دانشگاه شهید بهشتی
دانشکده علوم ریاضی
گروه علوم کامپیوتر

پایان‌نامه کارشناسی ارشد

عنوان

تشخیص ژست‌های حرکتی با کمک شبکه‌های ضربه‌ای کانولوشنال

نگارش
عارف مقدم مهر

استادان راهنما
آقای دکتر هادی فراهانی و آقای دکتر سعیدرضا خردپیشه

بهمن ماه ۱۳۹۹

تقدیم به مسافران پرواز ۷۵۲...

رسیدن، به دانش است و به کردار نیک...

و بی دانش به کردار نیک هم توان رسید، که نیکی را پیشتر باید شناختن، آنگاه بجای آوردن. پس دانش به همه حال می باید تا به رستگاری توان رسیدن. و چون دانش راه آمد، به بهترین چیزها که آدمی را تواند بودن. و در اوّل آفرینش حاصل نیست و بعضی از آن بی رنج و اندیشه حاصل شود، پس هر آینه مهمتر چیزی باشد که در حاصل کردنش عمر گذرانند، لیکن برخی هست که بی اندیشه حاصل آید و بعضی را ناچار به اندیشه حاجت بود، و آنچه به اندیشه حاصل شود دانسته ای خواهد که درو اندیشه کنند تا این نادانسته بدان اندیشه که در آن دانسته کنند دانسته شود، و از هر دانسته هر نادانسته را توان شناخت، بلکه هر نادانسته را به دانسته ای که در خور او بود توان شناخت. و منطق آن علم است که درو راه انداختن نادانسته به دانسته دانسته شود...

پس منطق ناگزیر آمد بر جوینده ی رستگاری.

[1]

[1] مقدمه‌ی رساله‌ی منطق دانشنامه‌ی علائی، شیخ‌الرئیس ابن‌سینا

سپاس‌گزاری...

در آغاز وظیفه‌ی خود می دانم که از زحمات بی دریغ، تلاش های بی وقفه و راهنمایی های حکیمانه استاد راهنمای محترم جناب آقای دکتر فراهانی کمال تشکر و قدردانی را داشته باشم. همچنین از استاد گرامی جناب آقای دکتر خردپیشه که مشاوره این پژوهش را پذیرا شدند و در این مسیر از هیچ همکاری و کمکی به اینجانب دریغ نفرمودند؛ صمیمانه تشکر و قدردانی می نمایم. از استادان گرامی که در طول دوران تحصیلم افتخار کسب علم را در محضرشان داشته ام، جناب آقایان دکتر اصلاحچی و دکتر کتانفروش نهایت سپاسگزاری را دارم.

همچنین کمال تشکر را از گروه ریاضی دانشکده‌ی ریاضی دانشگاه شهید بهشتی دارم، برای مهیا کردن بسترهای پردازشی مناسب و سرورهای پردازش که فرصت آموزش شبکه‌های عصبی پیچیده تر را به ما داد. بدون چنین بسترهای پردازشی، به ثمر رسیدن این رساله ممکن نبود.

و نیز از دوست عزیز و گرانقدرم خانم زهرا نوری، بابت حمایت‌های بی دریغش در پیشبرد این تحقیق و کمک در نگارش و بازخوانی این رساله قدردان و سپاسگزار هستم.

و در آخر بوسه می‌زنم بر دستان پر مهر پدر و مادر عزیزم که در این راه همواره کنارم بوده‌اند.

عارف مقدم مهر

بهمن ماه ۱۳۹۹

نام خانوادگی دانشجو: مقدم‌مهر نام: عارف

عنوان: تشخیص ژست‌های حرکتی با کمک شبکه‌های ضربه‌ای کانولوشنال

استادان راهنما: آقای دکتر هادی فراهانی و آقای دکتر سعیدرضا خردپیشه

مقطع تحصیلی: کارشناسی ارشد رشته: علوم کامپیوتر گرایش: الگوریتم و نظریه محاسبه

دانشگاه: شهید بهشتی علوم ریاضی
تاریخ فارغ‌التحصیلی: بهمن ماه ۱۳۹۹ تعداد صفحات: ۶۶

واژگان کلیدی: شبکه‌های ضربه‌ای، شبکه‌های کانولوشنال، تشخیص ژست‌های حرکتی، بینایی ماشین

چکیده

یکی از انواع شبکه‌های عصبی، شبکه‌های عصبی ضربه‌ای یا اسپایکی است. این شبکه‌ها با الهام گیری از نورون‌های بیولوژیکی، شبکه‌هایی درست کرده‌اند که به جای مقادیر حقیقی، با پالس‌های الکتریکی اطلاعات را پردازش و منتقل میکنند. شبکه‌های اسپایکی در سال‌های اخیر پیشرفت‌هایی زیادی داشته. که میتوان به توان پردازشی بالا و مصرف پایین آنها اشاره کرد. در این پایان نامه بر روی مسئله‌ی تشخیص ژست‌های حرکتی تمرکز شده که کاربردهای زیادی در زمینه‌های متنوع از جمله تعامل انسان و روبات دارد. این ژست‌ها توسط حسگر بینایی پویا تهیه شده‌اند و خود ماهیت اسپایکی دارند. با بهره گیری از شبکه‌های اسپایکی استفاده شده است و با تلفیق آنها با شبکه‌های کانولوشنال و همچنین پیاده سازی ابزارهای کمکی دقت یادگیری آنها را افزایش دادیم، و به دقت ۹۷٪ در مجموعه‌ی داده دست یافتیم.

فهرست مطالب

مقدمه ۱

۱ شبکه‌های اسپایکی ۳

۱.۱ ساختار مغز ۳

۲.۱ روش‌های کدگذاری اسپایکی ۵

۱.۲.۱ روش کد گذاری براساس نرخ اسپایک ۵

۲.۲.۱ روش کد گذاری بر اساس زمان اسپایک ۶

۳.۱ مدل‌های متداول نورون اسپایکی ۶

۱.۳.۱ مدل تجمیع_آتش با نشتی ۶

۲.۳.۱ مدل تجمیع_آتش ۸

۴.۱ آموزش شبکه‌های اسپایکی ۸

۱.۴.۱ روش‌های آموزش محلی ۸

۲.۴.۱ روش باینری کردن شبکه‌های عصبی مصنوعی ۹

۳.۴.۱ روش تبدیل شبکه‌های عصبی مصنوعی به اسپایکی ۹

۵.۱ روش آموزش نظارت شده ۹

۱.۵.۱ نرم کردن شبکه‌ی اسپایکی ۱۰

۲.۵.۱ استفاده از توابع اسپایکی نرم مشتق پذیر ۱۰

۳.۵.۱ گرادیان در مدل‌های احتمالی ۱۱

۴.۵.۱ گرادیان در شبکه‌های کدگذاری نرخ_اسپایک ۱۱

۵.۵.۱ گرادیان در شبکه‌های تک_اسپایک (کدگذاری زمان اسپایک) ۱۲

۶.۵.۱ روش گرادیان جایگزین ۱۲

۲ تعریف مسئله ۱۵
۱.۲ تشخیص ژست حرکتی ۱۵
۲.۲ مروری بر تحقیق‌های انجام شده ۱۷

۳ روش شناسی و پژوهش ۲۳
۱.۳ آماده‌سازی داده‌ها ۲۳
۱.۱.۳ تبدیل تصویر مبتنی بر پیکسل به تصویر اسپایکی ۲۴
۲.۱.۳ فریم بندی جریان اسپایک ۲۴
۳.۱.۳ تقویت و اضافه کردن داده ۳۰
۲.۳ معماری شبکه ۳۰
۱.۲.۳ لایه کامل ۳۱
۲.۲.۳ لایه کانولوشن ۳۲
۳.۲.۳ لایه‌های Max-Pooling و Avg-Pooling ۳۲
۴.۲.۳ عدم استفاده از ضریب جریان سیناپس (α) ۳۲
۵.۲.۳ نرمال کردن اسپایک‌های خروجی ۳۳
۶.۲.۳ وزن‌های اولیه ۳۴
۷.۲.۳ لایه DropOut ۳۵
۸.۲.۳ اتصالات جانبی ۳۶
۹.۲.۳ اتصالات بازگشتی ۳۶
۱۰.۲.۳ استفاده از مکانیزم Winner-Takes-All ۳۷
۳.۳ تصمیم گیری و محاسبه‌ی تابع خطا ۳۷
۱.۳.۳ تصمیم گیری در لایه خروجی ۳۷
۲.۳.۳ محاسبه‌ی تابع خطا ۳۹
۴.۳ پیاده سازی ۴۱
۵.۳ نتایج ۴۴
۱.۵.۳ آزمایش شبکه‌ی کامل ۴۴
۲.۵.۳ آزمایش روش‌های تولید تصویر مبتنی بر فریم ۴۶

۳.۵.۳ آزمایش طول زمانی داده‌ها ۴۸
۴.۵.۳ آزمایش نوع خروجی ۴۸
۵.۵.۳ آزمایش اتصالات کمکی ۴۸
۶.۵.۳ مقایسه شبکه‌ی اسپایکی با شبکه‌ی مصنوعی ۴۹
۷.۵.۳ نتیجه گیری ۴۹

واژه‌نامه فارسی به انگلیسی ۵۱

واژه‌نامه انگلیسی به فارسی ۵۵

نمایه ۵۹

مراجع ۵۹

مقدمه

هدف اصلی این پایان نامه، تحلیل و بررسی شبکه‌های عصبی اسپایکی و ارائه روشی برای آموزش ژست‌های حرکتی با استفاده از این شبکه‌ها است. همان طور که میدانیم در سال‌های اخیر، شبکه‌های عصبی پیشرفت‌های چشم گیری داشته‌اند و امروزه در مسائل مختلف کاربرد دارند. یکی از انواع شبکه‌های عصبی، شبکه‌های اسپایکی است که مدل نورونی آنها شباهت بیشتری به نورون‌های بیولوژیکی دارد. این شبکه‌ها از قدرت بالایی برخوردار هستند و در سال‌های اخیر مورد توجه قرار گرفته‌اند. در این پایان نامه، در بخش اول به توضیح این شبکه‌ها می‌پردازیم، سپس در بخش دوم به خلاصه‌ای از تحقیقات اخیر در این زمینه را مرور می‌کنیم. نهایتا در بخش پایانی به توضیح روش پیاده سازی شده و نتایج آن خواهیم پرداخت.

فصل ۱

شبکه‌های اسپایکی

۱.۱ ساختار مغز

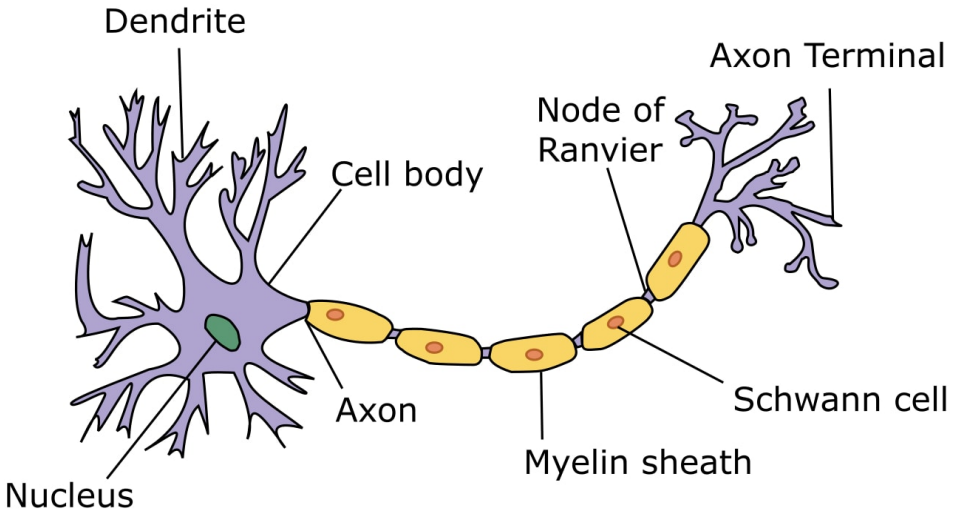
هدف این پایان نامه استفاده از شبکه‌های اسپایکی برای حل مسائل بینایی ماشین است. از آنجایی که این شبکه‌ها از سیستم عصبی بیولوژیکی الهام گرفته‌اند، داشتن اطلاعات در مورد این سیستم‌ها می‌تواند بسیار مفید باشد. به طور خلاصه سیستم عصبی موجودات زنده، (مغز یا همان سیستم پردازش اطلاعات) از مجموعه‌ای در هم تنیده‌ای از عصب‌ها تشکیل شده‌اند. به این سلول‌های عصبی نورون هم گفته می‌شود. این سلول‌ها در موجودات زنده شامل یک هسته‌ی مرکزی و تعدادی ترمینال ورودی [۱] که در این پایان نامه آنها را سیناپس خواهیم نامید و یک ترمینال خروجی[۲] می‌باشند. تصویر ساده شده‌ی یک نورون عصبی در شکل ۱.۱ قابل مشاهده است. این شبکه‌ها بر مبنای پالس‌های الکتریکی منتقل شده و بین آنها کار می‌کنند. به عبارت دیگر، در صورتی که ولتاژ ورودی از سیناپس‌ها، به حد مورد انتظار سلول برسد، سلول تحریک شده و یک پالس عصبی در ترمینال خروجی خود تولید می‌کند. سپس این پالس‌ها به سیناپس سلول‌های دیگر برخورد کرده و مجموعه‌ای از پالس‌های دیگر را در پی دارد که نهایتا منجر به عمل کرد خردمند مغز می‌شود.

قسمت بینایی مغز پستانداران، دارای ساختار سلسله مراتبی است [۱۲]. بدین معنی که اطلاعات را در طی لایه‌های متوالی تحلیل می‌کند و آن‌ها را از لایه‌ای به لایه بعدی انتقال می‌دهد. این لایه‌ها که در قسمت شکمی قشر بینایی یا Cortex Visual the of Pathway Ventral قرار دارد و هر کدام وظیفه تحلیل تصویر در سطح مشخصی را دارد [۸]. نورون‌ها در طول این سلسله مراتب‌ها، به مرور از تشخیص اشکال ساده مثل خطوط و زاویه‌های تصویر در قسمت‌های ابتدایی قشر بینایی تا تشخیص اشیاء در قشر تحلیلی می‌پردازد. [۷]

تفاوت عمده‌ی شبکه‌های عصبی مصنوعی و شبکه‌های عصبی اسپایکی در نحوه‌ی مدل سازی

[۱] به این ترمینال‌ها سیناپس یا دندریت هم گفته می‌شود.
[۲] به این ترمینال‌ها اکسون هم گفته می‌شود.

نورون‌ها می‌باشد. هرچند ساختار شبکه‌های عصبی مصنوعی الهام گرفته از مغز موجودات زنده است، [۱۸] ولی در واقع این شبکه‌ها شباهت خیلی زیادی به سیستم مغزی موجودات زنده ندارند چرا که در این شبکه‌ها کاملا از ساختار و مدل نورون‌ها صرف نظر شده است و فقط به تقلید ارتباط نورون‌ها و ساختار شبکه تمرکز داشته‌اند. نورون‌های مصنوعی دارای خروجی تابع پیوسته هستند در حالی که نورون‌های طبیعی دارای خروجی تابع ضربه یا اسپایک می‌باشند. تحقیقات انجام شده نشان می‌دهد که مغز پستانداران با نرخ بسیار آهسته‌ای اسپایک می‌زند. ولی کدگذاری اطلاعات در زمان اسپایک زدن، باعث می‌شود همین مقدار اسپایک برای انتقال اطلاعات کافی باشد. این امر باعث می‌شود مغز ما بسیار سریع و بسیار کم مصرف باشد [۱۶]. پیوند این عصب‌ها با یکدیگر، شبکه‌های عصبی پیچیده‌تر و قدرتمندتری را به وجود می‌آورد. این شبکه‌ها که امروزه به اسم شبکه‌های کامل، شبکه‌های رو به جلو یا شبکه‌های چند لایه شناخته می‌شوند، عموما از لایه‌های به هم چسبیده تشکیل شده‌اند و توانایی به مراتب بالاتری نسبت به یک عصب تنها را دارند.



نمودار ۱.۱: تصویر ساده شده‌ی یک سلول عصبی در مغز موجودات زنده.

یکی از زیر شاخه‌های شبکه‌های عصبی، شبکه‌های عصبی ضربه‌ای یا اسپایکی است که در سال‌های اخیر بسیار مورد توجه قرار گرفته‌اند. این شبکه‌ها در مقایسه با شبکه‌های عصبی مصنوعی [۳]، شباهت بیشتری به مغز بیولوژیکی دارند. برای درک بهتر موضوع می‌توانیم نورون‌های مصنوعی متداول و سلول‌های عصبی زیستی را با هم مقایسه کنیم.

تفاوت عمده‌ی این دو در نحوه‌ی تولید خروجی آنهاست. برخلاف نورون‌های مصنوعی که خروجی آنها پیوسته است، خروجی سلول‌های عصبی زیستی از نوع پالس‌های الکتریکی می‌باشد. در واقع این سلول‌ها با دریافت پالس‌های الکتریکی از سیناپس‌های ورودی خود، افزایش پتانسیل پیدا می‌کنند و پس از این که پتانسیل آنها به مقدار آستانه تحمل آنها رسید، سلول عصبی یک پالس الکتریکی منتشر می‌کند،

[۳] اگرچه استفاده از واژه‌ی شبکه عصبی مصنوعی درست نیست، چرا که هر دوی این شبکه‌ها مصنوعی هستند، ولی در این پایان نامه مقصود از شبکه‌ی عصبی مصنوعی، شبکه‌هایی هستند که به نسبت شباهت کمتری به مغز انسان دارند و در واقع دارای خروجی حقیقی و هستند و در مقابل آنها شبکه‌ی عصبی اسپایکی قرار دارد که شباهت بیشتری به مغز انسان دارد.

که باعث تخلیه شدن پتانسیل آن می‌شود. شبکه‌های عصبی اسپایکی هم بر همین مبنا مدل شده‌اند.

بنابراین، شاید مهم‌ترین وجه تمایز این دو شبکه همین است که تعریف شبکه‌های اسپایکی به وابسته به زمان است. شبکه‌های اسپایکی در بعد زمان تعریف می‌شوند و بر خلاف شبکه‌های مصنوعی که مسئله را در یک لحظه بررسی می‌کنند و جواب را برای همان لحظه تولید می‌کنند، شبکه‌های اسپایکی در طول زمان خروجی خود را تولید می‌کنند. برای مثال در لحظه‌ی t_0 شروع به ورودی دادن به شبکه می‌کنیم. اگرچه در این لحظه شبکه هیچ خروجی تولید نمی‌کند، ولی پس از گذشت $_1k$ واحد زمانی، شبکه اولین اسپایک خود را تولید می‌کند و پس از گذشت k_2 واحد زمانی به حد کافی اسپایک تولید کرده باشد تا بتوانیم برای خروجی تصمیم گیری و نتیجه گیری کنیم. همین امر باعث می‌شود تا دقت این شبکه‌ها بنا به نیاز، قابل تغییر باشند. به عبارت دیگر در صورتی که نیاز مبرم به تولید خروجی هرچند نادقیق داشته باشیم می‌توانیم در زمان زودتر اقدام به نتیجه گیری کنیم.

همان طور که قبلا اشاره شد، این شبکه‌ها الهام گرفته از مغز موجودات زنده هستند و به دلیل خاصیت زمانی نهادینه شده در آنها، این شبکه ها را می‌توان در طبقه بندی شبکه‌های بازگشتی هم قرار داد چرا که این دو، خاصیت‌ها و کاربردهای مشابه بسیاری دارند. در ادامه به بررسی جزئیات بیشتر این شبکه‌ها می‌پردازیم. این جزئیات شامل روش‌های کدگذاری داده‌ها و اسپایک‌ها و همچنین مدل دینامیکی درون نورون است.

۲.۱ روش‌های کدگذاری اسپایکی

برخلاف نورون‌های عصبی مصنوعی که دارای تابع فعال سازی با خروجی حقیقی هستند، نورون‌های اسپایکی دارای تابع فعال سازی با خروجی پالس یا ضربه می‌باشند که این پالس‌ها، دارای مقدار و طول یکسان برای تمام نورون‌ها هستند. بنابراین این مقدار به خودی خود نمی‌تواند اطلاعاتی را انتقال دهد. به همین دلیل شبکه‌های اسپایکی در بعد زمان کار می‌کنند تا اطلاعات را در زمان اسپایک زدن ذخیره کنند. نحوه‌ی ذخیره این اطلاعات نوع کد گذاری شبکه نامیده می‌شود که یکی از پارامترهای مهم و تعیین کننده‌ی شبکه می‌باشد. برای کد گذاری اطلاعات در این نورون‌ها دو روش کلی وجود دارد، روش کدگذاری بر اساس نرخ اسپایک و روش کدگذاری بر اساس زمان اسپایک.

۱.۲.۱ روش کد گذاری براساس نرخ اسپایک

در این روش، کد گذاری اطلاعات در فرکانس و یا در نرخ اسپایک ذخیره می‌شود. به عبارت دیگر هر چقدر نورون با فرکانس بالاتری اسپایک بزند به این معنا است که فعالیت بیشتری دارد. در واقع می‌توان فرکانس اسپایک در این روش را با مقدار خروجی حقیقی نورون‌های مصنوعی متناظر دانست. یکی از مثال‌های تبدیل خروجی نورون‌های ReLU به فرکانس نورون‌های اسپایکی است. بدیهی است که در این روش نورون‌ها می‌توانند بیش از یک بار اسپایک تولید کنند که به طبع آن، شبکه توانایی پردازش جریان

داده‌ها را خواهد داشت. یکی از مزیت‌های این روش این است که مدل انتشار اسپایک‌ها نسبت به روش زمان اسپایک، بیشتر شبیه مغز ما است. به همین دلیل این مدل‌ها کمک بیشتری در تحلیل مکانیزم مغز ما می‌کنند. همچنین این مدل‌ها توانایی پردازش متوالی داده‌ها بدون نیاز به بازنشاندن شبکه دارند.

۲.۲.۱ روش کد گذاری بر اساس زمان اسپایک

در مقابل، روش زمان اسپایک اطلاعات را از طریق زمان انتشار اسپایک‌ها منتقل می‌شود. بدین معنی که زمان اولین اسپایک نشان دهنده‌ی فعالیت نورون است. هر چقدر این زمان به صفر نزدیکتر باشد نشان دهنده‌ی فعالیت بیشتر نورون و هر چقدر این زمان بزرگتر باشد نشان دهنده‌ی فعالیت کمتر آن است. در صورتی که نورون هیچ اسپایکی تولید نکند نشان دهنده‌ی فعالیت صفر نورون است.

بدیهی است که در این روش تنها اولین اسپایک در نظر گرفته می‌شود پس اسپایک‌های بعدی هیچ ارزش اطلاعاتی ندارند. به همین علت در پیاده سازی‌های این روش تنها یک اسپایک برای هر نورون در نظر گرفته می‌شود و هر نورون حداکثر می‌تواند یک بار اسپایک تولید کند.

این مدل‌ها نسبت به مدل‌های نرخ اسپایک ساده تر می‌باشند و تعداد اسپایک کمتری را تولید می‌کنند. بنابراین می‌توان گفت این مدل‌ها دارای مصرف انرژی بهینه تری می‌باشند. ولی از طرف دیگر این مدل‌ها به خودی خود توانایی پردازش جریان داده‌ها را ندارند. برای حل این مشکل نیاز داریم، مکانیزمی برای بازنشاندن شبکه و نورون‌ها در نظر بگیریم. برای مثال پس از حل هر نمونه آزمایشی و یا بعد از تحلیل هر سیگنال، پتانسیل نورون‌ها را با سیگنال مشخصی، به صفر بازگردانیم تا برای اجرای دوباره آماده باشند.

۳.۱ مدل‌های متداول نورون اسپایکی

در این قسمت به بررسی مدل دینامیکی درون نورون‌ها پرداخته و نحوه‌ی کارکرد پتانسیل درونی آنها را بررسی می‌کنیم.

۱.۳.۱ مدل تجمیع‌ـآتش با نشتی

متداول ترین روش برای مدل نورون‌های اسپایکی مدل تجمیع‌ـآتش با نشتی یا Leaky Integrate-and-Fire می‌باشد [۱۱] که به طور خلاصه LIF نامیده می‌شوند. این نورون‌ها شباهت بسیار زیادی به مدل مغز ما دارد. معادله‌ی ۱.۱ مدل این نورون‌ها را در غیاب مکانیسم اسپایک که باعث بازگشتن پتانسیل به مقدار اولیه می‌شود را نشان می‌دهد.

$$\tau_{mem}\frac{dU_i}{dt} = RI_i - (U_i - U_{rest}) \qquad (۱.۱)$$

در این معادله U_i مقدار پتانسیل جاری نورون را نشان می‌دهد و U_{rest} مقدار پتانسیل استراحت نورون را نشان می‌دهد. مقدار استراحت مقداری است که نورون بدون داشتن ورودی یا خروجی و یا پس از اسپایک زدن به آن می‌رسد. مقدار $\frac{dU_i}{dt}$ نشان دهنده تغییرات لحظه‌ای مقدار پتانسیل است که با استفاده از ثابت τ_{mem} کنترل می‌شود. این ثابت در واقع مقدار نشتی پتانسیل نورون است. I_i مقدار جریان ورودی و R هم مقدار مقاومت آن ورودی می‌باشد. همچنین برای مدل کردن جریان ورودی نورون، از روش دنبال کردن توانی مقدار جریان استفاده میکنیم [۲۸]. برای این مدل از معادله‌ی ۲.۱ استفاده می‌کنیم.

$$\frac{dI_i}{dt} = -\frac{I_i(t)}{\tau_{syn}} + \sum_j W_{ij} S_j(t) \tag{۲.۱}$$

در این معادله، مانند معادله‌ی قبلی سعی داریم تغییرات مقدار جریان را مدل کنیم. این مقدار وابسته به مقدار قبلی جریان است که با ثابت τ_{syn} کنترل می‌شود و مقدار اسپایک‌های ورودی نورون که هر کدام در وزن متناظر خود W ضرب می‌شوند. می‌دانیم که پتانسیل نورون‌های عصبی پس از رسیدن آستانه‌ی تحمل باعث می‌شوند که نورون اسپایک تولید کند و مقدار پتانسیل کاهش پیدا کند. بنابراین با اضافه کردن مدل اسپایک به معادله‌ی ۱.۱ می‌توانیم به مدل ۳.۱ دست پیدا کنیم.

$$\frac{dU_i}{dt} = -\frac{1}{\tau_{mem}}(U_i - U_{rest}) + RI_i + S_i(t)(U_{rest} - \theta) \tag{۳.۱}$$

که در این معادله مقدار $S_i(t)$ بیانگر این است که آیا نورون مورد نظر در زمان t دارای اسپایک خروجی بوده یا نه. این مقدار به صورت ۴.۱ تعریف می‌شود. مقدار θ نیز نشان دهنده‌ی آستانه‌ی تحمل نورون است.

$$S(t) = \begin{cases} 1 & if \;\; U_i >= \theta \\ 0 & otherwise \end{cases} \tag{۴.۱}$$

برای ساده تر شدن محاسبات می‌توان معادلات ۱.۱ و ۲.۱ را به فرم گسسته و به صورت زیر نوشت:

$$I_i[n+1] = \alpha * I_i[n] + \sum_j W_i j S_j[n] \tag{۵.۱}$$

$$U_i[n+1] = \beta * U_i[n] + I_i[n] - S_i[n] \tag{۶.۱}$$

۲.۳.۱ مدل تجمیع ـ آتش

این مدل که به آن Integrate-and-Fire یا IF هم می‌گوییم کاملا مشابه مدل LIF است و تنها تفاوت آن این است که این مدل‌ها نشتی ندارند. برای بدست آوردن معادله این مدل تنها کافیست در مدل ۶.۱ مقدار β را یک در نظر بگیریم در نتیجه از نشتی نورون صرف نظر می‌شود. این مدل‌ها دارای مکانیسم ساده تری برای پردازش اطلاعات دارند و یکی از کاربردهای آنها استفاده در مدل‌های کدگذاری زمان اسپایک می‌باشد.

۴.۱ آموزش شبکه‌های اسپایکی

به دلیل متفاوت بودن دینامیک نورون‌های اسپایکی، و وابستگی آنها به بعد زمان و همچنین به دلیل مشتق پذیر نبودن تابع ضربه نمی‌توان به روش‌های مرسوم این نورون‌ها را آموزش داد. بنابراین روش‌هایی برای حل این مشکل به وجود آمده است که در ادامه به بررسی آنها می‌پردازیم.

۱.۴.۱ روش‌های آموزش محلی

یکی از رایج ترین روش آموزش محلی شبکه‌های عصبی اسپایکی، روش انعطاف پذیری وابسته به زمان اسپایک یا STDP می‌باشد. این روش که زیر مجموعه‌ی روش‌های آموزش بدون نظارت محسوب می‌شود شباهت بسیاری به روش‌های آموزش در مغز دارد. در این روش آموزش هر نورون وابسته به فعالیت‌های جریان ورودی آن در قبل و بعد از لحظه‌ای است که نورون اسپایک می‌زند. به این صورت که وزن ورودی‌هایی که قبل از لحظه‌ی اسپایک، تحریک شده باشند قوی تر می‌شود و وزن ورودی‌هایی که بعد از لحظه‌ی اسپایک تحریک شده باشند، تضعیف می‌شود. مدل ریاضی این روش در معادلات زیر نشان داده شده است.

$$\Delta W_{ij} = \begin{cases} a^{+} w_{ij}(1 - w_{ij}) & if \;\; t_j - t_i \leq 0 \\ a^{-} w_{ij}(1 - w_{ij}) & if \;\; t_j - t_i > 0 \end{cases} \tag{۷.۱}$$

در این معادله i و j به ترتیب سیگنال پسا سیناپسی یا Post-Synaptic (سیگنال مربوط به اسپایک‌های خروجی نورون) و سیگنال پیش سیناپسی یا Pre-Synaptic (سیگنال مربوط به جریان اسپایک‌های ورودی نورون) می‌باشد. وزن‌های a^{+} و a^{-} به ترتیب وزن‌های آموزش مثبت و منفی نورون می‌باشند که به عنوان پارامتر ورودی سیستم تعیین می‌شوند. در واقع این دو مقدار تغییرات نورون را کنترل می‌کنند. عبارت $w_{ij}(1 - w_{ij})$ باعث می‌شود تا مقادیر وزن‌ها در بازه‌ی صفر تا یک باقی بمانند. در واقع این عبارت باعث می‌شود تا مقدار تغییرات در انتهای بازه‌ی صفر تا یک به صفر میل کند.

از مزایای روش‌های STDP می‌توان به بدون نظارت بودن آن اشاره کرد. بنابراین این روش‌ها می‌توانند بدون داشتن برچسب داده‌ها، سعی به آموزش الگوی آنها کنند. اما از طرف دیگر این عدم توانایی یادگیری برچسب‌ها می‌تواند مشکلاتی را نیز به وجود آورد. برای مثال ممکن است شبکه سعی کند الگوهایی را یاد بگیرد که به حل مسئله کمک چندانی نمی‌کند. همچنین از دیگر مشکلات این روش می‌توان به عدم کارایی آن در شبکه‌هایی با لایه‌های عمیق تر نیز اشاره کرد.

۲.۴.۱ روش باینری کردن شبکه‌های عصبی مصنوعی

در این روش تنها تغییری که بر روی شبکه‌های عصبی مصنوعی می‌دهیم تابع فعال سازی آنها است. به طوری که تابع فعال سازی آنها را به تابع ضربه تغییر می‌دهیم. بنابراین این شبکه‌ها برخی از خواص شبکه‌های اسپایکی را دارند از جمله انتقال اطلاعات مابین نورون‌ها به صورت باینری خواهد بود که همین امر موجب افزایش کارایی و کاهش حافظه‌ی مورد نیاز آنها می‌شود.

۳.۴.۱ روش تبدیل شبکه‌های عصبی مصنوعی به اسپایکی

یکی دیگر از روش‌های موجود روش تبدیل شبکه‌های عصبی مصنوعی به شبکه‌های عصبی اسپایکی می‌باشد. برای این کار ابتدا شبکه را با استفاده از داده‌های موجود آموزش می‌دهیم. سپس با استفاده از قیدهای از پیش تعیین شده شبکه را به شبکه‌ی اسپایکی تبدیل می‌کنیم. برای مثال می‌توان تابع تبدیلی تعریف کرد که مقدار حقیقی تابع را به زمان اسپایک تبدیل کند. سپس وزن‌های شبکه را به گونه‌ای انتخاب می‌کنیم که شبکه‌ی عصبی اسپایکی خروجی متناظر با شبکه‌ی عصبی مصنوعی تولید کند.

نکته‌ی مهم و قابل تامل در این زمینه این است که تمام عناصر شبکه‌های مصنوعی قابل تبدیل به شبکه‌های عصبی اسپایکی نیستند. برای مثال لایه‌هایی همچون لایه‌ی Average-Pooling ویا توابع فعال سازی به غیر از تابع ReLU قابل تبدیل به عناصر اسپایکی نیستند. چرا که این عناصر ماهیت حقیقی دارند.

۵.۱ روش آموزش نظارت شده

در قسمت‌های قبلی به روش آموزش نظارت شده اشاره شده است. به طور خلاصه در این روش نیاز داریم تا تابع خطا را طوری تعیین کنیم که خطا روی خروجی مورد نظر کمینه شود. سپس، خطای تولید شده توسط هر نورون را محاسبه می‌کنیم تا بتوانیم وزن‌های نورون‌ها را طوری تغییر دهیم که این خطا کمینه شود و در نتیجه خروجی مورد نظر تولید شود. مسئله‌ای که در شبکه‌های اسپایکی وجود دارد، این است که تابع فعال ساز نورون‌های اسپایکی، تابع ضربه، مشتق پذیر نیست. در واقع مشتق آن در تمامی نقاط صفر می‌باشد و در لحظه‌ای که تابع فعال می‌شود و اسپایک تولید میکند، مشتق آن تعریف نشده است. بنابراین نمی‌توانیم از روش‌های معمول برای بهینه کردن شبکه استفاده کنیم. در قسمت ۲.۳.۳ به

بررسی این موضوع و تعریف دقیق تابع خطا خواهیم پرداخت. برای حل مسئله مشتق پذیر نبودن تابع، چهار راهبرد کلی پیشنهاد شده است:

۱. پیاده سازی کامل فرایندهای بیولوژیکی و آموزش از طریق این روش‌ها، برای مثال استفاده از روش‌های آموزش محلی مانند روش STDP.

۲. تبدیل شبکه‌های اسپایکی به شبکه‌های مصنوعی

۳. نرم کردن شبکه‌ی اسپایکی و استفاده از اسپایک‌های نرم

۴. استفاده از گرادیان جایگزین برای توابع اسپایکی

در قسمت‌های قبلی راجع به مورد اول و دوم به طور مختصر بحث شد. در ادامه به بررسی موارد سوم و چهارم می‌پردازیم.

۱.۵.۱ نرم کردن شبکه‌ی اسپایکی

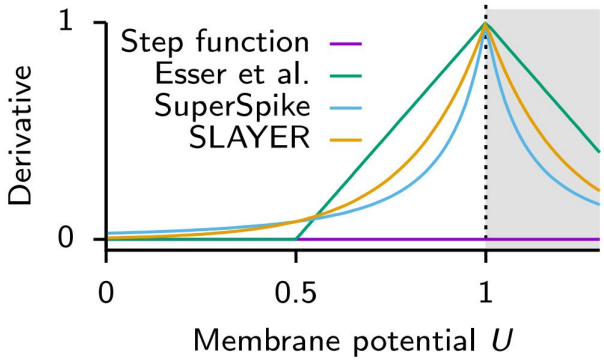
در واقع می‌توان گفت در روش نرم کردم شبکه‌ی اسپایکی از توابع و روش‌هایی استفاده می‌کنیم که مشتق پذیری شبکه را تضمین کنند. برای این کار چهار روش پیشنهاد شده است که عبارتند از: روش استفاده از توابع اسپایکی نرم مشتق پذیر، گرادیان در مدل‌های احتمالی، گرادیان در شبکه‌های کدگذاری نرخ_اسپایک، گرادیان در شبکه‌های تک_اسپایک (کدگذاری زمان اسپایک)، و روش گرادیان جایگزین.

۲.۵.۱ استفاده از توابع اسپایکی نرم مشتق پذیر

این روش را می‌توان بر روی تمامی شبکه‌های اسپایکی پیاده سازی کرد. در این روش به جای استفاده از تابع ضربه (تابع ضربه‌ی سخت) از توابع ضربه‌ی نرم استفاده می‌کنیم. بر خلاف تابع ضربه‌ی سخت که مقدار آن ناپیوسته است و در همه‌ی نقاط به جز $U = \theta$ مقدار صفر دارد، تابع ضربه‌ی نرم مقدار پیوسته‌ای دارد که در تمامی نقاط مشتق پذیر هستند. مقدار این تابع به صورت پیوسته به نقطه‌ی θ می‌رسد. نمونه‌ای از این توابع در شکل ۲.۱ نشان داده شده است.

مزیتی که این روش دارد این است که پس از تغییر تابع، به سادگی می‌توان از روش‌های آموزش شبکه‌های بازگشتی، مانند BPTT [۴] یا RTRL [۵] استفاده کرد و این شبکه‌ها را آموزش داد. مشکلی که این روش دارد این است که شبکه از حالت اسپایکی خارج می‌شود و نورون‌ها دیگر خاصیت باینری ندارند و همانطور که می‌دانیم داشتن خروجی باینری یکی از مهمترین ویژگی شبکه‌های اسپایکی است. بنابراین می‌توان گفت این شبکه‌ها دیگر برخی از خواص شبکه‌های اسپایکی را ندارند و بیشتر به نوعی به شبکه‌ی بازگشتی مصنوعی شباهت دارند.

[۴] Back Propagation Through Time
[۵] Real-Time Recurrent Learning



نمودار ۲.۱: تابع ضربه (مورد اول) و مقایسه‌ی آن با مثال‌هایی از توابع ضربه‌ای نرم. منبع [۲۸]

۳.۵.۱ گرادیان در مدل‌های احتمالی

روش بعدی استفاده از مدل‌های احتمالی برای محاسبه‌ی گرادیان است. در این روش تابع ضربه به صورت احتمالی تعریف می‌شود. بنابراین می‌توان گرادیان تابع را بر روی امید ریاضی به سادگی تعریف کرد که در عین حال تابع دارای خروجی باینری نیز می‌باشد. در مدل‌های [۱۰] ، [۵] و [۲۴] شبکه‌های یک لایه‌ای و چند لایه‌ای از این مدل، مورد بررسی قرار گرفته‌اند. تنها مسئله‌ی این روش این است که نویز وارد شده به سیستم، در هنگام نمونه گیری احتمالی اسپایک، باعث نرم شدن تابع ضربه و مشتق پذیری آن می‌شود، و از طرف دیگر گاهی باعث می‌شود مدل به سختی بهینه شود که همگرایی آن را تحت تاثیر قرار می‌دهد.

۴.۵.۱ گرادیان در شبکه‌های کدگذاری نرخ_اسپایک

یکی از روش‌های مرسوم دیگر برای بدست آوردن مقدار گرادیان در شبکه‌های اسپایکی استفاده مفهوم کدگذاری نرخ_اسپایک است. به طوری که فاکتورهای نورون‌های اسپایکی برای تولید اسپایک، حجم اسپایک ورودی آن نورون است. بنابراین می‌توان گفت که بین تعداد اسپایک‌های ورودی و خروجی رابطه ای برقرار است. این رابطه را می‌توانیم در قالب منحنی جریان_اسپایک یا f-I برای نورون ترسیم کنیم، به طوری که در یک محور جریان ورودی به نورون و در محور دیگر نرخ اسپایک‌های خروجی را رسم میکنیم. در این موارد مشتق منحنی f-I گزینه بسیار مناسب برای بهینه سازی نورون می‌تواند باشد. کارهای مختلفی در این زمینه انجام شده است که بین آنها می‌توان به [۱۵] و [۲۷] اشاره کرد. در هر کدام از این موارد از کدگذاری نرخ اسپایک استفاده کرده‌اند که بر روی مجموعه داده‌های CIFAR-10 و MNIST نتایج بسیار خوبی گرفتند. همچنین در [۱۹] که یک شبکه ی عمیق نشان داده شده است، بر اساس همین روش مشتق گیری آموزش می‌بینند. در این روش بر روی اسپایک‌های ورودی یک فیلتر low-pass قرار داده شده است که نتیجه ی آن را بهتر می‌کند.

روش‌های گرادیان کدگذاری نرخ_اسپایک، می‌توانند بسیار موثر باشند و نتایج خوبی را تولید کنند. هرچند این روش‌ها مشکل‌های دیگری نیز دارند. برای مثال دقت این مدل‌ها می‌توانند تحت شرایط

خاصی بسیار پایین بیاید چرا که برای محاسبه ی منحنی f-I نیاز داریم تا تعداد اسپایک‌های ورودی و خروجی را میانگین بگیریم. این میانگین گیری نیاز دارد تا مقداری زمان از شروع پردازش بگذرد چرا که نویز تولید شده توسط این پروسه به حداقل برسد.

۵.۵.۱ گرادیان در شبکه‌های تک_اسپایک (کدگذاری زمان اسپایک)

در یک تلاش دیگر برای بهینه کردن شبکه‌های اسپایکی، بدون اضافه کردن نویزهای ناخواسته و بدون استفاده از روش‌های کدگذاری نرخ_اسپایک، استفاده از روش کدگذاری تک اسپایک است. در این گونه شبکه‌ها که هر نورون، تنها یک اسپایک تولید می‌کند، هر اسپایک حجم بالایی از اطلاعات را حمل می‌کند. ایده ی اصلی برای این روش در [۴] ارائه شده است. در این روش زمان اسپایک زدن در لایه‌های پنهان شبکه محاسبه می‌شود و برای این مقدار یک تابع خطی را در نظر می‌گیریم. بنابراین می‌توان مشتق این تابع را به سادگی محاسبه کرد. ولی این روش هم مشکلات خود را دارد. اولین و مهمترین مشکل آن این است که این روش فقط برای شبکه‌هایی کارایی دارد که تک_اسپایک هستند. علاوه بر این، مشکل دیگر این شبکه‌ها نورون‌هایی هستند که اسپایک تولید می‌کنند و ساکت هستند. برای این نورون‌ها نمی‌توان مقدار گرادیان را محاسبه کرد و باید روش دیگری برای این نورون‌ها پیاده سازی کنیم.

۶.۵.۱ روش گرادیان جایگزین

روش‌های گرادیان جایگزین، راه حل مناسبی برای مسئله پیوسته نبودن تابع فعال سازی نورون‌های اسپایکی هستند. فلسفه ی اصلی این روش این است که به جای تغییر تابع و گرادیان، روشی پیشنهاد می‌دهیم که گرادیان را نرم تر کند و در نتیجه مقدار آن پیوسته می‌شود. مسئله مهم تر این است که این نرم کردن گرادیان فقط در مرحله ی بهینه سازی پارامترها کاربرد دارد و در مرحله اجرای شبکه تاثیری ندارد. یکی دیگر از خواص مهم این روش این است که می‌تواند شبکه را بدون نیاز به مشخص کردن روشهای کد گذاری و سایر پارامترها آموزش دهد. در واقع این روش نیاز به دانستن این پارامترها ندارد. روش گرادیان جایگزین این قابلیت را دارد تا همانند روش کاهش گرادیان معمولی بتواند از متدهایی مانند BPTT بهره بگیرد.

تا به حال تحقیق‌های متعددی از روش گرادیان جایگزین استفاده کرده‌اند تا بتوانند با استفاده از روش BPTT شبکه ی اسپایکی را آموزش دهند. این کار تنها با تغییر جزیی در شبکه به وجود می‌آید و آن این است که در هر حلقه ی اجرای الگوریتم آموزش، از مقدار گرادیان یک تابع دیگر به جای گرادیان تابع اسپایکی استفاده می‌کنیم. این روش را حتی می‌توان در ابزارهای جدید که از تکنیک‌های مشتق خودکار استفاده می‌کنند نیز استفاده کرد. یکی از اولین استفاده‌ی این روشها در [۳] پیاده سازی شده است. که در آن گرادیان تابع اسپایکی با گرادیان توابع چند جمله ای کوتاه شده جایگزین شده است که نتیجه ی آن مشتق تابع ReLU به وجود آمده است. همچنین در تحقیقات [۶] و [۹] نیز از روش گرادیان جایگزین الهام گرفته شده تا شبکه‌های بزرگتر را با لایه‌های کانولوشنال با تابع اسپایکی آموزش دهند. در [۳۳]

نیز یک روش آموزش سه فاکتوره ارائه کردند که از قسمت منفی تابع سیگموید استفاده می‌کند و با آن گرادیان جایگزین را می‌سازد. همچنین در [۲] نیز با استفاده از همین قسمت منفی تابع سیگموئید، شبکه ای با خاصیت زمانی را با موفقیت آموزش می‌دهد. در این تحقیق نویسنده پی می‌برد که می‌توان از این روش استفده کرد تا شبکه ی کانولوشنال با لایه‌های حافظه ای کوتاه_بلند مدت را آموزش داد. نهایتا در [۳۲] از تابع نمایی برای گرادیان جایگزین استفاده شده که در آن نویسنده به نتایج بسیار خوبی نسبت به روش‌های سخت افزارهای نورومورفیک پیشین، رسیده است.

به طور خلاصه در تحقیقات متعددی از روش گرادیان جایگزین استفاده کرده‌اند که در آنها از توابع فعال سازی مختلف و توابع جایگزین مختلفی استفاده شده است که در آنها شبکه‌هایی با ساختارهای گوناگون مورد بررسی قرار گرفته است. هر چند تمام این توابع فعال سازی، همگی یک وجه مشترک دارند و آن این است که همه آنها به طور یکنواخت نسبت به نقطه ی اسپایک افزایش مقدار می‌دهند.

فصل ۲

تعریف مسئله

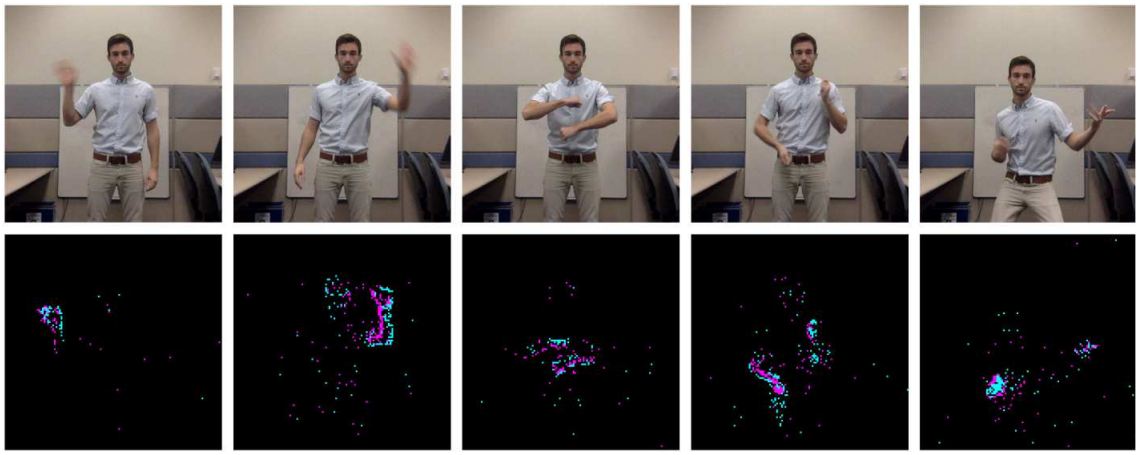
مسئله‌ی انتخاب شده برای این پایان نامه تشخیص ژست حرکتی توسط شبکه‌های اسپایکی می‌باشد. در این قسمت به بررسی صورت مسئله و هدف از حل این مسئله می‌پردازیم.

۱.۲ تشخیص ژست حرکتی

یکی از شاخه‌های بینایی ماشین، مبحث تشخیص ژست حرکتی می‌باشد که در آن به کمک روش‌های یادگیری ماشین قصد داریم ژست‌های حرکتی انسان را تشخیص دهیم. برای درک صورت مسئله ابتدا باید لغت ژست حرکتی را تعریف کنیم. در اینجا ژست که ترجمه‌ی لغت Gesture است، به معنی موقعیت فیزیکی جسم است که می‌تواند اطلاعاتی از وضعیت درونی آن جسم بیان کند. جسم فیزیکی مورد بررسی در اینجا انسان است و هدف از این ژست بیان اطلاعات و یا درخواست‌های او است. یک مثال ساده از این گونه ژست‌ها، لبخند می‌باشد که بیانگر حس شادی و شعف در فرد می‌باشد و می‌تواند اطلاعات مفیدی را در تعامل بین دو فرد منتقل کند. ما همواره در حال تعامل و انتقال اطلاعات با استفاده از این ژست‌ها هستیم.

تشخیص این ژست‌ها می‌تواند در زمینه‌های مختلفی کاربرد داشته باشد. یکی از مثال‌های این مسئله می‌تواند تحلیل حرکتی انسان بوده و حل این مسئله می‌تواند برای موارد پزشکی سودمند باشد، یکی از این مثال‌ها برنامه‌هایی هستند که سعی می‌کند تا با استفاده از تحلیل اسکلت بندی و حرکتی فرد به دنبال مشکل در سیستم حرکتی او شوند [۲۹] و [۲۶]. مثال دیگر کاربرد این مسائل می‌تواند در تعامل انسان و ربات بوده که خود شاخه‌ی بسیار بزرگی را تشکیل می‌دهد. در این گونه مسائل به دنبال این هستیم که ربات بتواند برداشتی از حرکت انسان داشته باشد. برای مثال دستوری را که سوژه‌ی انسانی مد نظر دارد را تشخیص دهد و آن را اجرا کند. [۲۲] و [۳۱]

در ادامه به بررسی ژست‌های حرکتی خواهیم پرداخت. این ژست‌ها به حرکت‌هایی گفته می‌شوند که هر بخش آن به تنهایی، و در حالت سکون، معنی دیگری ممکن است داشته باشد ویا معنی خاصی ندهد



نمودار ۱.۲: در این تصویر نمونه‌هایی از مجموعه داده‌ی IBM-DVSGesture مشاهده می‌شود. تصاویر بالا با دوربین مبتنی بر پیکسل عادی تهیه شده و تصاویر پایین مربوط به همان سوژه است که با حسگر DVS گرفته شده است. منبع: [۱]

ولی مجموعه‌ی به هم پیوسته‌ی این ژست‌ها معنی واحدی می‌دهند. برای مثال می‌توان تکان دادن دست در هوا را در نظر گرفت که معنی خداحافظی می‌دهد.

به طور کلی تشخیص ژست‌ها می‌توانند کاربردهای زیادی داشته باشند. مهمترین کاربرد آنها در تعامل ربات با انسان است. به عبارت دیگر این ژست‌ها می‌توانند یکی از درگاه‌های تعامل ما با دنیای هوش مصنوعی باشد. این تعامل‌ها می‌توانند شامل مثال‌های ساده‌ای همچون کنترل وسایل خانگی تا مثال پیچیده تری از تشخیص و تحلیل احساسات انسانی توسط هوش مصنوعی باشد.

تشخیص ژست‌های ایستا که دارای حرکت نیستند نسبت به ژست‌های حرکتی، کار ساده‌تری است، چرا که بعد زمان در این ژست‌ها در نظر گرفته نمی‌شود. این در حالی است که تشخیص ژست‌های حرکتی می‌تواند بسیار مفیدتر از ژست‌های ایستا باشد چرا که ژست‌های حرکتی می‌توانند اطلاعات خیلی بیشتری را در بر داشته باشند.

در این پایان نامه از دو مجموعه داده استفاده شده است. مجموعه داده‌ی اول توسط شرکت IBM جمع آوری شده است که در آن افراد در مقابل حسگر DVS [۱] به انجام تعداد مشخصی ژست حرکتی می‌پردازند. در این مجموعه ۲۹ نفر مختلف در ۳ شرایط نوری مختلف، ۱۱ ژست حرکتی را اجرا می‌کنند. شرایط نوری و ژست‌های حرکتی در جدول‌های ۱.۲ و ۲.۲ مشخص شده است. نکته‌ی متمایز کننده‌ی این مجموعه داده همین شرایط نوری مختلف است، چرا که این شرایط می‌توانند در الگوی تولید اسپایک در حسگر DVS تاثیر چشمگیری داشته باشد. برای مثال اسپایک‌هایی که در نور طبیعی تولید می‌شوند تقریبا یکدست هستند، در حالی که در حسگر در نور لامپ تقریبا اسپایک‌هایی با الگویی متناظر با فرکانس لامپ تولید می‌کند. این مجموعه داده IBM-DVSGesture نام دارد. [۱] نمونه‌ای از این داده‌ها را در تصویر ۱.۲ مشاهده میکنیم.

[۱] Dynamic Vision Sensor

شماره	نام ژست حرکتی
۱	clap hand
۲	wave hand right
۳	wave hand left
۴	clockwise arm right
۵	clockwise counter arm right
۶	clockwise arm left
۷	clockwise counter arm left
۸	roll arm
۹	drums air
۱۰	guitar air
۱۱	gestures other

جدول ۱.۲: جدول مربوط به ژست‌های حرکتی در مجموعه داده‌ی IBM-DVSGesture

شماره	شرایط نوری
۱	Fluorescent
۲	LED
۳	Light Natural

جدول ۲.۲: جدول مربوط به شرایط نوری در مجموعه داده‌ی IBM-DVSGesture

مجموعه‌ی داده‌ی دوم، که NavGesture نام دارد، شامل تعداد کمتری داده نسبت به مجموعه‌ی قبلی است. برای جمع آوری این داده، حسگر DVS بر روی تلفن همراه نصب شده و فرد مورد نظر را از این زاویه می‌نگرد و فرد مورد نظر فرمان‌هایی را به صورت ژست حرکتی به تلفن همراه می‌دهد. هدف این تحقیق، پیشرفت در راستای ارتباط انسان با تلفن‌های همراه است. نکته‌ی قابل توجه این تحقیق این است که در دو قسمت جداگانه وجود دارد. در قسمت اول سوژه‌ی مورد نظر نشسته و حرکت ندارد و در قسمت دوم سوژه در حال حرکت است. این حرکت باعث افزایش نویز محیط است که به واسطه‌ی حرکت دوربین به وجود می‌آید. [۲۱] نتایج این تحقیق‌ها در قسمت ۵.۳ بررسی شده است.

۲.۲ مروری بر تحقیق‌های انجام شده

دیتاست IBM-DVSGesture از مقاله ی [۱] استخراج شده است. محققان در این پایان نامه تلاش داشته‌اند تا با استفاده از شبکه عصبی اسپایکی که بر روی سخت افزارهای نورومورفیک پیاده سازی شده است مسئله ی تشخیص ژست حرکتی را حل کنند. همانطور که پیشتر گفته شد این مجموعه داده، توسط حسگر DVS تهیه شده است. علاوه بر این کد پیاده سازی شده در این تحقیق بر روی سخت افزار

TrueNorth بوده است که توسط IBM طراحی و توسعه یافته است. ساختار شبکه ی اسپایکی استفاده شده در این تحقیق شامل چهار بخش است که به طور خلاصه توضیح داده می‌شود.

بخش اول شبکه یک لایه فیلتر زمانی است که وظیفه ی نگهداری اسنپ شات‌های متوالی از داده‌های ورودی برای مدت زمان مشخصی را دارد. این لایه دارای k فیلتر است که در ابتدا فیلتر اول، سری اول از داده ورودی را در خود نگه می‌دارد. سپس در سیکل زمانی بعدی، لایه دوم، مقادیر لایه اول را در خود کپی می‌کند و سپس لایه اول سری دوم را از ورودی می‌خواند. و به همین ترتیب اطلاعات از لایه اول شروع تا به آخرین فیلتر زمانی می‌رسد. در واقع وظیفه ی این فیلتر تولید نوعی حافظه‌ی کوتاه مدت است. لازم به ذکر است که تمام این لایه‌ها به صورت همزمان به لایه‌های کانولوشن متصل می‌باشند.

بخش دوم، لایه‌های کانولوشن است. این لایه‌ها مشابه لایه‌های کانولوشنال در شبکه‌های مصنوعی می‌باشد و با این تفاوت که به جای تابع فعال سازی ReLU از تابع باینری استفاده شده است. همچنین وزن این لایه‌ها نیز به صورت سه تایی می‌باشد که می‌توانند وزن‌های صفر، یک یا منفی یک را بپذیرند. آموزش در این لایه به صورت آفلاین و خارج از سخت افزار نورومورفیک است. برای آموزش وزن‌ها هم از نوعی الگوریتم بازگشت خطا کمک گرفته شده است. در این الگوریتم یک کپی از وزن‌های شبکه به صورت مقدار صحیح نگهداری می‌شود و عمل بازگشت خطا و بهینه سازی روی این وزن‌ها اجرا می‌شود و سپس این وزن‌ها به وزن‌های سه گانه تبدیل می‌شوند. شبکه ی استفاده شده در این تحقیق دارای ۱۶ لایه کانولوشنال است.

بخش سوم لایه، winner-takes-all است. این بخش بعد از بخش کانولوشنال قرار دارد و دارای مکانیزمی است که به بعد از اسپایک زدن هر نورون، پتانسیل نورون‌های مجانب آن تقلیل می‌یابد. بنابراین بعد از اسپایک زدن هر نورون احتمال اسپایک زدن نورون‌های اطراف آن بسیار پایین می‌آید. این لایه نقش کلاس بندی شبکه را نیز ایفا می‌کند و یکی از کلاس‌ها را به عنوان خروجی می‌دهد. آخرین بخش فیلتر، sliding window است که خروجی بخش سوم را نرم می‌کند که نحوه پیاده سازی آن به صورت میانگین n فریم قبلی است.

در تحقیق انجام شده در [۱۶]، نشان داده شده است که می‌توان با استفاده از شبکه‌های اسپایکی به نتایج قابل قبول، در حد دقت شبکه‌های عصبی مصنوعی برسیم. ساختار شبکه به این صورت می‌باشد که در ابتدا، یک لایه‌ی کدگذاری زمانی قرار دارد. این لایه که از نوع تفاوت گاوسی می‌باشد، وظیفه دارد مقادیر پیکسل‌های تصویر که بیانگر شدت نور در این پیکسل است را به اسپایک تبدیل کند. این تبدیل به این صورت است که هر چه مقدار شدت نور در پیکسل بیشتر باشد اسپایک در آن پیکسل زودتر اتفاق می‌افتد و هر چه مقدار شدت نور کمتر باشد، اسپایک دیرتر و یا اصلا اتفاق نمی‌افتد.

بعد از لایه کدگذاری، تعدادی لایه‌ی کانولوشن قرار دارد. در مسئله‌ای که در این تحقیق مورد بحث قرار گرفته، از سه لایه کانولوشنال استفاده شده که بعد از هر لایه، یک لایه پولینگ قرار دارد. نهایتا بعد از لایه‌های کانولوشنال، یک لایه کلاس بندی قرار دارد که نتیجه‌ی لایه‌های کانولوشنال را تشخیص دهد و خروجی مسئله را تولید کند. لازم به ذکر است که نورون‌های لایه‌های کانولوشنال از نوع IF می‌باشند که به این معنی است که مقدار ولتاژ درونی آنها در طول زمان کاهش پیدا نمی‌کنند و فقط مقدار جریان

ورودی را جمع آوری می‌کنند.

نوع کدگذاری در این تحقیق از نوع کدگذاری زمان‌ـاسپایک می‌باشد. به این معنی که در لایه اول، پس از این که اسپایک‌های تصویر تولید شد، هر نورون ورودی حداکثر یک اسپایک تولید می‌کند. سپس لایه‌های کانولوشنال، این اسپایک‌ها را جمع آوری می‌کنند و پس از این که مقدار ولتاژ درونی آنها از حد مشخص گذر کرد اسپایک خروجی را تولید می‌کنند و دیگر اسپایک دیگری تولید نمی‌کنند. به عبارت دیگر، لایه‌های کانولوشنال بلافاصله پس از این که مقدار پتانسیل درونی آنها از آستانه‌ی تحمل رد شد، اسپایک تولید می‌کنند و همچنین در لایه‌های پولینگ نیز نورون‌ها پس از دریافت اولین اسپایک ورودی، اسپایک تولید می‌کنند و باقی اسپایک‌های ورودی را نادیده می‌گیرند.

یکی دیگر از مکانیزم‌هایی که در این تحقیق استفاده شده، مکانیزم winner-takes-all می‌باشد. این مکانیزم باعث می‌شود که در هر ناحیه از لایه‌ی کانولوشنال، صرف نظر از صفحه‌ی ویژگی، اگر اسپایکی پدیدار شود، دیگر در همسایگی آن نقطه هیچ نورونی نمی‌تواند اسپایک تولید کند. این امر باعث می‌شود شبکه، ویژگی‌های مهم و متداول را یاد بگیرد. در قسمت‌های بعدی به طور دقیق تر به آن خواهیم پرداخت. آموزش در این شبکه به صورت لایه به لایه و با استفاده از متد STDP انجام می‌شود و منظور از لایه به لایه این است که آموزش هر لایه، پس از پایان آموزش لایه قبل شروع می‌شود و به صورت نظارت نشده انجام می‌شود.

در [۱۷] از شبکه‌های کانولوشنال عمیق استفاده شده ولی با این تفاوت که در این تحقیق از آموزش نظارت شده استفاده نشده است و محققان از روش STDP برای آموزش شبکه استفاده کرده‌اند. [۱۳] نیز تلاش کرده است تا مسئله‌ی تشخیص ژست را حل کند. اگرچه در این تحقیق از ژست‌های بدون حرکت استفاده شده ولی در این مقاله از شبکه‌های اسپایکی استفاده شده است که با روش STDP آموزش می‌بینند. در این تحقیق از نورن‌های IF استفاده شده که تنها یک بار اسپایک تولید می‌کنند و همچنین این شبکه کانولوشنال نیست.

در مقاله‌ی [۳۳] روشی برای آموزش نورون‌های IF ارائه شده است که می‌توان شبکه‌های چند لایه را با این روش به صورت نظارت شده آموزش داد. این روش بر مبنای روش گرادیان جایگزین است که در آن با استفاده از قانون Hebbian Three-Factor Rule وزن‌های نورون‌ها را آموزش می‌دهیم. این روش به این صورت عمل می‌کند که نورون‌هایی که قطبیت آنها تغییر می‌کند، سیگنال آموزشی متناظر با ضریب آموزشی از پیش تعیین شده‌ای، دریافت می‌کنند. این عمل در زمان اجرا رخ می‌دهد و نیازی به بازگرداندن خطا نیست. در [۱۵] نیز، روش مشابه‌ای برای آموزش نورون‌های LIF ارائه شده است.

روش‌های ساده و متداول سیگنال‌های آموزشی را بر اساس مقدار فعالیت نورون به آنها انتقال می‌دهند. این روش‌ها اغلب با بزرگ شدن اندازه‌ی شبکه و با اضافه شدن تعداد لایه‌های آن کارایی خود را از دست می‌دهند. در مقاله‌ی [۲۰] روش جدیدی برای انتقال رو به عقب سیگنال‌های آموزشی ارائه شده که در آن بر روی بازگشت خطا تنها به نورون‌های که در تولید خطا نقش داشته‌اند تمرکز شده است. این مقاله بر روی مجموعه داده‌ی MNIST آزمایش شده و توانسته است در شرایط یکسان و تنها با استفاده از این روش خطای آموزشی شبکه‌ی مورد آزمایش را از ۷% به ۲% کاهش دهد.

همین مسئله در مقاله‌ی [۲۵] با روش متفاوتی حل شده. در این تحقیق که بر روی مجموعه داده‌ی CIFAR-10 بررسی شده است، هر نورون خطای آموزشی خود را مستقیماً با استفاده از خطای نهایی شبکه و با بهره گیری از مکانیزم Random Auxiliary Classifiers محاسبه می‌کند. و هر نورون با استفاده از این خطا و بدون وابستگی به سایر نورون‌ها و لایه‌ها آموزش می‌بینند. یکی از مزیت‌های این روش نیز این است که لایه‌ها به صورت همزمان می‌توانند آموزش ببینند.

مقاله‌ی [۱۹] به مسئله‌ی مشتق ناپذیر بودن تابع اسپایک پرداخته است. در این مقاله برای محاسبه‌ی سیگنال آموزشی، به جای مشتق گرفتن از تابع اسپایک، از مقدار مشتق پتانسیل داخلی نورون استفاده شده است. از آنجایی که این مقدار رابطه‌ی مستقیمی با فعالیت و مقدار اسپایک‌های خروجی نورون دارد، و همچنین مقدار آن به جز در زمان اسپایک، پیوسته است، مقدار مناسبی برای جایگزینی با مشتق تابع اسپایک می‌باشد. البته نکته‌ی قابل تامل این است که این مقدار در لحظه‌ی اسپایک دارای ناپیوستگی است که می‌توان آن را به عنوان اختلال در سیستم تلقی کرد و تاثیری در کارکرد کلی سیستم نخواهد داشت. این مقاله بر روی مجموعه داده‌ی MNIST پیاده سازی شده است.

در [۳۰] به طور کلی از مکانیزم بازگشت خطا صرف نظر شده، و به جای آن از روش Feedback Alignment استفاده شده. این روش که به شبکه‌های عصبی بیولوژیکی شباهت بیشتری دارد، خطای آموزشی را با استفاده از اتصالات تصادفی به طور مستقیم به لایه‌ها و نورون‌ها منتقل می‌کند و شبکه را آموزش می‌دهد. این مقاله بر روی مجموعه داده‌های MNIST و CIFAR پیاده سازی شده است.

به طور مشابه‌ی شبکه‌ی مورد استفاده در [۲۴]، از مدل نورونی احتمالاتی استفاده شده است. برای آموزش این مدل، به جای استفاده از روش‌های بازگشت خطا، از مکانیزم winner-takes-all استفاده شده است. در این مکانیزم، نورونی که زودتر اسپایک تولید کند برنده محسوب می‌شود و پتانسیل نورون‌هایی که در مجاورت این نورون قرار دارند کاهش می‌یابند. به این ترتیب احتمال اسپایک زدن این نورون‌ها در سیکل‌های آتی کمتر خواهد بود. در نتیجه نورون‌ها سعی می‌کنند تفاوت‌هایی که از هم فاصله‌ی بیشتری دارند را یاد بگیرند. این تحقیق بر روی مجموعه داده‌ی MNIST پیاده سازی شده است.

در شبکه‌های عصبی بیولوژیکی، سیگنال‌های عصبی بسته به فاصله‌ی فیزیکی نورون‌ها ممکن است با تاخیر به مقصد برسند. یکی از روش‌های بهبود کارایی شبکه‌های اسپایکی پیاده سازی این تاخیر سیگنال‌های عصبی است. در مقاله‌ی [۳۲] همزمان با وزن‌های شبکه، ضریب شبیه سازی تاخیر سیگنال‌های عصبی نیز بهینه می‌شود که نشان داده شده می‌تواند توانایی شبکه را بهبود ببخشد. در این تحقیق از معماری کانولوشنال برای یادگیری مجموعه داده‌های MNIST، TIDigitsو DVS-Gesture استفاده شده. برای مدل سازی نورون‌ها نیز از مدل احتمالاتی نورون اسپایکی استفاده گردیده است که بر روی سخت افزارهای نورومورفیک، آزمایش و بررسی شده‌اند.

یکی از معماری‌های مهم شبکه‌های عصبی، شبکه‌های بازگشتی است. در مقاله‌های [۱۴] و [۵] بر روی نورون‌های بازگشتی تحقیق کردند. در مدل نورونی که توسط مقاله‌ی [۵] ارائه گردیده، نورون‌های بازگشتی در قالب شبکه‌ی کامل اقدام به یادگیری مسئله می‌کنند. در این روش از مدل آماری نورون‌های اسپایکی استفاده شده که با کمینه کردن کران بالای مقدار Kullback-Leibler ، نورون‌ها آموزش می‌بینند.

همچنین، در مقاله‌ی [۱۴] روشی برای آموزش و بهینه کردن نورون‌های بازگشتی، بر اساس روش گرادیان جایگزین ارائه شده است. در این مقاله به جای تابع اسپایک، از مقادیر پتانسیل و جریان‌های ورودی نورون‌ها، سیگنال آموزشی محاسبه و نورون‌ها آموزش می‌بینند. به طور مشابه‌ی نیز در [۲] از شبکه‌هایی با معماری بازگشتی استفاده شده است. با این تفاوت که این مقاله یک قدم نیز فراتر رفته و مکانیزم LSTM را پیاده سازی کرده است و آن را با استفاده از روش BPTT آموزش می‌دهد.

اغلب مقاله‌های فوق، دارای کدگذاری نرخ اسپایک هستند. در مقاله‌ی [۲۳] روشی برای آموزش شبکه‌های مبتنی بر کدگذاری زمان اسپایک، در زمان اجرا ارائه شده است. در این مقاله وجود رابطه‌ی میان جریان ورودی و خروجی نورون‌ها نشان داده شده که در تمامی نقاط مشتق پذیر است و می‌توان سیگنال آموزشی را بر اساس این مقادیر محاسبه کرد. این مقاله نیز بر روی مجموعه داده‌ی MNIST پیاده سازی شده است.

در مقاله‌ی [۶] شبکه‌ی عصبی مصنوعی دودویی و روشی برای آموزشی آن ارائه شده. اگرچه این شبکه‌ها از بسیاری از مزیت‌های شبکه‌های عصبی اسپایکی محروم هستند، ولی به دلیل استفاده از عملگرهای دودویی، از نظر مقدار حافظه و توان پردازشی مورد نیاز، بسیار بهینه‌تر از شبکه‌های عصبی مصنوعی متداول هستند و به دلیل نداشتن ماهیت اسپایکی به سادگی بر روی GPU قابل شبیه سازی هستند. در این مقاله که بر روی مجموعه داده‌های MNIST، CIFAR-10 و SVHN پیاده سازی شده است، روشی برای محاسبه‌ی گرادیان نورون‌ها در هنگام اجرای شبکه نیز ارائه گردیده.

فصل ۳

روش شناسی و پژوهش

در این پژوهش ما از دو منبع به عنوان کد پایه برای این پروژه استفاده کردیم: پروژه ی SpyTorch [۱] و پروژه ی S2Net [۲].

پروژه ی SpyTorch بر روی مجموعه داده Fashion-MNIST توسعه پیدا کرده است. این مجموعه داده متشکل از تصاویر ۴۸ در ۴۸ پیکسلی سیاه و سفید می‌باشد که از البسه ی مختلفی گرفته شده است. به دلیل این که این شبکه بر روی داده‌های تصویری مبتنی بر پیکسل کار می‌کند، به لایه تبدیل تصویر مبتنی بر پیکسل، به تصویر مبتنی بر اسپایک کار می‌کند. همچنین این شبکه دارای کد گذاری نرخ_اسپایک است. این پروژه فقط دارای پیاده سازی لایه‌های کامل است و از لایه‌های کانولوشنال استفاده نمی‌کند.

در قدم بعدی کار S2Net را مورد بررسی قرار دادیم. این پروژه بر روی داده‌های صوتی_اسپایکی متمرکز شده است. در این پروژه لایه‌های ساده ی کانولوشنی پیاده سازی شده است. ولی پس از بررسی‌های انجام شده این شبکه به تنها قادر به یادگیری داده‌های ژست حرکتی نیست. و در اولین قدم نیاز است که بعد لایه‌های کانولوشن را افزایش دهیم. همچنین این شبکه فاقد لایه‌های Batch-Norm، DropOut، Pooling و... است.

۱.۳ آماده‌سازی داده‌ها

در اولین قدم پردازش نیاز است که داده‌ها پیش پردازش شوند و آماده سازی شوند. این مرحله در بسیاری از شبکه‌های عصبی به خصوص مواردی که بر روی تصویر کار می‌کنند مرحله‌ای رایج است. اغلب این مرحله شامل کارهایی از قبیل تمیز کردن تصویر و حذف نویزهای اضافه‌ی آن، درست کردن‌اندازه‌ی تصویر و... می‌باشد. اما از آنجایی که داده‌ی استفاده شده در این پایان نامه به صورت اسپایک می‌باشد مرحله‌ی آماده سازی داده کمی‌متفاوت تر می‌باشد. مهم ترین قسمت این مرحله تغییر شکل داده‌ها

[۱] https://github.com/fzenke/spytorch
[۲] https://github.com/romainzimmer/s2net

از جریان اسپایک به مجموعه اسپایک‌های مبتنی بر فریم است. در ادامه به جزئیات بیشتر این مرحله می‌پردازیم.

۱.۱.۳ تبدیل تصویر مبتنی بر پیکسل به تصویر اسپایکی

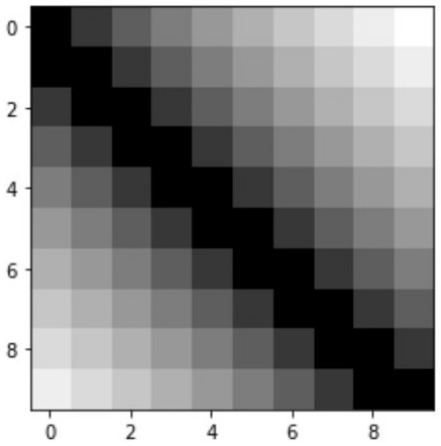
همانطور که قبل تر گفته شد، ساختار این پروژه بر اساس تحقیق [۲۸] بنا نهاده شده است. اما تفاوت اصلی کار ما با این پروژه در این است که مجموعه داده‌ای که ما استفاده کرده‌ایم تشکیل شده از جریان اسپایک است. این در حالی است که این تحقیق از تصویرهایی بر مبنای پیکسل استفاده می‌کند و از آنجایی که شبکه‌ی استفاده شده از نوع اسپایکی است، نیاز است داده‌های تصویر به اسپایک تبدیل شوند. این کار را لایه‌ی تبدیلی که در ابتدای شبکه قرار دارد انجام می‌دهد. نحوه‌ی کار کرد این لایه در معادله‌ی زیر قابل مشاهده است:

$$T(x) = \begin{cases} T_{max} & if \;\; x < threshold \\ \tau \log(\frac{x}{x - threshold}) & otherwise \end{cases} \tag{۱.۳}$$

در این معادله، تابع T، نگاشت عددی شدت نوری پیکسل‌ها به زمان اسپایک است. پارامتر مهم این معادله مقدار $threshold$ است. این مقدار بیانگر حداقل مقداری است که پیکسل باید مقدار داشته باشد تا مجاز به تولید اسپایک کند. مقدارهایی که کمتر از این $threshold$ مقدار دارند اسپایکی تولید نمی‌کنند. به عبارت دیگر در زمان T_{max}اسپایک تولید می‌کند که در واقع زمانی است که شبکه هرگز به آن نمی‌رسد و قبل از پردازش داده متوقف می‌شود. پارامتر بعدی مقدار tau است که ثابت زمانی غشا نورون LIF است که مشخص می‌کند نورون در چه زمانی شارژ می‌شود. در واقع با افزایش این مقدار، اسپایک‌ها بیشتر در انتهای اسپکتروم زمانی تشکیل می‌شوند و با کم کردن آن، اسپایک‌ها به زمان صفر نزدیک تر می‌شوند. نهایتا T_{max}حداکثر تعداد سیکلی است که شبکه کار می‌کند. شبکه هیچگاه به زمان T_{max}نخواهد رسید و قبل از آن متوقف خواهد شد. نمونه‌ای از مقدار تصویر، و مقدار زمان اسپایک آن در شکل ۱.۳ قابل مشاهده است. لازم به ذکر است که مقدار x قبل از این معادله باید به مقدار بین صفر تا یک نرمال شود تا کدینگ‌های مختلف تصویر در این معادله تاثیر منفی نگذارند. از آنجایی که داده‌ی استفاده شده در این پایان نامه به صورت اسپایکی می‌باشد، نیازی به اضافه کردن این لایه تبدیل داده‌ها در این پایان نامه نبوده است.

۲.۱.۳ فریم بندی جریان اسپایک

همان طور که در قسمت قبلی گفته شد، پردازش داده‌ها در کد [۲۸] به صورت مبتنی بر فریم است. ما نیز در پروژه‌ی خود از پردازش مبتنی بر فریم استفاده کردیم. اگرچه پردازش مبتنی بر فریم برای پیاده سازی در سخت افزارهای نورومورفیک مناسب نیست و نیز هدف اصلی استفاده از حسگرهای DVS



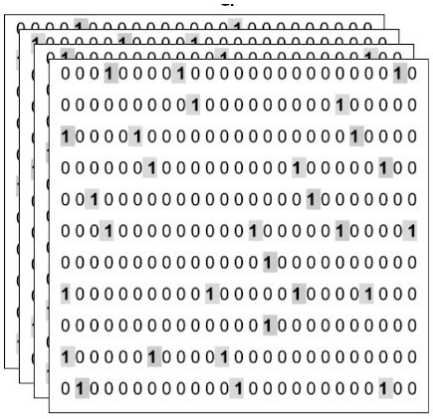
(آ) تصویر مبتنی بر پیکسل

```
[255, 199, 161, 130, 103,  79,  57,  36,  17,   0]
[255, 255, 199, 161, 130, 103,  79,  57,  36,  17]
[199, 255, 255, 199, 161, 130, 103,  79,  57,  36]
[161, 199, 255, 255, 199, 161, 130, 103,  79,  57]
[130, 161, 199, 255, 255, 199, 161, 130, 103,  79]
[103, 130, 161, 199, 255, 255, 199, 161, 130, 103]
[ 79, 103, 130, 161, 199, 255, 255, 199, 161, 130]
[ 57,  79, 103, 130, 161, 199, 255, 255, 199, 161]
[ 36,  57,  79, 103, 130, 161, 199, 255, 255, 199]
[ 17,  36,  57,  79, 103, 130, 161, 199, 255, 255]
```

(ب) مقادیر عددی شدت نوری پیکسل‌ها

```
[ 12,  16,  20,  26,  34,  47,  74, 167,   -,   -]
[ 12,  12,  16,  20,  26,  34,  47,  74, 167,   -]
[ 16,  12,  12,  16,  20,  26,  34,  47,  74, 167]
[ 20,  16,  12,  12,  16,  20,  26,  34,  47,  74]
[ 26,  20,  16,  12,  12,  16,  20,  26,  34,  47]
[ 34,  26,  20,  16,  12,  12,  16,  20,  26,  34]
[ 47,  34,  26,  20,  16,  12,  12,  16,  20,  26]
[ 74,  47,  34,  26,  20,  16,  12,  12,  16,  20]
[167,  74,  47,  34,  26,  20,  16,  12,  12,  16]
[  -, 167,  74,  47,  34,  26,  20,  16,  12,  12]
```

(ج) زمان اسپایک هر نقطه از تصویر

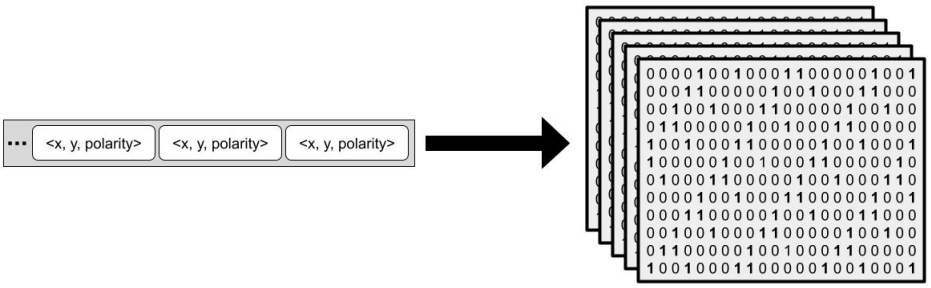


(د) ماتریس اسپایک‌ها در طول زمان

نمودار ۱.۳: در شکل ۱.۳آ نمونه‌ی تصویر مبتنی بر پیکسل نمایش داده شده است. در شکل ۱.۳ب مقدار پیکسل‌های قسمت کوچکی از تصویر نمایش داده شده است. لازم به ذکر است که در کدینگ uint8 مقدار پیکسل‌ها بین ۰ تا ۲۵۵ می‌باشد. مقدار ۰ بیانگر رنگ سیاه (پیکسل خاموش) و ۲۵۵ بیانگر رنگ سفید (پیکسل روشن) است. شکل ۱.۳ج مقدار تبدیل شده‌ی تصویر به زمان اسپایک را نشان می‌دهد. هر درایه ماتریس، زمانی که در آن پیکسل اسپایک می‌زند را مشخص می‌کند و مقدارهای inf نشان دهنده‌ی این است که پیکسل اسپایکی در زمان کار کردن شبکه تولید نمی‌کند. شکل ۱.۳د نشان دهنده‌ی مقدار اسپایکی پیکسل‌ها را نشان می‌دهد که در هر سیکل شبکه می‌تواند مقدار یک یا صفر بگیرد که نشان دهنده‌ی این است که در آن سیکل اسپایک تولید کرده است یا نه.

بهره مندی از خاصیت پراکنده بودن داده‌ها است، ولی در این پایان نامه، از این قابلیت صرف نظر کردیم و این داده‌ها را به صورت ماتریسی، در قالبی منظم که به آن تصاویر اسپایکی مبتنی بر فریم می‌گوییم، تبدیل و پردازش می‌کنیم. دلیل این تصمیم این است که شبیه سازی شبکه‌هایی که با داده‌های منظم کار کنند در GPUها، و به طور کلی در سخت افزارهای von-neumann بسیار ساده‌تر خواهد بود، و پردازش آنها به مراتب سریع‌تر انجام خواهد شد. به همین منظور و همچنین به دلیل عدم دسترسی به پردازنده‌های نورومورفیک، این تبدیل را انجام داده‌ایم تا هم سرعت پردازش و آموزش شبکه را بالا ببریم و هم در زمان کوتاه تر بتوانیم سیستم را توسعه دهیم.

بنابراین، اولین قدم در پیش پردازش داده‌ها تبدیل جریان اسپایک‌های ورودی از حسگر DVS به فریم‌های قابل پردازش است. شمای کلی این روش در شکل ۲.۳ نشان داده شده. داده‌های حسگر DVS به صورت جریان اسپایک می‌باشد. به عبارت دیگر، داده‌های این حسگر به صورت پکت‌های متوالی هستند که هر کدام شامل اطلاعات اسپایک هستند. مهم ترین اطلاعاتی که برای ما مهم است، مختصات اسپایک x) و y اسپایک در مختصات تصویر) و جهت اسپایک است. عملکرد حسگر DVS به این صورت است که با تغییر شدت نوری هر نقطه ی حسگر، اسپایکی تولید می‌شود که شامل جهت این تغییرات (مثبت یا منفی) و مختصات آن است. بنابراین، در صورتی که تصویر ثابت باشد و تغییری نداشته باشد اسپایکی تولید نمی‌شود و به طبع تنها نقاطی از تصویر که تغییرات دارند اسپایک تولید می‌کنند.



نمودار ۲.۳: همانطور که مشاهده می‌شود، داده‌ی ورودی به صورت قطاری از بسته‌های اسپایک است که به طور پیوسته به سیستم تزریق می‌شوند. هر کدام از این بسته‌ها دارد مختصات و جهت اسپایک هستند. خروجی الگوریتم نیز مجموعه‌ای از تصاویر اسپایکی میباشند که مقدار هر درایه آن، صفر یا برابر با جهت اسپایک در آن نقطه است.

برای تبدیل جریان اسپایک به فریم از الگوریتم ۱ استفاده می‌کنیم. در این الگوریتم، فرض بر این است که جریان اسپایک‌ها از نظر زمانی مرتب هستند، اگر چه در عمل اینطور نیست. به طور کلی، روش کار این الگوریتم به این صورت است که، اسپایک‌ها را به ترتیب شروع به پردازش می‌کنیم. سپس با در نظر گرفتن ثابت frame-length مدت زمان هر فریم را تعیین می‌کنیم. سپس اسپایک‌ها را به بازه‌هایی به طول frame-length افراز می‌کنیم. برای این کار از الگوریتم ۱ استفاده می‌کنیم. در نتیجه متغیر frame-set

مجموعه ای است از افراز جریان اسپایک‌ها که هرکدام نشان دهنده ی اسپایک‌های هر فریم است.

Algorithm 1: تبدیل جریان اسپایک به تصویر مبتنی بر فریم

Result: مجموعه داده‌ی فریم بندی شده.

$frameSet \leftarrow \emptyset$;
$frameStartTime \leftarrow$ Time of first spike in system;
// Assuming spikes are sorted in time.
while *There is spike to process* **do**
  $currentTime < -frameStartTime$;
  while $currentTime \leq frameStartTime + frameLength$ **do**
    $spike \leftarrow getNextSpike()$;
    $frameSet \leftarrow frameSet + \{spike\}$;
    $currentTime \leftarrow spikeTimeStamp$;
  end
  $frameStartTime \leftarrow currentTime$;
end

در قدم بعدی، هر کدام از این افرازها را به فریم تبدیل می‌کنیم. برای این کار، یک فریم خالی (ماتریس صفر به ابعاد تصویر) در نظر می‌گیریم. سپس کافیست اسپایک‌ها را پردازش کنیم و مقدار نقاط فریم را که در آنها اسپایک رخ داده را افزایش یا کاهش دهیم.

برای فریم بندی داده‌ها دو پارامتر مهم وجود دارد: تعداد فریم‌ها و طول فریم‌ها. طول فریم‌ها که در الگوریتم قبلی توضیح داده شد. این پارامتر بیانگر مدت زمانی است که صبر می‌کنیم تا اسپایک‌های تصویر برسد و آن را وارد فریم می‌کنیم. پارامتر دوم، تعداد فریم‌ها است. این پارامتر بیانگر تعداد فریم‌هایی است که داده ی مجموعه دارد.

نکته ی مهم در بریدن جریان اسپایک‌ها توجه به برچسب زمانی داده‌ها است. در این مجموعه داده‌ها برچسب داده‌ها به صورت جدولی از شروع و پایان هر ژست در فایلی ذخیره شده است. برای بریدن داده‌ها، باید طوری عمل کنیم که در یک افراز از داده، دو ژست مختلف وجود نداشته باشد. برای حل این مشکل، در مرحله ی تولید افراز، هنگامی که اسپایک‌ها را بررسی می‌کنیم و به هم می‌چسبانیم، به زمان که اسپایک رخ داده توجه می‌کنیم. با توجه به زمان رخ داد اسپایک می‌توان برچسب آن را تعیین کرد. در صورتی که برچسب اسپایک مورد بررسی، برابر با برچسب اولین اسپایک افراز باشد، به این معنی است که هر دو اسپایک به یک ژست تعلق دارند. بنابرین هر دو را به افراز اضافه می‌کنیم، و در صورتی که برچسب آنها متفاوت باشد تمام اسپایک‌های افراز را صرف نظر کرده و شروع به تولید افراز جدید می‌کنیم. دلیل این امر هم این است که طول این افراز کمتر از مقدار مورد نظر خواهد بود، و نمی‌توان از آن در مجموعه داده استفاده کرد.

قدم بعدی که یکی از مهم ترین قدم‌ها بعد از پیش پردازش داده است، این است که پس از فریم بندی داده‌ها آنها را در فایل‌هایی در قالب فایل numpy ذخیره می‌کنیم. دلیل این کار این است که این قسمت

از پیش پردازش، بسیار زمانبر تر است و نیازی به انجام آن در هر مرحله نیست. در این پایان نامه داده‌ها با طول‌های ۲۰، ۸۰ و ۱۲۰ فریم بررسی شده است که طول فریم هر کدام حدود ۵۰ میکرو ثانیه است.

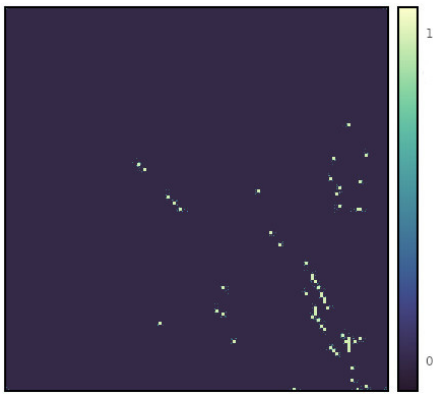
قطبیت یا جهت تغییرات هر اسپایک می‌تواند مثبت یا منفی باشد. ولی اسپایک‌های بیولوژیکی فقط مثبت هستند، و به طبع آن در مدل ما هم تنها اسپایک‌های مثبت تولید می‌شود. لذا تعریف دقیق تری از ورودی‌های شبکه نیاز داریم داشته باشیم. می‌توانیم از قطبیت ورودی‌ها صرف نظر کنیم و یا تعبیری برای ورودی‌های منفی داشته باشیم. در این پایان نامه چهار روش پیاده سازی برای این موضوع پیشنهاد می‌دهیم:

۱ـ صرف نظر کردن از قطبیت اسپایک‌ها: ساده ترین روش برای پردازش اسپایک‌ها صرف نظر کردن از قطبیت آنهاست. به عبارت دیگر برای شبکه فقط مهم است که در نقطه ی مورد نظر فقط تغییرات وجود داشته باشد و جهت تغییرات مهم نیست. پردازش به این روش را می‌توان معادل پردازش لبه‌های تصویر در نظر گرفت. در این گونه پردازش گرادیان جهت لبه‌ها اهمیتی ندارند و تنها خود لبه‌ها ارزش دارند. برای پیاده سازی این روش از قطبیت اسپایک‌ها صرف نظر می‌کنیم و در هر نقطه از تصویر که اسپایکی در آن وجود دارد مقدار ۱ و سایر نقاط را ۰ در نظر می‌گیریم.

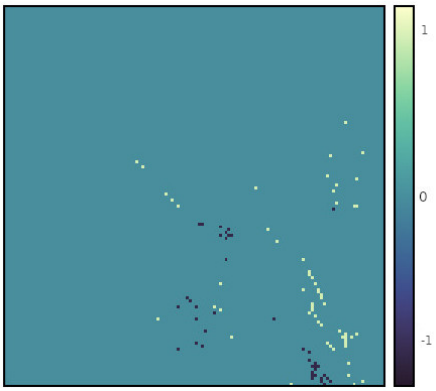
۲ـ روش تک لایه: در این روش همانند روش قبلی، صفحه ی فریم را به صورت صفر در نظر می‌گیریم و اسپایک‌ها را روی آن قرار می‌دهیم. تنها تفاوت این روش در این است که مقدار اسپایک‌ها را مقدار قطبیت آنها قرار می‌دهیم. یعنی، اسپایک‌های مثبت را مثبت یک و اسپایک‌های منفی را، منفی یک قرار می‌دهیم. این روش از نظر تعداد و حجم داده ی ورودی با روش قبلی تفاوتی ندارد و تنها تفاوت قابلیت منفی بودن این اسپایک‌ها است. نکته ای که باعث می‌شود این روش، روش مطلوبی برای فریم بندی جریان اسپایک‌ها نباشد این است که در این روش اسپایک‌های منفی داریم که در دنیای واقعی و در نورون‌های بیولوژیکی وجود ندارد. بنابراین مدل را، از شبیه بودن به نورون‌های مغزی دور می‌کند.

۳ـ روش دو لایه: همانند روش اول، در این روش هم تمام اسپایک‌ها مقدار مثبت یک دارند و تمام نقاط دیگر مقدار صفر دارند. تنها تفاوت این روش این است که در آن دو لایه مجزا در نظر گرفته می‌شود. لایه اول برای اسپایک‌ها با قطبیت مثبت و لایه دوم برای اسپایک‌ها با قطبیت منفی است. دلیل این کار این است که اسپایک‌ها از نظر قطبیت از هم جدا می‌شوند، از طرفی در سیستم اسپایک با مقدار منفی یک نخواهیم داشت. و همچنین در این روش حجم داده‌های ورودی دو برابر می‌شود.

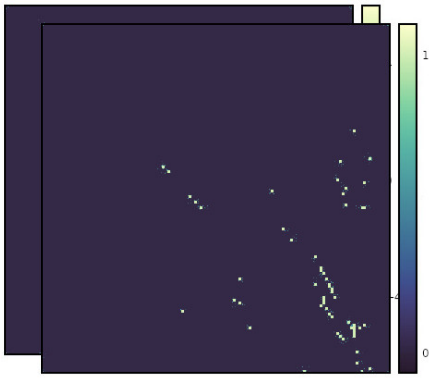
۴ـ روش جمع کردن اسپایک‌ها: مشکل تمام روش‌های فوق این است که تعداد اسپایک‌های یک نقطه را در نظر نمی‌گیرد و فقط وجود و عدم وجود اسپایک را بررسی می‌کند. برای مثال اگر در یک نقطه بیش از یک اسپایک وجود داشته باشد از اسپایک‌های دوم به بعد صرف نظر می‌شود. به همین دلیل حجم زیادی از اطلاعات در این مرحله از بین می‌رود. برای حل این مشکل فریم را طوری در نظر می‌گیریم که به جای نشان دادن اسپایک‌ها، انتگرال اسپایک در طول بازه ی زمانی فریم را نشان دهد. بدین شکل دیگر مقادیر فریم، صفر و یک نیستند و می‌توانند مقدارهای حقیقی مثبت یا منفی داشته باشند. از طرفی این روش در شبیه سازی‌های کامپیوتری به مدل واقعی اسپایکی شبیه تر است.



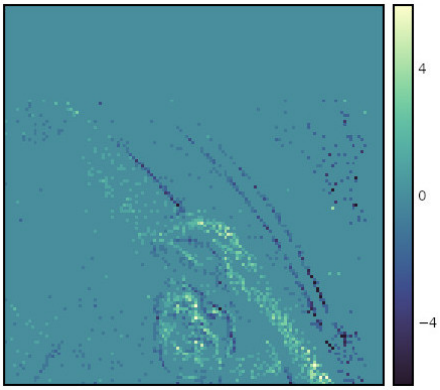
(آ) روش صرف نظر از جهت اسپایک



(ب) روش تک لایه



(ج) روش دولایه



(د) روش جمع اسپایک

نمودار ۳.۳: تصویرهایی از یک فریم زمانی بریده شده، در روش‌های مختلف فریم بندی را نشان داده است. شکل ۳.۳آ روش صرف نظر از قطبیت را نشان می‌دهد. شکل ۳.۳ب روش تک لایه را نشان می‌دهد. شکل ۳.۳ج روش دو لایه را نشان می‌دهد. و شکل ۳.۳د روش جمع اسپایک‌ها را نشان می‌دهد. همانطور که مشاهده می‌شود در روش جمع اسپایک‌ها، شکل ملموس تری تشکیل می‌شود.

۳.۱.۳ تقویت و اضافه کردن داده

قدم بعدی در پیش پردازش داده‌ها، تولید و اضافه کردن داده‌های جدید بر اساس داده‌های موجود به مجموعه داده و تقویت آن است. اگر چه این قدم اختیاری است، ولی انجام این کار باعث تقویت مجموعه داده خواهد شد. به عبارت دیگر با انجام این کار تعداد داده‌های قابل آموزش در مجموعه افزایش پیدا می‌کند و به سبب آن می‌توان شبکه را بهتر آموزش داد و به نتایج بهتری دست یافت.

برای تولید داده‌های تصویری روش‌های گوناگونی وجود دارد که نمی‌توان همه ی آنها را در تصاویر اسپایکی به کار برد. برای مثال در تصاویر مبتنی بر پیکسل می‌توان از تغییر رنگ، بریدن یا تغییر اندازه‌ی تصویر و... استفاده کرد. ولی از این روش‌ها در تصاویر اسپایکی نمی‌توان استفاده کرد. برای مثال بریدن تصویر در فریم اسپایکی امکان پذیر نیست چرا که نمی‌توان فریم بریده شده را تغییراندازه داد.

روشی که ما برای تقویت مجموعه داده استفاده کردیم صرفا جابه جایی تصویر است. به عبارت دیگر مختصات هر مجموعه تصویر با نویز تصادفی جمع می‌شود که در طول زمان آن داده تغییر نمی‌کند. این کار مانند این است که سوژه ی مورد نظر کمی جابه جایی داشته باشد. بنابراین در این روش توزیع اسپایک‌ها تغییر نمی‌کند. و تنها شکل فریم تغییر می‌کند و می‌توان گفت داده ی جدید تولید کرده ایم.

قدم نهایی در مرحله ی پیش پردازش داده‌ها، تقسیم آنها به داده‌های آموزش و تست است. که در این قسمت مجموعه فریم‌های بدست آمده را، به صورت تصادفی به دو قسمت آموزش و تست تقسیم کردیم. تنها نکته ی مهم در این قسمت این است که در مجموعه‌های تست که مبنای محاسبه ی دقت سیستم و سایر متریک‌های آن این است که هیچ گونه روش تقویت و اضافه کردنی استفاده نشده است. چرا که این داده‌ها نباید نویز تصادفی داشته باشند تا بتوان نتیجه قابل تولید مجدد داشت.

۲.۳ معماری شبکه

در این قسمت به بررسی لایه‌ها و تکنیک‌های استفاده شده در معماری شبکه می‌پردازیم. این تکنیک‌ها شامل لایه‌های مختلف پیاده سازی شده در شبکه و همچنین تکنیک‌هایی برای بهبود کارکرد شبکه است. اما قبل از آن به نکته‌ی قابل تامل باید اشاره کرد. یکی از اهداف ما در مورد استفاده از شبکه‌های عصبی اسپایکی، نزدیک تر شدن به ساختار بیولوژیکی مغز است. همان طور که می‌دانیم، معماری‌هایی همچون لایه‌های کامل و لایه‌های کانولوشنال، با نظم خاصی چیده شده‌اند. این در حالی است که شبکه‌های عصبی بیولوژیکی دارای ساختار پیچیده و در هم تنیده‌ای هستند که به صورت تکاملی پدید آمده‌اند. به دلیل همین پیچیدگی، شبیه سازی این ساختارها در GPU، بسیار دشوار است، و توان پردازشی بسیار بالایی نیاز دارد. در حالی که ساختارهای منظم، دارای پیاده سازی ساده‌تری هستند و از سرعت پردازش بالاتری نیز برخوردارند. همچنین، به دلیل ساختار ساده‌تر، آموزش این شبکه‌ها هم ساده‌تر است، چرا که سیگنال‌های آموزشی در این شبکه‌ها دارای مدل ریاضی ساده‌تری هستند. به همین منظور، در این پایان نامه، و در اغلب تحقیق‌ها و مقالات مشابه نیز از همین ساختارها استفاده شده است.

۱.۲.۳ لایه کامل

اولین و مهمترین لایه در شبکه‌های عصبی لایه کامل است. همانطور که می‌دانیم این لایه از اتصال تمام نورون‌ها به تمام نورون‌های لایه قبلی یا به طور دقیق تر از اتصال کامل تمام سیناپس‌ها به تمام ورودی‌ها تشکیل می‌شود. نحوه ی کارکرد این لایه در الگوریتم a نشان داده شده است.

Algorithm 2: پردازش لایه‌های کامل اسپایکی

```
// this has the shape of (batch, nb-frames, input-shape)
x ← inputSpikes;
y ← 0;
syn ← 0;
mem ← 0;
spk ← 0;
for i = 0 to Number Of Frames do
    h ← x[i].W;
    rst ← spk;
    syn ← alpha * syn + h;
    mem ← beta * mem + syn − rst;
    spk ← heavySide(mem);
    y[i] = spk;
end
return y;
```

این الگوریتم به ازای هر داده اجرا می‌شود و نورون‌های لایه جاری را شبیه سازی می‌کند. لازم به ذکر است که همان طور که قبل تر گفته شد، برای سادگی و سرعت شبیه سازی شبکه، آن را به صورت لایه به لایه انجام می‌دهیم. بنابراین، از اولین لایه شروع به شبیه سازی می‌کنیم و پس از این که لایه را برای تمام زمان‌ها اجرا کردیم به لایه بعدی می‌رویم. بنابراین، در این لایه، مقدار ورودی، یا همان x دارای یک بعد زمان هم است. که برای پردازش لایه در این بعد پیمایش می‌کنیم. همانطور که قابل مشاهده است، متغیر i در بعد زمان پیمایش می‌کند تا لایه را شبیه سازی کند.

سپس ورودی لحظه ی i در وزن‌های شبکه ضرب می‌شود و مقدار جریان ورودی را تشکیل می‌دهد که در متغیر h قرار دارد. h در واقع همان متغیر مشاهده شده در شبکه‌های عصبی مصنوعی می‌باشد. در قدم بعدی مقدار جریان ورودی به نورون محاسبه می‌شود. این مقدار در متغیر syn ذخیره می‌شود. سپس پتانسیل داخلی نورون که mem نام دارد محاسبه می‌شود. این مقدار با استفاده از syn افزایش و با استفاده از rst کاهش پیدا می‌کند. همچنین ضریب beta در طول زمان کاهش می‌ابد. نهایتا این مقدار با استفاده از تابع heavy-side فیلتر می‌شود و مقادیر بزرگتر از یک منجر به اسپایک می‌شوند.

البته دو مکانیز دیگر، نرمال کردن وزن‌ها و عدم استفاده از وزن ضریب جریان باعث تغییری در الگوریتم فوق می‌شود که در ادامه به آنها می‌پردازیم.

۲.۲.۳ لایه کانولوشن

پیاده سازی نورون‌های این لایه همانند نورون‌های لایه کامل است و تنها تفاوت در نحوه ی دریافت ورودی آنها است. به عبارت دیگر می‌توان گفت نحوه ی کارکرد مکانیزم پتانسیل و اسپایک زدن نورون‌ها در این لایه کاملا مشابه با لایه کامل است. چرا که هر دو از نورون‌های تجمیه_آتش با نشتی استفاده می‌کنند. تنها تفاوت این دو لایه در نحوه ی پردازش ورودی می‌باشد که این قسمت با سایر شبکه‌های عصبی مصنوعی تفاوتی ندارد.

در این قسمت، ابتدا وزن‌های هسته را تشکیل می‌دهیم، سپس آنها را بر روی دیتای ورودی می‌لغزانیم تا داده‌ی دیده شده یا همان h بدست بیاید. تنها نکته ی قابل تامل این لایه‌ها در ورودی آنهاست. بر خلاف لایه‌های کامل که انتظار ورودی به شکل [تعداد داده ورودی، زمان شبیه سازی، اندازه دسته] ورودی را به صورت [عرض تصویر، طول تصویر، زمان شبه سازی، کانال‌های تصویر، اندازه دسته][۳] دریافت می‌کنند.

در این پروژه این لایه‌ها در قالب‌های یک، دو و سه بعدی پیاده سازی شده اند. البته لایه‌های یک بعدی در این مسئله کاربردی ندارد. همچنین پس از انجام آزمایش به این نتیجه رسیدیم که مناسب ترین نوع پردازش برای این مسئله، پردازش آن در سه بعد است. با این تفاوت که طول کرنل استفاده شده در بعد زمان (بعد سوم) یک است. دلیل استفاده از لایه‌های سه بعدی امر سرعت بیشتر پردازش آن و همچنین قابلیت پردازش در بعد زمان در صورت نیاز است.

۳.۲.۳ لایه‌های Max-Pooling و Avg-Pooling

در ادامه ی لایه‌های کانولوشن، لایه‌های پولینگ نیز پیاده سازی شده‌اند. این لایه‌ها، با استفاده از ابزارهای pytorch پیاده سازی شده‌اند و نیازی به تغییر دیگری ندارند. این لایه‌ها هیچ گونه پارامتر قابل آموزش ندارند. لایه ی Max-Pooling دارای خروجی اسپایکی است و در بازه ی پردازشی نورون، در صورت دیدن هر ورودی دارای اسپایک، خود نیز اسپایک تولید می‌کند. ولی لایه ی Avg-Pooling خاصیت اسپایکی خود را از دست می‌دهد، چرا که داده‌ها را از حالت صفر_یک خارج کرده و به صورت عدد حقیقی مقدار دهی می‌کند. به همین دلیل ما در شبکه‌های خود از این لایه استفاده نکردیم و پیاده سازی آن صرفا جهت بررسی و آزمایش بوده است.

۴.۲.۳ عدم استفاده از ضریب جریان سیناپس (α)

پس از انجام آزمایش‌های متعدد، به این نتیجه رسیدیم که برای ساده تر شدن مدل شبکه و همچنین برای سریع تر شدن آن، از وزن‌های سیناپس یا همان α استفاده نکنیم. بینش پس این تصمیم این است که، شبکه با استفاده از پارامتر β می‌تواند وزن‌های خود را بهینه کند و در واقع نرم کردن جریان ورودی،

[۳] این ترتیب مناسب برای لایه‌های کانولوشن سه بعدی می‌باشد.

مزیت افزوده ای برای نورون نخواهد داشت. به همین دلیل دو خط ۹ و ۱۰ الگوریتم ۲ را که قبل تر به آن اشاره کردیم را می‌توان به صورت زیر بازنویسی کرد:

Algorithm 3: پردازش لایه‌ی کامل بدون ثابت سیناپسی

$x \leftarrow inputSpikes$;
$y \leftarrow 0$;
$syn \leftarrow 0$;
$mem \leftarrow 0$;
$spk \leftarrow 0$;
for $i = 0$ *to Number of Frames* **do**
  $h \leftarrow x[i].W$;
  $rst \leftarrow spk$;
  $mem = (mem - rst) * beta + h * (1 - beta)$;
  $spk \leftarrow heavySide(mem)$;
  $y[i] = spk$;
end
return$y$;

بدین ترتیب دیگر نیازی به متغیر syn نخواهیم داشت و مقدار h به صورت مستقیم در mem تاثیر گذار خواهد بود. تنها نکته آن این است که مقدار h با استفاده از ضریب ۱ ـ beta کنترل می‌شود تا مقدار mem از یک تجاوز نکند.

۵.۲.۳ نرمال کردن اسپایک‌های خروجی

یکی از تکنیک‌هایی که موجب آموزش بهتر شبکه‌های عصبی می‌شود، استفاده از لایه‌های نرمال سازی دسته ای یا Batch-Norm است. استفاده از لایه در شبکه‌های عصبی مصنوعی باعث می‌شود دقت آموزش شبکه افزایش چشمگیری پیدا کند. دلیل این امر این است که داده‌های ورودی پس از شکسته شدن به دسته‌های کوچکتر توزیع متفاوت تری نسبت به داده ی اولیه پیدا می‌کنند. گاهی این توزیع می‌تواند به قدری متفاوت باشد که آموزش شبکه را دچار مشکل کند. همچنین در لایه‌های بالاتر، شبکه خروجی لایه‌ها می‌تواند خیلی متفاوت باشد که باعث کم شدن یا زیاد شدن بیش از حد سیگنال آموزشی شود که این امر باعث عدم آموزش شبکه می‌شود. استفاده از لایه‌های نرمال سازی، باعث می‌شود خروجی هر لایه در بازه ی نرمال قابل انتظاری قرار بگیرد و در نتیجه لایه بعدی می‌تواند به راحتی و با دقت بیشتری آموزش پیدا کند.

روش متداول نرمال سازی یا استفاده از لایه‌های Batch-Norm، بر روی شبکه‌های عصبی اسپایکی به خودی خود قابل اجرا نیست. چرا که این لایه‌ها مقادیر خروجی خود را به صورت پیوسته و حقیقی تولید می‌کنند. به عبارت دیگر عمل نرمال سازی متغیرهای اسپایکی را به متغیرهای حقیقی تبدیل می‌کند.

برای حل این مسئله، عمل نرمال سازی را، درون نورون پیاده سازی کرده ایم. پیاده سازی این روش در الگوریتم ۳ قابل مشاهده است. این الگوریتم همانند الگوریتم ۴ است و تنها تفاوت آن در محاسبه ی مقدار آستانه یا همان mthr است. این مقدار یک مقدار نرمال شده از پتانسیل نورون است. و در این الگوریتم، به جای محاسبه ی اسپایک‌ها از مقدار پتانسیل، اسپایک از مقدار نرمال شده ی mthr محاسبه می‌گردند.

Algorithm 4: پردازش لایه کامل همراه با نرمال‌ـ سازـ دسته‌ای

$nrom := ||w||2;$
$x \leftarrow inputSpikes;$
$y \leftarrow 0;$
$syn \leftarrow 0;$
$mem \leftarrow 0;$
$spk \leftarrow 0;$
for $i = 0$ *to Number of Frames* **do**
  $h \leftarrow x[i].W;$
  $rst \leftarrow spk * b * nrom;$
  $syn \leftarrow alpha * syn + h;$
  $mem \leftarrow beta * mem + syn - rst;$
  $mthr \leftarrow mem/norm + b;$
  $spk \leftarrow heavySide(mthr);$
  $y[i] = spk;$
end
return$y$;

در این الگوریتم علاوه بر وزن‌های آموزشی w، و beta یک متغیر دیگر نیز آموزش پیدا می‌کند که متغیر نرمال سازی یا همان b نام دارد که بین صفر و یک می‌تواند تغییر کند. در این الگوریتم ترم ریست، rst، به جای این که مستقیم با مقدار اسپایک‌های سیکل گذشته جایگزین شود، در b و نرمال دوم وزن‌ها ضرب می‌شود. بنابراین مقدار نرم تری نسبت به گذشته پیدا می‌کند، و بر اساس این که در سیکل گذشته تعداد اسپایک‌ها کم بوده یا زیاد، مقدار پتانسیل این سیکل لایه، کنترل می‌شود و در نتیجه می‌توان گفت، همواره پتانسیل نورون‌ها در رنج‌های معقولی قرار خواهند داشت.

۶.۲.۳ وزن‌های اولیه

تعیین وزن‌های اولیه در شبکه‌های اسپایکی از اهمیت بسیار زیادی برخوردار است و عدم تعیین درست آنها می‌تواند سبب کاهش دقت شبکه یا حتی آموزش ناپذیر شدن آن شود. دلیل این مسئله این است که خطای آموزش محاسبه شده در این شبکه‌ها با استفاده از اسپایک‌های خروجی شبکه محاسبه می‌گردد. و در صورتی که وزن سیناپس‌های نورون کم باشد ممکن است نورون هیچ اسپایکی تولید نکند و در این

صورت هیچ سیگنال آموزشی تولید نمی‌شود و شبکه نمی‌تواند برای تصحیح وزن‌های خود کاری کند. همچنین در صورتی که بازه ی تصادفی وزن‌ها بزرگ باشد، باعث می‌شود تا نورون‌هایی با پتانسیل کمتر در عمل نرمال کردن محو شوند و آموزش پیدا نکنند.

در آزمایش‌ها و پیاده سازی‌های انجام شده به این نتیجه رسیدیم که وزن‌ها می‌بایست میانگین صفر داشته باشند و انحراف معیار بین 0.001 تا 0.01 برای لایه‌های کانولوشنال و مقدار 0.4 تا 0.6 برای لایه‌های کامل که بهترین نتیجه را تولید می‌کند و باعث می‌شود شبکه در زمان مناسب به بیشتر دقت آموزشی برسد. لازم به ذکر است که مقدار وزن‌های شبکه با استفاده از توزیع نرمال، مقدار دهی می‌شوند و مقدار انحراف از معیار این توزیع از معادله ۲.۳ و ۳.۳ برای لایه‌های کانولوشن، و ۴.۳ و ۵.۳ برای لایه‌های کامل بدست می‌آید. مقدار std در این معادله همان مقدار انحراف از معیار است که قبل تر توضیح داده شد.

$$std_w = std.\sqrt{\frac{1}{c.\prod k_i}} \tag{۲.۳}$$

$$std_v = std.\sqrt{\frac{1}{c}} \tag{۳.۳}$$

در این معادله std_w انحراف از معیار نرمال مربوط به وزن‌های شبکه (وزن‌های کانولوشنال)، و std_v مربوط به وزن‌های بازگشتی (در صورت وجود) است. مقدار c تعداد کانال‌های خروجی لایه است و مقدار k_i اندازه ی کرنل در بعد iام را نشان می‌دهد.

$$std_w = std.\sqrt{\frac{1}{L^{i-1}}} \tag{۴.۳}$$

$$std_v = std.\sqrt{\frac{1}{L}} \tag{۵.۳}$$

در این معادله، L تعداد نورون‌های لایه فعلی یا به عبارت دیگر تعداد خروجی‌های لایه را مشخص می‌کند. و مقدار L^{i-1} تعداد نورون‌های لایه قبل یا همان تعداد ورودی‌های لایه را مشخص می‌کند.

۷.۲.۳ لایه DropOut

یکی دیگر از ایده‌های بهینه کردن شبکه و افزایش کارکرد آن، اضافه کردن لایه‌های حذف‌ـ کننده‌ـ تصادفی یا DropOut است. ایده اصلی لایه‌های DropOut، این است که در مرحله آموزش نورون به صورت تصادفی از عمل بازگشت خطا و بهینه شدن حذف می‌شود و این کار باعث افزایش دقت شبکه می‌شود. تعبیر این کار این است، که با این کار در واقع تعداد زیادی شبکه ی عصبی مختلف و موازی را همزمان در شبکه خواهیم داشت که با هم کار می‌کنند. ما در پیاده سازی خود از لایه‌های DropOut موجود در کتابخانه‌های استاندارد یادگیری ماشین استفاده کرده ایم.

۸.۲.۳ اتصالات جانبی

ایده ی استفاده از اتصالات جانبی، ایده ی تازه ای نیست، و در گذشته شبکه‌هایی ارایه شده‌اند که از این اتصالات بهره مند شده‌اند. به طور کلی این اتصالات می‌تواند کمک کند تا شبکه بتواند با لایه‌های کمتری نتایج مشابهی را تولید کند و قدرت مشابهی داشته باشد. اتصالات جانبی، اتصالاتی است مابین نورون‌های هر یک لایه، در نتیجه معادله ی جریان ورودی نورون به صورت معادله ی ۶.۳ تغییر پیدا می‌کند.

$$a_i^m = \sum_j W_{ij}^{m-1} z_j^{m-1} + \sum_{k \neq i} V_{ik}^m z_k^m \tag{۶.۳}$$

در این معادله، W وزن‌های شبکه و V وزن‌های بین نورون‌ها یا همان وزن‌های جانبی است. همچنین برای پیاده سازی این اتصالات در الگوریتم نورون، می‌بایست الگوریتم را به شکل زیر تغییر داد. ابتدا متغیر d را به صورت زیر در نظر می‌گیریم:

$$d = W.W^T \tag{۷.۳}$$

سپس برای محاسبه ی مقدار rst مقدار اسپایک‌های سیکل قبلی را در d ضرب می‌کنیم.

۹.۲.۳ اتصالات بازگشتی

اتصالات بازگشتی به ما اجازه می‌دهند تا شبکه‌هایی با معماری بازگشتی را پیاده سازی کنیم. صرف نظر از نوع لایه، در تمامی نورون‌ها قابلیت اتصالات بازگشتی پیاده سازی شده است بنابراین می‌توان این معماری را با لایه‌های کامل و لایه‌های کانولوشنال همزمان پیاده سازی کرد. پیاده سازی این لایه‌ها به این صورت است که در هر سیکل پردازش نورون، اسپایک‌های لایه را در سیکل زمانی قبلی در نظر می‌گیریم، و آن را در ورودی زمان حال نورون جمع می‌کنیم. برای درک این موضوع معادله ۸.۳ را در نظر می‌گیریم.

$$h(t) = I(t) + S(t-1).V \tag{۸.۳}$$

در این معادله I ورودی نورون از لایه قبلی است، S اسپایک‌های خروجی نورون، V وزن‌های مربوط به روابط بازگشتی و h جریان ورودی به رابطه ی پتانسیل نورون می‌باشد.

۱۰.۲.۳ استفاده از مکانیزم Winner-Takes-All

یکی از مکانیزم‌های استفاده شده در لایه تصمیم گیری شبکه، مکانیزم winner-takes-all است. این مکانیزم به طور خلاصه به این صورت عمل می‌کند که در لایه تصمیم گیری، در صورتی که پتانسیل یک نورون به آستانه ی تحمل آن برسد و اسپایک تولید کند، پتانسیل نورون‌های مجاور آن نیز کاهش پیدا می‌کند. به عبارت دیگر نورونی که زودتر اسپایک تولید کند سیگنال آموزشی دریافت می‌کند. بنابراین نورون‌ها برای زودتر اسپایک زدن با هم رقابت می‌کنند. از طرف دیگر استفاده از این مکانیزم باعث کاهش کلی اسپایک‌های تولید و در نتیجه خلوت شدن خروجی این لایه می‌شود.

۳.۳ تصمیم گیری و محاسبه‌ی تابع خطا

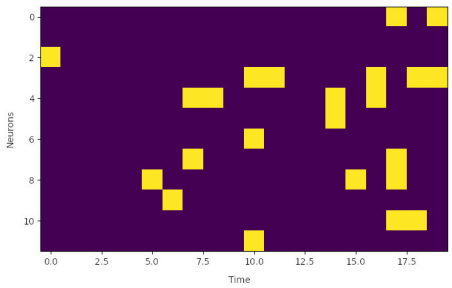
در این قسمت به بررسی ساختارهای پیاده سازی شده برای لایه تصمیم گیری و محاسبه خطا و همچنین برای بهینه سازی می‌پردازیم.

به طور کلی این شبکه را با دو دیدگاه پیاده سازی کردیم. دیدگاه اول این است که جریان داده را افراز کنیم و به صورت مجموعه داده ای با برچسب با آن برخورد کنیم. و دیدگاه دوم این است که شبکه را بر روی جریان داده آموزش دهیم، همانند روش‌های آموزش تقویت شده. هر کدام از این روش‌ها مزیت و معایب خود را دارند. دیدگاه اول به نظر روش بسیار مناسبی برای پردازش جریان داده‌ها است، ولی در عوض مسائل دشواری را وارد سیستم می‌کند. شاید مهمترین مشکل این روش نحوه ی پیاده سازی روش‌های بازگشت خطا است، و همچنین عدم توانایی شبیه سازی نورون‌ها به صورت لایه لایه است که این امر باعث شد تا این روش بسیار کند شود و نهایتا ما از دیدگاه دوم برای حل این مسئله استفاده کردیم.

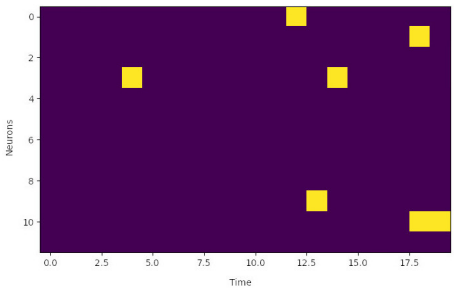
دیدگاه دوم روشی است که داده‌ها را به قسمت‌های مساوی افراز می‌کنیم تا هر داده تعداد مشخصی فریم زمانی داشته باشد. این روش در واقع همان روشی است که در قسمت‌های قبل برای تهیه و آماده سازی داده‌ها توضیح داده شد. مزیت این روش ساده تر بودن آن برای شبیه سازی در GPU است. و در عمل در آزمایش‌های انجام شده این روش کارایی بهتری را از خود نشان داده است.

۱.۳.۳ تصمیم گیری در لایه خروجی

نکته‌ای که در محاسبه‌ی تابع هزینه وجود دارد این است که محاسبه‌ی خروجی و برداشت آن از شبکه بدیهی نیست. به بیان دیگر، در لایه‌ی آخر شبکه‌ی اسپایکی، یک لایه‌ی کامل می‌باشد که در آن مقادیر پتانسیل نورون‌های آن در طول زمان تغییر می‌کند. برای تعیین یک کلاس مشخص از مقادیر خروجی این نورون‌ها، روش‌های متعددی را می‌توان تعریف کرد. روش‌هایی که در این پایان نامه پیاده سازی شده‌اند عبارتند از:



(آ) روش تعداد اسپایک



(ب) روش زمان اولین اسپایک

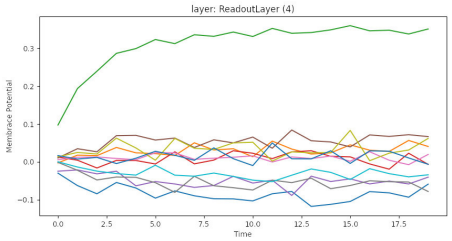
نمودار ۴.۳: در این نمودار فعالیت نورون‌های لایه‌ی آخر شبکه را مشاهده میکنیم. شکل بالا مربوط به حالتی است که شبکه بر اساس تعداد اسپایک های تولید کرده ارزیابی و آموزش داده شده است. شکل پایین مربوط شبکه در حالتی است که بر اساس کلاس اولین اسپایک تولید شده آموزش دیده است.

روش‌های مبتنی بر اسپایک زدن

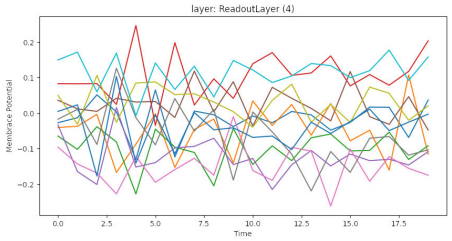
این روش‌ها خود شامل دو بخش می‌شوند. روش اول تعداد اسپایک و روش دوم زمان اولین اسپایک. در روش اول، کلاس خروجی کلاسی است که متعلق به نورونی است که بیشترین تعداد اسپایک را داشته و روش دوم، خروجی متعلق به نورونی است که اولین اسپایک را تولید کرده است. بعد از بررسی‌ها به این نتیجه رسیدیم که مشکل روش اول، تولید بسیار زیاد اسپایک است، که این امر سبب از بین رفتن بهینگی و دقت شبکه می‌شود. و روش دوم کارایی بهتری را نشان داده چرا که شبکه سعی می‌کند اسپایک‌های کمتری تولید کند. ولی از طرف دیگر، مشکل این روش این است که اسپایک‌ها با تاخیر بیشتری تولید می‌شوند. همین امر باعث کندتر شدن فرایند آموزش می‌شود.

روش‌های مبتنی بر پتانسیل

این روش‌ها بر این اساس بنا شده‌اند که نیازی به شمارش اسپایک‌های خروجی نداشته باشیم و با بررسی مقادیر پتانسیل نورون‌ها، کلاس خروجی را انتخاب کنیم. در این دسته نیز دو روش پیاده سازی شده است. روش اول، محاسبه‌ی میانگین پتانسیل نورون‌ها و انتخاب نورون با بیشترین میانگین و روش دوم، محاسبه‌ی بیشینه پتانسیل هر نورون (در طول مدت شبیه سازی شبکه بر روی داده) و انتخاب نورون با بیشترین بیشینه پتانسیل است. بین این دو روش نیز روش دوم، کارایی بهتری را نشان داده است چرا



(آ) روش بیشینه پتانسیل



(ب) روش میانگین پتانسیل

نمودار ۵.۳: در این نمودار پتانسیل نورون‌ها در طول زمان پس از آموزش دیدن نشان داده شده. شکل بالا نشان دهنده‌ی لایه خروجی با تصمیم گیرنده‌ی بیشینه است و شکل پایین نشان دهنده‌ی همین لایه با تصمیم گیرنده‌ی میانگین است. همانطور که مشاهده می‌شود، تصمیم گیرنده‌ی بیشینه فشار بیشتری برای پایین نگاه داشتن پتانسیل وارد میکند.

که این روش فشار بیشتری (خطای بیشتر) را بر روی سایر نورون‌ها وارد می‌کند تا در هیچ نقطه‌ای مقدار زیاد نداشته باشند، ولی در روش اول فقط کافی بود که سایر نورون‌ها در بیشتر نقاط، مقدار کمی داشته باشند و می‌توانستند گاهی مقدار بالا بگیرند.

۲.۳.۳ محاسبه‌ی تابع خطا

پس از مشخص شدن نوع خروجی، باید مقدار خطای شبکه را محاسبه کنیم. همانطور که قبل تر نیز گفته شد، قدم اول در این زمینه تعریف تابع خطا یا تابع هزینه است. از آنجایی که هدف ما در این پایان نامه تولید شبکه با قابلیت کلاس بندی است، تابع خطای احتمالی لگاریتمی منفی به عنوان تابع هزینه انتخاب شده است. در این تابع سعی می‌کنیم تا اختلاف مقدار خروجی را با مقدار مورد نظر که هر دو بین صفر و یک هستند را با شیب لگاریتمی در مقدار خطا وارد کنیم. این مقدار در معادله‌ی ۹.۳ نشان داده شده است.

$$L = -\frac{1}{n}\sum_{x}\sum_{k=1}^{K} y_k \log(a_k) \tag{۹.۳}$$

در این معادله x داده‌های ورودی است که مقدار تابع هزینه را روی آن محاسبه می‌کنیم. K تعداد کل کلاس‌های مسئله است و y_k مقدار مشخص شده برای کلاس kام در مجموعه داده است. مقدارa نیز

بیانگر مقدار خروجی مدل است. لازم به ذکر است که مقدار y_k می‌تواند صفر یا یک باشد و مقدار a بین صفر تا یک می‌تواند مقدار بگیرد. بنابراین در این فرمول در واقع برای هر داده از مجموعه داده، تنها یک مقدار $\log(a_k)$ را محاسبه خواهیم کرد (مقدار متناظر با کلاس مربوطه) و سایر مقادیر در تابع هزینه لحاظ نمی‌شوند چرا که مقدار y_k آنها صفر است. در ادامه برای بهینه کردن وزن‌های شبکه از معادله‌ی ۱۰.۳ استفاده می‌کنیم.

$$W_{ij} \leftarrow W_{ij} - \eta \frac{\partial L}{\partial W_{ij}} \tag{۱۰.۳}$$

سپس برای وزن‌های لایه خروجی خواهیم داشت:

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial O_i}\frac{\partial O_i}{V_i}\sum_t \frac{\partial V_i}{\partial U_i(t)}\frac{\partial U_i(t)}{\partial W_{ij}} = (O_i - 1)\sum_t \frac{\partial U_i(t)}{\partial W_{ij}} \tag{۱۱.۳}$$

که در این معادله برای محاسبه‌ی $\frac{\partial U_i(t)}{\partial W^b_{ij}}$ از معادله‌ی ۱۴.۳ تا ۱۶.۳ استفاده می‌کنیم. در لایه‌های بعدی خطای انتقال داده شده را با δ نشان می‌دهیم. بنابراین خواهیم داشت:

$$\frac{\partial U_i(t)}{\partial W_{ij}} = \sum_t \frac{\partial L}{\partial S_i(t)}\frac{\partial S_i(t)}{\partial W_{ij}} = \sum_t \delta^h_i \frac{\partial S_i(t)}{W_{ij}} \tag{۱۲.۳}$$

$$\delta^o_i = (O_i - 1) \tag{۱۳.۳}$$

در معادله‌ی ۱۲.۳ برای محاسبه‌ی $\frac{\partial S_i}{\partial W_{ij}}$ می‌توانیم از معادله‌ی زیر استفاده کنیم.

$$\frac{\partial S_i(t)}{\partial W_{ij}} = \Theta'(U_i(t))\frac{\partial U_i(t)}{\partial W_{ij}} \tag{۱۴.۳}$$

$$\frac{\partial U_i(t)}{\partial W_{ij}} = \beta\frac{\partial U_i(t-1)}{W_{ij}} + \frac{\partial I_i(t-1)}{W_{ij}} \tag{۱۵.۳}$$

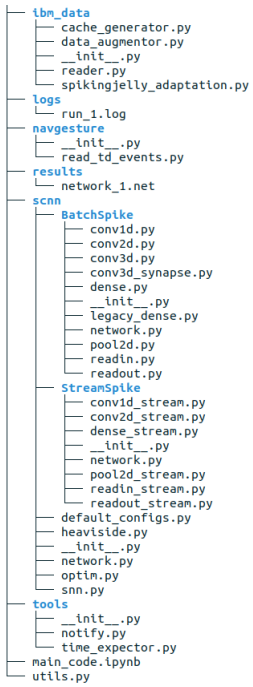
$$\frac{\partial I_i(t)}{W_{ij}} = \alpha\frac{I_i(t-1)}{W_{ij}} + S_j[t-1] \tag{۱۶.۳}$$

تنها مقدار باقی مانده در این معادله مقدار $\Theta(U)$ است. همانطور که قبل تر گفته شد، مشتق تابع Θ قابل محاسبه نیست چرا که تابع ضربه در نقطه‌ی صفر ناپیوسته و در سایر نقاط مشتق صفر دارد. برای حل این موضوع می‌توان از روش گرادیان جایگزین ارائه شده در [۲۸] استفاده کرد.

۴.۳ پیاده سازی

همانطور که قبل‌تر نیز گفته شد، پیاده‌سازی این پایان نامه در زبان پایتون ۳ [۴] و در پلتفرم پایتورچ [۵] انجام شده. دلیل استفاده از پایتون وجود کتابخانه‌های فراوان در زمینه‌های علوم داده، پردازش داده و پردازش تصویر بوده و دلیل استفاده از پلتفرم پایتورچ بهره مندی از ابزارهای مربوط به شبکه‌های عصبی و یادگیری ماشین از جمله قابلیت‌های محاسبه‌ی اتوماتیک گرادیان توابع، ابزارهای بهینه سازهای مانند Adam است.

برای بررسی پیاده سازی، بهتر است نگاهی به ساختار فایل بندی پروژه کنیم. همانطور که در ۶.۳ قابل مشاهده است، پروژه دارای یک پوشه‌ی اصلی به نام scnn است که در آن فایل‌های مربوط به پیاده سازی شبکه وجود دارد. در کنار آن پوشه‌های navgesture و ibm-data قرار دارد که محل قرار گیری فایل‌های پیاده سازی رابط داده‌ها [۶] (رابط کاربری، مربوط به کار با مجموعه داده‌های متناظر آنها) قرار دارد. در پوشه‌ی tools نیز ابزارهای جانبی که مربوط به اجرای نرم‌افزار هستند قرار گرفته‌اند. نهایتا دو پوشه‌ی logs و results نیز پس از اجرای شبکه تکمیل خواهند شد که حاوی فایل‌های خروجی شبکه هستند. فایل‌هایی نیز در پوشه‌ی اصلی پروژه قرار دارند که مهم‌ترین آنها main-code.py و main-code.ipynb هستند، که به ترتیب فایل اجرای پروژه در قالب پایتون و دفترچه‌ی ژوپیتر [۷] هستند.



نمودار ۶.۳: شمای پوشه‌ها و فایل‌های پروژه.

---

[۴] Python
[۵] PyTorch
[۶] Data Interface
[۷] Jupiter Notebook

در پوشه‌ی scnn دو زیر پوشه‌ی BatchSpike و StreamSpike قرار دارند که نمونه‌ی مشابه‌ی از پیاده سازی شبکه‌ی اسپایکی هستند، با این تفاوت که BatchSpike برای پردازش به صورت دسته‌ای[۸] و دیگری برای پردازش به صورت پیوسته‌ی[۹] داده‌ها می باشد. پردازش دسته‌ای برای افزایش عملکرد شبکه مناسب‌تر است در حالی که پردازش پیوسته به مراتب شباهت بیشتری به ساختار مغزی دارد. همانطور که در بخش‌های گذشته گفته شد، در شبیه سازی بر روی GPU، پردازش دسته‌ای دارای بهره وری بالاتری است. به همین دلیل در این پایان نامه، برای این که بتوانیم از حداکثر توان پردازشی موجود استفاده نماییم، بر روی پردازش دسته‌ای تمرکز بیشتری کرده‌ایم. در ادامه خلاصه‌ای از کلاس‌های پیاده سازی شده در این پوشه‌ها را بررسی می نماییم.

معماری استفاده شده در این پیاده سازی، به گونه‌ای است که کاربر بتواند به سادگی شبکه‌های مختلف تولید کند و به سادگی بتواند لایه‌های شخصی سازی شده را نیز به شبکه‌ی خود اضافه کند. برای همین منظور، از یک کلاس پایه برای لایه‌ها به نام neuron-base استفاده شده است و تمامی لایه‌های پیاده سازی شده نیز از این کلاس ارث بری می‌کنند. در نتیجه هر کلاسی که از این کلاس ارث بری کرده باشد میتواند به شبکه اضافه شود. این کلاس انتزاعی، مدل پایه‌ای از یک لایه‌ای از نورون‌های اسپایکی است. یکی از مهمترین این پارامترهای این کلاس، تعداد نورون‌های ورودی و خروجی لایه، تعداد کانال‌های ورودی و خروجی آن (برای نورون‌های کانولوشنال) و نوع تابع اسپایک است. همچنین مکانیزم‌های Dropout ، Weight-Decay و اتصالات بازگشتی هم به دلیل داشتن پیاده سازی یکسان برای انواع لایه‌ها، در این کلاس پیاده سازی شده‌اند.

برای پیاده سازی و اضافه کردن لایه‌ی جدید در شبکه تنها کافی است از کلاس neuron-base ارث بری کرده و توابع زیر را در لایه‌ی مورد نظر پیاده سازی شود. همچنین سه پارامتر IS-CONV ، IS-SPIKING و HAS-PARAM نیز باید مقدار دهی و مشخص شوند. پارامتر اول نشان دهنده‌ی کانولوشنال بودن لایه است. به کمک این پارامتر، شبکه به صورت خودکار بین لایه‌های کانولوشنال و کامل داده‌ها را به یک بعد کاهش می‌دهد[۱۰]. پارامتر دوم نشان دهنده‌ی اسپایکی بودن لایه است، و نهایتا پارامتر آخر نشان دهنده‌ی این است که لایه نیاز به آموزش دارد یا خیر. بعد از مشخص کردن پارامترها کافی است توابع زیر را پیاده سازی کرده تا بتوانیم لایه را در شبکه بارگذاری کنیم:

- ___init___: این تابع که تابع سازنده در زبان پایتون بوده و مربوط به ساختار دهی کلاس می باشد. در این تابع، باید تابع ___init___ از کلاس والد فراخوانی شود. پس از آن می‌توان از مقادیری ورودی کاربر و همچنین، مقادیر تعداد نورون‌ها و کانال‌ها استفاده کرد.
- forward-function: تابع اصلی مدل نورونی هر لایه است. این تابع در هر بار اجرای شبکه، برای هر لایه یک بار فراخوانی می‌شود. در این تابع باید معادلات مربوط به اجرا و پردازش نورون‌ها پیاده سازی شود.

[۸] Batch Processing
[۹] Stream Processing
[۱۰] در کتابخانه‌های یادگیری ماشین مانند پایتوچ، این عملگر عموما با نام Flatten شناخته می‌شود

- train-able این تابع لیستی از وزن‌های قابل آموزش را باز می‌گرداند تا کلاس network بتواند این پارامترها را شناسایی و آموزش دهد.
- ___str___ این تابع شمای لایه را به صورت نوشتاری باز می‌گرداند.
- reset-parameters این تابع وزن‌های لایه را با مقادیر تصادفی که مشخص شده مقدار دهی می‌کند.
- draw این تابع، برای رسم نمودارهای مربوط تحلیل شبکه و نورون‌ها فراخوانی می‌شود.

در مرحله‌ی بعد، دو لایه‌ی readin و readout قرار دارند. لایه‌ی readin نقش شکل دهی به داده‌های ورودی را دارد به طوری دارای پارامترهای مشابه‌ی نسبت به سایر لایه‌ها باشند. لایه‌ی readout نقش تفسیر خروجی شبکه را دارد. نحوه‌ی عملکرد این لایه در فصل ۱.۳.۳ به طور مفصل توضیح داده شده.

کلاس‌های dense و legacy-dence هر دو پیاده سازی لایه‌ی کامل هستند و تنها تفاوت این دو در پیاده سازی است. استفاده از مکانیزم وزن‌های سیناپسی در لایه‌ی dense صرف نظر شده، چرا که در در آزمایش‌های انجام شده نشان دادیم که این مکانیزم تاثیر چندانی روی کارکرد شبکه نداشته است..

در این پیاده سازی از لایه‌های کانولوشنال یک، دو و سه بعدی استفاده شده است که هر کدام، از لایه‌ی قبلی خود ارث بری کرده‌اند. وجه تمایز این کلاس‌ها در تابع forward-function آنها است. اگرچه این تابع برای هر سه نوع به طور مشابه‌ی عمل می‌کند، و تفاوت آنها در ضرب‌های ماتریسی آنها، و نوع عملگر کانولوشن آنهاست. در این کلاس‌ها برای عملگر کانولوشن، از توابع موجود در پایتورچ استفاده شده است. دلیل این امر یکسان بودن این عملگر بودن در شبکه‌های اسپایکی و شبکه‌های غیر اسپایکی است.

در این پوشه‌ی همه کلاس‌ها به استثنای کلاس network، مربوط به پیاده سازی لایه‌ها هستند. کلاس network که از ماژول torch.nn.Module واقع در پایتورچ ارث بری کرده است، وظیفه‌ی ذخیره و نگهداری لایه‌های شبکه و فراخوانی توابع آنها در صورت لزوم را دارد. توابع کمکی به همین منظور پیاده سازی شده‌اند. به عنوان مثال تابع predict که به ترتیب تمامی لایه‌های شبکه را اجرا میکند و خروجی لایه‌ی آخر را باز می‌گرداند. یکی دیگر از وظایف مهم این کلاس، پیاده سازی تابع fit است. فراخوانی این تابع منجر به آموزش شبکه می‌شود. این تابع در ابتدا تنظیمات مورد نیاز برای آموزش شبکه را انجام داده، سپس به تعداد ایپاک مورد نیاز عمل آموزش لایه‌ها را تکرار می‌کند.

پیاده سازی روش گرادیان جایگزین، در کلاس heavyside انجام شده، برای این کار تابع پله (تابع ضربه) را با ارث بری از کلاس autograd یا همان گرادیان خودکار، موجود در کتابخانه‌ی پایتورچ پیاده سازی کردیم. این کلاس همانند یک تابع اسپایک سخت کار می‌کند، با این تفاوت که در هنگام فراخوانی، مقادیر ورودی را ذخیره کرده و نورون‌هایی که اسپایک تولید کرده‌اند را در حافظه نگاه می‌دارد. سپس با باز نویسی تابع مشتق (در اینجا تابع backward) مشتق مقادیر لایه را با استفاده از اسپایک‌هایی که در طول زمان‌های گذشته تولید شده‌اند محاسبه می‌کند.

آخرین کلاس از مجموعه‌ی اصلی کد، کلاس SNN می‌باشد. این کلاس رابط کاربری شبکه محسوب گردیده و نقطه‌ای است که در آن کاربر با شبکه در تعامل است. این کلاس شامل توابعی برای اضافه کردن لایه‌های مختلف و ساختن شبکه، توابعی برای ذخیره سازی و بازیابی شبکه‌ها از حافظه، و همچنین توابعی برای رسم نمودارهای مورد نیاز برای تحلیل شبکه است. درعمل این کلاس، پوششی برای کلاس Network است.

در کنار پوشه‌ی scnn دو پوشه‌ی دیگر برای کار با مجموعه داده‌ها وجود دارد، پوشه‌ی ibm-data و پوشه‌ی navgesture. پوشه‌ی اول برای کار با مجموعه داده‌ی IBM-DVSGesture و پوشه‌ی navgesture برای کار با مجموعه داده‌ی NavGesture است. لازم به ذکر است که هر دوی این پوشه‌ها حاوی تابع یکسانی به نام load-all واقع در فایل __init__.py هستند که دارای ورودی و خروجی یکسانی است. بنابراین کاربر بدون نیاز به دانستن نحوه‌ی کارکردن این ماژول‌ها می‌تواند از آنها استفاده کند. هر دو ماژول وظیفه‌ی خواندن داده‌ها از دیسک، پیش پردازش آنها و بارگذاری آنها در حافظه‌ی رم برای استفاده‌ی شبکه را دارد. ماژول ibm-data دارای دو پیاده سازی: پیاده سازی بر مبنای کتابخانه‌ی SpikingJelly و پیاده سازی بومی. پیاده سازی SpikingJelly مربوط به روش ۲.۱.۳ بوده و پیاده سازی بومی مربوط به سه روش دیگر است.

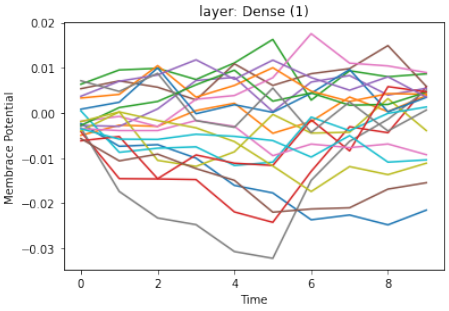
۵.۳ نتایج

در قدم اول به بررسی نورون‌های شبکه می‌پردازیم و اطمینان حاصل می‌کنیم که شبکه به درستی کار می‌کند. برای این کار، ابتدا یک شبکه کوچک، متشکل از دو لایه،که لایه اول شامل ۲۰ نورون اسپایکی و لایه دوم نیز لایه تصمیم گیرنده که شامل دو نورون است را تشکیل می‌دهیم. در این مثال، ورودی به صورت مجموعه داده‌ی کوچکی از تصاویر مبتنی بر فریم اسپایکی می‌باشند که هر کدام شامل ۱۰ فریم هستند. و تصاویر این مجموعه به دو دسته‌ی اسپایک‌های روی قطر اصلی و اسپایک‌های روی قطر فرعی تصویر تقسیم بندی می‌شوند. فعالیت نورون‌های لایه اول و دوم این شبکه، قبل و بعد از آموزش دیدن در شکل ۷.۳ مشاهده می‌شود. همانطور که مشاهده می‌شود، این شبکه پس از آموزش دیدن به سادگی می‌تواند این تصاویر را تفکیک کند.

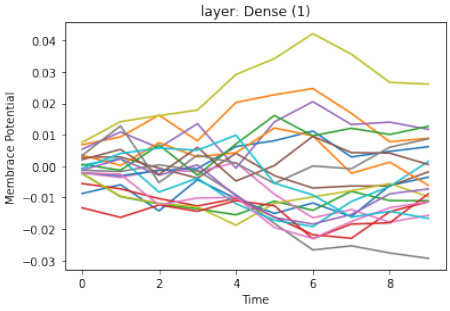
در مرحله‌ی بعد، نوبت به انجام همین آزمایش برای لایه‌های کانولوشنال است. تنها تغییر مورد نیاز برای این کار، جایگزینی لایه اول شبکه‌ی قبلی با یک لایه‌ی کانولوشنال، و سپس اجرای مجدد شبکه است. این لایه‌ها نیز به سادگی قادر به یادگیری این مجموعه هستند.

۱.۵.۳ آزمایش شبکه‌ی کامل

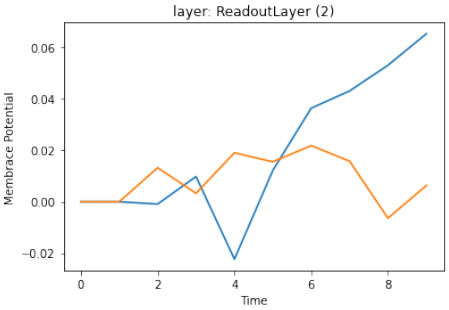
در قدم بعدی، یک شبکه‌ی کامل، با یک لایه با ۱۲۸ نورون اسپایکی و یک لایه تصمیم گیری را با استفاده از مجموعه داده‌ی IBM-DVSGesture آموزش دادیم. داده‌ها با طول ۲۰ فریم از مجموعه داده



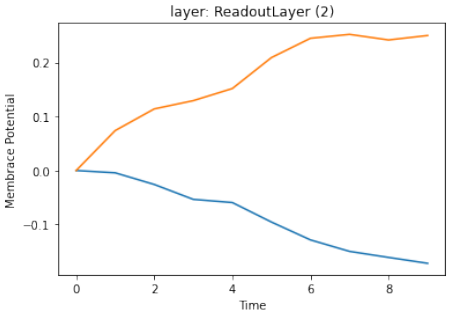
(آ) لایه اول قبل از آموزش



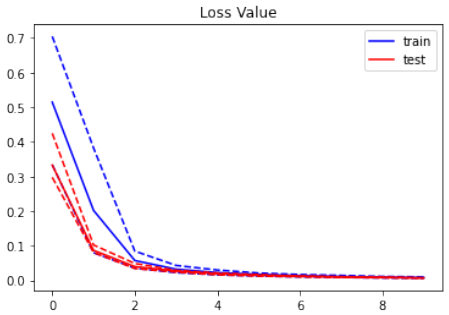
(ب) لایه اول بعد از آموزش



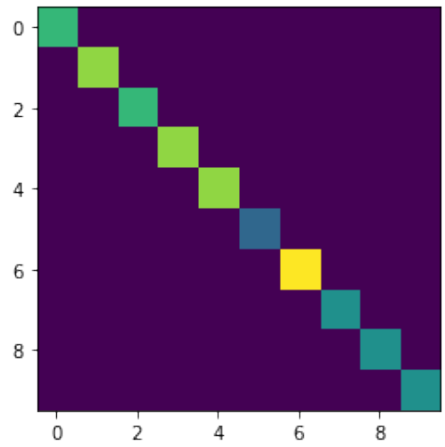
(ج) لایه تصمیم گیری قبل از آموزش



(د) لایه تصمیم گیری بعد از آموزش



(ه) نمودار تابع خطا در طول فرایند آموزش



(و) نمونه‌ای از داده‌ی آموزشی

نمودار ۷.۳: در دو شکل اول پتانسیل تعدادی نورون تصادفی از لایه‌ی اول مشاهده می‌شود و همانطور که مشاهده می‌شود، نورون‌های آموزش دیده رفتار تفکیک شده تری دارند. در دو شکل بعدی هم، همین امر را در مورد نورون‌های لایه‌ی تصمیم گیری مشاهده می‌کنیم.

استخراج شده‌اند و در دسته‌های ۱۶تایی آموزش پیدا کردند. روش استخراج قطبیت اسپایک‌ها نیز به روش جمع جهت اسپایک‌ها بوده است. همچنین در این آزمایش شبکه در ۴۰ سیکل آموزشی با ضریب آموزشی 10^{-4} آموزش پیدا کرده است. در تصویر ۸.۳ مشاهده می‌شود، این شبکه توانسته است با دقت ٪۱۰۰ داده‌های آموزشی، و با دقت نزدیک به ٪۷۰ داده‌های آزمایشی را به درستی شناسایی کند. این آزمایش نشان دهنده‌ی کارکرد اجزای شبکه در کنار هم می‌باشد. ولی همانطور که انتظار می‌رود، شبکه در این حالت دچار overfitting شده است. در ادامه به بررسی دقیق تر پارامترهای شبکه و تاثیر هر کدام بر روی کارکرد آن می‌پردازیم.

۲.۵.۳ آزمایش روش‌های تولید تصویر مبتنی بر فریم

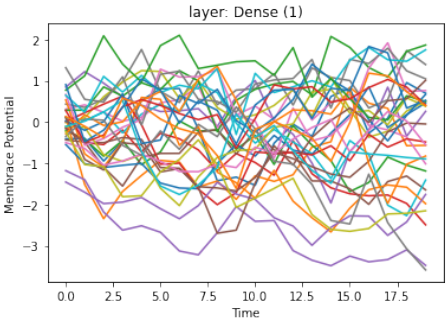
در این قسمت روش‌های استخراج و افراز اسپایک‌ها از جریان اسپایک و تولید تصویر مبتنی بر فریم را بررسی می‌کنیم. این روش‌ها همانطور که در بخش‌های قبلی بررسی شده‌اند، شامل روش‌های زیر می‌شوند:

- روش صرف نظر کردن از قطبیت اسپایک
- روش تک‌لایه (ذخیر مقدار قطبیت اسپایک در یک کانال تصویر)
- روش دولایه (ذخیره‌ی اسپایک در دولایه‌ی مجزا بر اساس قطبیت آنها)
- روش تجمعی (جمع کردن مقادیر قطبیت اسپایک‌ها)

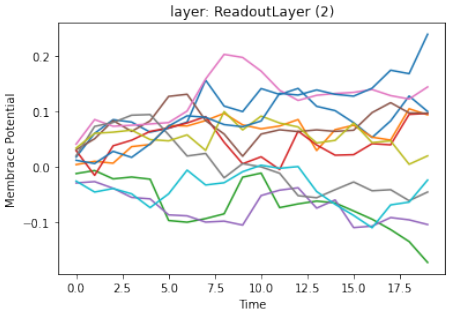
در این آزمایش از چهار شبکه مختلف استفاده شده، که شامل دو شبکه‌ی کامل و دو شبکه‌ی کانولوشنال می‌باشند. با این کار، این چهار روش در هر دو معماری ساده و کانولوشنال بررسی می‌شوند. لازم به ذکر است این آزمایش بر روی مجموعه داده‌ی NavGesture انجام شده است و تنها از ۳۰ فایل داده‌ی آموزشی به جای کل این مجموعه استفاده است. همانطور که در جدول ۱.۳ مشاهده می‌شود، روش تجمیع قطبیت کارکرد بهتری را نشان می‌دهد. دلیل این افزایش کارکرد، احتمالا این است که در سایر روش‌ها مقداری از اطلاعات از بین می‌رود. چرا که اسپایک‌ها روی هم انباشته نمی‌شوند.

شبکه	صرف نظر از قطبیت	روش تک لایه	روش دو لایه	روش تجمیع قطبیت
Dense.۱۲۸	۵۱%	۵۰%	۷۰%	۶۸%
Dense.۵۱۲	۸۰%	۸۴%	۸۳%	۸۷%
CNN.۶۴_۶۴	۸۲%	۸۳%	۸۴%	۹۶%
CNN+Dense	۷۹%	۸۳%	۸۱%	۹۷%

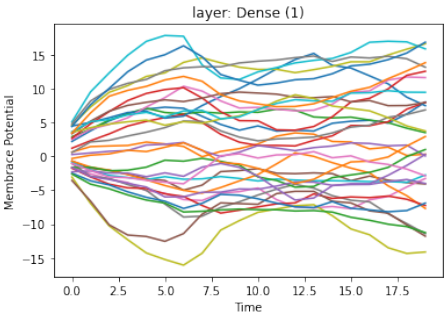
جدول ۱.۳: نتایج آزمایش‌های روش تولید تصویر مبتنی بر فریم



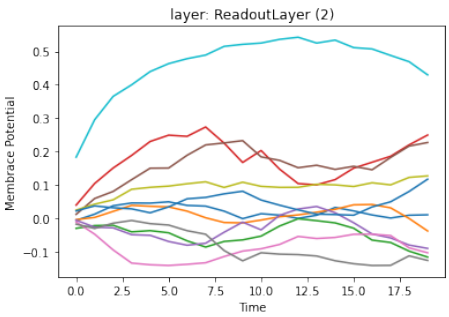
(آ) لایه اول قبل از آموزش



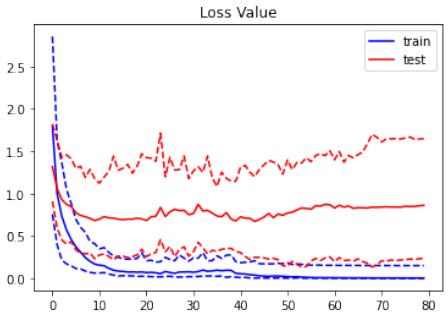
(ب) لایه اول بعد از آموزش



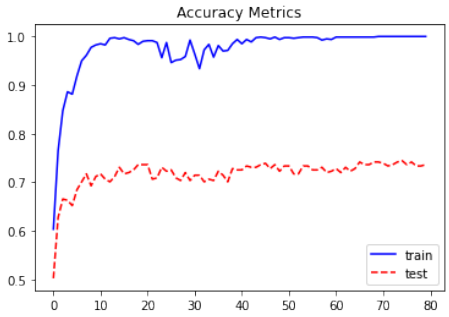
(ج) لایه تصمیم گیری قبل از آموزش



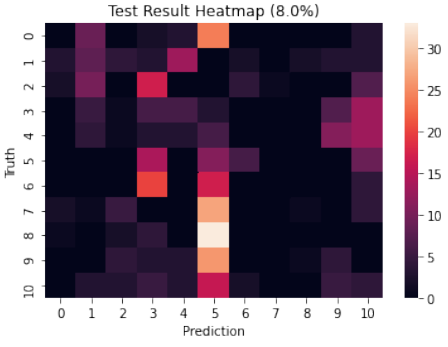
(د) لایه تصمیم گیری بعد از آموزش



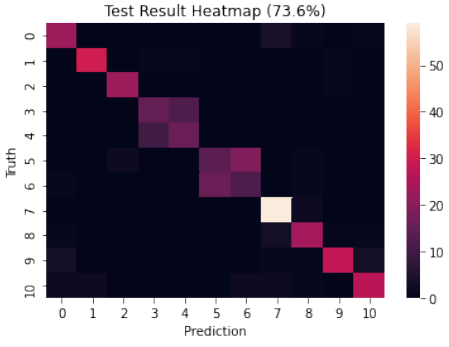
(ه) نمودار تابع خطا در طول فرایند آموزش



(و) نمودار درصد دقت شبکه در طول فرایند آموزش



(ز) ماتریس پراکندگی جواب‌ها قبل از آموزش



(ح) ماتریس پراکندگی جواب‌ها بعد از آموزش

نمودار ۸.۳: در این نمودار، پتانسیل لایه‌های شبکه را قبل و بعد از آموزش مشاهده می‌کنیم. همچنین بر اساس ۸.۳ز و ۸.۳ح می‌توانیم به این نتیجه برسیم که شبکه تا حد قابل قبولی توانایی تشخیص کلاس‌های درست را دارد.

شبکه	IBM-DVSGesture			NavGesture		
	۲۰ فریم	۸۰ فریم	۱۲۸ فریم	۲۰ فریم	۸۰ فریم	۱۲۸ فریم
Dense.۱۲۸	۷۸%	۷۵%	۷۸%	۶۸%	۸۶%	۸۴%
Dense.۲۵۶	۷۱%	۷۵%	۷۵%	۸۷%	۸۸%	۸۶%

جدول ۲.۳: نتایج آزمایش‌های طول فریم

شبکه	میانگین پتانسیل	بیشینه پتانسیل	تعداد اسپایک	زمان اولین اسپایک
Dense.۱۲۸	۷۵%	۷۸%	۶۳%	۶۸%
Dense.۵۱۲	۷۴%	۷۱%	۶۴%	۷۰%

جدول ۳.۳: نتایج آزمایش نوع خروجی

۳.۵.۳ آزمایش طول زمانی داده‌ها

در این آزمایش، طول زمانی داده‌ها و تاثیر آنها بر روی کارکرد شبکه بررسی شده است. همانند آزمایش قبلی، در این آزمایش نیز از شبکه‌هایی با پارامترهای مشابه استفاده شده است، و تنها تفاوت این آزمایش در این است که از هر دو مجموعه داده استفاده شده است.

۴.۵.۳ آزمایش نوع خروجی

در این آزمایش نوع لایه‌های تصمیم گیری با یکدیگر مقایسه شده‌اند. برای انجام این آزمایش از شبکه‌هایی با پارامترهای مشابه و از مجموعه داده‌ی IBM-DVSGesture استفاده کرده‌ایم. تنها تفاوت این شبکه‌ها در نوع لایه‌ی تصمیم گیری آنها است. همانطور که مشاهده می‌شود، حالت‌های تصمیم گیری بر اساس پتانسیل، کارکرد چشمگیری نسبت به حالت‌های تصمیم گیری بر اساس اسپایک دارند و همچنین تصمیم گیری بر اساس بیشینه پتانسیل نیز، نسبت به سایر حالت‌ها دارای کارکرد بهتری است.

۵.۵.۳ آزمایش اتصالات کمکی

در این قسمت نیز انواع اتصالات بازگشتی و جانبی در کنار یکدیگر مقایسه شده‌اند. این آزمایش نیز تنها بر روی مجموعه داده‌ی IBM-DVSGesture انجام شده. همانطور که در این آزمایش مشاهده می‌شود، اتصالات بازگشتی و جانبی تاثیر چشمگیری در کارکرد شبکه ندارند و تنها باعث افزایش حجم پردازش و افزایش حافظه‌ی مصرفی شبکه می‌گردند. البته در مورد اتصالات تحقیقات بیشتری نیاز است، چرا که ممکن است این اتصالات در مورد مسائل دیگر کارکرد خوبی از خود نشان دهند.

شبکه	بدون اتصالات کمکی	با اتصالات بازگشتی	اتصالات جانبی	با هر دو نوع اتصال
Dense.۱۲۸	۷۸%	۷۱%	۷۲%	۷۲%
Dense.۵۱۲	۷۱%	۷۷%	۷۵%	۷۷%

جدول ۴.۳: نتایج آزمایش‌های اتصالات کمکی

شبکه	شبکه‌ی اسپایکی		شبکه‌ی مصنوعی	
	دقت	محاسبات اعشاری	دقت	محاسبات اعشاری
Dense.۱۲۸	۹۴%	$2.89e4$	۹۱%	$1.04e6$
Dense.۲۵۶	۹۶%	$5.8e4$	۸۹%	$2.09e6$
CNN.۱۲۸	۸۳%	$4.38e6$	۹۷%	$2.2e8$
CNN.۳۲_۳۲	۹۷%	$2.7e5$	۹۴%	$1.2e7$

جدول ۵.۳: نتایج آزمایش‌های مقایسه‌ی شبکه‌های اسپایکی و شبکه‌های مصنوعی

۶.۵.۳ مقایسه شبکه‌ی اسپایکی با شبکه‌ی مصنوعی

در ادامه مباحثی که در بخش‌های قبلی در رابطه با مزیت‌های استفاده از شبکه‌های اسپایکی گفته شد، در این قسمت نتایج این شبکه‌ها را با شبکه‌های مصنوعی مشابه آنها مقایسه کرده‌ایم. همانطور که در جدول ۵.۳ مشاهده می‌شود، شبکه‌های اسپایکی می‌توانند با دقتی مشابه شبکه‌های مصنوعی داده‌های این مجموعه را یاد بگیرند. این در حالی است که به طور میانگین ۲ درصد شبکه‌های مصنوعی محاسبات اعشاری دارند که به این معنا است که دارای مصرف انرژی بسیار پایین تر می‌باشند. همچنان در صورتی که بر روی سخت افزارهای نورومورفیک اجرا شوند زمان پاسخ دهی سریع تری نیز خواهند داشت. کمتر بودن تعداد محاسبات اعشاری این شبکه به دلیل پراکندگی ماتریس اسپایک خروجی نورون‌ها می‌باشد که در نتیجه از محاسبه‌های اضافی بعدی جلوگیری می‌شود.

۷.۵.۳ نتیجه گیری

هدف اصلی در این پایان نامه استفاده از شبکه‌های اسپایکی برای حل مسئله‌ی تشخیص ژست حرکتی است. دلیل انتخاب این شبکه‌ها نیز این است که ساختار و مکانیزم کاری نورون‌های این شبکه‌ها به مغز ما شبیه تر هستند، و در نتیجه می‌توان گفت با مصرف انرژی کمتر کارکرد بهتری خواهند داشت. بنابراین، می‌توان گفت که احتمالا در آینده‌ای نه چندان دور، شاهد استفاده‌ی گسترده تری از این شبکه‌ها در صنعت خواهیم بود. کما این که در حال حاضر نیز بسیاری از صنایع تکنولوژی در حال پیاده سازی بسترهای استفاده از این شبکه‌ها هستند.

در این پایان نامه ما به بررسی این شبکه‌ها و چالش‌های آنها و همچنین به بررسی تاثیر مکانیزم‌ها و ابزارهای موجود در این شبکه‌ها و نقاط قوت و ضعف آنها پرداختیم. همچنین، در این پایان نامه پیاده

سازی کاملی از شبیه ساز این شبکه‌ها انجام دادیم که پلتفرم خوبی برای تحقیقات آینده در این زمینه است.

شبکه‌های اسپایکی، اگرچه در محیط‌های شبیه سازی کندتر عمل می‌کنند و شبیه سازی آنها به نسبت به شبکه‌های مصنوعی دشوارتر است، ولی صرف نظر از سربار شبیه سازی آنها، این شبکه بسیار سریع تر از شبکه‌های مصنوعی عمل می‌کنند و مصرف انرژی آنها بسیار کمتر است. دلیل اصلی این امر نیز، استفاده از وزن‌های باینری و تولید خروجی پالس در این شبکه‌ها است. به عبارت دیگر، به دلیل پراکنده بودن ماتریس پالس‌های خروجی، این شبکه‌ها خروجی‌های کمتری تولید می‌کنند.

نهایتا به عنوان زمینه‌های کاری آینده، می‌توان به پیاده سازی این شبکه‌ها در سخت افزارهای نورومورفیک اشاره کرد. از آنجایی که ما در این پایان نامه فرصت تهیه و استفاده از این سخت افزارها را نداشتیم، محدود به استفاده از شبیه سازی در محیط کامپیوتری بودیم. استفاده از سخت افزارهای نورومورفیک احتمالا بتواند نتایج این تحقیق را به شکل چشمگیری تحت تاثیر قرار دهد.

علاوه بر این به عقیده‌ی من پیاده سازی‌های مکانیزهای آموزشی استفاده شده در این پایان نامه جای بهبود دارند. به عبارت دیگر روش گرادیان جایگزین محدودیت‌هایی در زمینه انتقال خطا در شبکه‌هایی که لایه‌های زیادی دارند به دنبال دارد. همچنین، این روش، در شرایطی که مجموعه داده به اندازه‌ی کافی بزرگ نباشد دچار مشکل می‌شود. یکی از راه‌هایی که به عقیده‌ی بنده می‌تواند راه گشای این مسئله باشد، استفاده توام از روش‌های آموزش بدون نظارت مانند STDP به عنوان مکانیزم‌های پیش آموزشی شبکه می‌باشد.

واژه‌نامه فارسی به انگلیسی

آموزش با نظارت ..... Supervised Learning
آموزش بدون نظارت ..... Unsupervised Learning
آموزش تقویت شده ..... Reinforcement Learning
آموزش شبه نظارتی ..... Semi-Supervised Learning
اتصالات جانبی ..... Lateral Connections
اختصاص اعتبار ..... Credit Assignment
ارابه‌ـ عمود ..... Cart-Pool
اسپایک ..... Spike
اسپایک سخت ..... Hard Spike
اسپایک نرم ..... Soft Spike
الگوریتم‌های ژنتیک ..... Genetics Algorithms
بازگشت خطا ..... Error Backpropagation, Backprop
بر مبنای پیکسل ..... Pixel-Based Image
بینایی ماشین ..... Computer Vision
تابع خطا ..... Loss Function
تابع خطای احتمالی لگاریتمی منفی ..... Negative Log-Likelihood Loss (NLLLoss)
تابع فعال ساز ..... Activation Function
تابع هزینه ..... Cost Function
تجمیع‌ـ آتش ..... Integrate-and-Fire (IF)
تجمیع‌ـ آتش با نشتی ..... Leaky Integrate-and-Fire (LIF)
تشخیص ژست حرکتی ..... Gesture Recognition
تصویر مبتنی بر فریم ..... Frame-Based Image
تعامل انسان و روبات ..... Human-Computer Interaction (HCI)
تفاوت گاوسی ..... Gaussian Differential

تقویت و اضافه کردن داده ... Data Augmentation and Reinforcement
ثابت زمانی قشاع ... Membrane Time Constant
جریان اسپایک ... Spike Train
جریان داده ... Data Stream
جهت اسپایک ... Spike Polarity
حافظه‌ی کوتاه‌ـبلند مدت ... Long-Short Term Memory
حذف‌ـکننده‌ی‌ـتصادفی ... Dropout
دنبال کردن توانی جریان ... Exponential Current Following
روش انعطاف پذیری وابسته به زمان ضربه ... Spike-Timing-Dependent Plasticity (STDP)
رگرسیون ... Regression
شبکه‌های بیزی ... Bayesian Network
شبکه‌ی رو به جلو ... Feed Forward Network
شبکه‌ی چند لایه ... Multi-Layer Perceptron
شبکه‌ی کامل ... Fully Connected Network, Densely Connected Network
شبکه‌ی کانولوشنال ... Convolutional Network
شدت نوری پیکسل ... Pixel Intensity
ضربه ... Spike
فیلتر زمانی ... Temporal Filter
قسمت شکمی قشر بینایی ... Ventral Pathway of the Visual Cortex
قشر تحلیلی ... inferotemporal cortex
لایه‌ی کشنده ... Pooling Layer
ماشین بردار پشتیبان ... Support Vector Machine
مقدار نشتی ... Leakage
میزان ارزندگی ... Fitness
نرمال‌ـساز‌ـدسته‌ای ... Batch-norm
پرسپترون ... Perceptron
پسا سیناپسی ... Post-Synaptic
پیش سیناپسی ... Pre-Synaptic
کانولوشنال ... Convolutional
کاهش گرادیان ... Gradient Decent
کد گذاری بر اساس زمان اسپایک ... Spike-time coding
کد گذاری بر اساس نرخ‌ـاسپایک ... Spike-rate coding

کد گذاری زمانی ..... Temporal Coding

کدگذاری اطلاعات در زمان اسپایک ..... Spike-Time-Coding

کشنده‌ـ بیشینه ..... Max-Pool

کشنده‌ـ میانگین ..... Average-Pool, Avg-Pool

کلاس بند ..... Classifier

کلاس بندی ..... Classification

گرادیان جایگزین ..... Surrogate Gradient

یادگیری عمیق ..... Deep Learning

واژه‌نامه انگلیسی به فارسی

تابع فعال ساز ..... Activation Function
کشنده‌ـ میانگین ..... Average-Pool
کشنده‌ـ میانگین ..... Avg-Pool
نرمال‌ـ ساز‌ـ دسته‌ای ..... Batch-norm
شبکه‌های بیزی ..... Bayesian Network
ارابه‌ـ عمود ..... Cart-Pool
کلاس بندی ..... Classification
کلاس بند ..... Classifier
بینایی ماشین ..... Computer Vision
کانولوشنال ..... Convolutional
شبکه‌ی کانولوشنال ..... Convolutional Network
تابع هزینه ..... Cost Function
اختصاص اعتبار ..... Credit Assignment
تقویت و اضافه کردن داده ..... Data Augmentation and Reinforcement
جریان داده ..... Data Stream
یادگیری عمیق ..... Deep Learning
شبکه‌ی کامل ..... Densely Connected Network
حذف‌ـ کننده‌ی‌ـ تصادفی ..... Dropout
بازگشت خطا ..... Error Backpropagation
دنبال کردن توانی جریان ..... Exponential Current Following
شبکه‌ی رو به جلو ..... Feed Forward Network
میزان ارزندگی ..... Fitness
تصویر مبتنی بر فریم ..... Frame-Based Image
شبکه‌ی کامل ..... Fully Connected Network

تفاوت گاوسی ..... Gaussian Differential
الگوریتم‌های ژنتیک ..... Genetics Algorithms
تشخیص ژست حرکتی ..... Gesture Recognition
کاهش گرادیان ..... Gradient Decent
اسپایک سخت ..... Hard Spike
تعامل انسان و روبات ..... Human-Computer Interaction (HCI)
تجمیع ـ آتش ..... Integrate-and-Fire (IF)
اتصالات جانبی ..... Lateral Connections
مقدار نشتی ..... Leakage
تجمیع ـ آتش با نشتی ..... Leaky Integrate-and-Fire (LIF)
حافظه‌ی کوتاه ـ بلند مدت ..... Long-Short Term Memory
تابع خطا ..... Loss Function
کشنده ـ بیشینه ..... Max-Pool
ثابت زمانی قشاع ..... Membrane Time Constant
شبکه‌ی چند لایه ..... Multi-Layer Perceptron
تابع خطای احتمالی لگاریتمی منفی ..... Negative Log-Likelihood Loss (NLLLoss)
پرسپترون ..... Perceptron
شدت نوری پیکسل ..... Pixel Intensity
بر مبنای پیکسل ..... Pixel-Based Image
لایه‌ی کشنده ..... Pooling Layer
پسا سیناپسی ..... Post-Synaptic
پیش سیناپسی ..... Pre-Synaptic
رگرسیون ..... Regression
آموزش تقویت شده ..... Reinforcement Learning
آموزش شبه نظارتی ..... Semi-Supervised Learning
اسپایک نرم ..... Soft Spike
ضربه یا اسپایک ..... Spike
جهت اسپایک ..... Spike Polarity
جریان اسپایک ..... Spike Train
کدگذاری اطلاعات در زمان اسپایک ..... Spike-Time-Coding
روش انعطاف پذیری وابسته به زمان ضربه ..... Spike-Timing-Dependent Plasticity (STDP)
کد گذاری بر اساس نرخ ـ اسپایک ..... Spike-rate coding

کد گذاری بر اساس زمان اسپایک ..... Spike-time coding
آموزش با نظارت ..... Supervised Learning
ماشین بردار پشتیبان ..... Support Vector Machine
گرادیان جایگزین ..... Surrogate Gradient
کد گذاری زمانی ..... Temporal Coding
فیلتر زمانی ..... Temporal Filter
آموزش بدون نظارت ..... Unsupervised Learning
قسمت شکمی قشر بینایی ..... Ventral Pathway of the Visual Cortex
قشر تحلیلی ..... inferotemporal cortex

مراجع

[۱] Nayak، T. Nolfo، Di C. McKinstry، J. Melano، T. Berg، D. Taba، B. A.، Amir، event- fully power، low A .(۲۰۱۷) al. et Mendoza، M. Garreau، G. Andreopoulos، A. Com- on Conference IEEE the of Proceedings In system. recognition gesture based ۱۷ ،۱۶ .۷۲۵۲–۷۲۴۳ pp. Recognition، Pattern and Vision puter

[۲] Long .(۲۰۱۸) Maass W. and Legenstein، R. Subramoney، A. Salaj، D. G.، Bellec، Advances neurons. spiking of networks in learning-to-learn and memory short-term ۲۱ ،۱۳ .۷۹۷–۷۸۷ ،۳۱ Systems Processing Information Neural in

[۳] predictive fractionally of networks in Error-backpropagation .(۲۰۱۱) M. S. Bohte، —۶۰ pp. Networks، Neural Artificial on Conference International In neurons. spiking ۱۲ Springer. .۶۸

[۴] tem- in Error-backpropagation .(۲۰۰۲) Poutre La H. and Kok، N. J. M.، S. Bohte، ۱۲ .۳۷–۱۷ ،(۴ـ۱)۴۸ Neurocomputing neurons. spiking of networks encoded porally

[۵] sequence in storage and recall Matching .(۲۰۱۳) Pfister J.-P. and Senn، W. J.، Brea، .۹۵۷۵–۹۵۶۵ ،(۲۳)۳۳ neuroscience of Journal networks. neural spiking with learning ۲۰ ،۱۱

[۶] Bi- .(۲۰۱۶) Bengio Y. and El-Yaniv، R. Soudry، D. Hubara، I. M.، Courbariaux، activations and weights with networks neural deep Training networks: neural narized ۲۱ ،۱۲ .arXiv:۱۶۰۲.۰۲۸۳۰ preprint arXiv or-۱. ۱ to+ constrained

[۷] Trends recognition. object invariant Untangling .(۲۰۰۷) Cox D. D. and J. J. DiCarlo، ۳ .۳۴۱–۳۳۳ ،(۸)۱۱ sciences cognitive in

[۸] visual solve brain the does How .(۲۰۱۲) Rust C. N. and Zoccolan، D. J.، J. DiCarlo، ۳ .۴۳۴–۴۱۵ ،(۳)۷۳ Neuron recognition؟ object

[۹] An- A. Appuswamy، R. Cassidy، S. A. Arthur، V. J. Merolla، A. P. K.، S. Esser، Con- .(۲۰۱۶) al. et Barch، R. D. Melano، T. McKinstry، L. J. Berg، J. D. dreopoulos، Proceedings computing. neuromorphic energy-efficient fast، for networks volutional ۱۲ .۱۱۴۴۶–۱۱۴۴۱ ،(۴۱)۱۱۳ sciences of academy national the of

[۱۰] en- spatiotemporally Learning .(۲۰۱۵) Grüning A. and Sporea، I. B.، Gardner، compu- Neural networks. neural spiking structured in transformations pattern coded ۱۱ .۲۵۸۶–۲۵۴۸ ،(۱۲)۲۷ tation

[۱۱] Paninski L. and Naud، R. Kistler، M. W. W.، Gerstner، .(۲۰۱۴) dynam- Neuronal University Cambridge cognition. of models and networks to neurons single From ics: ۶ Press.

[۱۲] Malach R. and K. Grill-Spector، .(۲۰۰۴) The human visual cortex. Annu. Rev. Neurosci، ۲۷، ۶۴۹–۶۷۷. ۳

[۱۳] Korondi P. and Botzheim، J. Domonkos، M. M.، N. Gyöngyössy، .(۲۰۱۹) Super- net- neural spiking by recognition gesture for set training small with learning vised pp. (SSCI)، Intelligence Computational on Series Symposium IEEE ۲۰۱۹ In works. ۱۹ IEEE. .۲۲۰۱–۲۲۰۶

[۱۴] Sejnowski J. T. and D. Huh، .(۲۰۱۷) Sejnowski networks. neural spiking for descent Gradient preprint arXiv:۱۷۰۶.۰۴۶۹۸ .arXiv ۲۰، ۲۱

[۱۵] Eliasmith C. and E. Hunsberger، .(۲۰۱۵) Eliasmith neurons. lif with networks deep Spiking preprint arXiv:۱۵۱۰.۰۸۸۲۹ .arXiv ۱۱، ۱۹

[۱۶] Masquelier T. and Thorpe، J. S. Ganjtabesh، M. R.، S. Kheradpisheh، .(۲۰۱۸) Stdp- Neural recognition. object for networks neural convolutional deep spiking based Networks ۹۹، ۵۶–۶۷. ۴، ۱۸

[۱۷] Masquelier T. and Ganjtabesh، M. Ghodrati، M. R.، S. Kheradpisheh، .(۲۰۱۶) recogni- object invariant in vision feed-forward human resemble can networks Deep reports Scientific tion. ۶، ۳۲۶۷۲. ۱۹

[۱۸] Hinton G. and Bengio، Y. Y.، LeCun، .(۲۰۱۵) Deep nature learning. ۵۲۱(۷۵۵۳)، ۴۳۶–۴۴۴. ۴

[۱۹] Pfeiffer M. and Delbruck، T. H.، J. Lee، .(۲۰۱۶) net- neural spiking deep Training neuroscience in Frontiers backpropagation. using works ۱۰، ۵۰۸. ۱۱، ۲۰

[۲۰] Akerman J. C. and Tweed، B. D. Cownden، D. P.، T. Lillicrap، .(۲۰۱۶) Random Nature learning. deep for backpropagation error support weights feedback synaptic communications ۷(۱)، ۱–۱۰. ۱۹

[۲۱] Benosman R. and Ieng، S.-H. J.-M.، Maro، .(۲۰۲۰) Benosman recog- gesture Event-based capa- computational smartphone using suppression background dynamic with nition Neuroscience in Frontiers bilities. ۱۴، ۲۷۵. ۱۷

[۲۲] Acharya T. and S. Mitra، .(۲۰۰۷) Acharya Transactions IEEE survey. A recognition: Gesture Reviews) and (Applications C Part Cybernetics، and Man، Systems، on ۳۷(۳)، ۳۱۱–۳۲۴. ۱۵

[۲۳] H. Mostafa، .(۲۰۱۷) Supervised neural spiking in coding temporal on based learning systems learning and networks neural on transactions IEEE networks. ۲۹(۷)، ۳۲۲۷–۳۲۳۵. ۲۱

[۲۴] Cauwenberghs G. and H. Mostafa، .(۲۰۱۸) .A framework learning for winner-take-all networks with stochastic synapses. Neural computation ۳۰(۶)، ۱۵۴۲–۱۵۷۲. ۱۱، ۲۰

[۲۵] Cauwenberghs G. and Ramesh، V. H.، Mostafa، .(۲۰۱۸) .Deep supervised learning using local errors. Frontiers in neuroscience ۱۲، ۶۰۸. ۲۰

[۲۶] Anderson، J. R.، Garaigordobil، M. M.، Mozaz، and Crucian، P. G. Rothi، G. J. L. Heilman M. K. .(۲۰۰۶) .Posture recognition in Brain and cogni- tion ۶۲(۳)، ۲۴۱–۲۴۵. ۱۵

[۲۷] Detorakis G. and Paul، S. Augustine، C. O.، E. Neftci، .(۲۰۱۷) .Event-driven ran- dom back-propagation: Enabling neuromorphic deep learning machines in Frontiers neuroscience ۱۱، ۳۲۴. ۱۱

[۲۸] Zenke F. and Mostafa، H. O.، E. Neftci، .(۲۰۱۹) .Surrogate gradient learning in spiking networks. IEEE Signal Processing Magazine ۳۶، ۶۱–۶۳. ۷، ۱۱، ۲۴، ۴۰

[۲۹] C. A. d. N. Christovão، L. C. T. Ferreira، B. A. L. Grecco، C. A. L. P.، H. Neto، Oliveira S. C. and Duarte، .(۲۰۱۵) .Clinical analysis and baropodometric evaluation in foot posture: A clinical trial of abnormal diagnosis of Journal of Bodywork and Move- ment Therapies ۱۹(۳)، ۴۲۹–۴۳۳. ۱۵

[۳۰] A. Nøkland، .(۲۰۱۶) .Direct feedback alignment provides learning in deep neural networks. preprint arXiv:۱۶۰۹.۰۱۵۹۶. arXiv networks. ۲۰

[۳۱] Yuan J. and Meng، J. Z.، Ren، .(۲۰۱۱) .Depth based hand gesture recog- nition and its applications in human-computer-interaction. ۲۰۱۱ ۸th International Conference on Information، Communications & Signal Processing. pp. ۱–۵. IEEE. ۱۵

[۳۲] Orchard G. and B. S. Shrestha، .(۲۰۱۸) .Slayer: Spike layer error reassignment in time. Advances in neural information processing systems ۳۱، ۱۴۱۲–۱۴۲۱. ۱۳، ۲۰

[۳۳] Ganguli S. and F. Zenke، .(۲۰۱۸) .Superspike: Supervised learning in multilayer spiking neural networks. Neural computation ۳۰(۶)، ۱۵۱۴–۱۵۴۱. ۱۲، ۱۹

Surname: Moqadam Mehr Name: Aref

Title: Gesture Recognition via Spike-Convolutional Neural Networks (S-CNN)

Supervisors: Dr. Hadi Farahani and Dr. Saeed Reza Kheradpisheh

Degree: Master of Science Subject: Department of Computer Science
Field: Algorithms and Computation Theory

Shahid Beheshti University Faculty of Mathematical Sciences
Date: 2020 Number of pages: 66

Keywords: Spiking Neural Networks, Convolutional Neural Networks, Human Gesture Recognition, Dynamic Vision Sensor, Computer Vision, Machine Learning

Abstract

Biological neurons use spikes to process and learn temporally dynamic inputs in energy and computationally efficient way. Unlike conventional Artificial Neural Networks, Spiking Neural Networks (SNN) tries to imitate this characteristic of the biological neurons to have advantages, such as energy efficiency. However, applying state-of-the-art gradient-based supervised algorithms to spiking neural networks is a challenge due to the non-differentiability of the activation function of spiking neurons. Employing surrogate gradients is one of the leading solutions to overcome this challenge. Although SNNs naturally work in the temporal domain, recent studies have focused on developing SNNs to solve static image categorization tasks. This paper employs a surrogate gradient descent learning algorithm to recognize twelve human hand gestures recorded by dynamic vision sensor (DVS) cameras.

Shahid Beheshti University

Faculty of Mathematical Sciences

Department of Computer Science

Master of Science Thesis

Gesture Recognition via Spike-Convolutional Neural Networks (S-CNN)

by

Aref Moqadam Mehr

Supervisors

Dr. Hadi Farahani and Dr. Saeed Reza Kheradpisheh

2020