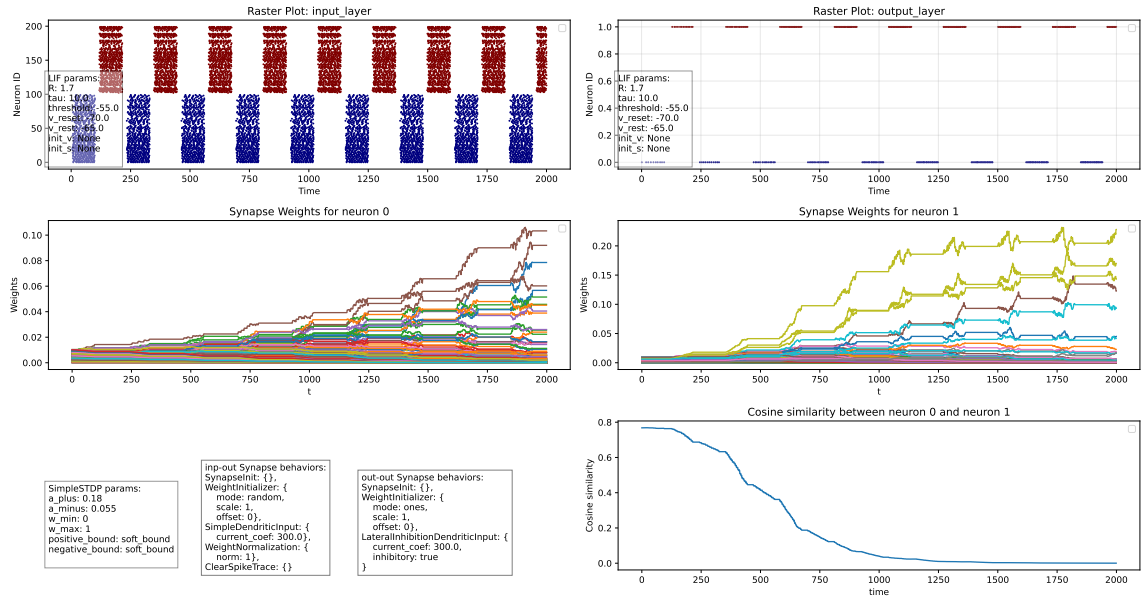


(ج) شبکه ساده به همراه ساز و کار k-winners-take-all: نمودار ولتاژ دو نورون لایه خروجی.

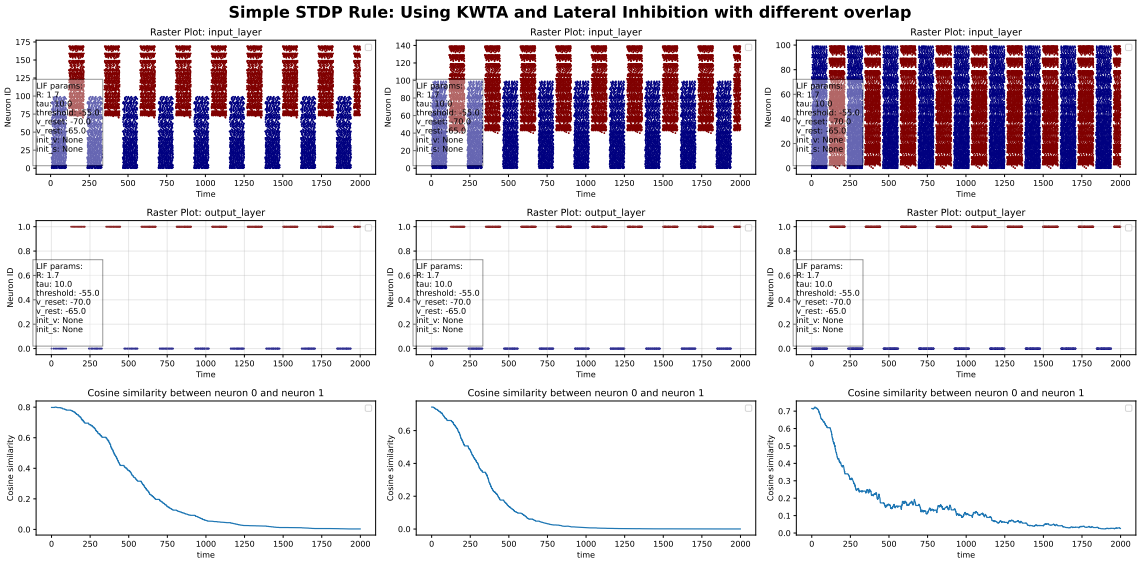
شکل ۱۲.۱: **مقایسه ولتاژ در شبکه با ساز و کار مهار جانبی یا k-winners-take-all.** در شکل بالا، مقایسه ولتاژ در دو شبکه با قانون STDP که یکی دارای ساز و کار مهار جانبی و دیگری دارای ساز و کار k-winners-take-all هستند آمده است. مطابق شکل ملاحظه می‌کنیم که در شبکه ای که مهار جانبی وجود دارد، به دلیل اینکه سرکوب کردن ضربه زدن نورون های همسایه توسط جریان صورت می‌گیرد، ملاحظه می‌کنیم که ولتاژ در زمان هایی که ورودی های مختلف داده می‌شود، یکی درمیان به علت گرفتن جریان منفی زیاد، تا -90 درجه کم می‌شوند در حالی که در ساز و کار k-winners-take-all ولتاژ به طور ناگهانی در پیاده سازی برابر با ولتاژ حالت استراحت مقدار می‌شود.



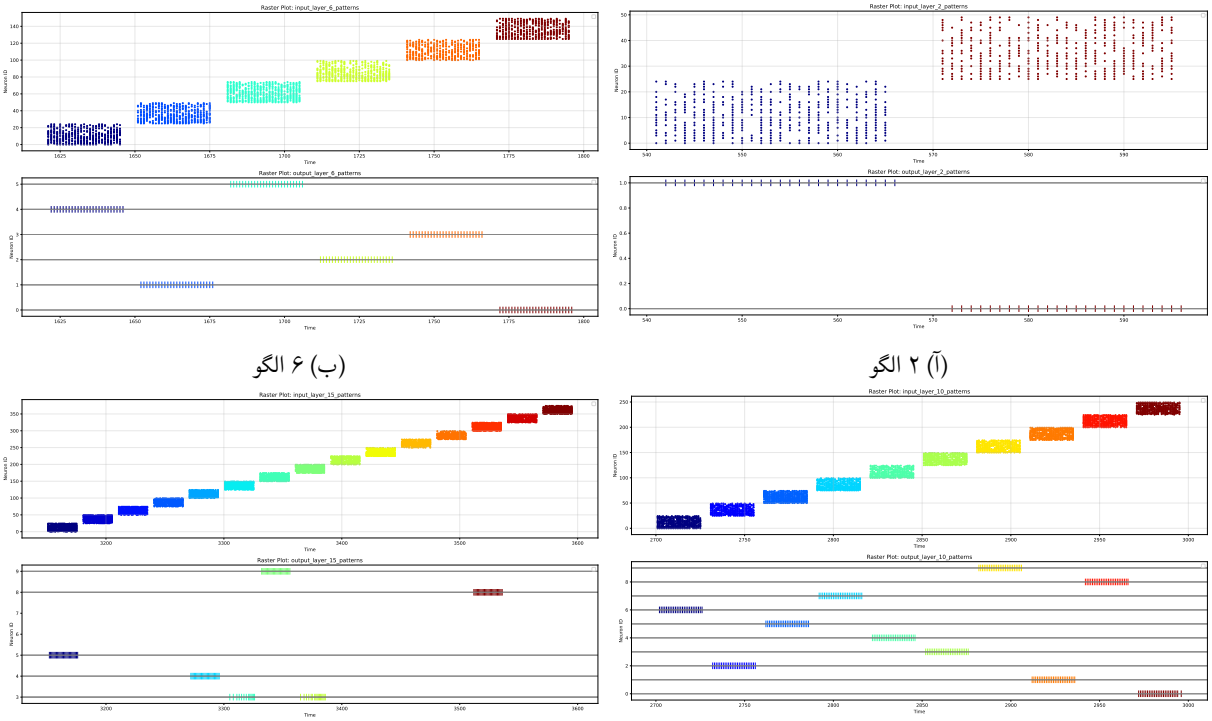
شکل ۱۳.۱: **اضافه کردن ساز و کار k-winners-take-all به همراه ساز و کار مهار جانبی به لایه خروجی.** مشاهده می‌کنیم که با داشتن هر دو ساز و کار k-winners-take-all و مهار جانبی، مدل به خوبی توانسته است دو الگو را تشخیص دهد. مطابق شکل، دریافت می‌شود که در مراحل اولیه، مدل در یادگیری الگو ها دچار مشکل بوده است با این حال، توانسته است پس از مراحلی، الگو ها را یاد بگیرد. در حالی که هنگامی که این شرایط برای مدل های دیگر پیش می‌آید، مدل حتی پس از طی کردن مراحل بیشتر شبیه سازی نیز موفق به یادگیری الگو ها نمی‌شود. همچنین پس از طی مراحلی، وزن ها به خوبی نسبت به الگو های ورودی تنظیم شده اند و مشاهده می‌کنیم که شباهت کسینوسی نیز به صفر میل کرده است. نکته مهمی که در این مدل وجود دارد این است که پایدار است و هر چقدر آن را شبیه سازی کنیم، در تشخیص الگو ها موفق است.



شکل ۱۴.۱: **اضافه کردن ساز و کار k-winners-take-all به همراه ساز و کار مهار جانبی به لایه خروجی: داده های پیچیده‌تر.** مطابق شکل ملاحظه می‌کنیم مدل از عهده تشخیص الگو های پیچیده‌تر نیز برمی‌آید. در این حالت نیز وزن ها به خوبی تنظیم می‌شوند و شباهت کسینوسی نیز به صفر میل می‌کند. نکته مهم اینکه مدل پایدار است و با تکرار آزمایش نیز می‌تواند الگو ها را تشخیص دهد.



شکل ۱۵.۱: **اضافه کردن ساز و کار k-winners-take-all به همراه ساز و کار مهار جانبی به لایه خروجی:میزان اشتراک متفاوت.** همانطور که انتظار داشتیم، شبکه‌ای که هر دو ساز و کار را دارد، نسبت به انواع میزان اشتراک الگو ها پایدار است و در هر سه حالت توانسته است الگو ها را تشخیص دهد. هر چند هنگامی که آزمایش می‌شد، برای اشتراک با میزان کم یا متوسط، همیشه مدل درست تشخیص می‌داد، ولی هنگامی که حداکثر اشتراک را داشتیم، در بعضی شبیه سازی ها مدل نمی‌توانست به درستی دو الگو را تشخیص دهد.



(آ) ۲ الگو

(ب) ۶ الگو

(ج) ۱۰ الگو

(د) ۱۵ الگو

شکل ۱۶.۱: **آزمایش مدل با تعداد الگو های متفاوت.** در این آزمایش یک مدل ترین شده روی الگو ها را، به اندازه یک بار ورودی دادن همه الگو ها شبیه سازی می‌کنیم. (تعداد باری که هر الگو را مدل دیده است در هر آزمایش برابر است.) مطابق شکل مشاهده می‌کنیم که مدل توانسته است هنگامی ۱۰ الگو ورودی می‌گیرد هم به خوبی تشخیص دهد که نسبت به حالتی که ساز و کار k-winners-take-all را نداشتیم، پیشرفت محسوب می‌شود. نکته دیگر اینکه با تکرار آزمایش نیز مدل با احتمال کمتری ممکن است توانایی تشخیص خود را از دست بدهد. همچنین زیاد شدن تعداد الگو ها، مثلا ۱۵ عدد مدل نمی‌تواند بعضی از الگو ها را تشخیص دهد و بعضی نورون ها حتی در طول زمانی ورودی دادن ضربه‌ای نمی‌زنند. در بخش بعدی، ساز و کار هم‌ایستایی را برای حل این مشکل به مدل اضافه خواهیم کرد.

بخش ۲

هم‌ایستایی یا هومئوستازی (Homeostasis)

۱.۲ ساز و کار هم‌ایستایی

هم‌ایستایی یا هومئوستازی [۱] در زیست‌شناسی به معنای حفظ پایداریِ محیط داخلی بدن و ثابت نگه داشتن شرایط فیزیکی و شیمیایی جاندار است. عملکرد بهینه جاندار در گرو این ویژگی است که متغیرهای زیادی از جمله دما و تعادل مایعات بدن را در محدوده‌ای از پیش تعیین شده نگه می‌دارد (محدوده هومئوستاتیک). پی‌اچِ مایعات برون‌سلولی، غلظت یون‌های سدیم، پتاسیم و کلسیم و سطح قند خون نیز بخشی از این متغیرهاست که پیوسته کنترل می‌شود. جاندار، علی‌رغم تغییرات محیط، نوع رژیم غذایی و مقدار فعالیت بدنی، تعادل متغیرهای بدنش را، هرکدام با یک یا چند سازوکار هومئوستاتیک به‌طوری پایدار حفظ می‌کند که تمام این فرایندهای تنظیمی باهم حیات را تدوام می‌بخشد. پلاستیسیته هموستاتیک نیز در زمینه تولیدکننده الگوهای مرکزی بسیار مهم است. در این زمینه، خواص عصبی در پاسخ به تغییرات محیطی به منظور حفظ خروجی عصبی مناسب تعدیل می‌شوند. در این قسمت، اضافه کردن هر دو نوع این ساز و کار را به لایه دوم بررسی می‌کنیم.

۲.۲ ساز و کار هم‌ایستایی براساس فعالیت

ابتدا، به عنوان اولین آزمایش، به یک شبکه ساده، این ساز و کار را اضافه کرده، و سپس به شبکه های بخش قبلی نیز آن را اضافه می‌کنیم. همانطور که در شکل ۱.۲ نیز مشاهده می‌کنیم، اضافه کردن این ساز و کار به تنهایی به یک شبکه ساده با قانون یادگیری STDP توانسته است نسبت به حالت ساده، بهتر الگو ها را تشخیص دهد. هر چند با تکرار شبیه سازی، ممکن است این یادگیری انجام نشود چرا که ساز و کار هومئوستازی بیشتر در کنار ساز و کار های دیگر برای بهبود و حفظ پایداری می تواند کاربرد داشته باشد.

حال این ساز و کار را به شبکه قسمت قبل که دارای ساز و کار های مهار جانبی و k-winners-take-all بود اضافه می‌کنیم. انتظار داریم که اضافه کردن این ساز و کار بتواند پایداری شبکه قبل را بهتر کند. همانطور که مطابق شکل ۲.۲ نیز مشاهده می‌کنیم، افزودن این ساز و کار به شبکه باعث می‌شود که شبکه توانایی یادگیری خود را حفظ کند و نسبت به قبل پایدار تر شود. در این مدل، ما پارامتر های ساز و کار هومئوستازی را به نحوی دادیم که در یک پنجره زمانی که ورودی داده می‌شود، فقط نصف نورون ها فعالیت داشته باشند. از این رو مطابق شکل نیز ملاحظه می‌کنیم که تعداد ضربه ها هنگامی که یک ورودی داده می‌شود نسبت به مدل های قبلی بیشتر شده است و گویی که نورون مخصوص به یک الگو، فقط و فقط در زمان ورودی دادن آن الگو فعال می‌شود.

۱.۲.۲ افزایش تعداد الگو ها

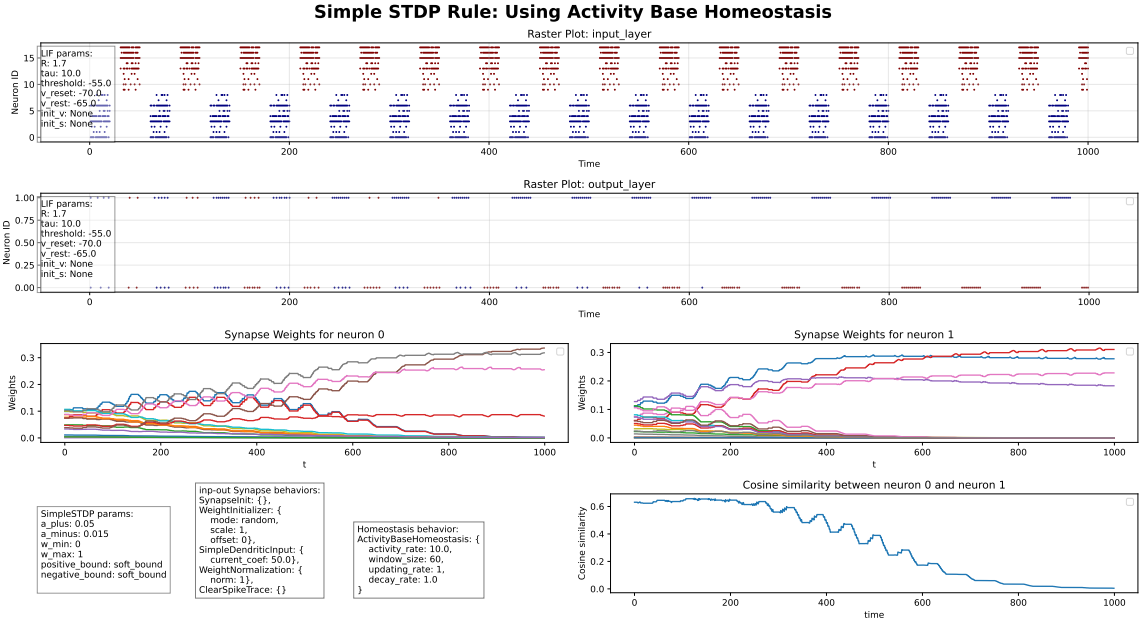
حال در این مرحله، تعداد الگو های ورودی و در نتیجه تعداد نورون های خروجی را افزایش می‌دهیم. برای اینکار از مجموعه داده مخزن دانشگاه واترلو و همچنین یک عکس تصادفی (من در اینجا عکس خودم را به عنوان نمونه قرار داده ام) استفاده می‌کنیم. (شکل ۳.۲)

حال ساز و کار مورد نظر را به مدل اضافه می‌کنیم و مطابق شکل ۸.۲ ملاحظه می‌کنیم که مدل برای تعداد الگوهای بیشتر نیز می‌تواند توانایی خود در تشخیص آن ها را حفظ کند. حتی با تکرار شبیه سازی با دفعات زیاد نیز مدل همچنان در تشخیص الگو ها موفق بوده و پایداری خود را حفظ می‌کند. فقط دقت شود که به دلیل جلوگیری از شلوغی نمودار ها و تحلیل بهتر، در اینجا اندازه الگو های ورودی را کمی کاهش دادیم.

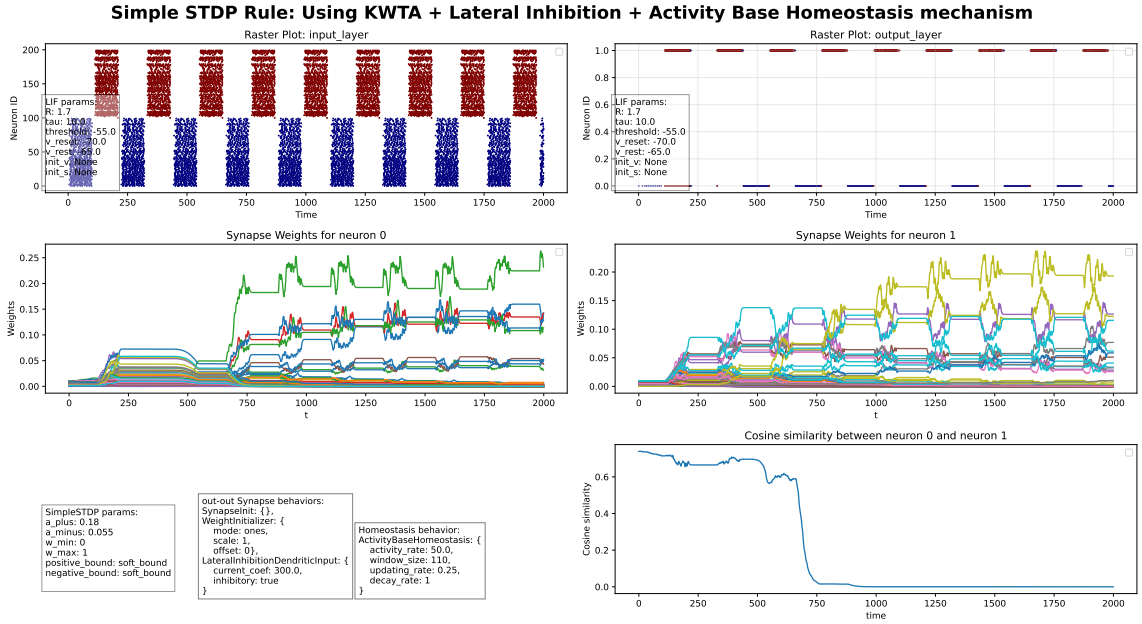
۲.۲.۲ نتایج

حال که با ساز و کار هم‌ایستایی به عنوان ساز و کاری که میزان فعالیت نورون ها را در یک بازه زمانی کنترل می‌کند آشنا شدیم، ممکن است به این فکر کنیم که این ساز و کار چقدر می‌تواند کارایی مدلمان را نسبت به بخش قبل بهبود دهد. خوشبختانه مطابق شکل ۵.۲ ملاحظه می‌کنیم که

[۱] Homeostasis



شکل ۱.۲: **اضافه کردن ساز و کار هومئوستازی براساس فعالیت به یک شبکه ساده.** همانطور که در شکل بالا نیز مشاهده می‌کنیم، اضافه کردن این ساز و کار به تنهایی برای یک شبکه ساده، می‌تواند نسبت به حالتی که وجود نداشته باشد، یادگیری را بهتر کن.(یادگیری زودتر اتفاق بیوفتد یا تعداد نورون های فعال برای الگو بیشتر شود) اما این ساز و کار به تنهایی نمی‌تواند یادگیری مدل را تنظیم کند و بهتر است درکنار ساز و کار های دیگر شبکه مورد استفاده قرار گیرد.

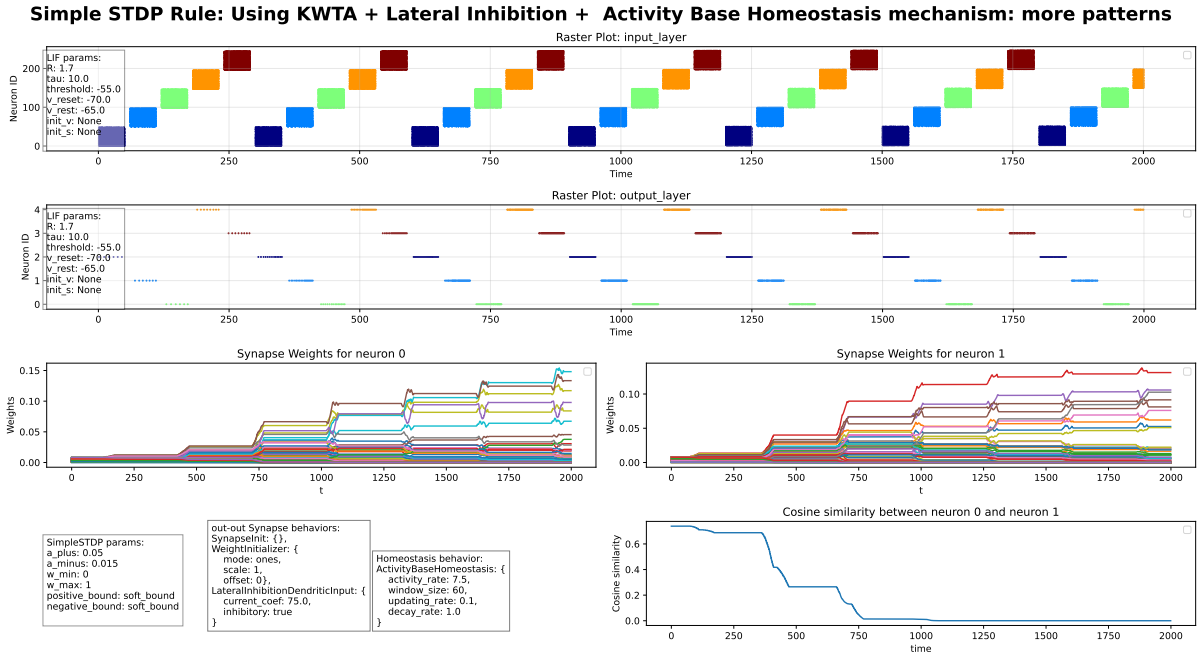


شکل ۲.۲: **ساز و کار، مهار جانبی، k-winners-take-all و هومئوستازی براساس فعالیت در یک شبکه.** همانطور که در شکل بالا نیز مشاهده می‌کنیم، اضافه کردن این ساز و کار هومئوستازی به شبکه قسمت قبل که هم مهار جانبی داشت و هم k-winners-take-all توانسته است الگوهای ورودی را به خوبی تشخیص دهد. همچنین ساز و کار هومئوستازی باعث شده که میزان فعالیت نورون ها در یک پنجره زمانی (در اینجا پنجره زمانی که ورودی داده می‌شود) متعادل شود و ما شاهد فعالیت بیشتر نورون های لایه خروجی برای الگو متعلق به خود هستیم.

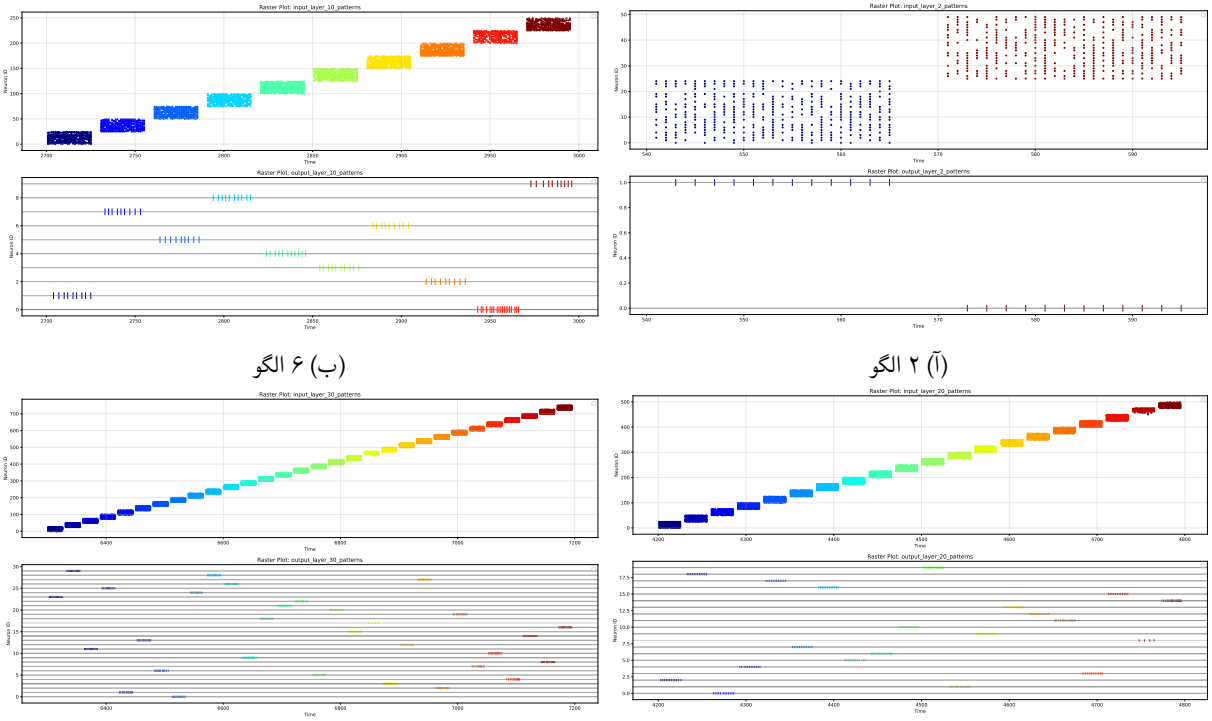
توانایی مدل در تشخیص الگو ها به میزان قابل توجهی افزایش یافته است.

(آ) (ب) (ج) (د) (ه)

شکل ۳.۲: مجموعه داده مورد استفاده



شکل ۴.۲: **ساز و کار، مهار جانبی، k-winners-take-all و هومئوستازی براساس فعالیت در یک شبکه: الگو های بیشتر** مطابق شکل ملاحظه می‌کنیم که مدل توانسته است از عهده تشخیص الگو های بیشتر نیز بربیاید. هر نورون توانسته است یکی از الگو ها را یاد بگیرد. در این آزمایش، با تکرار شبیه سازی، پایداری مدل حفظ می‌شود و هر نورون لایه خروجی، یک الگو را تشخیص می‌دهد.



(آ) ۲ الگو

(ب) ۶ الگو

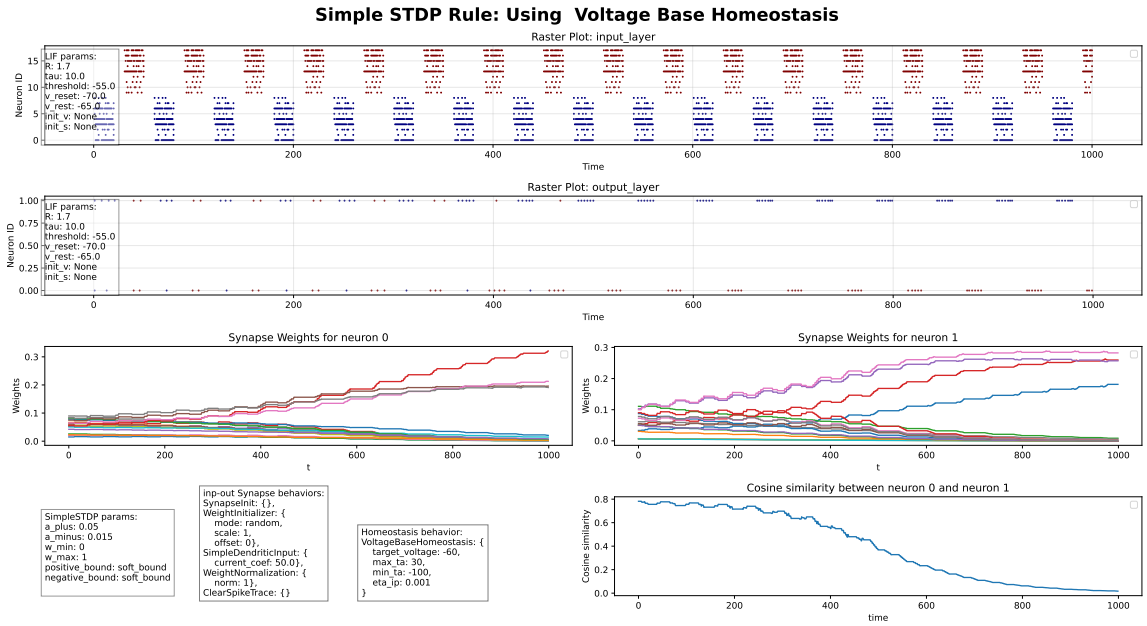
(ج) ۲۰ الگو

(د) ۳۰ الگو

شکل ۵.۲: **آزمایش مدل با تعداد الگو های متفاوت.** در این آزمایش یک مدل ترین شده روی الگو ها را، به اندازه یک بار ورودی دادن همه الگو ها شبیه سازی می‌کنیم. (تعداد باری که هر الگو را مدل دیده است در هر آزمایش برابر است.) مطابق شکل مشاهده می‌کنیم که مدل حتی توانسته است تا ۳۰ الگو را به طور کامل یاد بگیرد و با تکرار آزمایش نیز در یادگیری موفق است. مشاهده می‌کنیم که مشکلی که در مدل قبلی (مدل شامل مهار جانبی و k-winners-take-all) باعث می‌شد برخی نورون ها فعالیتی از خود نشان ندهند و در نتیجه برخی الگو ها تشخیص داده نشوند، با اضافه شدن هم‌ایستایی مبتنی بر فعالیت رفع شده است. چرا که در این ساز و کار، ما میزان فعالیت نورون ها را کنترل می‌کنیم و نمی‌گذاریم که از حدی کمتر شود (یک نورون هیچ الگوی را تشخیص ندهد) یا از حدی بیشتر شده که یک نورون به ازای دو ورودی ضربه بزند. من این آزمایش را برای تعداد بیشتر الگو ها هم انجام دادم ولی به دلیل محدودیت زمان و منابع محاسباتی، و همچنین شلوغ شدن تصاویر، تا ۳۰ الگو را در اینجا آورده‌ام.

۳.۲ ساز و کار هومئوستازی براساس ولتاژ (امتیازی)

حال در این قسمت، ساز و کار هومئوستازی براساس ولتاژ را به مدل اضافه می‌کنیم. کلیت مفهوم این ساز و کار نیز همانند ساز و کار قبلی است، اما تفاوت آن در چیزی است که آن را حفظ می‌کند. در این ساز و کار، سعی می‌شود که ولتاژ نورون را در محدوده‌ای که به عنوان پارامتر می‌گیرد، تنظیم کند. ابتدا این ساز و کار را به تنهایی و به همراه ساز و کار های دیگر به شبکه اضافه می‌کنیم و در نهایت آن را با نوع قبلی هومئوستازی مقایسه می‌کنیم. مطابق شکل ۶.۲ مشاهده می‌کنیم که افزودن ساز و کار هومئوستازی بر پایه ولتاژ توانسته است مدل را قادر سازد الگو های ورودی را تشخیص دهد. نکته جالبی که در این آزمایش وجود داشت این است که حتی با تکرار آزمایش نیز ،مدل به احتمال بیشتری نسبت به حالتی که ساز و کاری وجود نداشت توانایی خود در تشخیص الگو ها را حفظ می‌کرد.



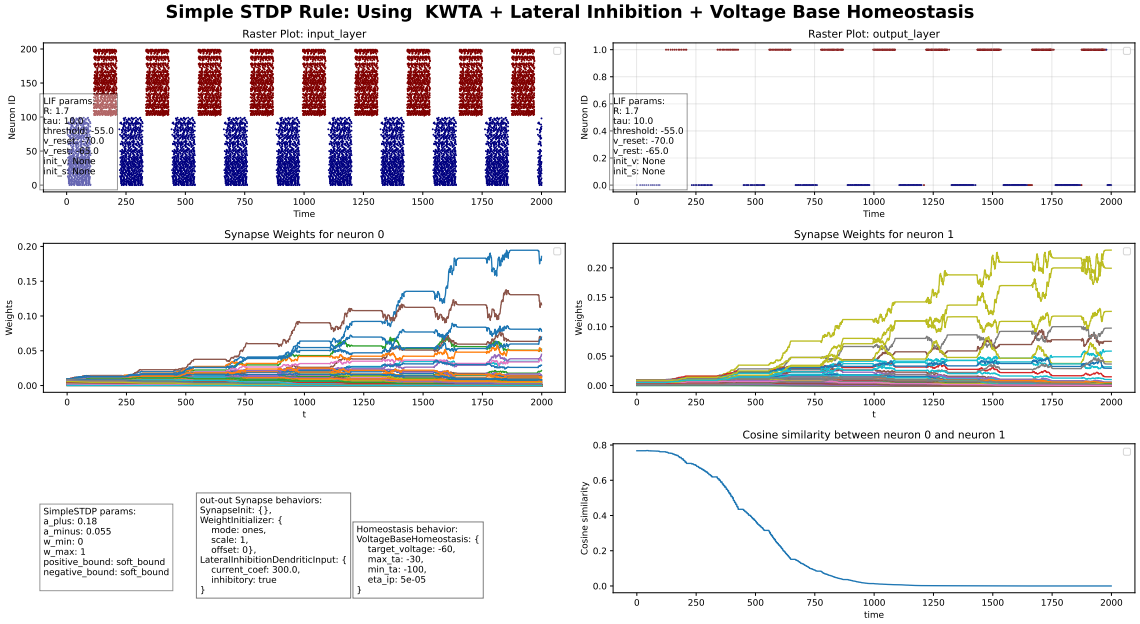
شکل ۶.۲: **اضافه کردن ساز و کار هومئوستازی براساس ولتاژ در یک شبکه ساده.** مطابق شکل ملاحظه می‌کنیم که با اضافه کردن این ساز و کار نیز، مدل توانسته است الگو های ورودی را تشخیص دهد. هر چند هنوز نیز با تکرار آزمایش مدل ممکن است در تشخیص الگو ها گاهی ناموفق باشد، ولی نسبت به حالتی که شبکه ساز و کاری ندارد شاهد پایداری بیشتری هستیم.

حال که تاثیر این ساز و کار را بر مدل ملاحظه کردیم، به سراغ اضافه کردن آن به شبکه‌ای که در نهایت در بخش دوم ساخته شد می‌رویم. همانطور که در شکل ۷.۲ نیز ملاحظه می‌کنیم، اضافه کردن این ساز و کار به شبکه نیز مانند قسمت قبل است و مدل توانسته است الگو ها را تشخیص دهد. نکته‌ای که در مقایسه این ساز و کار با ساز و کار قبلی وجود دارد این است که ما در ساز و کار قبلی شاهد بودیم که در پنجره زمانی که ورودی ها داده می‌شدند، فعالیت نورون های لایه خروجی متراکم تر بود در حالی که چنین چیزی را در اینجا شاهد نیستیم.

مقایسه پیاده سازی

در این قسمت می‌خواهیم پیاده سازی دو ساز و کار را با یکدیگر مقایسه کنیم.

هومئوستازی مبتنی بر فعالیت در هومئوستازی مبتنی بر فعالیت، تنظیم سطح فعالیت نورون ها بر اساس رفتار spiking آنها در یک پنجره زمانی تعریف شده طراحی شده است. این رفتار، یک پارامتر به نام `activity_rate` دریافت می‌کند که نشان دهنده نرخ ضربه زدن نورون ها در پنجره زمانی مورد نظر است. این پنجره زمانی نیز خود با یک پارامتر به نام `window_size` مقدار دهی می‌شود. همچنین یک پارامتر به نام `updating_rate` نیز گرفته می‌شود که عامل مقیاس‌پذیری برای به‌روزرسانی آستانه‌های نورون‌ها است. علاوه بر آن پارامتر `decay_rate` نیز وجود دارد تا نرخ به‌روزرسانی را در طول زمان کاهش می‌دهد تا تغییرات ممکن را تثبیت کند. نحوه کارکرد این ساز و کار به این صورت است که ضربه ها را در یک پنجره مشخص جمع می کند و آستانه نورون را بر اساس انحراف از فعالیت هدف تنظیم می کند. اگر فعالیت خیلی زیاد باشد، آستانه افزایش می یابد و اگر خیلی کم باشد، کاهش می یابد. به‌روزرسانی آستانه‌ها با `updating_rate` مقیاس‌بندی می‌شود، و این نرخ پس از هر پنجره با `updating_rate` تغییر می‌کند.



شکل ۷.۲: **ساز و کار، مهار جانبی، k-winners-take-all و هومئوستازی براساس ولتاژ در یک شبکه.** همانطور که در شکل بالا نیز مشاهده می‌کنیم، اضافه کردن این ساز و کار هومئوستازی به شبکه قسمت قبل که هم مهار جانبی داشت و هم k-winners-take-all توانسته است الگوهای ورودی را به خوبی تشخیص دهد. همچنین ساز و کار هومئوستازی باعث شده که میزان فعالیت نورون ها در یک پنجره زمانی (در اینجا پنجره زمانی که ورودی داده می‌شود) متعادل شود و ما شاهد فعالیت بیشتر نورون های لایه خروجی برای الگو متعلق به خود هستیم.

هومئوستازی مبتنی بر ولتاژ این ساز و کار ولتاژ نورون ها را تنظیم می کند تا آن را در محدوده دلخواه نگه دارد. برای هدایت تنظیمات خود از آستانه های ولتاژ هدف، حداقل (`min_ta`) و حداکثر (`max_ta`) استفاده می کند. همچنین یک پارامتر `eta_ip` نیز وجود دارد که مربوط به قدرت یا نرخ تنظیمی است که تغییرات در آن اعمال می شود. این ساز و کار بررسی می‌کند که آیا ولتاژ نورون از آستانه‌های تعیین‌شده فراتر می‌رود یا از آن پایین‌تر می‌آید و بر این اساس تنظیم می‌کند. تنظیمات به طور مستقیم از ولتاژ نورون کم می شود، (تحت تاثیر پارامتر `eta_ip`).

مقایسه

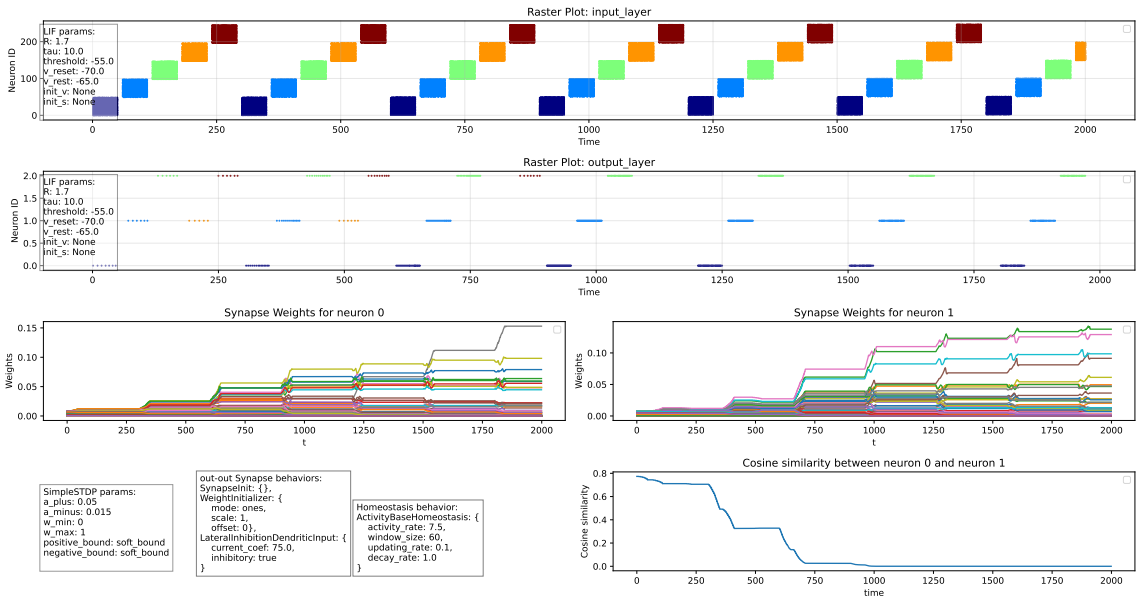
- **تمرکز:**هومئوستازی مبتنی بر فعالیت بر روی سطح فعالیت (نرخ زدن) نورون ها تمرکز دارد، در حالی که هومئوستازی مبتنی بر ولتاژ بر حفظ ولتاژ نورون در محدوده های خاص تمرکز دارد.
- **تطبیق پذیری:** هومئوستازی مبتنی بر فعالیت دارای یک نرخ تطبیقی است (`updating_rate` که با `decay_rate` به‌روزرسانی می شود)، که می تواند آن را پویاتر کند و به تغییرات طولانی مدت در رفتار نورون پاسخ دهد. هومئوستازی مبتنی بر ولتاژ از یک نرخ ثابت (`eta_ip`) استفاده می کند، که ممکن است در طول زمان کمتر به تغییرات پاسخ دهد مگر اینکه به صورت دستی تنظیم شود. (این مورد را زمانی که شبیه سازی را تا مراحل بیشتری ادامه می‌دهیم شاهد هستیم. شکل ۷.۲ را ببینید.)
- **پیچیدگی:** رویکرد مبتنی بر فعالیت به دلیل نیاز به شمارش ضربه ها و محاسبه به روزرسانی ها ممکن است به طور بالقوه پیچیده تر باشد. رویکرد مبتنی بر ولتاژ مستقیماً ولتاژ جریان را با محدوده‌های تنظیم شده مقایسه می‌کند، که ممکن است از نظر محاسباتی ساده‌تر باشد.
- **کاربرد:** انتخاب بین این روش ها به نیازهای خاص شبیه سازی بستگی دارد. به عنوان مثال، اگر تمرکز بر پایداری شبکه در برابر تغییرات ناگهانی در فعالیت ضربه ها باشد، روش مبتنی بر فعالیت ممکن است ترجیح داده شود. اگر اطمینان از عملکرد نورون‌ها در محدوده ولتاژ حیاتی‌تر باشد (احتمالاً برای جلوگیری از اشباع یا نورون‌های غیرفعال)، روش مبتنی بر ولتاژ مناسب‌تر خواهد بود. (مثلا هنگامی که یک جریان یا فعالیت زمینه داریم.)

۴.۲ (امتیازی) آزمایش با نسبت های مختلف الگو ها و لایه خروجی

در این قسمت، آزمایش را به ازای نسبت های مختلف تعداد نورون های لایه خروجی و تعداد الگو ها انجام می‌دهیم. در حالت کلی، سه حالت داریم:

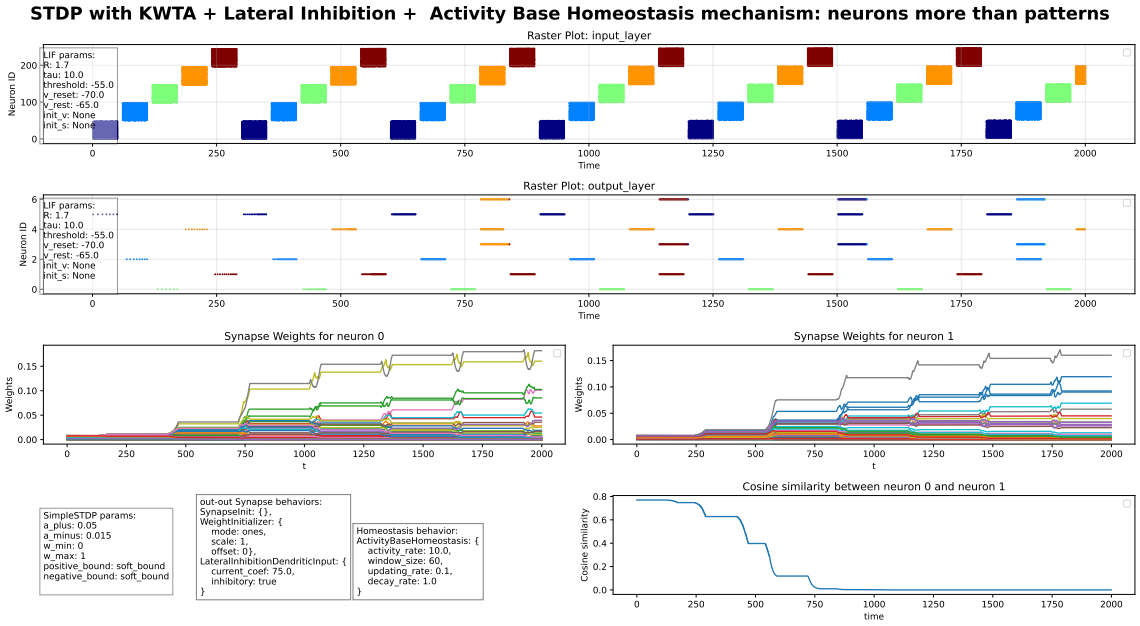
- تعداد الگو ها بیشتر از تعداد نورون های لایه خروجی باشد
- تعداد الگو ها با نورون های لایه خروجی برابر باشد. (این آزمایش به صورت ضمنی در بخش های قبل انجام شد)
- تعداد الگو ها کمتر از تعداد نورون های لایه خروجی باشد

ابتدا حالتی را بررسی می‌کنیم که تعداد الگو ها کمتر از تعداد نورون های لایه خروجی باشد. برای این آزمایش، از یک شبکه با دو لایه و یک سیناپس از لایه ورودی به خروجی که از قانون یادگیری STDP استفاده می‌کند و همچنین ساز و کار مهار جانبی و k-winners-take-all دارد استفاده می‌کنیم. ساز و کار هومئوستازی مبتنی بر فعالیت نیز در لایه خروجی استفاده می‌شود. مطابق شکل ۸.۲ مشاهده می‌کنیم که در این حالت، در ابتدا بعضی نورون ها به ازای دو الگو ضربه می‌زنند و رفته رفته برای یکی از این دو الگو حساس تر شده و فقط هنگامی که آن الگو ورودی داده می‌شود فعال می‌شوند.



شکل ۸.۲: **ساز و کار، مهار جانبی، k-winners-take-all و هومئوستازی مبتنی بر فعالیت: تعداد الگو ها ۵ عدد و تعداد نورون های خروجی ۳ عدد است.** مطابق شکل مشاهده می‌کنیم که در این حالت، در مراحل اولیه شبیه سازی بعضی از نورون های لایه خروجی به ازای دو الگو فعال می‌شوند و هر چه مراحل بیشتری می‌گذرد نسبت به یکی از این دو الگو حساس تر شده و در نهایت فقط هنگامی که آن الگو ورودی داده می‌شود ضربه می‌زنند.

حال آزمایش را برای حالتی که تعداد الگو ها از تعداد نورون های لایه خروجی کمتر است تکرار می‌کنیم و نتیجه را بررسی می‌کنیم. در این حالت مطابق شکل ۹.۲ مشاهده می‌کنیم که بیشتر بودن تعداد نورون ها نسبت به الگو ها باعث می‌شود که به اندازه تعداد الگو ها، نورون های خروجی به الگو های متناظر حساس شوند و نورون های دیگر نیز ممکن است به یک الگوی دیگر مجددا حساس شده یا کلا ضربه‌ای نزنند و غیر فعال شوند.



شکل ۹.۲: **ساز و کار، مهار جانبی، k-winners-take-all و هومئوستازی مبتنی بر فعالیت: تعداد الگو ها ۵ عدد و تعداد نورون های خروجی ۷ عدد است.** مطابق شکل مشاهده می‌کنیم که بیشتر بودن تعداد نورون ها نسبت به الگو ها باعث می‌شود که به اندازه تعداد الگو ها، نورون های خروجی به الگو های متناظر حساس شوند و نورون های دیگر نیز ممکن است به یک الگوی دیگر مجددا حساس شده یا کلا ضربه‌ای نزنند و غیر فعال شوند.

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] Lateral inhibition. Wikipedia [Link]

[۵] Unsupervised Learning of Phase-Change-Based Neuromorphic Systems. Wozniak, Stanislaw Andrzej