بسم الله الرحمن الرحیم

دانشگاه شهید بهشتی

دانشکده علوم ریاضی

گروه علوم داده‌ها و کامپیوتر

پایان نامه کارشناسی ارشد

عنوان

یادگیری خودنظارتی در شبکه های عصبی اسپایکی(ضربه‌ای)

نگارش

یگانه بهاری اصل

استادان راهنما

دکتر هادی فراهانی و دکتر سعیدرضا خردپیشه

شهریور ماه ۱۴۰۲

كلية حقوق اعم از چاپ و تكثير، نسخه برداري ، ترجمه، اقتباس و ... از اين پايان نامه براي دانشگاه شهيد بهشتي محفوظ است. نقل مطالب با ذكر مأخذ آزاد است.

با ارادت و ادب، از تلاش‌های بی‌دریغ اساتید محترم، جناب آقای دکتر خردپیشه و جناب آقای دکتر فراهانی که در انجام این پایان‌نامه با بی‌دریغی راهنمایی‌های بسیاری ارائه داده‌اند، کمال قدردانی را دارم.

همچنین، از همسر عزیز، پدر گرامی و مادربزرگ مهربانم که با محبت و پشتیبانی‌های بی‌اندازه، دلسوزانه در کنارم بوده‌اند، تشکر عمیق را ابراز می‌کنم.

یگانه بهاری اصل

شهریور ماه ۱۴۰۲

تقدیم به مادرم...

نام خانوادگی دانشجو: بهاری اصل نام: یگانه

عنوان: یادگیری خودنظارتی در شبکه های عصبی اسپایکی(ضربه‌ای)

استادان راهنما: دکتر هادی فراهانی و دکتر سعیدرضا خردپیشه

مقطع تحصیلی: کارشناسی ارشد رشته: علوم داده‌ها و کامپیوتر گرایش: داده‌کاوی

دانشگاه: شهید بهشتی علوم ریاضی

تاریخ فارغ‌التحصیلی: شهریور ماه ۱۴۰۲ تعداد صفحات: ۷۶

واژگان کلیدی: شبکه های عصبی اسپایکی(ضربه‌ای) ، یادگیری خودنظارتی ، دسته بندی داده

چکیده

در این پایان‌نامه، به بررسی و معرفی شبکه‌های عصبی اسپایکی (ضربه‌ای) به‌عنوان نسل جدیدی از شبکه‌های عصبی می‌پردازیم. این شبکه‌ها، الهام گرفته از ساختار مغز، اطلاعات را از طریق قطارهای اسپایکی که به‌طور دقیق زمان‌بندی شده‌اند، کدگذاری و پردازش می‌کنند. برخلاف باقی شبکه‌های عصبی، در این نوع از شبکه نورون‌ها به جز مواقع تولید اسپایک در حال استراحت هستند و این خاصیت باعث بهینه بودن ساختار این شبکه‌ها و کم‌ترین استفاده از حافظه می‌شود. در این پژوهش، به بررسی روش‌های یادگیری خودنظارتی در شبکه‌های عصبی اسپایکی می‌پردازیم. این روش قابلیت دارد که با استفاده از تکنیک‌هایی مانند افزایش داده ، یادگیری را در شبکه بدون نیاز به داده برچسب دار انجام دهد. در نهایت، هدف از این پژوهش ساخت یک شبکه عصبی اسپایکی بهینه با قابلیت استفاده از روش‌های یادگیری خودنظارتی است تا بدون نیاز به برچسب داده، نمایش‌هایی از مجموعه داده‌های MNIST و CIFAR۱۰ تولید کرده و از این نمایش‌ها برای وظایف مختلفی مانند دسته‌بندی نظارت شده استفاده شود. این پژوهش تلاش می‌کند تا با ساخت این نمایش‌ها، در نهایت دقت بیشتری را در عملکرد شبکه ساخته شده بر روی روش‌های دسته‌بندی روی مجموعه داده‌ها ایجاد کند.

فهرست مطالب

مقدمه ۱

۱ شبکه‌های عصبی ضربه‌ای ۳

۱.۱ الهام از روش یادگیری مغز ۳

۲.۱ کد گذاری اسپایک‌های ورودی ۶

۱.۲.۱ کد گذاری بر اساس نرخ اسپایک‌های تولید شده ۶

۲.۲.۱ کد گذاری بر اساس زمان اسپایک‌های تولید شده ۷

۳.۱ مدلسازی نورون اسپایکی ۸

۱.۳.۱ مدل نورونی نشت کننده و ادغام آتش ۹

۴.۱ شبکه عصبی ضربه‌ای ۱۴

۱.۴.۱ یادگیری در شبکه های عصبی ضربه‌ای ۱۵

۲.۴.۱ روش گرادیان جایگزین در شبکه های عصبی ضربه‌ای ۱۷

۲ یادگیری خودنظارتی ۱۹

۱.۲ معرفی روش یادگیری خودنظارتی ۱۹

۱.۱.۲ نقاط قوت روش یادگیری خودنظارتی ۲۰

۲.۱.۲ نقاط ضعف روش یادگیری خودنظارتی ۲۱
۲.۲ الگوریتم های یادگیری خودنظارتی ۲۲
۱.۲.۲ مدل مبتنی بر انرژی ۲۲
۲.۲.۲ معماری تعبیه مشترک ۲۳
۳.۲.۲ الگوریتم‌های متضاد ۲۳
۴.۲.۲ روش های تبعیض نمونه ۲۴
۳.۲ روش های افزایش داده و تولید نمونه های جدید ۲۷
۴.۲ مفهوم انتقال یادگیری در یادگیری خود نظارتی ۲۹
۵.۲ توابع هزینه در یادگیری خودنظارتی ۳۱

۳ تعریف مساله ۳۵
۱.۳ پیاده سازی روش یادگیری خودنظارتی بر روی شبکه های عصبی اسپایکی ۳۵
۲.۳ پیشینه تحقیق ۳۷
۳.۳ چالش‌ها ۳۹

۴ روش پیشنهادی ۴۱
۱.۴ آماده سازی داده ورودی ۴۱
۱.۱.۴ مجموعه داده MNIST ۴۱
۲.۱.۴ مجموعه داده CIFAR۱۰ . . . . . ۴۲
۳.۱.۴ افزایش داده ۴۳
۲.۴ ساخت معماری شبکه ۴۵
۳.۴ انتخاب الگوریتم خودنظارتی مناسب ۴۸

۴.۴ انتخاب تابع هزینه ۴۹
۵.۴ انتخاب بهینه‌ساز ۵۰
۶.۴ الگوریتم نزدیک ترین همسایه(KNN) ۵۱
۷.۴ دسته‌بندی کننده ۵۲

۵ پیاده سازی و نتایج ۵۵
۱.۵ پیاده سازی مدل نورونی ۵۵
۲.۵ پیاده سازی یادگیری خودنظارتی روی شبکه عصبی اسپایکی ۵۶
۳.۵ پیاده سازی الگوریتم نزدیک ترین همسایه ۵۷
۴.۵ پیاده سازی دسته بندی کننده و نتایج ۵۸
۵.۵ چشم انداز پژوهشی ۶۰

واژه‌نامه فارسی به انگلیسی ۶۳

واژه‌نامه انگلیسی به فارسی ۶۵

نمایه ۶۸

مراجع ۶۸

کتاب‌نامه ۶۹

مقدمه

شبکه‌های عصبی اسپایکی به دلیل توانایی پردازش اطلاعات زمانی، مصرف کم انرژی و قابلیت بیولوژیکی بالا، در حال رشد هستند. این شبکه‌ها با الهام از مکانیزم نورون‌های بیولوژیکی، به جای مقادیر حقیقی، با استفاده از پالس‌های الکتریکی و اسپایک‌ها، اطلاعات را منتقل می‌کنند. [۳۰] یکی از اصلی‌ترین دلایل استفاده از شبکه‌های عصبی اسپایکی، شباهت این شبکه‌ها به نورون‌های بیولوژیکی انسان و حیوانات است. این شبکه‌ها به معماری و عملکرد نورون‌های واقعی پایبند بوده و توانایی در نمایش پردازش اطلاعات به صورت الکتریکی و با تولید اسپایک‌ها (پالس‌های الکتریکی کوتاه) دارند. این شباهت به نورون‌های بیولوژیکی به ما امکان می‌دهد تا رفتارها و ویژگی‌هایی که در مغز و سیستم عصبی واقعی مشاهده می‌شود را به دقت بیشتری شبیه‌سازی کنیم. [۲۶] یکی از مزایای مهم شبکه‌های عصبی اسپایکی این است که به دلیل فرآیند انتقال اطلاعات با اسپایک‌های الکتریکی، مصرف کمتری از انرژی نیاز دارند. این ویژگی بسیار مهم است، به ویژه در کاربردهایی مانند رباتیک که نیاز به سیستم‌های با مصرف انرژی پایین دارند . همچنین این شبکه‌ها از نظر ساختاری و عملکردی به بیولوژی نزدیک‌تر هستند و این قابلیت برای تحقیقات در زمینه علوم اعصاب بسیار ارزشمند است. [۲]

از طرفی یادگیری خودنظارتی یک حوزه پژوهشی در حال رشد است که در سال‌های اخیر با نتایج چشمگیر، به ویژه در زمینه پردازش تصویر، در حال پیشروی است. این روش یادگیری بر روی داده‌های بدون برچسب اعمال می‌شود و از طریق یک فرآیند افزایش داده بر روی داده‌های ورودی، سعی در ایجاد نمایشی از هر داده دارد که بیشترین تمایز از سایر داده‌ها را داشته باشد. به این صورت که در هر مرحله، شبکه آموزش می‌بیند که نمایش‌های یکسان از حالت‌های یک داده را ایجاد کند، در حالی که نمایش ایجاد شده برای هر داده، از دیگر داده‌ها متمایز باشد. [۷]

با توجه به ماهیت نوآورانه روش‌های یادگیری خودنظارتی و هم‌چنین شبکه‌های عصبی اسپایکی به عنوان نسل جدیدی از شبکه‌های عصبی، تاکنون پژوهش‌های زیادی در حوزه پیاده سازی این روش بر روی شبکه‌های عصبی اسپایکی صورت نگرفته و در هر یک از این دو حوزه به صورت مستقل مطالعاتی انجام شده است. در روش‌های یادگیری خودنظارتی با توجه به اینکه این روش نیاز به داده برچسب‌دار را تا حد خوبی کم می‌کند در حوزه‌های متنوعی به خصوص در داده‌های پزشکی که فرآیند برچسب زدن داده بسیار پرهزینه است، مطالعات جالب توجهی انجام شده است. [۱۳] [۲۸] از طرفی از شبکه‌های عصبی اسپایکی نیز با توجه به ماهیت بیولوژیکی آن‌ها و بهینه بودن در مصرف انرژی، در حوزه‌هایی مانند رباتیک و مهندسی نورومورفیک و هم‌چنین تشخیص الگو در داده‌های مغزی پژوهش‌هایی صورت گرفته است. [۱۲] [۲۷]

در این پروژه، قصد داریم با مدل‌سازی یک شبکه عصبی اسپایکی و استفاده از داده ورودی به عنوان یک جریان ثابت به شبکه، و شبیه‌سازی نورون‌ها با یک مدار الکتریکی، و پیاده‌سازی مدل نورونی یک‌پارچه و نشتی آتش[۱]، و با استفاده از روش گرادیان مجازی[۲] در یادگیری و پیاده‌سازی روش یادگیری خودنظارتی بر روی این شبکه یک ابزار قدرتمند و بهینه برای تولید نمایش‌های اسپایکی از داده‌های بدون برچسب بسازیم. به این صورت که اطلاعات اصلی هر نمونه از داده در این نمایش‌ها ذخیره شود و به گونه‌ای ساخته شود که هر نمایش بیشترین تمایز را از نمایش دیگر داده‌ها داشته باشد و در نهایت این نمایش‌های اسپایکی ساخته شده می‌توانند در انجام وظایف متنوعی مفید باشند.

این ابزار قادر است وظایفی مانند دسته بندی نظارت‌شده و تشخیص الگو در داده‌های تصویر بدون نیاز به برچسب انجام دهد و بسیار در کاهش هزینه و زمان مفید باشد. علاوه بر این با توجه به ماهیت شبکه‌های عصبی اسپایکی می‌توان از این ابزار ساخته شده برای وظایفی مانند ساخت ابزارهای نورومورفیک و پردازش داده‌های مغزی، بدون نیاز به هزینه و زمان برای ساخت داده‌ی برچسب‌دار استفاده کرد .

[1] Leaky integerate and fire

[2] Surrogate gradient

فصل ۱

شبکه‌های عصبی ضربه‌ای

۱.۱ الهام از روش یادگیری مغز

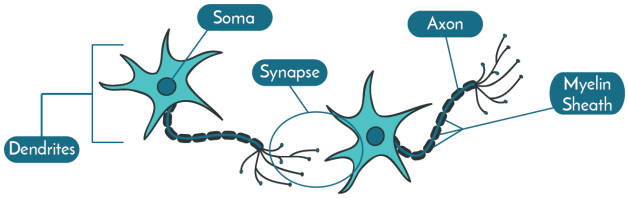
در مغز انسان، فرآیند یادگیری و انتقال اطلاعات به شکلی بسیار پیچیده و اهمیت‌بخش اتفاق می‌افتد. سلول‌های عصبی که به نام نورون‌ها[1] نیز شناخته می‌شوند، از ساختاری حساس و پیچیده تشکیل شده‌اند. هر نورون دارای یک هسته مرکزی و تعدادی ترمینال ورودی است. در شبکه‌های عصبی مغز، وظیفه اصلی این نورون‌ها دریافت و پردازش اطلاعات می‌باشد. این اطلاعات به صورت پالس‌های الکتریکی با نام اسپایک‌ها[2] منتقل می‌شوند. وقتی پتانسیل یک نورون از آستانه‌ای تعیین شده عبور می‌کند، یک اسپایک الکتریکی تولید می‌شود که حاوی اطلاعات است. این اسپایک سپس به عنوان ورودی به نورون‌های بعدی انتقال داده می‌شود، و این فرآیند ادامه می‌یابد. مهم‌ترین ویژگی این اسپایک‌ها این است که به عنوان واحدهای انتقال اطلاعات عمل می‌کنند و نورون‌ها توسط این اسپایک‌ها با یکدیگر ارتباط برقرار می‌کنند. [۲۵]

بیشتر نورون‌ها در شبکه مغز در حالت استراحت قرار دارند و تنها در صورتی که پتانسیل آن‌ها از حد آستانه‌ای تعیین شده عبور کند، اسپایک تولید می‌شود. این ویژگی باعث بهبود کارایی و مصرف انرژی در سیستم عصبی می‌شود. ساختار یک سلول عصبی شامل یک هسته مرکزی و تعدادی ترمینال ورودی

[1] Neuron

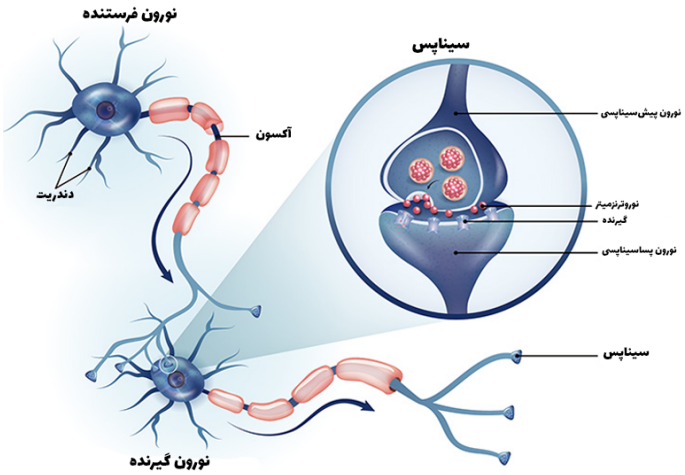
[2] Spikes

است که وظیفه دریافت و انتقال اطلاعات را از نورون‌های قبلی به نورون‌های بعدی دارد. ساختار ساده شده یک سلول عصبی در شکل ۱.۱ نمایش داده شده است .



شکل ۱.۱: یک سلول عصبی با نمایش اجزا اصلی

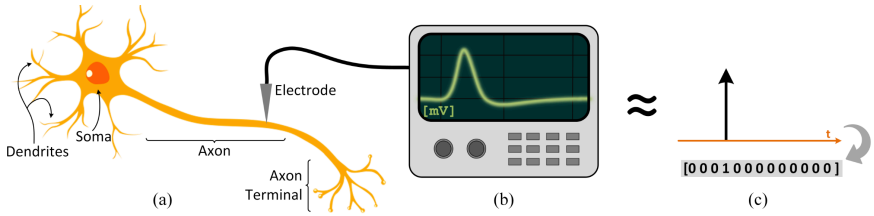
نورون‌ها در شبکه مغز به وسیلهٔ درگاه اتصالی به نام سیناپس به یکدیگر متصل هستند و انتقال اطلاعات بین نورون‌ها به وسیلهٔ سیناپس [۳] اتفاق می‌افتد. در صورتی که ولتاژ ورودی از سیناپس‌ها به حد مورد انتظار سلول برسد و پتانسیل سلول از آستانه عبور کند، سلول تحریک شده و یک اسپایک در ترمینال خروجی خود تولید می‌کند. به این سلول نورون پیش‌سیناپسی [۴] گفته می‌شود. این نورون اسپایک خروجی را در قالب یون‌های مثبت و منفی به صورت پالس‌های الکتریکی حاوی جریان به سیناپس‌های سلول‌های دیگر که به آن نورون پساسیناپسی [۵] گفته می‌شود، منتقل می‌کند و به عنوان یک جریان وارد نورون می‌شود. در شکل ۲.۱، یک ساختار ساده‌شده از سیناپس و اتصال دو نورون پیش‌سیناپسی و پساسیناپسی نشان داده شده است. [۱۶]



شکل ۲.۱: اتصال نورون فرستنده و گیرنده به وسیله سیناپس

[3] Synapse
[4] Pre-synaptic neuron
[5] Post-synaptic neuron

در این پایان‌نامه، قصد داریم یک شبکه عصبی اسپایکی الهام‌گرفته شده از ساختار شبکه مغز را پیاده‌سازی کنیم. همانطور که گفته شد، انتقال اطلاعات در مغز توسط اسپایک‌ها انجام می‌شود و نورون‌ها در بیشتر مواقع در حال استراحت هستند، به این معنا که خروجی اسپایکی ندارند. در راستای پیاده‌سازی شبکه الهام‌گرفته از مغز، در ابتدا به این دو مولفهٔ شبکه، یعنی اسپایک و پراکنده بودن[۶]، توجه می‌کنیم. در شکل ۳.۱ نشان داده شده است که اگر یک الکترود را به نورون متصل کنیم، ساختار اسپایک خروجی همانند یک پالس الکتریکی است که در ساختار دادهٔ پراکنده ذخیره می‌شود. [۳۱]



شکل ۳.۱: خروجی اسپایکی نورون و خاصیت پراکندگی

اسپایک‌ها: سلول‌های عصبی زیستی اطلاعات را از طریق اسپایک‌ها (شاره‌های الکتریکی)منتقل می‌کنند که بیشترین ولتاژ آنها حدود ۱۰۰ میلی‌ولت است. در بسیاری از مدل‌های محاسباتی نورون‌ها این گسیل شدن ولتاژ را به یک رویداد گسسته و یک‌بیتی ساده تبدیل می‌کنند: '۱' یا '۰'. این نسخه ساده‌تر در سخت‌افزار نمایش دادن آن نسبت به یک مقدار با دقت بالا، بهترین کارایی را دارد.

اسپایک‌ها: سلول‌های عصبی زیستی اطلاعات را از طریق اسپایک‌ها (شاره‌های الکتریکی) منتقل می‌کنند که بیشترین ولتاژ آن‌ها حدود ۱۰۰ میلی‌ولت است. در بسیاری از مدل‌های محاسباتی نورون‌ها، این گسیل شدن ولتاژ را به یک رویداد گسسته و یک‌بیتی ساده تبدیل می‌کنند: '۱' یا '۰'. این نسخه ساده‌تر در سخت‌افزار نمایش دادن آن نسبت به یک مقدار با دقت بالا، بهترین کارایی را دارد.

پراکندگی: نورون‌ها بیشتر زمان خود را در حال استراحت می‌گذرانند و اگر خروجی اسپایک را به عنوان عدد یک و زمان‌هایی که نورون خروجی ندارد را صفر در نظر بگیریم، بردارهای مربوط به خروجی نورون‌ها در طول زمان پراکنده‌گرا هستند؛ به این معنا که بیشتر عناصر این بردارها صفر است. این بردارها بسیار ساده و ارزان‌تر در ذخیره‌سازی هستند و برای بسیاری از محاسبات نیازی به خواندن بخش‌های صفر از حافظه نداریم که این خاصیت سرعت محاسبات را در این شبکه‌ها افزایش می‌دهد.

[۶] Sparsity

یکی دیگر از مولفه‌های مهم شبکه‌های عصبی اسپایکی و وجه تمایز آن‌ها با دیگر شبکه‌های عصبی وجود مولفه زمان است. به این معنا که در شبکه‌های اسپایکی، تولید اسپایک در طول زمان اتفاق می‌افتد. برخلاف دیگر شبکه‌های عصبی مصنوعی که اطلاعات را در یک لحظه پردازش کرده و در همان لحظه خروجی را محاسبه می‌کند، شبکه‌های عصبی اسپایکی الهام گرفته شده از مغز اسپایک‌های ورودی خود را که خروجی نورون قبلی بوده، در طول گام‌های زمانی دریافت کرده و هرزمان که پتانسیل آن به حد آستانه رسید اسپایک خروجی می‌دهد.

در ادامه به بررسی جزئی تر این شبکه‌ها و ساختار داده ورودی و مدل‌های نورونی و نوع یادگیری در شبکه‌های عصبی اسپایکی می‌پردازیم.

۲.۱ کد گذاری اسپایک‌های ورودی

همان‌طور که اشاره شد، داده‌های ورودی و خروجی نورون‌ها از نوع اسپایک هستند و نورون‌ها به صورت پالس‌هایی با طول و مقدار یکسان خروجی تولید می‌کنند. در نتیجه، پالس‌ها در خود تفاوتی ندارند و لازم است اطلاعات در زمان ارسال این پالس‌ها ذخیره و کدگذاری شود. دو روش اصلی و مهم برای کدگذاری اسپایک وجود دارد که به توضیح آن‌ها می‌پردازیم:

۱.۲.۱ کد گذاری بر اساس نرخ اسپایک‌های تولید شده

در این روش پردازش اطلاعات منتقل شده توسط نورون‌ها بر اساس تعداد یا فرکانس اسپایک‌های خروجی انجام می‌شود. به این معنا که هرچقدر تعداد اسپایک‌های خروجی از یک نورون بیشتر باشد، تأثیر آن نورون در شبکه بیشتر می‌شود و وزن بیشتری به آن اختصاص می‌یابد. این به این معناست که نورون فعالیت بیشتری دارد.

برای مثال، اگر فرض کنیم داده ورودی شبکه اسپایکی از نوع یک فریم عکس باشد، می‌توان برای کدگذاری هر فریم ورودی بر اساس نرخ اسپایک‌ها به صورت زیر عمل کرد:

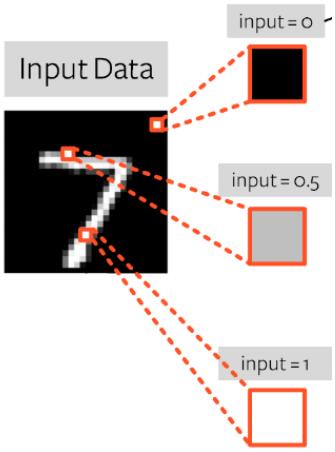
اگر هر پیکسل با نرمال‌سازی را به عنوان احتمال رخداد یک اسپایک در نظر بگیریم که در یک گام

زمانی به عنوان ورودی به شبکه داده شده است، می‌توان با استفاده از توزیع برنولی[۷] ورودی را بر اساس نرخ اسپایک‌ها کدگذاری کرد. معادله توزیع برنولی به صورت زیر است :

$$P(X=x)=\begin{cases} p & \text{if} x=1 \\ 1-p & \text{if} x=0 \end{cases} \tag{۱.۱}$$

بر اساس شدت احتمال در هر پیکسل از داده ورودی با استفاده از توزیع برنولی می‌توان یک عدد به آن نسبت داد که برابر با نرخ اسپایک خروجی آن پیکسل خواهد شد.

در شکل ۴.۱ یک نمونه از پیکسل‌های مربوط به یک نمونه داده ورودی نشان داده شده است. بر اساس میزان شدت هر پیکسل و احتمال نسبت داده شده به آن می‌توان کدگذاری براساس نرخ اسپایک انجام داد. شکل ۵.۱

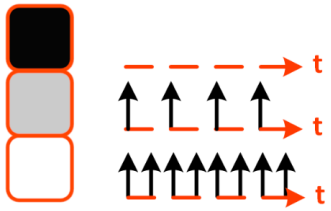


شکل ۴.۱: یک نمونه از داده ورودی شبکه اسپایکی و تبدیل هر پیکسل از داده برای کدگذاری بر اساس نرخ اسپایک

۲.۲.۱ کد گذاری بر اساس زمان اسپایک‌های تولید شده

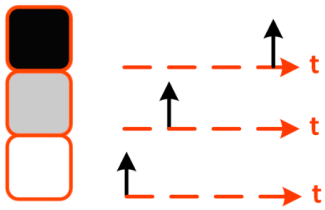
در روش کدگذاری بر اساس زمان اسپایک میزان فعالیت نورون و وزن اهمیت آن در شبکه به زمان اولین اسپایک بستگی دارد و هرچه زمان اسپایک زدن نورون کوتاه تر باشد به معنای فعالیت بیشتر آن

[7] Bernoulli distribution



شکل ۵.۱: سه نوع از پیکسل های داده ورودی به عنوان نمونه و تبدیل آن ها به کدگذاری برا ساس نرخ اسپایک

نورون است. در این مدل تنها اولین اسپایک خروجی از نورون اهمیت دارد و زمان آن به عنوان پارامتر اصلی فعالیت نورون در نظر گرفته میشود و باقی اسپایک ها اهمیتی ندارند. در شکل ۶.۱ یک نمونه از کدگذاری براساس زمان نشان داده شده است



شکل ۶.۱: سه نوع از پیکسل های داده ورودی به عنوان نمونه و کدگذاری آن ها به اسپایک بر اساس زمان

۳.۱ مدلسازی نورون اسپایکی

در این بخش به توضیح روش‌های مدلسازی نورون اسپایکی می‌پردازیم. در مدلسازی این نورون‌ها یک طیف گسترده از مدل‌های دقیق بیوفیزیک تا مدل‌هایی که نورون مصنوعی بسیار ساده را پیاده‌سازی کرده‌اند وجود دارد. یک مدل نورونی که ویژگی های بسیار دقیق بیوفیزیکی را پیاده‌سازی کرده‌است مدل نورونی هاجکین ـ هاکسلی[۸] است. [۱۰] اگرچه این مدل‌های بیوفیزیکی می‌توانند نتایج الکتروفیزیولوژیکی را با دقت بسیار بالا تولید کنند، اما پیچیدگی آن‌ها باعث می‌شود که در حال حاضر استفاده از آن‌ها دشوار باشد.

در طرف مقابل طیف نورون‌های مصنوعی وجود دارند که در آن‌ها ورودی‌ها با وزن‌های مربوطه

[8] Hodgkin-Huxley

ضرب می‌شوند و از طریق تابع فعال‌سازی عبور می‌کنند. این ساده‌سازی به محققان یادگیری عمیق اجازه داده‌است تا عملکردهای شگفت‌انگیزی در بینایی ماشین[۹]، پردازش زبان طبیعی [۱۰]و بسیاری از وظایف دامنه‌های یادگیری ماشین دیگر انجام دهند.

ما بین طیف مدلسازی‌های بیوفیزیکی و مدلسازی نورون‌های مصنوعی نوعی از مدل نورونی وجود دارد که تا حد ممکن به خواص بیوفیزیکی وفادار است و از طرفی پیچیدگی‌های محاسباتی آن در سطحی است که می‌توان به راحتی پیاده‌سازی کرد. نام این مدلسازی که در ادامه به توضیح آن می‌پردازیم مدل نورونی نشت کننده و ادغام آتش[۱۱] است.

۱.۳.۱ مدل نورونی نشت کننده و ادغام آتش

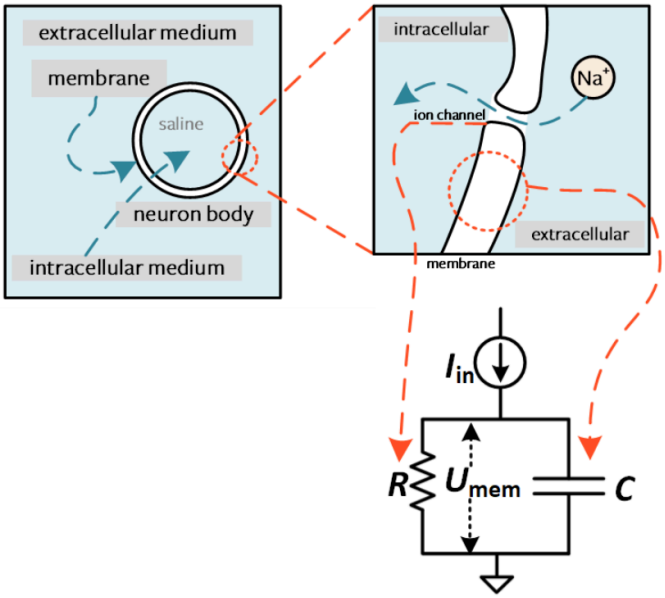
نوعی مدل نورون مشترک میان دو نوع مدل مصنوعی و هاجکین_هاکسلی، مدل نورون نشت‌کننده و ادغام آتش [۱۲]است. این مدل مانند نورون مصنوعی مجموع ورودی‌های وزن‌دار را می‌گیرد، اما به جای ارسال آن مستقیماً به تابع فعال‌سازی، ورودی را مانند یک مدار خازن مقاومت [۱۳]با یک نشتی زمانی ادغام می‌کند، اگر مقدار ادغام‌شده از مقدار پتانسیل آستانه نورون عبور کند نورون LIF یک اسپایک از جنس پالس الکتریکی ایجاد می‌کند.

نورون LIF در میان واقعیت زیستی و کاربردپذیری قرار می‌گیرد و هم سعی می‌کند خاصیت های بیوفیزیک نورون را شبیه سازی کند و هم قابلیت پیاده سازی داشته باشد.مدلسازی نورون‌های LIF با توجه به ساختار نورون‌های مغز اتفاق می‌افتد. نورون‌های مغز مانند تمام سلول‌ها توسط یک غشای نازک احاطه شده است. این غشاء یک لایه‌دهی لیپیدی است که داخل نورون را از محیط خارجی جدا می‌کند. از نظر الکتریکی، این غشا مانند ک خازن عمل می‌کند که دارای یک ظرفیت خاص می‌باشد. در شبیه سازی نورون‌های اسپایکی هدف محاسبه تغییرات پتانسیل غشا در طول زمان است که در این نوع شبیه سازی به عنوان یک خازن در نظر گرفته می‌شود. با محاسبه تغییرات پتانسیل و مشخص کردن آستانه و

[9] Computer vision
[10] Natural Language Processing
[11] Leaky integrate-and-fire
[12] LIF
[13] Resistor and Capacitor circuit

در نتیجه گسیل کردن اسپایک می‌توان عملکرد یک نورون اسپایکی را مدلسازی کرد. به این صورت که با محاسبه تغییرات پتانسیل در طول زمان اگر پتانسیل نورون از آستانه خود عبور کند (مانند خازنی که ظرفیت آن پر شده و تخلیه می‌شود.)نورون یک اسپایک گسیل می‌کند و پتانسیل آن به حالت استراحت باز می‌گردد و این فرایند دوباره تکرار می‌شود. [۲۲]

یکی دیگر از وظایف این غشاء، کنترل آنچه که به داخل و خارج از این سلول می‌رود است. این غشاء به طور معمول نسبت به یون‌ها نفوذناپذیر است و مانع ورود و خروج آن‌ها از بدنه نورون می‌شود. اما در غشاء کانال‌های مشخصی وجود دارند که با تزریق جریان به نورون، فعال می‌شوند. این جابجایی بار به طور الکتریکی توسط یک مقاومت مدلسازی می‌شود. با توجه به توضیحات داده شده یک نورن LIF توسط یک مدار دارای یک خازن که نقش غشا نورون را دارد و یک مقاومت برای مدلسازی کانال های غشا نورون پیاده سازی می‌شود. در شکل ۷.۱شبیه سازی غشا نورونی به مدار RC نشان داده شده است.



شکل ۷.۱: شبیه سازی غشا لیپیدی یک نورون به یک مدار خازن مقاومت الکتریکی.غشا دولایه مانند یک خازن رفتار می‌کند که پتانسیل آن نقش ظرفیت خازن را دارد. غشا نورون دارای کانال‌های یونی برای انتقال یون‌ها به نورون می‌باشد که رفتار آن مشابه مقاومت در مدار است.

برای مدلسازی ریاضی نورون LIF ابتدا از حل معادله دیفرانسیل مربوط به تغییرات پتانسیل غشا در طول زمان آغاز می‌کنیم:

$$\tau_m \frac{dU(t)}{dt} = -(U(t) - U_{rest}) + R \cdot I_{in}(t) \tag{۲.۱}$$

که در این معادله:

τ_m : ثابت زمانی برای پتانسیل غشا سلول

$U(t)$: پتانسیل غشا سلول

U_{rest} : پتانسیل غشا در حال استراحت

R : مقاومت

$I(t)$: جریان تابع زمان

پاسخ کلی معادله دیفرانسیل غشا به صورت زیر خواهد شد :

$$U(t) = U_{rest} + (U(0) - U_{rest}) \cdot e^{-\frac{t}{\tau_m}} + \frac{R}{\tau_m} \int_0^t e^{-\frac{t-s}{\tau_m}} \cdot I_{in}(s)\, ds \tag{۳.۱}$$

با حل تحلیلی معادله دیفرانسیل یک راه حل برای بدست آوردن مقدار پتانسیل غشا در طول زمان بدست آمد اما برای اینکه بتوانیم از این مفهوم در یادگیری شبکه‌های عصبی استفاده کنیم باید یک فرم گسسته و بازگشتی از معادله را بازسازی کنیم. برای این کار از روش اویلر برای حل معادلات دیفرانسیل استفاده می‌کنیم و این فرم را می‌توانیم مستقیم در روند یادگیری شبکه عصبی اسپایکی به کار بریم.

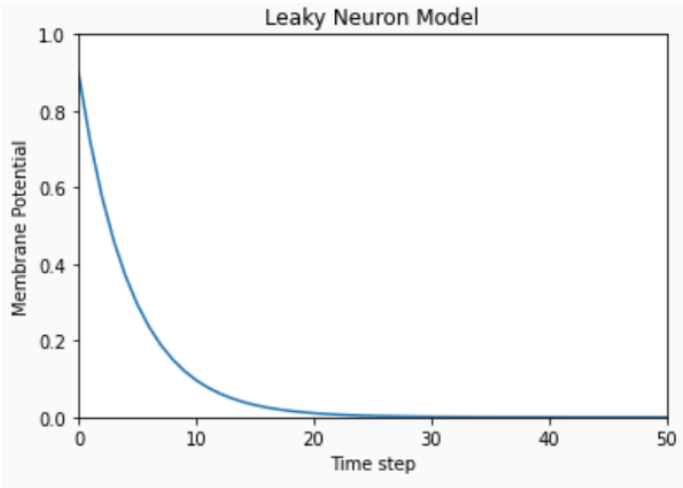
فرم گسسته شده معادله دیفرانسیل به صورت زیر نوشته می‌شود :

$$U[t+1] = U[t] + \frac{1}{\tau_m} \left(-(U[t] - U_{rest}) + R \cdot I_{in}[t]\right) \Delta t \tag{۴.۱}$$

اگر بازه زمانی یک مقدار گسسته بسیار کوچک در نظر گرفت معادله را به روش اویلر می‌توان به صورت زیر بازنویسی کرد :

$$u(t+\delta t) = u(t) + \delta t \left(-\frac{1}{\tau} \cdot (u(t) - u_{\text{rest}}) + I_{\text{in}}(t) \cdot R \right) \tag{۵.۱}$$

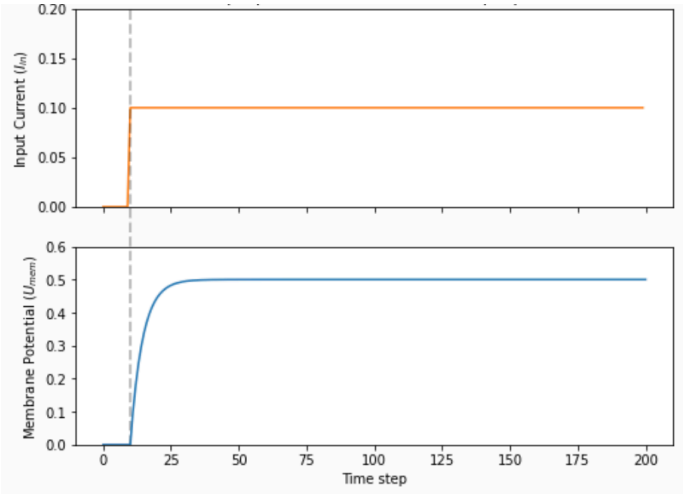
با حل این معادله در گام‌های زمانی و با مقدار اولیه U=۰.۹v و جریان ورودی صفر شکل ۸.۱ تغییرات پتانسیل را در طول زمان نشان می‌دهد. همانطور که در شکل مشخص است پتانسیل غشا از یک مقدار اولیه آغاز شده و در طول زمان به صورت نمایی افت کرده است.در این حالت فرض کردیم که جریانی به غشا وارد نشده است.



شکل ۸.۱: شبیه سازی معادله دیفرانسیل گسسته سازی شده مربوط به تغییرات پتانسیل غشا در طول زمان بدون جریان ورودی

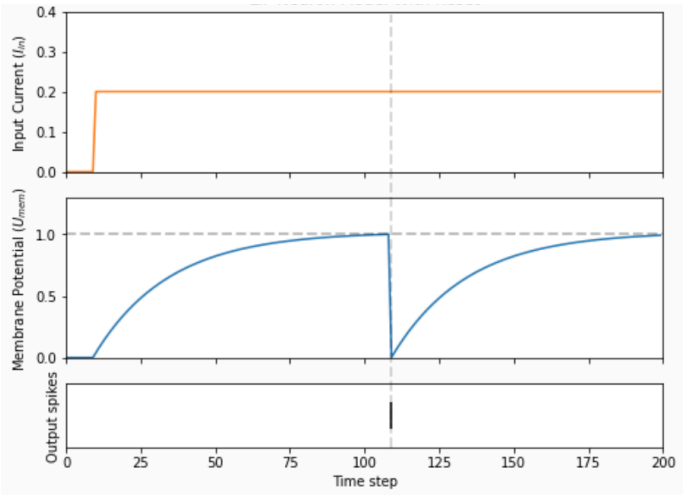
حال اگر بخواهیم مقدار جریان اولیه را به عنوان ورودی به نورون که در این شبیه سازی نقش خازن در مدار را دارد اضافه کنیم و اگر فرض کنیم پتانسیل از مقدارا اولیه برابر با صفر شروع خواهد شد و مقدار جریان ورودی برابر با ۱۰۰mA است تغییرات پتانسیل شبیه سازی‌شده در گام‌های زمانی در شکل ۹.۱ نشان داده شده است.

همانطور که گفته شد نورون‌ها اسپایک را به عنوان ورودی می‌گیرند و برای اینکه اسپایک گسیل کنند باید مقدار پتانسیل آن‌ها از یک آستانه عبور کند در این صورت نورون یک اسپایک گسیل کرده و پتانسیل به حالت استراحت بازمی‌گردد. شکل ۱۰.۱ یک شبیه سازی اولیه از پتانسیل یک نورون مدل شده به وسیله



شکل ۹.۱: شبیه سازی معادله دیفرانسیل گسسته سازی شده مربوط به تغییرات پتانسیل غشا در طول زمان با جریان ورودی ثابت

مکانیسم نشت کننده و ادغام آتش است :



شکل ۱۰.۱: تغییرات پتانسیل غشا در طول زمان با جریان ورودی ثابت با در نظر گرفتن مکانیسم نشت کننده و ادغام آتش

همانطور که مشاهده می‌شود در ابتدا یک جریان ثابت به نورون وارد شده پتانسیل آن شروع به افزایش کرده و هنگامی که به آستانه رسیده یک اسپایک گسیل کرده و سپس پتانسیل به حالت استراحت نشت کرده است.

در این مرحله می‌توانیم از مدل نورونی ساخته‌شده در یک شبکه عصبی استفاده کنیم.

۴.۱ شبکه عصبی ضربه‌ای

همانطور که گفته شد برای استفاده از مدل نورونی نشت کننده و ادغام آتش در شبکه‌های عصبی باید یک فرم بازگشتی گسسته از معادله دیفرانسیل ارائه دهیم :

اگر بتا را به عنوان نرخ کاهش پتانسیل غشا در نظر بگیریم :

$$\beta = 1 - \frac{1}{\tau} \tag{۶.۱}$$

$$U[t+1] = \beta \cdot U[t] + (1-\beta) I_{in}[t+1] \tag{۷.۱}$$

در یادگیری عمیق پارامترهای وزن معمولا در طول فرایند یادگیری آموزش داده می‌شوند. در این مرحله جریان ورودی را به دو بخش که یک پخش آن پارامتر ورودی شبکه است و بخش دیگر یک پارامتر قابل یادگیری است جداسازی می‌کنیم:

$$WX[t] = I_{in}[t] \tag{۸.۱}$$

در این صورت پارامتر ورودی مربوط به جریان را می‌توان همان اسپایک وارد شده به نورون در نظر گرفت. برای اسپایک ورودی به نورون باید یک تابع تعریف کنیم که تغییرات پتانسیل را نیز نشان دهد تابع را به صورت زیر تعریف می‌کنیم:

$$\mathrm{S}_{\mathrm{out}}[t] = \begin{cases} 1, & \text{if} U[t] \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{۹.۱}$$

در نتیجه تغییرات پتانسیل را به صورت معادله زیر بازنویسی می‌کنیم:

$$U[t] = \underset{\text{decay}}{\beta U[t-1]} + \underset{\text{input}}{WX[t]} - \underset{\text{reset}}{S_{out}[t-1]} \cdot \theta \tag{۱۰.۱}$$

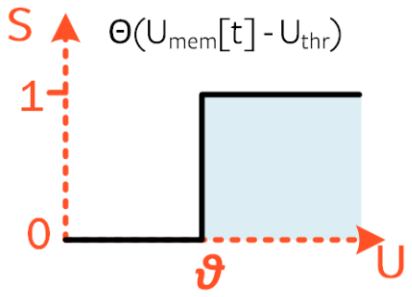
این معادله را می‌توان مستقیم در شبکه عصبی به کار برد.

همانطور که گفته شد داده ورودی در شبکه‌های اسپایکی باید دارای پارامتر زمان باشد در نتیجه اگر داده در زمان یک ساختار ثابت دارد آن را به صورت یک جریان ثابت به عنوان ورودی در چند گام زمانی به شبکه می‌دهیم و فعالیت نورون و تغییرات پتانسیل آن و همچنین اسپایک‌های خروجی از نورون را به عنوان خروجی مشاهده و بررسی می‌کنیم. در نتیجه خروجی شبکه اسپایکی همواره یک بعد زمان خواهد داشت.

۱.۴.۱ یادگیری در شبکه های عصبی ضربه‌ای

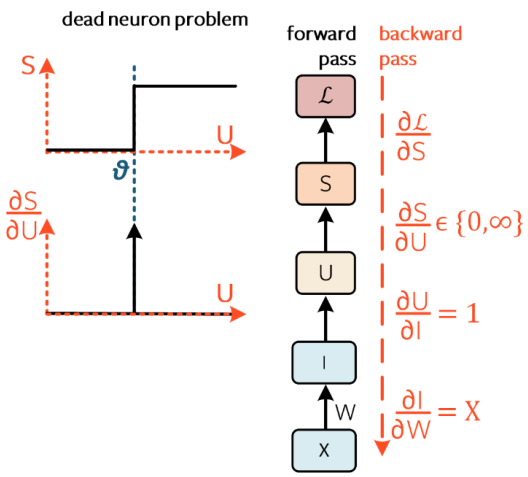
برای استفاده از مدل نورونی اسپایکی نشت کننده و ادغام آتش در شبکه‌های عصبی یک فرم بازگشتی گسسته از تغییرات پتانسیل نورون بازسازی کردیم. در این معادله یک تابع برای گسیل اسپایک در نظر گرفته شد. در شکل ۱۱.۱ نحوه عملکرد تابع تعریف شده برای گسیل اسپایک نمایش داده شده است .

همانطور که می‌دانیم یادگیری در شبکه‌های عصبی به وسیله کمینه کردن مشتق زنجیره‌ای توابع اتفاق می‌افتد. در این مساله تابع تعریف شده برای گسیل اسپایک مشتق پذیر نیست در شکل۱۲.۱ فرآیند مشتق گیری و مساله تابع گسیل اسپایک نشان داده شده‌است. همانطور که مشاهده می‌کنیم که تابع گسیل



شکل ۱۱.۱: تابع تعریف شده برای فرآیند گسیل اسپایک توسط نورون

اسپایک همانند تابع دلتای دیراک[۱۴] است که همواره برابر با صفر است به جز مواقع رد کردن آستانه پتانسیل نورون که به بینهایت میل می‌کند که به این معناست که گرادیان قابل محاسبه نیست و فرایند یادگیری به درستی اتفاق نمی‌افتد. به این مساله در مدلسازی نورون‌های اسپایکی مساله نورون مرده [۱۵] می‌گویند. برای حل این مساله و پیاده‌سازی یادگیری در شبکه‌های عصبی اسپایکی روش‌های مختلفی استفاده می‌شود . یکی از روش‌های مرسوم استفاده از یک تابع جایگزین است که در بخش بعد به توضیح آن می‌پردازیم.



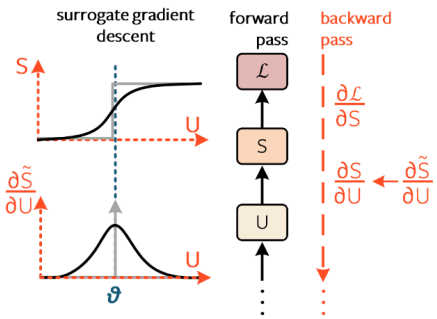
شکل ۱۲.۱: فرایند مشتق گیری زنجیره ای در یادگیری شبکه عصبی و مساله مشتق پذیر نبودن تابع تعریف شده برای گسیل اسپایک

[14] Delta Dirac function

[15] Dead neuron problem

۲.۴.۱ روش گرادیان جایگزین در شبکه های عصبی ضربه‌ای

همانطور که گفته‌شد تابع گسیل اسپایک یک تابع پله است که گرادیان آن به صورت یک تابع دلتای دیراک بدست می‌آید که فریاند مشتق پذیری در یادگیری برای کمینه کردن وزن‌ها را مختل می‌کند. یک راه حل برای این مساله استفاده از یک تابع مشتق پذیر با عملکرد مشابه تابع اصلی می‌باشد که این روش را گرادیان جایگزین می‌نامیم. [۱۶] یکی از توابع مورد استفاده برای جایگزین کردن تابع اصلی گسیل اسپایک تابع سیگموید [۱۷]است. در شکل ۱۳.۱ فرایند گرادیان گیری تابع سیگموید و عملکرد آن قابل مشاهده است.



شکل ۱۳.۱: تابع جایگزین برای حل مساله مشتق پذیر نبودن تابع گسیل اسپایک

به وسیله جایگزین کردن این تابع با تابع اصلی فرایند یادگیری و کمینه کردن وزن‌ها در شبکه به درستی اتفاق می‌افتد و می‌توان از شبکه عصبی اسپایکی ساخته شده در مسائل یادگیری ماشین استفاده کرد.

[16] Surrogate gradient discent

[17] Sigmoid function

فصل ۲

یادگیری خودنظارتی

۱.۲ معرفی روش یادگیری خودنظارتی

یادگیری خود نظارتی یک الگوی یادگیری ماشینی است که در آن مدل با استفاده از داده‌های بدون برچسب در طی فرآیند یادگیری یک نمایش [۱] از داده ها را می سازد که می‌توان از این نمایش در مسائل مختلف از جمله دسته بندی داده ها و پیش بینی برچسب داده ها استفاده کرد . ایده اصلی در یادگیری های خودنظارتی بی نیاز شدن از داده های برچسب دار و تولید برچسب داده توسط خود مدل و سپس استفاده از آن برای حل مسائل نظارت شده ای مانند طبقه‌بندی تصویر، شناسایی اشیاء، تقسیم‌بندی معنایی یا تقسیم‌بندی نمونه است. [۱۳] یادگیری‌خودنظارتی در مقایسه با یادگیری نظارت‌شده[۲] و یادگیری بدون نظارت [۳] دارای تفاوت است به این صورت که یادگیری نظارت‌شده شامل آموزش یک مدل با داده‌هایی است که دارای برچسب با کیفیت بالا هستند تا در طی فرآیند یادگیری شبکه وزن‌های خود را با برچسب داده مطابقت دهد. اما یادگیری خودنظارتی شامل آموزش یک مدل با داده‌ها و برچسب‌هایشان است، که در اینجا برچسب‌ها توسط خود مدل تولید می‌شوند و در ابتدای آموزش در دسترس نیستند. یادگیری بدون نظارت بر روی مجموعه‌داده‌هایی که برچسب در دسترس ندارند کار می‌کند و این الگوی یادگیری سعی

[1] representation

[2] Supervised learning

[3] Unsupervised learning

می‌کند مفهوم‌سازی داده‌ها را بدون استفاده از برچسب در هر مرحله از آموزش انجام دهد.[۳۳] بنابراین می‌توان نتیجه گرفت که یادگیری خود نظارتی زیرمجموعه از یادگیری بدون نظارت است زیرا هر دوی این روش‌ها تنها با داده‌های بدون برچسب ارائه شده همراه هستند. با این حال، یادگیری بدون نظارت به سمت خوشه‌بندی[۴] ، گروه‌بندی و کاهش بعد[۵] می‌رود، در حالی که یادگیری خودنظارتی وظایف قطعی مانند دسته‌بندی[۶] ، تقسیم‌بندی و رگرسیون[۷] را مانند هر مدل نظارت‌شده انجام می‌دهد.

اگرچه یادگیری تحت نظارت به طور گسترده در حوزه های کاربردی گسترده موفق است، مشکلات متعددی با آن مرتبط است.

یادگیری نظارت شده شدت به حجم زیادی از داده‌های برچسب گذاری شده با کیفیت بالا متکی است که به دست آوردن آنها بسیار پرهزینه و وقت گیر است. این یک محدودیت بزرگ در حوزه هایی مانند تصویربرداری پزشکی است، جایی که فقط متخصصان پزشکی متخصص می توانند داده ها را به صورت دستی حاشیه نویسی کنند. علاوه بر این، مدل‌های یادگیری نظارت‌شده زمانی بهینه عمل می‌کنند که هر دسته از داده‌ها تعداد نمونه‌های کم و بیش مساوی داشته باشند. عدم تعادل تعداد اعضای هر دسته بر عملکرد مدل تأثیر منفی می گذارد. و با این حال، به دست آوردن داده‌های کافی برای طبقات نادر دشوار است. یادگیری خودنظارتی همانطور که گفته شد نیاز به برچسب را در داده ورودی از بین می‌برد و خود الگوریتم برای داده‌ها برچسب تولید می‌کند. [۲۱]

۱.۱.۲ نقاط قوت روش یادگیری خودنظارتی

۱. مقیاس‌پذیری [۸]

یکی از مزایای قابل توجه یادگیری خودنظارتی، مقیاس‌پذیری آن است. الگوریتم‌های یادگیری نظارت شده برای آموزش به داده‌های برچسب‌دار با کیفیت بالا نیاز دارند. با این حال، جمع‌آوری و برچسب‌گذاری چنین داده‌هایی ممکن است نیازمند مصرف منابع و زمان زیادی باشد. بر خلاف این روش‌ها، یادگیری

[4] Clustering
[5] Dimention reduction
[6] Classification
[7] Regression
[8] Scalability

خودنظارتی نیازی به داده‌های برچسب‌دار ندارد. به جای آن، از ساختار ذاتی موجود در داده‌های بدون برچسب بهره می‌برد و این امکان را فراهم می‌کند که از مقادیر انبوهی از داده‌های بدون برچسب برای آموزش استفاده کند. این قابلیت مقیاس‌پذیری به مدل‌های یادگیری خودنظارتی اجازه می‌دهد تا احتمالاً نمایش‌های غنی‌تر و تعمیم‌پذیرتری را از مجموعه‌های داده گسترده یاد بگیرند و به کارایی آن‌ها در وظایف مختلف کمک کنند.

۲. درک چگونگی عملکرد ذهن انسان

یادگیری نظارت‌شده برای آموزش مدل‌ها به برچسب‌های تولید شده توسط انسان نیاز دارد. در اینجا، رایانه سعی می‌کند تا یاد بگیرد که انسان‌ها چگونه فکر می‌کنند از طریق نمونه‌هایی که قبلاً برچسب‌گذاری شده‌اند. اما، همانطور که گفته شد، برچسب‌گذاری چنین حجم زیادی از داده‌ها همیشه امکان‌پذیر نیست. یادگیری خودنظارتی با تولید خودکار برچسب‌ها بدون در حلقه هوش مصنوعی، توانایی ماشین را برای تفکر مستقل - مانند انسان - بررسی می‌کند. خود مدل باید تصمیم بگیرد که آیا برچسب‌های تولید شده قابل اعتماد هستند یا خیر، و بر این اساس از آن‌ها در تکرار بعدی برای تنظیم وزن خود استفاده کند.

۳. قابلیت‌های جدید هوش مصنوعی

یادگیری خودنظارتی برای اولین بار در زمینه پردازش زبان طبیعی استفاده شد. از آن زمان، برای حل انواع وظایف بینایی کامپیوتری مانند طبقه‌بندی تصویر، پیش‌بینی فریم ویدیو، و غیره گسترش یافته است. [۳۲]

۲.۱.۲ نقاط ضعف روش یادگیری خودنظارتی

۱. به قدرت محاسباتی زیادی نیاز دارد:

در روش یادگیری خودنظارتی، مدل باید داده‌های بدون برچسب ارائه شده را معنا کند، و همچنین برچسب‌های مربوطه را تولید کند. این مساله نیازمند منابع محاسباتی زیادی است که می‌تواند باعث افزایش هزینه‌های محاسباتی و زمانی در طول فرآیند آموزش مدل گردد. از طرف دیگر، پیچیدگی محاسباتی می‌تواند محدودیت‌هایی در اجرای این روش در مقیاس‌های بزرگتر ایجاد کند و این مسئله به ویژه در مواقعی که داده‌های بزرگ و پیچیده مورد استفاده قرار می‌گیرند، مشهودتر می‌شود. بنابراین، در طراحی

و پیاده‌سازی این روش‌ها، نیازمندی به محاسبات کارآمد و مدیریت منابع محاسباتی به عنوان یک نقطه حیاتی مطرح می‌شود.

۲. دقت پایین:

مدل‌های یادگیری خودنظارتی برچسب‌های خود را برای مجموعه داده تولید می‌کنند، و ما هیچ پشتیبانی خارجی نداریم که بتواند به مدل در تعیین درستی محاسباتش کمک کند. بنابراین، نمی‌توان انتظار داشت که مدل‌های یادگیری خودنظارتی به اندازه مدل‌های یادگیری نظارت شده دقیق باشند. [۳۲]

۲.۲ الگوریتم های یادگیری خودنظارتی

در یادگیری خود نظارتی، چندین الگوریتم برای یادگیری نمایش های معنادار از داده‌های بدون برچسب پیشنهاد شده است. این الگوریتم‌ها در رویکردهای خود برای ایجاد نمایش از داده‌های ورودی متفاوت هستند. در ادامه به توضیح چند الگوریتم رایج مورد استفاده در یادگیری خود نظارتی می‌پردازیم:

۱.۲.۲ مدل مبتنی بر انرژی

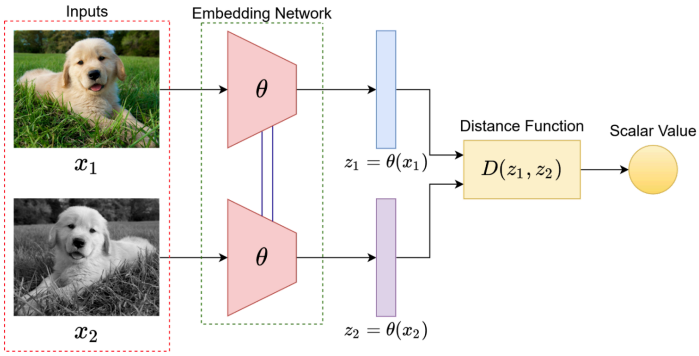
مدل‌های مبتنی بر انرژی [۹] سعی می‌کنند سازگاری بین دو ورودی داده شده را با استفاده از یک تابع ریاضی محاسبه کنند. هنگامی که دو ورودی داده می شود، اگر یک مدل مبتنی بر انرژی خروجی انرژی پایینی تولید کند، به این معنی است که ورودی ها سازگاری بالایی دارند. خروجی انرژی بالا نشان دهنده ناسازگاری بالا است.

برای مثال، دو نسخه تقویت‌شده از یک تصویر، مثلاً از یک سگ، وقتی به عنوان ورودی به مدل مبتنی بر انرژی داده می‌شود، انرژی خروجی پایینی تولید می‌کند، در حالی که تصویر یک سگ و تصویر یک گربه که به عنوان ورودی داده می‌شود، باید انرژی بالایی تولید کند. [۱۱]

[9] Energy Based Models(EBM)

۲.۲.۲ معماری تعبیه مشترک

معماری تعبیه مشترک [۱۰] یک شبکه دو شاخه است که شاخه ها در ساختار یکسان هستند. دو داده ورودی به هر یک از شاخه ها برای محاسبه نمایش آنها ارائه می شود. یک ماژول در سر شبکه وجود دارد که دو بردار تعبیه شده را به عنوان ورودی می گیرد و فاصله بین آنها را در فضای فاز[۱۱] محاسبه می کند. [۱] بنابراین، هنگامی که دو ورودی مشابه یکدیگر هستند ، فاصله محاسبه شده باید کوچک باشد. پارامترهای شبکه را می توان به راحتی تنظیم کرد تا اطمینان حاصل شود که ورودی ها در فضای فاز به یکدیگر نزدیک هستند. در شکل ۱.۲ دو نسخه افزوده شده از یک تصویر سگ نشان داده شده که به عنوان ورودی به شبکه داده‌می‌شود شبکه در طی فرایند یادگیری فاصله محاسبه شده این دو ورودی را کاهش می‌دهد.



شکل ۱.۲: معماری تعبیه مشترک

۳.۲.۲ الگوریتم‌های متضاد

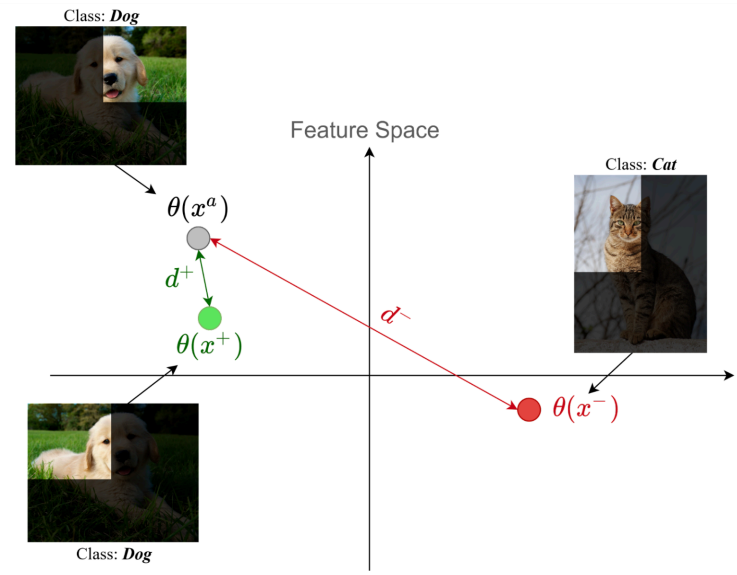
الگوریتم‌های متضاد [۱۲] با مقایسه و تضاد دیدگاه‌های مختلف از یک داده کار می‌کنند. این مدل آموزش داده شده است تا نقاط داده مشابه را در فضای بازنمایی آموخته شده به هم نزدیک کند در حالی که نقاط داده غیرمشابه را از هم جدا کند. هدف الگوریتم‌های متضاد، تضاد دیدگاه‌های مختلف از داده‌های مشابه برای یادگیری نمایش‌های معنادار است. آنها جفت‌های مثبت و منفی نمونه‌های داده را ایجاد می‌کنند و مدل را تشویق می‌کنند تا نمونه‌های مشابه را نزدیک‌تر کند در حالی که نمونه‌های غیرمشابه

[10] Joint embedding architecture

[11] Phase space

[12] Contrastive learning

را در فضای نمایش آموخته‌شده از هم جدا می‌کنند. [۱۸] در شکل ۲.۲ دو نمونه از داده مربوط به تصویر سگ به عنوان نمونه مشابه و یک تصویر گربه به عنوان نمونه غیر مشابه در فضای برداری نشان داده شده است که مدل یادگیری متضاد سعی می‌کند فاصله نمونه‌های مشابه را به حداقل و فاصله نمونه‌های غیر مشابه را به حداکثر در فضای برداری برساند. [۶]



شکل ۲.۲: الگوریتم یادگیری متضاد

۴.۲.۲ روش های تبعیض نمونه

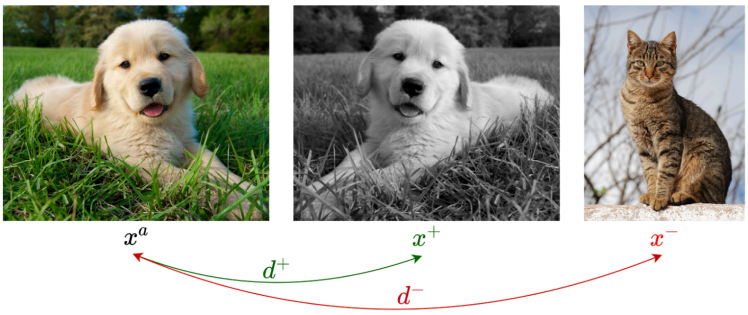
روش‌های تبعیض نمونه [۱۳] از ایده کلی یادگیری متضاد، برای کل نمونه‌های داده (مانند یک تصویر کامل) استفاده می کنند.

به عنوان مثال، دو نسخه چرخانده یا برگردان شده از یک تصویر سگ می‌توانند به عنوان جفت مثبت عمل کنند، در حالی که یک نسخه چرخانده/برگردان شده از یک تصویر گربه می‌تواند به عنوان نمونه منفی باشد. اکنون، مشابه اصل اساسی، فاصله بین جفت مثبت باید به حداقل برسد، در حالی که فاصله بین جفت منفی باید حداکثر شود.

ایده اصلی پشت این تکنیک این است که ورودی، که دستخوش برخی تبدیل‌های داده‌های اساسی شده است باید همچنان از همان دسته باشد، یعنی یک مدل یادگیری عمیق باید نسبت به تبدیل‌ها تغییر ناپذیر باشد. هنگامی که تصویر یک سگ به صورت عمودی چرخانده می‌شود و به مقیاس خاکستری

[13] Instance discrimination method

تبدیل می‌شود، همچنان کلاس "سگ" را نشان می دهد. [۲۹] در این دسته از روش‌ها، یک تصویر تصادفی گرفته می‌شود و تبدیل تصادفی برای ایجاد نمونه مثبت روی آن اعمال می شود (مانند چرخش، برش و غیره). اکنون، چندین تصویر دیگر از مجموعه داده به عنوان نمونه‌های منفی گرفته می شود و یک تابع هزینه طراحی شده است تا فاصله بین جفت‌های نمونه منفی را به حداکثر برساند. شکل ۳.۲



شکل ۳.۲: روش‌های تبعیض نمونه

یکی از روش‌های محبوب در دسته روش‌های تبعیض نمونه الگوریتم یادگیری متضاد ساده[۱۴]) است که در ادامه به توضیح این روش می‌پردازیم

SimCLR (یادگیری متضاد ساده)

یک روش یادگیری خود نظارت برای تصاویری است که از یادگیری متضاد برای یادگیری نمایش داده‌ها بدون تکیه بر داده‌های برچسب‌گذاری شده استفاده می‌کنند. این شامل آموزش یک شبکه عصبی برای پیش‌بینی این است که کدام دو تصویر، از یک جفت تصویر، شبیه‌تر هستند. نمایش‌هایی که شبکه یاد می‌گیرد می‌تواند به عنوان جاسازی ویژگی برای کارهای پایین دستی مانند دسته‌بندی تصویر استفاده شود. [۶] روش یادگیری متضاد ساده از اجزای کلیدی زیر تشکیل شده است:

۱.افزایش داده‌ها[۱۵] :

هر نمونه ورودی دو بار افزوده می‌شود تا دو نمای متفاوت از یک نقطه داده ایجاد شود.

۲.شبکه رمزگذار [۱۶] :

[14] Simplified contrastive learning(simclr)

[15] Data augmentation

[16] Encoder

یک شبکه عصبی (معمولاً یک شبکه عصبی پیچشی عمیق) برای استخراج ویژگی از نماهای تقویت‌شده داده استفاده می‌شود.

۳.سر نمایش دهنده [۱۷] :

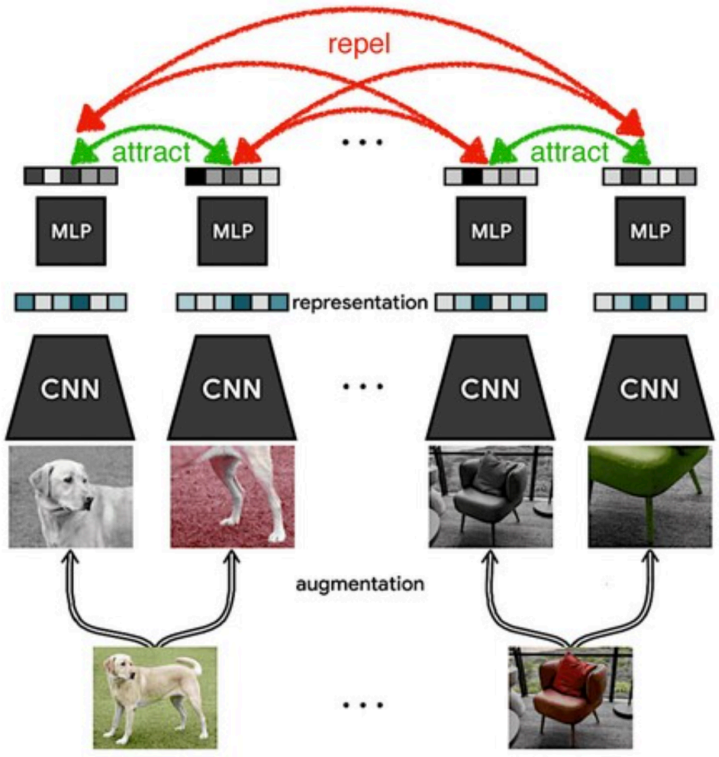
یک پرسپترون چندلایه [۱۸] با یک لایه پنهان است که فقط بردارهای نمایش را قبل از تابع هزینه کوچک‌تر می‌کند.

۴.تابع هزینه متضاد[۱۹] :

هدف مدل این است که نمایش‌های جفت‌های مثبت (نماهای تقویت‌شده همان نمونه) را نزدیک‌تر کند و نمایش‌های جفت‌های منفی (نماهای تقویت‌شده نمونه‌های مختلف) را در فضای ویژگی دورتر کند. تابع هزینه متضاد مدل را تشویق می‌کند تا بازنمایی‌های متمایزکننده‌ای را بیاموزد که ویژگی‌های اساسی داده‌ها را ضبط می‌کند.

در شکل ۴.۲ روند کلی الگوریتم یادگیری متضاد ساده نشان داده شده است به این صورت که دو داده افزوده شده به شبکه داده می‌شود و شبکه می آموزد نمایش داده های مربوط به یک کلاس را مشابه هم و نمایش داده‌های کلاس‌های مختلف را متفاوت تولید کند.

[17] Projection head
[18] Multilayer preceptron
[19] Contrastive loss function



شکل ۴.۲: روش یادگیری متضاد ساده

۳.۲ روش های افزایش داده و تولید نمونه های جدید

افزایش داده‌ها [۲۰] یک تکنیک مهم در حوزه یادگیری ماشین است که با استفاده از تغییرات مختلف بر روی داده‌های اصلی، حجم و تنوع مجموعه آموزشی را افزایش می‌دهد. هدف اصلی این تکنیک، کاهش اهمیت بار آموزشی و جلوگیری از بیش‌برازش مدل‌ها به داده‌های آموزشی است. با افزایش تنوع داده‌ها، مدل‌های یادگیری ماشین قادر به بهترین فراگیری الگوها و ویژگی‌ها می‌شوند و در مواجهه با داده‌های جدید و ناشناخته، عملکرد بهتری از خود نشان می‌دهند. [۲۳] افزایش داده‌ها از روش‌های متنوعی مانند تغییرات هندسی، تبدیل‌های رنگی، برش‌ها و چرخش‌ها، نویز اضافه کردن به داده‌ها و... استفاده می‌کند. با اعمال این تغییرات، داده‌ها به شکل مصنوعی تغییر می‌کنند و مجموعه آموزشی با داده‌های جدید و تا حدودی ناشناخته تکمیل می‌شود. این تغییرات همچنین باعث افزایش دقت و پایداری مدل‌ها در مواجهه با شرایط مختلفی می‌شود که ممکن است در داده‌های ورودی واقعی وجود داشته باشند.

[20] Data augmentation

در انجام افزایش داده‌ها، اهمیت استفاده از روش‌های مناسب و متعادل برای تغییر داده‌ها وجود دارد. از یک سو، باید مطمئن شویم که داده‌های تغییر یافته معتبر و معنادار هستند و با ویژگی‌های داده‌های اصلی همخوانی دارند. از سوی دیگر، تغییرات اعمال شده نباید اطلاعات اصلی و اساسی داده‌ها را به طور قابل ملاحظه‌ای تغییر دهد تا از از دست دادن اطلاعات مهم جلوگیری شود. به کارگیری تکنیک‌های افزایش داده‌ها باعث بهبود عملکرد و کارایی مدل‌های یادگیری ماشین در حل مسائل مختلف می‌شود و در عمل، جزء مهمی از روش‌های موفق در حوزه یادگیری ماشین به شمار می‌آید.

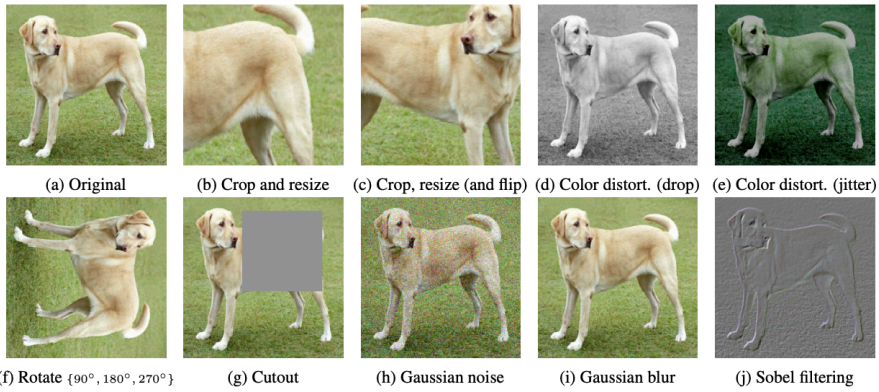
افزایش داده‌ها یکی از روش‌های موثر در بهبود عملکرد مدل‌های یادگیری ماشین است. کتابخانه PyTorch ابزارهای متنوعی برای انجام افزایش داده‌ها فراهم می‌کند که به ما اجازه می‌دهد داده‌ها را به صورت مصنوعی تغییر داده و تنوع آن‌ها را افزایش دهیم. در ادامه، به برخی از روش‌های افزایش داده‌ها در PyTorch پرداخته می‌شود:

۱. تغییرات هندسی: این روش‌ها شامل تغییر اندازه، برش، چرخش و انعکاس داده‌ها است. با اعمال این تغییرات، می‌توانیم داده‌ها را به صورت مختلف در فضای هندسی قرار دهیم و از این طریق تنوع مجموعه آموزشی را افزایش دهیم.

۲. تبدیل‌های رنگی: این روش‌ها شامل تغییر رنگ، کنتراست و روشنایی تصاویر است. با اعمال تبدیل‌های رنگی، می‌توانیم داده‌ها را به صورت مصنوعی تغییر رنگ دهیم و با ایجاد تصاویر مختلف، مجموعه آموزشی را متنوع کنیم.

۳. افزودن نویز: با افزودن نویز به داده‌ها، می‌توانیم شباهت آن‌ها را کاهش داده و مدل‌ها را در برابر داده‌های جدید و ناشناخته آماده کنیم. افزودن نویز به تصاویر و ویدئوها از جمله روش‌های متداول این دسته است.

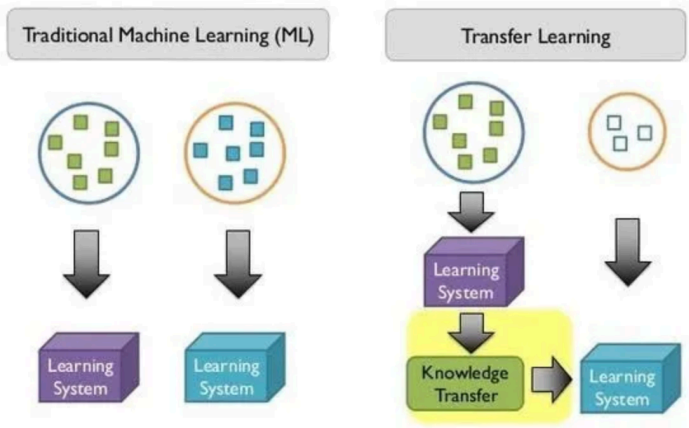
در شکل ۵.۲ نمونه‌هایی از اعمال روش‌های مختلف افزایش داده نشان داده شده است که همانطور که گفته شد برای پیاده سازی الگوریتم‌های یادگیری خودنظارتی به این تکنیک‌های افزایش داده نیازمند هستیم.



شکل ۵.۲: روش‌های افزایش داده

۴.۲ مفهوم انتقال یادگیری در یادگیری خود نظارتی

انتقال یادگیری [۲۱] یکی از روش‌های مؤثر در یادگیری خودنظارتی است که به کمک آن می‌توان اطلاعات آموزش‌دیده شده از یک مسئله را به مسئله‌ی دیگری منتقل کرد. در یادگیری خودنظارتی، معمولاً از اطلاعات بدون نظارت در داده‌ها برای آموزش مدل‌ها استفاده می‌شود. اما با انتقال یادگیری، می‌توان از اطلاعات آموزش‌دیده شده در یک مسئله برای حل مسئله‌ی دیگری بهره‌برداری کرد. در شکل ۶.۲ مفهوم انتقال یادگیری در یادگیری ماشین در مقایسه با یادگیری ساده نشان داده شده است.



شکل ۶.۲: انتقال یادگیری در مقایسه با یادگیری ساده

یکی از روش‌های معمول استفاده از انتقال یادگیری در یادگیری خودنظارتی، استفاده از مدل‌های پیش‌آموزش‌دیده [۲۲] است. در این روش، یک مدل با داده‌های بدون برچسب آموزش داده می‌شود. سپس

[21] Transfer learning

[22] Pre-trained models

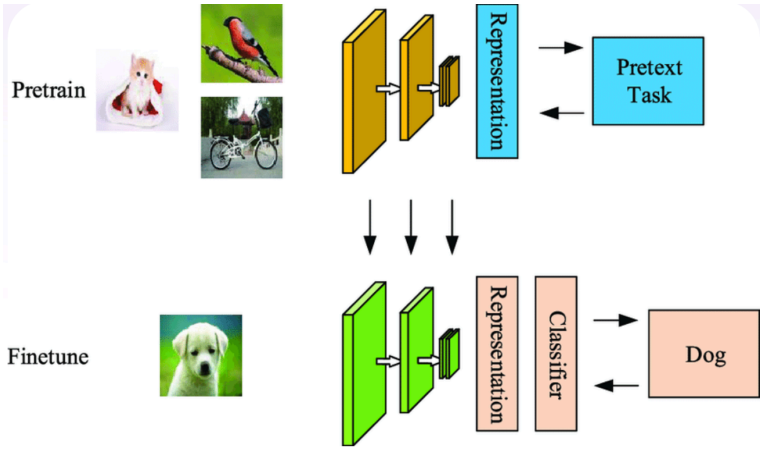
وزن‌های این مدل به عنوان وزن‌های اولیه در حل مسئله‌ی هدف استفاده می‌شود. این انتقال وزن‌ها امکان دسترسی به اطلاعات بدون نظارت از داده‌های پیشین و بهره‌گیری از آن‌ها برای بهبود عملکرد مدل را فراهم می‌کند. [۱۹] همچنین در انتقال یادگیری در یادگیری خودنظارتی، می‌توان از لایه‌های مشترک میان دو مسئله استفاده کرد. این لایه‌های مشترک می‌توانند اطلاعات کلیدی را که در هر دو مسئله به کار می‌روند، از داده‌های پیشین یاد بگیرند و از آن‌ها برای حل مسئله‌ی هدف استفاده کنند. این روش از نظر محدودیت داده‌ها و منابع محاسباتی مؤثر است و می‌تواند عملکرد مدل‌ها را بهبود بخشد.

تمایز اصلی انتقال یادگیری از یادگیری معمولی این است که در انتقال یادگیری، مدل از داده‌ها و دانش موجود در یک مسئله به مسئله‌ی دیگری منتقل می‌شود. این روش بسیار مؤثر در مواردی است که داده‌های کافی برای آموزش مدل در مسئله‌ی هدف موجود نیست یا هزینه‌ی جمع‌آوری داده‌های جدید بسیار بالاست.

انتقال یادگیری در حوزه‌های مختلفی از جمله بینایی ماشین، پردازش زبان طبیعی، تشخیص الگو و بازیابی اطلاعات استفاده می‌شود و به دلیل کارایی و عملکرد بالا، از محبوبیت بسیاری برخوردار است. استفاده از انتقال یادگیری در یادگیری خودنظارتی باعث افزایش سرعت آموزش مدل‌ها، بهبود کیفیت نتایج و کاهش نیاز به داده‌های برچسب‌دار می‌شود. در این روش شبکه ابتدا با داده‌های اولیه [۲۳] بدون برچسب آموزش می‌بیند سپس شبکه آموزش دیده شده برای حل مساله اصلی با داده‌های برچسب دار استفاده می شود. در شکل ۷.۲ ابتدا شبکه بر روی داده‌های بدون برچسب آموزش می‌بیند و یک نمایش از داده‌ها تولید می‌کند سپس شبکه آموزش دیده شده برای حل یک مساله نظارت شده استفاده می‌شود و در طی یادگیری بر روی داده‌های جدید برچسب دار به جای از ابتدا آموزش دیدن صرفا تنظیم دقیق [۲۴] می شود که بسیار سریع و بهینه است. [۲۰]

[23] Pretext

[24] Finetune



شکل ۷.۲: انتقال یادگیری در فرایند یادگیری خودنظارتی

۵.۲ توابع هزینه در یادگیری خودنظارتی

در یادگیری خودنظارتی متضاد، تابع خطای مقایسه‌ای برای آموزش مدل به کار می‌رود. هدف اصلی این تابع خطا، ایجاد فضای نمونه در مدل است که نمونه‌های مشابه در این فضا به هم نزدیک شوند، در حالی که نمونه‌های متفاوت از یکدیگر فاصله داشته باشند. با ایجاد این فضای نهان، مدل حساس به تفاوت‌های بین داده‌ها می‌شود و می‌تواند الگوهای مشترک بین داده‌ها را برای تمایز دادن از هم استخراج کند. [۶] برای تعریف تابع خطا به طور کلی از تابع هزینه رتبه بندی حاشیه [۲۵] استفاده می‌شود. در این روش، برای هر نمونه اصلی، دو نمونه دیگر انتخاب می‌شوند: یک نمونه مثبت (که مشابه نمونه‌ی اصلی است) و یک نمونه منفی (که با نمونه‌ی اصلی تفاوت دارد). سپس فاصله‌ی نمونه‌ی مثبت از نمونه‌ی اصلی (به عنوان فاصله مثبت) و فاصله‌ی نمونه‌ی منفی از نمونه‌ی اصلی (به عنوان فاصله منفی) محاسبه می‌شود. هدف این تابع خطا، مطمئن شدن از اینکه فاصله‌ی مثبت کمتر از فاصله‌ی منفی باشد و یک حداقل فاصله بین آن‌ها به عنوان لبه وجود داشته باشد. اگر فاصله‌ی مثبت از فاصله‌ی منفی بزرگتر از لبه باشد، تابع خطا مقدار آن را کاهش می‌دهد، در غیر این صورت، تابع خطا مقدار را افزایش می‌دهد.

در تعریف تابع خطای مقایسه‌ای، می‌توان از انواع مختلفی از فاصله‌ها استفاده کرد، از جمله فاصله اقلیدسی و یا شباهت کسینوسی. همچنین می‌توان مقدار فاصله لبه و انتخاب نمونه‌های مثبت و منفی را به طور دلخواه تنظیم کرد تا بهترین نتایج را بدست آورد.

[25] Margin ranking

با استفاده از تابع خطای مقایسه‌ای در یادگیری خودنظارتی متضاد، مدل قادر به استخراج ویژگی‌های معنادار از داده‌ها و یادگیری نمایشی مناسب برای مسائل مختلف خواهد بود. این تابع خطا می‌تواند برای مسائل تشخیص الگو، دسته‌بندی و مدل‌سازی مقایسه‌ای داده‌ها با موفقیت استفاده شود. [۱۴] از میان تابع‌های هزینه خودنظارتی، سه تابع رایج عبارتند از:

۱. تابع هزینه مقایسه‌ای [۲۶]: این تابع برای مسائل مدل‌سازی از نظر مشابهت داده‌ها به کار می‌رود. در این روش، دو نمونه از یک دسته به عنوان ورودی به مدل داده می‌شوند و مدل باید بتواند دو نمونه مشابه را از یکدیگر تشخیص دهد. اگر دو نمونه متفاوت باشند، مدل باید بتواند بین آن‌ها تفاوت را تشخیص دهد. یک فاصله برای لبه در نظر گرفته می شود و با استفاده از محاسبه فاصله اقلیدسی بین نمونه‌های مشابه و غیر مشابه تابع هزینه به صورت زیر تعریف می‌شود :

$$\mathcal{L}_{\text{contrastive}} = [\mathrm{d}_{\text{positive}} - \mathrm{m}_{\text{positive}}] + [\mathrm{m}_{\text{negative}} - \mathrm{d}_{\text{negative}}] \tag{۱.۲}$$

۲. تابع هزینه آنتروپی متقاطع با مقیاس دمایی نرمال شده [۲۷] این تابع برای مسائل تشخیص الگو و مدل‌سازی مقایسه‌ای داده‌ها مورد استفاده قرار می‌گیرد. در این روش، نمونه‌های مختلف از یک دسته با هم مقایسه می‌شوند و مدل باید بتواند نمونه‌های مشابه را از دیگر نمونه‌ها تمایز دهد. تابع ابتدا شباهت کسینوسی یک جفت داده مثبت را محاسبه کرده و سپس آن را تقسیم بر همین مقدار برای کل جفت‌های مثبت در آن دسته داده می‌کند. [۶]

$$\mathcal{L}_{\text{NTXent}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\text{sim}(z_i, z_{\text{pos}})/\tau)}{\sum_{j=1}^{2N}\exp(\text{sim}(z_i, z_j)/\tau)} \tag{۲.۲}$$

۳. تابع هزینه سه‌قلو [۲۸]: این تابع برای مسائل دسته‌بندی و مدل‌سازی مقایسه‌ای داده‌ها مورد استفاده

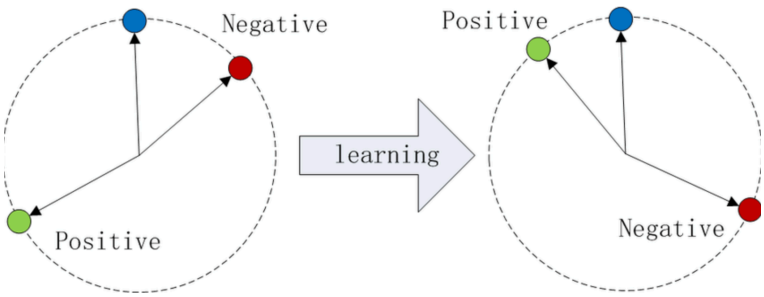
[26] Contrastive Loss

[27] Normalized Temperature-scaled Cross Entropy

[28] Triplet loss

قرار می‌گیرد. به طور کلی مشابه تابع هزینه مقایسه‌ای رفتار می‌کند با این تفاوت که در این روش، سه نمونه از یک دسته به عنوان ورودی به مدل داده می‌شوند: یک نمونه مثبت (که مشابه نمونه‌ی اصلی است)، یک نمونه منفی (که با نمونه‌ی اصلی تفاوت دارد) و نمونه‌ی اصلی. مدل باید بتواند نمونه‌های مشابه را از دیگر نمونه‌ها تمایز دهد و از نمونه منفی فاصله بگیرد. فاصله نمونه‌های مثبت باید برای تمامی نمونه‌ها حداقلی باشد تا مدل به طور کلی درک درستی از دسته‌ها داشته باشد. تابع هزینه سه‌قلو معمولاً از فاصله اقلیدسی یا شباهت کسینوسی رای محاسبه فاصله‌ها استفاده می‌کند. در شکل ۸.۲ نحوه محاسبه و عملکرد این تابع هزینه نشان داده‌شده‌است. [۱۷]

$$L = \max(d(a,p) - d(a,n) + \text{margin}, 0) \tag{۳.۲}$$



شکل ۸.۲: عملکرد تابع هزینه سه‌قلو

فصل ۳

تعریف مساله

۱.۳ پیاده سازی روش یادگیری خودنظارتی بر روی شبکه های عصبی اسپایکی

در این پروژه، قصد داریم از روش یادگیری متضاد ساده استفاده کنیم تا با ترکیب شبکه‌های عصبی اسپایکی با مدل نورونی نشت کننده و ادغام آتش، یادگیری خودنظارتی را بر روی داده‌ها انجام دهیم. از طریق این یادگیری خودنظارتی، قصد داریم نمایش‌های با کیفیتی از داده‌ها استخراج کنیم که می‌توانند در وظایف مختلفی مانند دسته‌بندی استفاده شوند.

با توجه به ماهیت بیولوژیکی شبکه‌های عصبی اسپایکی و پایبند بودن آن‌ها به روش عملکرد نورون‌های واقعی و توانایی در نمایش اطلاعات پردازش شده به صورت الکتریکی و با تولید اسپایک‌ها، استفاده ا این شبکه‌ها این امکان را می‌دهد تا رفتارها و ویژگی‌هایی که در مغز و سیستم عصبی واقعی مشاهده می‌شود را به دقت بیشتری شبیه‌سازی کنیم. از طرفی بهینه بودن شبکه‌های عصبی اسپایکی در مصرف انرژی و حافظه استفاده از این شبکه‌ها را در حوزه‌های متنوعی به خصوص حوزه‌های سخت‌افزاری نورومورفیک و رباتیک و غیره ممکن می‌سازد.

از سوی دیگر مزیت‌های چشمگیر روش یادگیری خودنظارتی و بی‌نیاز شدن از داده‌های برچسب‌دار برای آموزش شبکه در پیاده‌سازی این روش، با توجه به هزینه‌بر بودن فرایند تولید برچسب و هم‌چنین کمبود داده برچسب‌دار در بسیاری از حوزه‌ها، باعث شد به پیاده‌سازی روش یادگیری خودنظارتی بر روی

شبکه‌های عصبی اسپایکی بپردازیم.

به این نتیجه رسیدیم که الگوریتم یادگیری خودنظارتی را بر روی یک شبکه عصبی اسپایکی پیاده‌سازی کنیم و پس از آموزش دیدن شبکه با استفاده از روش انتقال یادگیری، شبکه آموزش دیده را بر روی داده‌های برچسب‌دار برای انجام وظایفی مانند دسته‌بندی نظارت شده آزمایش کنیم.

قبل از اعمال یادگیری خودنظارتی بر روی شبکه‌های عصبی اسپایکی، اقدام به توسعه و افزایش داده‌ها می‌کنیم. این عمل با استفاده از روش‌های افزایش داده مانند چرخش، انتقال، انعکاس و تغییر مقیاس انجام می‌شود. همانطور که در توضیحات روش یادگیری متضاد ساده گفته شده، برای پیاده‌سازی این روش باید از هر داده دو نمونه افزوده شده با یکی از تغییرات گفته شده برای ورودی شبکه تولید کنیم.

پس از انجام افزایش داده، اقدام به اعمال یادگیری خودنظارتی با روش یادگیری متضاد ساده بر روی شبکه می‌کنیم. این روش یادگیری به ما امکان می‌دهد که از معماری‌های پیچیده‌تری برای استخراج نمایندگان با کیفیت از داده‌ها استفاده کنیم. به عبارت دیگر، می‌توانیم اطلاعات کاربردی را از داده‌ها بدون نیاز به برچسب‌های دسته‌بندی شده، استخراج کنیم و از این اطلاعات برای تقویت کارایی شبکه‌های عصبی اسپایکی بهره‌برداری کنیم.

در طی فرآیند یادگیری، هر یک از نمونه‌های افزوده شده به عنوان جریان اسپایکی در گام‌های زمانی به شبکه داده می‌شود و تغییرات پتانسیل نورون‌ها و هم‌چنین آرایه اسپایک‌های خروجی هر نورون در طول زمان به عنوان خروجی شبکه در نظر گرفته می‌شود. با استفاده از توابع هزینه خودنظارتی، شبکه اسپایکی می‌آموزد که برای خروجی جفت‌های مشابه از داده، با الگوی یکسان و برای جفت‌های متفاوت ، با الگویی با بیشترین تفاوت اسپایک، خروجی تولید کند.

پس از آموزش شبکه برای ساخت نمایش اسپایکی از هر داده، شبکه آموزش دیده را بر روی داده‌های برچسب‌دار با تعداد کم(۲۰٪ از حجم کل داده) از دو مجموعه MNIST و CIFAR۱۰ منتقل می‌کنیم و میزان عملکرد شبکه را روی مساله دسته‌بندی نظارت شده بررسی می‌کنیم. مشاهده می‌شود نمایش‌های تولید شده توسط شبکه از روی داده‌های بدون برچسب، در حل مساله دسته‌بندی با داده‌های برچسب‌دار با تعداد کم، موثر است و دقت عملکرد شبکه در این مساله بیشتر می‌شود. در نتیجه با پیاده‌سازی روش یادگیری خودنظارتی بر روی شبکه‌های عصبی اسپایکی یک ابزار جدید و قدرتمند محاسباتی ساخته

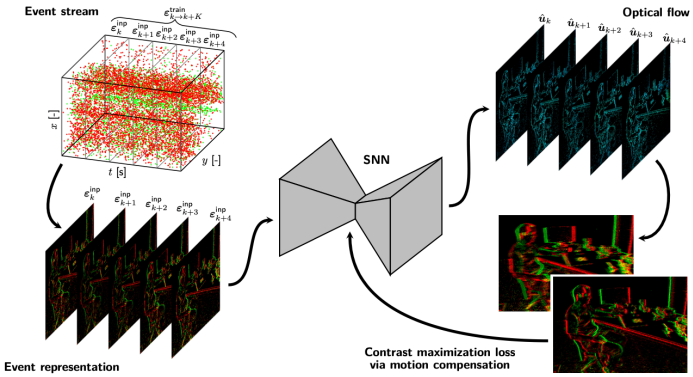
می‌شود که از یک سو هزینه برچسب زدن داده را کاهش می‌دهد و نیاز به داده برچسب‌دار برای محاسبه را کم می‌کند و از سوی دیگر با توجه به مزایای شبکه‌های اسپایکی، در حوزه‌های مختلف علوم اعصاب کاربرد بسیاری دارد.

۲.۳ پیشینه تحقیق

با توجه به نوآورانه بودن این حوزه، تاکنون تحقیقات قبلی در زمینه یادگیری خودنظارتی بر روی شبکه‌های عصبی اسپایکی کمتر متعارف بوده است و اکثر آن‌ها به طور مستقل در یکی از این دو حوزه انجام شده است. با این حال، چندین تلاش پژوهشی در این حوزه به عمل آمده است که می‌توانند به نتایج جالب و مفید منجر شوند.

در یک پژوهش در حوزه تکنولوژی نورومورفیک و شبکه‌های عصبی اسپایکی به بررسی امکانات و پیشرفت‌هایی می‌پردازد که در آن از شبکه‌های عصبی اسپایکی برای انجام وظایف پیچیده‌ای در زمینه بینایی ماشین استفاده می‌شود. این مقاله با بهره‌گیری از شبکه‌های عصبی اسپایکی در یک مسئله پیچیده مانند تخمین جریان نوری نشان می‌دهد که این نوع شبکه‌ها به کمک تصمیم‌گیری خودمتحرک توانسته‌اند عملکرد بهتری در مقایسه با شبکه‌های عصبی معمولی داشته باشند.

یکی از نکات مهم این تحقیق، تلاش برای کاهش نیاز به تطبیق زمانی در ورودی‌های این شبکه‌ها است. به این معنا که شبکه‌های عصبی اسپایکی توانسته‌اند ورودی‌هایی را طراحی کنند که بدون نیاز به اطلاعات زمانی دقیق، عملکرد خوبی در مواجهه با داده‌های پویا و بدون ساختار داشته باشند. این مسئله بسیار مهم است چرا که در بسیاری از موارد در بینایی ماشین، داده‌ها به صورت زمانی تغییر می‌کنند و این توانایی بدون نیاز به تطبیق زمانی می‌تواند عملکرد سیستم را بهبود بخشد. در شکل ۱.۳ ایده کلی پیاده‌سازی انجام شده قابل مشاهده است.

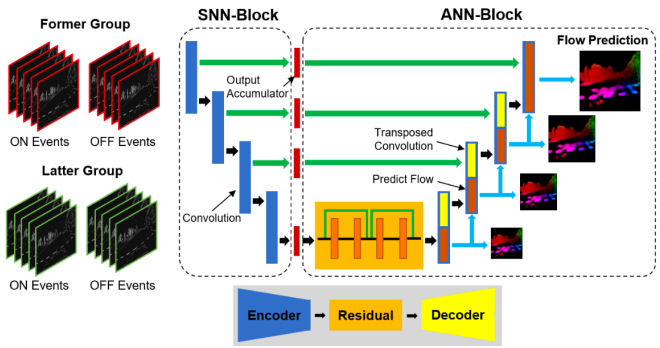


شکل ۱.۳: تخمین جریان نوری بر مبنای واقعه‌های خودآموز برای شبکه‌های عصبی اسپایکی عمیق. جریان واقعه به بخش‌های کوچک با تعداد یکسانی از واقعه‌ها تقسیم می‌شود، سپس به شکل مناسبی قالب‌بندی می‌شود و سپس به ترتیب به شبکه وارد می‌شود. برای هر بخش، نقشه جریان نوری پیش‌بینی می‌شود که هر واقعه ورودی را با یک بردار حرکت مرتبط می‌کند. هنگامی که تعداد کافی از واقعه‌ها پردازش شده باشد، با استفاده از تابع هزینه متضاد یک مرحله یادگیری انجام می‌شود..

دریک تحقیق دیگر، تخمین جریان نوری براساس داده‌های واقعه‌ای با شبکه‌های عصبی اسپایکی به بررسی استفاده از دوربین‌های مبتنی بر واقعه [1] برای وظایفی مانند تشخیص حرکت با سرعت بالا و ملاحظه در محیط‌های کم نور که دوربین‌های معمولی در آن‌ها با مشکلات مواجه می‌شوند، می‌پردازد. این توانایی به دلیل داشتن وضوح زمانی بالا، دامنه پویایی بالا و مصرف انرژی پایین این دوربین‌ها امکان‌پذیر می‌شود. با این حال، روش‌های بینایی ماشین معمولی و شبکه‌های عصبی مصنوعی عمیق مناسب برای کار با خروجی‌های ناهمگام و گسسته دوربین مبتنی بر واقعه نیستند. شبکه‌های عصبی اسپایکی به عنوان شبکه‌های مناسبی برای کار با خروجی‌های دوربین مبتنی بر واقعه عمل می‌کنند، اما این شبکه‌ها به تنهایی به دلیل پدیده ناپدید شدن اسپایک‌ها در لایه‌های عمیق عملکرد مطلوبی ندارند.

برای حل این مشکلات، در این مقاله یک مدل ارائه شده است، که یک معماری شبکه عصبی هیبریدی عمیق است که شبکه‌های عصبی معمولی را با شبکه‌های عصبی اسپایکی به صورت یکپارچه ترکیب می‌کند تا به بهترین شکل ممکن از خروجی‌های دوربین مبتنی بر واقعه برای تخمین جریان نوری بهره ببرد، در حالی که کارایی را حفظ می‌کند. این شبکه با یادگیری خودنظارتی ر روی یک مجموعه داده آموزش داده شده و در پیش‌بینی جریان نوری عملکرد بهتری نسبت به مدل‌های شبکه‌های عصبی معمولی دارد و همچنین کارایی محاسباتی قابل توجهی را فراهم می‌کند. در شکل ۲.۳ معماری ایده پرداخته شده در پژوهش نشان داده‌شده است.

[1] Event-based



شکل ۲.۳: معماری شبکه ساخته شده : تصاویر ورودی با چهار کانال، به ترتیب از طریق شبکه هیبریدی عبور می‌کنند. بلوک اسپایکی شامل لایه‌های ورودی و سپس تجمع خروجی است، در حالی که بلوک شبکه عصبی مصنوعی شامل لایه‌های باقی‌مانده است. پس از انتقال همه چارچوب‌های واقعه ورودی پشت‌سرهم در داخل پنجره زمانی، تابع خطا ارزیابی می‌شود و روند یادگیری طی می‌شود.

به طور کلی، تاکنون تحقیقات انجام شده در حوزه یادگیری خودنظارتی بر روی شبکه‌های عصبی اسپایکی اثبات می‌کند که این روش‌ها می‌توانند به عنوان یک راهکار موثر برای بهبود عملکرد شبکه‌ها در وظایف مختلف مورد استفاده قرار گیرند. همچنین، این تحقیقات نشان می‌دهد که استفاده از دانش به‌دست‌آمده از مسائل خودنظارتی می‌تواند در تسریع یادگیری و بهبود کیفیت نمایش‌ها بسیار مؤثر باشد و پیاده‌سازی این روش بر روی شبکه‌های عصبی اسپایکی با توجه به بهینه بودن و ساختار این شبکه‌ها می‌تواند در حوزه‌های مختلف بینایی ماشین نقش مهمی را ایفا کند.

۳.۳ چالش‌ها

چالش‌ها و مشکلاتی که در پیاده‌سازی یادگیری خودنظارتی بر روی شبکه‌های عصبی سنگین اسپایکی وجود دارند می‌توانند به شکل زیر باشند:

بالاخره، این چالش‌ها در پیاده‌سازی یادگیری خودنظارتی بر شبکه‌های عصبی سنگین اسپایکی می‌توانند به شرح زیر توضیح داده شوند:

۱. نرخ اسپایک‌ها:

در شبکه‌های عصبی اسپایکی، اطلاعات به صورت گسسته و در قالب اسپایک‌ها منتقل می‌شوند. این معمولاً به معنای این است که اطلاعاتی تنها در زمان وقوع یک اسپایک متنقل می‌شوند. بنابراین، تعیین

کردن کیفیت و اهمیت این اسپایک‌ها و همچنین انتخاب زمان مناسب برای آنها می‌تواند چالش‌هایی ایجاد کند.

۲. انتخاب تابع هزینه:

انتخاب یک تابع هزینه مناسب در یادگیری خودنظارتی مهم است. این تابع باید بتواند اطلاعات مهم و قابل استفاده را از داده‌ها استخراج کند و در طی فرآیند یادگیری نمایش‌های مناسبی از داده بسازد به این صورت که نمایش‌های مربوط به یک داده افزوده شده بیشترین شباهت و با نمایش دیگر داده‌ها بیشترین تفاوت را داشته باشد.

۳. پیچیدگی محاسباتی:

یادگیری خودنظارتی معمولاً نیازمند محاسبات پیچیده‌ای است و این محاسبات ممکن است نیاز به توانایی‌های محاسباتی قوی در سخت‌افزار داشته باشد. بنابراین، سخت‌افزارها باید مناسب برای انجام این محاسبات باشند و ممکن است فرآیند یادگیری زمان‌بر باشد.

۴. دقت پایین:

یادگیری خودنظارتی معمولاً روی داده‌های بدون برچسب انجام می‌شود و این می‌تواند به دقت پایین‌تری در مدل‌های یادگیری خودنظارتی منجر شود. این دقت پایین ممکن است چالش بزرگی را در مسائلی مانند تشخیص اشیا یا تصویربرداری از رخدادها ایجاد کند.

با توجه به نوآورانه بودن این حوزه، وجود چالش در محاسبات و پیاده‌سازی امری طبیعی است و به پشتکار و تلاش مداوم برای حاصل شدن پیشرفت نیاز دارد. در این تحقیق تلاش نموده‌ایم با پیاده‌سازی روش یادگیری خودنظارتی بر روی یک شبکه عمیق اسپایکی با چالش‌ها روبه‌رو شویم و یک نوآوری در این حوزه انجام داده و یک ابزار مفید برای انجام وظایف مختلف در حوزه یادگیری ماشین بسازیم.

فصل ۴

روش پیشنهادی

۱.۴ آماده سازی داده ورودی

مجموعه داده‌های MNIST[1] و CIFAR-۱۰ [2] دو مجموعه داده معروف و پرکاربرد در حوزه تشخیص الگو هستند که به عنوان محیط‌های آزمایشی برای بسیاری از مدل‌ها و الگوریتم‌های یادگیری عمیق استفاده می‌شوند.

۱.۱.۴ مجموعه داده MNIST

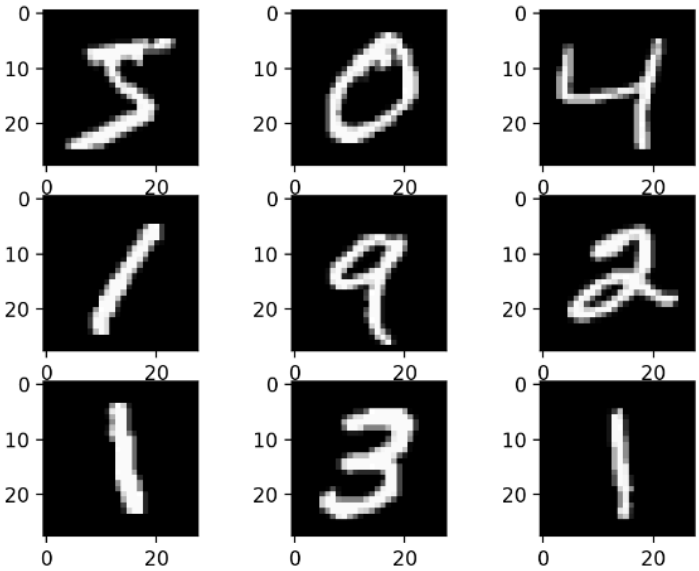
این مجموعه داده شامل تصاویر دست‌نویسی از ارقام از ۰ تا ۹ است که به صورت ۲۸در۲۸ پیکسل و به صورت خاکستری نمایش داده می‌شوند. مجموعه داده MNIST به عنوان یک مجموعه آموزشی کلاسیک در تشخیص ارقام مورد استفاده قرار می‌گیرد و بسیاری از مدل‌های یادگیری عمیق از جمله شبکه‌های عصبی پیچشی [3]و شبکه‌های عصبی بازگشتی [4] بر روی آن آموزش می‌بینند. در این پروژه از این مجموعه داده برای آزمون میزان یادگیری خودنظارتی در شبکه عصبی اسپایکی استفاده شده است. در شکل ۱.۴ یک نمونه از مجموعه داده MNIST نشان داده شده است.

[1] Modified National Institute of Standards and Technology database

[2] Canadian Institute For Advanced Research

[3] Convolutional neural network

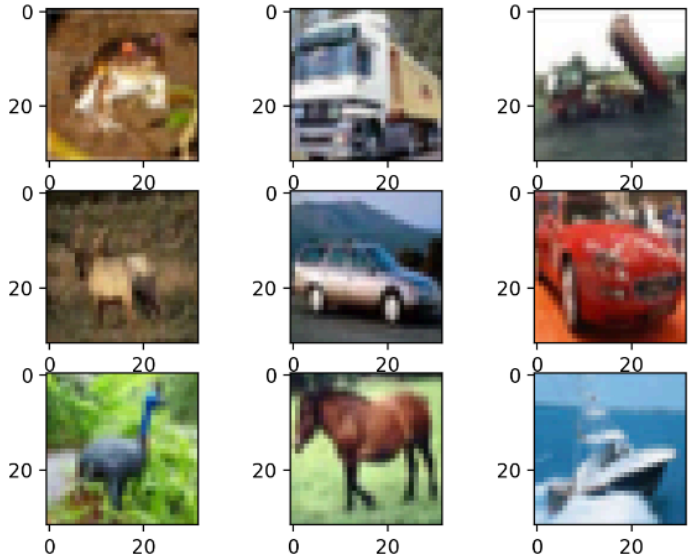
[4] Recurrent neural network



شکل ۱.۴: نمونه مجموعه داده MNIST

۲.۱.۴ مجموعه داده CIFAR۱۰

مجموعه داده CIFAR۱۰ شامل تصاویر رنگی از ده کلاس مختلف از اشیاء مختلف است. هر تصویر از این مجموعه داده ابعادی به اندازه ۳۲در۳۲ پیکسل و سه کانال رنگی (قرمز، سبز، آبی) دارد. این مجموعه داده به عنوان یک مجموعه آموزشی و ارزیابی محبوب در حوزه تشخیص الگو، تصویربرداری، و بینایی ماشین استفاده می‌شود. شبکه‌های عصبی پیچشی به خوبی برای طبقه‌بندی تصاویر در این مجموعه داده مورد استفاده قرار می‌گیرند . در این پروژه از این مجموعه داده به عنوان داده مورد تایید برای سنجش کارآیی مدل ساخته شده با شبکه عصبی اسپایکی استفاده می‌شود. در شکل ۲.۴ یک نمونه از مجموعه داده CIFAR۱۰ قابل مشاهده است.



شکل ۲.۴: نمونه مجموعه داده CIFAR۱۰

این دو مجموعه داده از مجموعه داده‌های کلاسیک برای آموزش و ارزیابی مدل‌های یادگیری عمیق به شمار می‌آیند و با توجه به سادگی و قدرت تشخیصی‌شان، همچنان در مطالعات بسیاری از پژوهش‌های مرتبط با هوش مصنوعی و یادگیری ماشین مورد استفاده قرار می‌گیرند.

۳.۱.۴ افزایش داده

برای استفاده از داده‌های مجموعه‌های MNIST و CIFAR۱۰ در یادگیری خودنظارتی، افزایش حجم داده‌ها از اهمیت بسیاری برخوردار است. با اعمال تکنیک‌های مختلف افزایش حجم داده به داده‌های آموزشی، می‌توان تنوع بیشتری ایجاد کرد و عملکرد مدل‌ها را بهبود بخشید.

۱. چرخش[۵] :

با اعمال چرخش‌های مختلف به تصاویر، امکان ایجاد تصاویری با زوایای مختلف به وجود می‌آید. این تغییرات می‌تواند به مدل‌ها کمک کند تا به الگوها و تفاوت‌هایی که در زوایای مختلف وجود دارد، حساس‌تر شوند.

۲. برش[۶]:

[5] Rotation

[6] Crop

با برش بخش‌های تصاویر اصلی، می‌توان تصاویر جدید با ابعاد مختلف و محتواهای مختلف ایجاد کرد. این تغییرات به افزایش تنوع داده‌ها کمک می‌کند و می‌تواند اثرات مثبتی در یادگیری مدل‌ها داشته باشد.

۳. مات [۷] :

اعمال اثر مات به تصاویر می‌تواند جزئیات غیرضروری را حذف کند و برای مدل‌ها کار کردن با داده‌های نویزی را آموزش دهد. این تغییرات می‌تواند مدل‌ها را به صورت مستقل از تفاوت‌های جزئی در داده‌ها آموزش دهد.

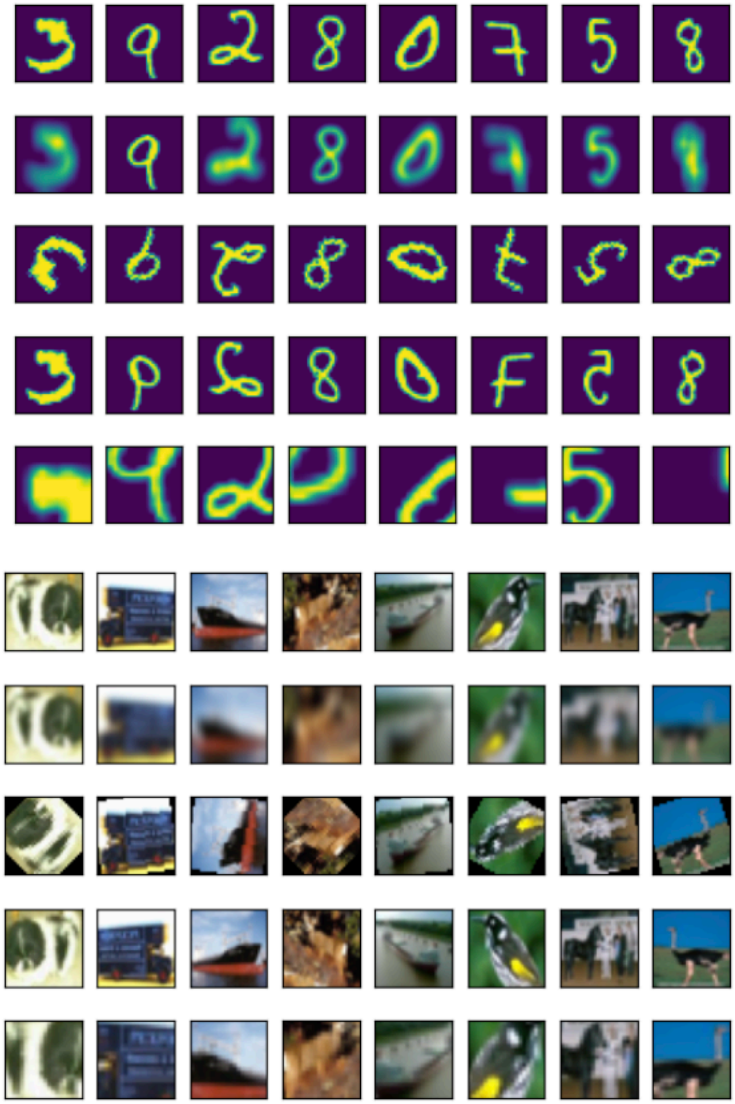
۴. وارونه [۸] :

انعکاس تصاویر از روش‌های ساده و موثر برای افزایش حجم داده‌هاست. با اعمال این تغییر، تصاویر جدید با جهت‌های مختلف ایجاد می‌شوند که به مدل‌ها کمک می‌کند تا به الگوها و تفاوت‌های جهتی حساس‌تر شوند.

با اعمال این تغییرات به داده‌های MNIST و CIFAR۱۰، می‌توان تعداد زیادی داده‌آموزشی جدید ایجاد کرد و مدل‌ها را با داده‌های متنوعی آموزش داد. این تنوع می‌تواند به افزایش دقت و کارایی مدل‌ها در مراحل بعدی یادگیری کمک کند و عملکرد آن‌ها را بهبود بخشد. با استفاده از داده‌های افزایش‌یافته، می‌توان به بهبود عملکرد یادگیری خودنظارتی بر روی شبکه‌های عصبی اسپایکی با اجرای الگوریتم‌های پیچیده‌تری در مراحل بعدی پرداخت. در این پروژه با استفاده از روش‌های مختلف افزودن داده روند یادگیری خودنظارتی شبکه عصبی اسپایکی را مورد بررسی قرار دادیم. در شکل۳.۴ پیاده سازی روش‌های مختلف افزایش داده بر روی داده‌های MNIST و CIFAR۱۰ قابل مشاهده است.

[7] Gaussian Blur

[8] Flip



شکل ۳.۴: روش های افزودن داده . به ترتیب روش های مات، چرخش ، وارون افقی و برش پیاده‌سازی شده‌است.

۲.۴ ساخت معماری شبکه

در ساخت شبکه عصبی اسپایکی برای مدلسازی نورون نشت کننده و ادغام آتش از پکیج snntorch [۹] استفاده شده و پیاده سازی اتصالات بین نورون ها به وسیله pytorch انجام شده است. هر نورون در مدل نورونی snntorch.leaky یک پارامتر به عنوان جریان و یک پارامتر به عنوان پتانسیل گام زمانی قبلی دریافت می‌کند و دو لیست با عنوان‌های اسپایک تولید شده توسط نورون(اگر پتانسیل از آستانه عبور کند برابر با یک و در غیر این صورت برابر با صفر خواهد بود.) و پتانسیل گام زمانی فعلی نورون را به

[9] https://snntorch.readthedocs.io/en/latest/index.html

عنوان خروجی می‌دهد. [۵]

با توجه به اینکه مفهوم زمان در شبکه‌های عصبی اسپایکی مورد توجه است و داده‌ها در طول زمان به صورت جریان ثابت وارد شبکه شده و فرآیند تغییر پتانسیل و تولید اسپایک نیز در طول زمان اتفاق می افتد، خروجی مدل نورونی یک بعد زمان نیز دارد و داده‌ها که به صورت فریم‌های عکس در قالب جریان به نورون‌ها وارد می‌شوند در طول چند گام زمانی چندین بار وارد شبکه شده و تغییرات پتانسیل و گسیل اسپایک از نورون در گام‌های زمانی ضبط می‌شود. ساختار خروجی نورون‌ها به صورت :

$$[\text{time step} \times \text{batch size} \times \text{feature dimension}]$$

خواهد بود. در فرایند آموزش شبکه تابع هزینه را برای خروجی هر یک از نورون‌ها در تمام گام‌های زمانی محاسبه کرده و در نهایت مجموع آن را به عنوان مقدار هزینه در نظر می‌گیریم و سعی در کمینه کردن آن داریم.

برای اتصالات بین نورون‌ها از لایه‌های پیچشی و همچنین از لایه‌های کاملا متصل استفاده کرده‌ایم. با توجه به اینکه معماری‌های شبکه‌های عصبی پیچشی برای یادگیری خودنظارتی مناسب هستند معماری شبکه عصبی اسپایکی را با الهام از شبکه‌های پیچشی پیاده‌سازی کرده و مدل نورونی را نشت کننده و ادغام آتش قرار دادیم.

در ادامه به توضیح بیشتر معماری مدل نورونی و اتصال لایه‌ها در شبکه می‌پردازیم :

لایه‌های پیچشی [۱۰]:

شبکه با یک سری لایه پیچشی شروع می‌شود. این لایه‌ها عملیات پیچش دو بعدی را روی داده‌های ورودی انجام می دهند. این لایه‌ها به گونه‌ای طراحی شده‌اند که با اعمال مجموعه‌ای از فیلترهای قابل یادگیری (هسته‌ها[۱۱]) ویژگی‌ها را در تصاویر ورودی ثبت کنند. این فیلترها روی تصویر می لغزند و نقشه‌های ویژگی را تولید می کنند که جنبه‌های مختلف ورودی را برجسته می کند. [۲۴]

لایه‌های عصبی نشت کننده و ادغام آتش (LIF):

[10] Convolutional layers

[11] Kernel

بعد از هر لایه پیچشی، یک لایه نورون نشت‌کننده و ادغام و آتش (LIF) مربوطه اعمال می شود. این لایه‌ها مدل نورون LIF را پیاده‌سازی می‌کنند. نورون‌های LIF رفتار نورون‌های واقعی را با ادغام اسپایک‌های ورودی در طول زمان شبیه سازی می کنند. اگر پتانسیل غشای یکپارچه از یک آستانه خاص فراتر رود، نورون یک اسپایک خروجی ساطع می کند. مدل LIF یک پارامتر نشتی را نیز معرفی می‌کند که کاهش تدریجی پتانسیل غشا را در طول زمان نشان می‌دهد.

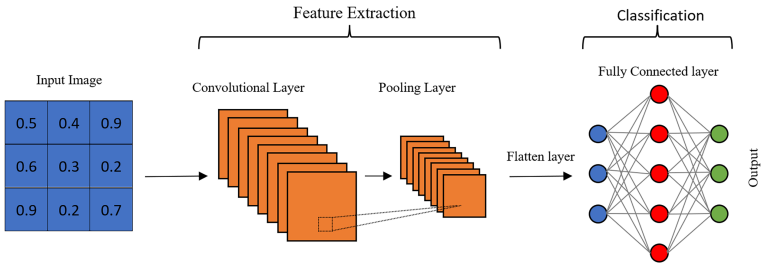
حداکثر ادغام [۱۲]:

در سراسر شبکه، لایه‌های حداکثر ادغام پس از لایه‌های پیچشی و LIF اعمال می‌شوند. این لایه‌ها نقشه‌های ویژگی به‌دست‌آمده از لایه‌های قبلی را پایین می‌آورد و ابعاد فضایی آن‌ها را کاهش می‌دهد. این عملیات به استخراج ویژگی‌های کلیدی از داده‌ها کمک می‌کند و در عین حال پیچیدگی محاسباتی را کاهش می‌دهد.

لایه‌های کاملا متصل [۱۳]:

در انتهای شبکه، از لایه‌های کاملاً متصل برای کاهش بعد نقشه‌های ویژگی به‌دست‌آمده از لایه‌های قبلی و تبدیل آن‌ها به یک بردار تک بعدی استفاده می‌شود. این لایه‌ها تبدیل‌های خطی روی داده‌ها انجام می‌دهند. لایه‌های نورون LIF مشابه لایه‌های قبلی بعد از لایه‌های کاملا متصل نیز اعمال می‌شوند.

در شکل ۴.۴ یک نمونه از نوع اتصالات لایه‌ها در شبکه نشان داده‌شده است. لازم به ذکر است لایه‌های نورونی اسپایکی در بین هریک از اتصالات شبکه قرار گرفته و خروجی اسپایکی نورون‌ها به وسیله اتصالات در شبکه فرآیند یادگیری را طی می‌کنند.



شکل ۴.۴: شکل کلی اتصالات لایه‌های پیچشی، حداکثر ادغام و کاملا متصل در معماری شبکه ساخته شده. لایه های پیچشی و حداکثر ادغام وظیفه اس تخراج ویژگی و لایه‌های ماکلا متصل وظیفه دسته‌بندی را به عهده دارند. نورون‌ها با مدل نورونی نشت‌کننده و ادغام آتش در بین تمام لایه‌های اتصال پیاده‌سازی شده‌اند.

[12] Max-pooling

[13] Fully connected

تابع رو به جلو [۱۴]:

در مسیر یادگیری رو به جلو، داده‌های ورودی از طریق لایه‌های پیچشی، حداکثر ادغام و کاملا متصل به صورت گام به گام پردازش می شوند. پتانسیل‌های غشایی نورون‌های LIF در طول مراحل زمانی به روز می‌شوند و اگر پتانسیل غشایی یک نورون از آستانه عبور کند، یک اسپایک خروجی ایجاد می‌شود. اسپایک‌های خروجی در چندین مرحله زمانی جمع‌آوری می‌شوند و به صورت دنباله‌ای از تانسورهای اسپایک بازگردانده می‌شوند.

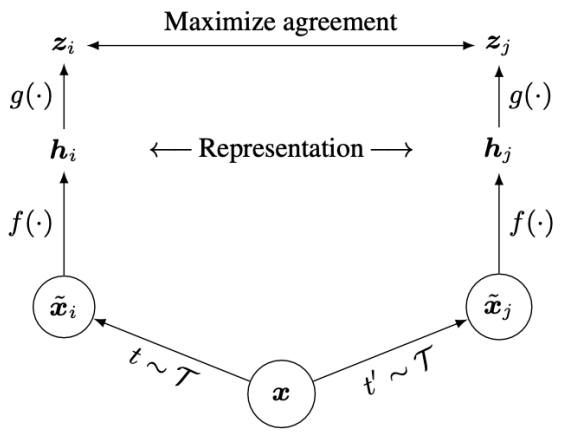
به طور خلاصه، این شبکه عصبی اسپایکی از لایه‌های پیچشی برای استخراج ویژگی‌ها، لایه‌های نورون LIF برای شبیه‌سازی رفتار عصبی، لایه‌های حداکثر ادغام برای کاهش بعد نمونه‌برداری و لایه‌های کاملاً متصل برای دسته‌بندی تشکیل شده است.

در نهایت، با استفاده از لایه‌های پیچشی و مدل‌های نورونی اسپایکی، مدل عصبی می‌تواند ویژگی‌های مهم از داده‌های MNIST و CIFAR-۱۰ استخراج کند و افزایش حجم داده با استفاده از تکنیک‌های گفته‌شده نیز می‌تواند بهبود قابل توجهی در یادگیری مدل ایجاد کند و دقت و عملکرد آن را افزایش دهد و همزمان نیز نیاز به داده برچسب‌دار را کاهش دهد.

۳.۴ انتخاب الگوریتم خودنظارتی مناسب

انتخاب روش SimCLR برای پیاده‌سازی بر روی شبکه عصبی اسپایکی نسبت به روش‌های خودنظارتی دیگر، دلایل متعددی دارد. از جمله مهم‌ترین دلایل این انتخاب، کارایی و اثربخشی روش SimCLR در یادگیری نمایش‌ها و استخراج ویژگی‌های معنادار از داده‌ها است. SimCLR از دو مرحله اصلی تشکیل شده است که شامل تولید نمونه‌های توسعه‌یافته و استفاده از تابع هدف مبتنی بر تفاوت نمونه‌ها می‌شود. این دو مرحله باعث می‌شوند که مدل با افزایش حجم داده‌ها و تلاش برای افزایش شباهت نمونه‌های مشابه و کاهش تفاوت نمونه‌های مختلف، ویژگی‌های با اهمیت و معنادار از داده‌ها استخراج کند. در شکل ۵.۴ روش کلی پیاده سازی این الگوریتم نشان داده شده است.

[14] Forward function



شکل ۵.۴: الگوریتم simclr با کمینه کردن فاصله میان نمونه‌های افزوده شده از یک داده در فضای نمونه، نمایش‌های مناسب از داده بدون برچسب‌را تولید می‌کند.

به علاوه، SimCLR به عنوان یکی از روش‌های معتبر یادگیری خودنظارتی، به‌طور گسترده‌ای مورد آزمایش و ارزیابی قرار گرفته است و نتایج بسیار خوبی در استخراج ویژگی‌ها از داده‌ها و بهبود کارایی مدل‌های عصبی حاصل کرده است. این روش می‌تواند بر روی مجموعه‌های داده‌های کوچکتر نیز به‌کار گرفته شود و نیاز به برچسب‌گذاری داده‌ها ندارد .

با توجه به مزیت‌ها و نتایج امتحان‌شده روش SimCLR در حوزه‌های دیگر یادگیری خودنظارتی، انتخاب این روش برای استفاده در شبکه عصبی اسپایکی می‌تواند به عنوان یک انتخاب مناسب و ارزشمند محسوب شود که می‌تواند به بهبود کارایی و عملکرد کلی مدل بسیار کمک کند.

۴.۴ انتخاب تابع هزینه

در پیاده سازی الگوریتم simclr از تابع هزینه آنتروپی متقاطع با مقیاس دمایی نرمال شده استفاده می شود:

$$\mathcal{L}_{\text{NTXent}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\text{sim}(z_i, z_{\text{pos}})/\tau)}{\sum_{j=1}^{2N}\exp(\text{sim}(z_i, z_j)/\tau)} \tag{۱.۴}$$

برای پیاده سازی این تابع هزینه با توجه به ساختار نمایش‌های ساخته شده توسط شبکه از کتابخانه

learning metric pytorch استفاده کرده ایم. به این صورت که تابع دو نمونه افزوده شده از داده اصلی را به عنوان ورودی می‌گیرد و مقدار شباهت کسینوسی آن را محاسبه کرده و تقسیم بر مجموع همین مقدار برای تمام جفت‌های مثبت می‌کند و در نهایت لگاریتم این مقدار را ذخیره می‌کند.

۵.۴ انتخاب بهینه‌ساز

بهینه‌ساز یک عنصر کلیدی در فرآیند یادگیری ماشین و شبکه‌های عصبی است. به‌طور کلی، بهینه‌سازها ابزارهایی هستند که در فرآیند تغییر پارامترهای مدل با هدف کمینه کردن تابع هزینه یا خطا به‌کار می‌روند. این پارامترها به‌طور معمول وزن‌ها و بایاس‌های موجود در شبکه‌های عصبی و دیگر پارامترهای مدل یادگیری ماشین را شامل می‌شوند.

هدف اصلی بهینه‌سازها این است که پارامترها را به گونه‌ای تغییر دهند که تابع هزینه کمینه شود. این کار معمولاً با محاسبه گرادیان تابع هزینه نسبت به پارامترها آغاز می‌شود. گرادیان نشان دهنده جهت و شیب تابع در نقاط مختلف است و اطلاعاتی درباره روش تغییرات پارامترها را ارائه می‌دهد.

نوع بهینه‌سازی مورد استفاده می‌تواند تأثیر زیادی بر سرعت و کیفیت یادگیری داشته باشد. بهینه‌سازها ممکن است معیارهای مختلفی را در نظر بگیرند تا در جلوگیری از بیش‌برازش [۱۵]مدل کمک کنند.

استفاده از بهینه‌ساز مناسب می‌تواند به تسریع فرآیند یادگیری، افزایش دقت مدل و جلوگیری از مشکلات مانند بیش‌برازش کمک کند. انتخاب بهینه‌ساز باید با توجه به مساله موردنظر، نوع داده‌ها و معماری مدل انجام شود.

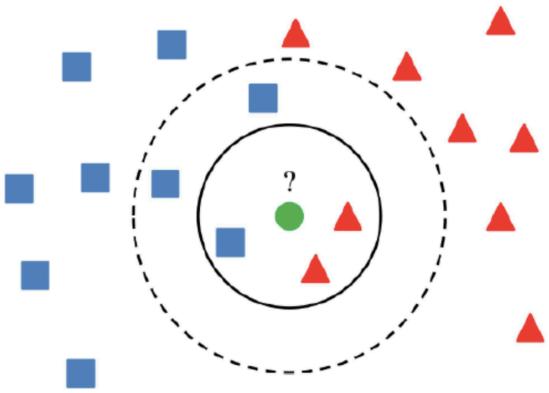
بهینه‌ساز Adam یک روش پیشرفته و موثر در فرآیند یادگیری ماشین و شبکه‌های عصبی است. یکی از ویژگی‌های منحصر به فرد ،Adam استفاده از نرخ یادگیری انطباقی است. این به معنای تنظیم خودکار نرخ یادگیری بر اساس ویژگی‌های داده‌ها و مساله است. این ویژگی باعث می‌شود Adam به طور خودکار با تغییرات متغیری در داده‌ها، به سرعت به نقاط بهینه همگرا شود. Adam به طور معمول به سرعت به نقاط بهینه همگرا می‌شود و در مواجهه با داده‌های نویزی نیز خوب عمل می‌کند. از این رو، این بهینه‌ساز به‌طور گسترده در زمینه‌های مختلف یادگیری ماشین و شبکه‌های عصبی مورد استفاده قرار می‌گیرد.

[15] Overfitting

در این پروژه برای پیاده سازی شبکه عصبی اسپایکی با روش یادگیری خودنظارتی از بهینه ساز Adam استفاده شده است. و با استفاده از کتابخانه pytorch پیاده سازی شده است. [۱۵]

۶.۴ الگوریتم نزدیک ترین همسایه(KNN)

با توجه به اینکه به وسیله روش یادگیری خودنظارتی شبکه می‌آموزد نمایش هایی از داده اصلی تولید کند برای بررسی اینکه این نمایش های تولید شده چه میزان قابل اعتماد هستند و شبکه تا چه حد آموخته که نمایش مناسبی از داده تولید کند، الگوریتم نزدیک ترین همسایه را برای نمایش‌های تولید شده توسط شبکه پیاده سازی کرده و مشاهده می‌کنیم برای نمونه‌های تصادفی از نمایش‌های تولید شده از داده اصلی نزدیک ترین همسایه‌ها تا چه حد مشابه نمایش‌های تولید شده از داده اصلی هستند. معمولا در الگوریتم نزدیک ترین همسایه با توجه به اکثریت همسایه‌های نمونه که در واقع کمترین معیار فاصله از داده را در فضای نمونه دارند، خود داده اصلی در گروه اکثریت همسایه‌ها با همان برچسب قرار می‌گیرد اما در این پروژه استفاده از الگوریتم نزدیک ترین همسایه برای آزمایش نمایش‌های تولید شده توسط شبکه با الگوریتم یادگیری خودنظارتی است و نتایج آن صرفا برای این هدف مشاهده می‌شوند. در شکل ۶.۴عملکرد کلی این الگوریتم برای یافتن نزدیک ترین همسایه نشان داده شده است. [۸]



شکل ۶.۴: الگوریتم نزدیک ترین همسایه

الگوریتم نزدیک‌ترین همسایه (KNN) یکی از الگوریتم‌های یادگیری ماشینی بدون نظارت است که برای دسته‌بندی و پیش‌بینی داده‌ها به کار می‌رود. اصول عمل این الگوریتم بسیار ساده است و بر مفهوم نزدیکی داده‌ها در فضای ویژگی‌ها بنا شده است.

راهکار الگوریتم KNN در پیدا کردن همسایه‌های نزدیک:

فرض کنید ما مجموعه‌ای از داده‌ها را داریم و می‌خواهیم برای تعدادی نمونه تصادفی که به ما داده می‌شود، نزدیک‌ترین همسایه‌های آن را پیدا کنیم. راهکار الگوریتم KNN به این صورت است:

۱. انتخاب تعداد همسایه :(K)

ابتدا باید تعداد همسایه‌هایی که می‌خواهیم پیدا کنیم را انتخاب کنیم. این تعداد به عنوان K شناخته می‌شود.

۲. محاسبه فواصل:

برای هر داده ما باید فاصله‌ی آن با تمام داده‌های موجود در مجموعه‌داده را محاسبه کنیم. این فواصل معمولاً با استفاده از معیارهای مختلفی مانند فاصله اقلیدسی یا فاصله کسینوسی محاسبه می‌شوند.

۳. انتخاب همسایه‌ها:

سپس تعداد همسایه‌ای که کمترین فواصل را با داده جدید دارند به عنوان همسایه‌های نزدیک انتخاب می‌شوند. و در نهایت نتیجه الگوریتم را بر روی نمایش های تولید شده توسط شبکه مشاهده و بررسی می‌کنیم.

۷.۴ دسته‌بندی کننده

در زمینه یادگیری خودنظارتی، وجود دسته‌بندی کننده [۱۶] یک جزء مهم است که برای ارزیابی کیفیت نمایش‌های آموخته‌شده استفاده می‌شود. این یک لایه شبکه عصبی اضافی است که به ویژگی‌های آموخته شده استخراج شده از شبکه اصلی متصل است. هدف اصلی دسته‌بندی‌کننده، تبدیل نمایش‌های تولیده شده توسط شبکه به برچسب‌های کلاس یا سایر پیش‌بینی‌های مرتبط است که می‌تواند برای وظایف نظارت‌شده استفاده شود.

دسته‌بندی کننده با استفاده از وظیفه خود نظارتی آموزش داده می شود. این به این معنی است که در طول مرحله آموزش خودنظارتی، سر دسته‌بندی کننده برای پیش‌بینی برخی ویژگی‌های خاص داده‌های

[16] Classifier head

ورودی که به روشی خاص تبدیل یا تقویت شده‌اند، آموزش داده می‌شود. این ویژگی می تواند چرخش، رنگ آمیزی و غیره باشد، بسته به روش خود نظارتی انتخاب شده اند. ابتدا شبکه عصبی پیاده‌سازی شده نمایش‌ها را از داده تقویت شده تولید می‌کند و می آموزد فاصله نمایش های داده‌های یکسان و در یک دسته را بدون برچسب طی فرایند یادگیری خودنظارتی کمینه کند سپس از شبکه آموزش دیده برای به کار گیری در وظایفی مانند دسته بندی نظارت شده استفاده می‌شود. در این مرحله یک دسته بندی کننده که در این پروژه از یک لایه کاملا متصل استفاده می‌شود بعد نمایش خروجی شبکه را به تعداد دسته‌های نهایی مورد نظر کاهش می‌دهد و از بردار خروجی برای وظایف نظارت‌شده استفاده می‌کند.

به طور خلاصه، دسته‌بندی کننده در یادگیری خود نظارتی به عنوان پلی بین وظیفه خود نظارت شده و وظایف نظارت‌شده عمل می کند و به شبکه اجازه می دهد تا نمایش های معنی دار را از داده‌های بدون برچسب یاد بگیرد و سپس این نمایش‌ها را برای کارهای خاص تنظیم یا تطبیق دهد.

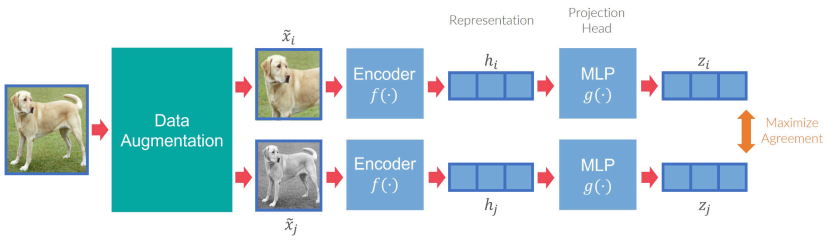
در این پروژه ابتدا شبکه با داده بدون برچسب با روش یادگیری خودنظارتی نمایش‌هایی از داده تولید می‌کند و در ادامه شبکه به علاوه دسته‌بندی کننده با داده با حجم کم برچسب‌دار طی یک فرآیند یادگیری نظارت شده یک عملیات دسته بندی را انجام می‌دهد به این صورت که یک بار شبکه اصلی به عنوان یک ستون [۱۷] به دسته‌بندی کننده متصل می‌شود و با نرخ یادگیری پایین بر روی داده برچسب‌دار با حجم کم آموزش می‌بیند. و یک بار نیز وزن‌های شبکه آموزش دیده شده را ثابت نگه داشته و تنها دسته‌بندی کننده آموزش می‌بیند و نتایج را مشاهده کرده و با یادگیری از ابتدا نظارت شده مقایسه می‌کنیم.

امید است فرایند یادگیری خودنظارتی در افزایش دقت شبکه در عملیات نهایی دسته بندی با داده برچسب دار با حجم کم موثر واقع شود. در نتیجه در انتهای فرآیند یادگیری شبکه ، یک بار نیز همان شبکه را با حجم داده کم و از ابتدا نظارت شده آموزش می‌دهیم تا نتیجه هر دو روش را با هم مقایسه کرده و میزان تاثیر فرایند خودنظارتی را بررسی کنیم.

در شکل ۷.۴ یک نمونه از معماری شبکه در یادگیری خودنظارتی نمایش داده شده است که ابتدا یک افزایش داده پیاده‌سازی شده و سپس یک شبکه عصبی اسپایکی با لایه‌های پیچشی و حداکثر ادغام وظیفه تولید نمایش‌های داده را به عهده دارد و در نهایت نیز لایه کاملا متصل دسته‌بندی را روی نمایش‌های

[17] Backbone

تولید شده اعمال کرده و تابع هزینه بر روی آن محاسبه می‌شود.



با توجه به توضیحات داده شده برای پیاده سازی و روش شناسی حل مساله در فصل بعد نتایج پیاده سازی را مشاهده و بررسی می‌کنیم.

فصل ۵

پیاده سازی و نتایج

در این بخش ابتدا پیاده‌سازی مدل نورونی را انجام داده سپس اول از نورون ساخته‌شده در یک شبکه عصبی پیشرو استفاده می‌کنیم و در مرحله بعد معماری شبکه عصبی اسپایکی پیچشی را با مدل نورونی نشت کننده و ادغام آتش پیاده سازی کرده و در نهایت الگوریتم خودنظارتی را به ساختار شبکه اضافه می‌کنیم و نتایج را برای داده های MNIST و CIFAR۱۰ مشاهده و بررسی خواهیم کرد.

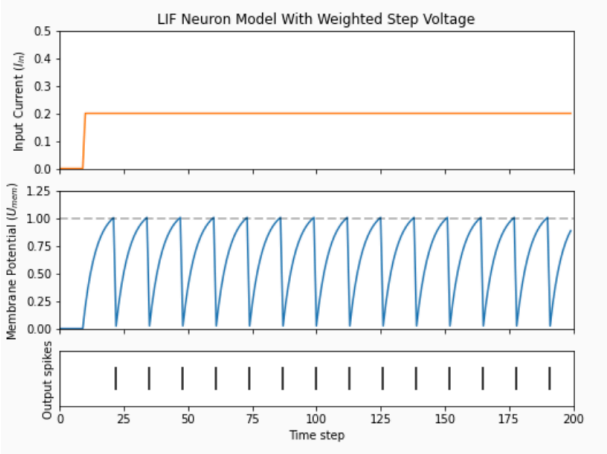
۱.۵ پیاده سازی مدل نورونی

در ابتدا مدل نورونی را با توجه به آنچه در فصل یک توضیح داده شد پیاده‌سازی کرده و تغییرات پتانسیل و فرآیند تولید اسپایک را بررسی می‌کنیم. همانطور که گفته شد بعد خروجی مدل نورونی برابر است با :

$$[\text{step time} \times \text{size batch} \times \text{dimension feature}]$$

و با توجه به اینکه بعد زمان در تانسور خروجی وجود دارد و داده به عنوان جریان ثابت به شبکه در طول گام های زمانی داده شده است، برای بررسی روند تولید اسپایک، تغییرات پتانسیل هر نورون را در طول گام‌های زمانی جمع بسته و بررسی می‌کنیم آیا پتانسیل از آستانه عبور کرده یا خیر. در شکل ۱.۵ یک

نمونه از جریان ثابت ورودی و یک نمونه از خروجی نورون که شامل تغییرات پتانسیل و گسیل اسپایک می‌شود نشان داده‌شده است.

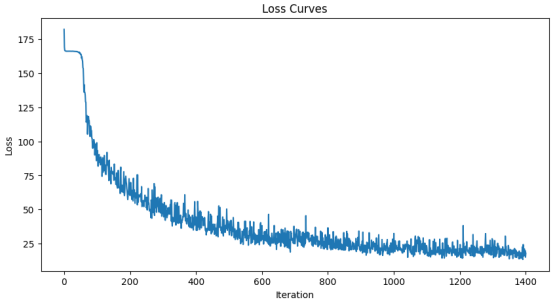


شکل ۱.۵: پیاده‌سازی مدل نورونی نشت‌کننده و ادغام آتش ونمایش تولید اسپایک در صورت عبور پتانسیل از آستانه

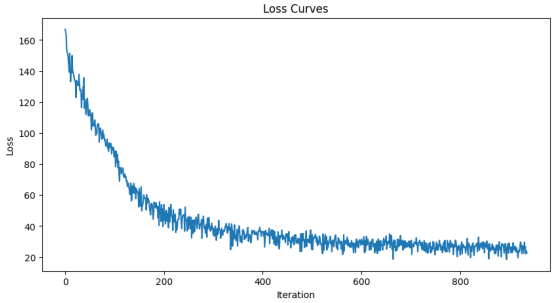
۲.۵ پیاده سازی یادگیری خودنظارتی روی شبکه عصبی اسپایکی

برای ساخت اتصالات بین نورون ها از اتصالات پیچشی و کاملا متصل استفاده می‌کنیم به این صورت که لایه‌های اسپایکی خروجی‌های اسپایک و تغییرات پتانسیل غشا را تولید می‌کنند و لایه‌های پیچشی و کاملا متصل نوع اتصال نورون ها را مشخص می‌کنند.

برای پیاده سازی بخش خودنظارتی هر بار دو نمونه از داده افزوده‌شده را در دسته‌های با اندازه‌های مختلف به شبکه داده و تابع هزینه را روی خروجی شبکه که در طول زمان جمع بسته شده اعمال می‌کنیم و فرآیند آموزش را طی می‌کنیم تا شبکه بیاموزد نمایش‌های مناسب از داده را تولید کند. نمودار تغییر تابع هزینه را در طول زمان مشاهده می‌کنیم و به این معناست که شبکه در حال آموختن است. با توجه به اینکه در این مرحله فرایند آموزش با داده بدون برچسب اتفاق می‌افتد تنها تغییر تابع هزینه بر روی مجموعه داده آموزشی در شکل‌های ۲.۵ و ۳.۵ قابل مشاهده است.



شکل ۲.۵: نمودار تابع هزینه شبکه برای مجموعه داده mnistدر فرایند یادگیری خودنظارتی

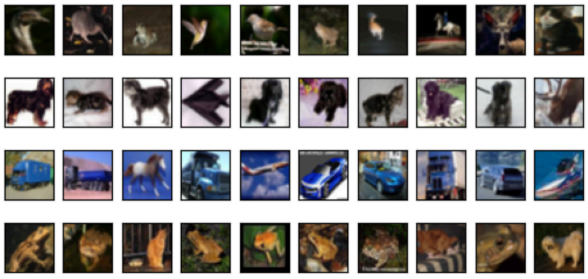


شکل ۳.۵: نمودار تابع هزینه شبکه برای مجموعه داده cifar فرایند یادگیری خودنظارتی

سپس برای آزمایش نمایش‌های تولید شده و آزمودن عملکرد شبکه یک بار نمایش‌ها را به عنوان ورودی به الگوریتم نزدیک‌ترین همسایه می‌دهیم و بررسی می‌کنیم و یک بار نیز دسته‌بند را به عنوان لایه آخر به شبکه اضافه می‌کنیم و بعد نمایش خروجی را به اندازه تعداد دسته داده که در این پروژه برای هر دو مجموعه داده MNIST ، CIFAR۱۰ تعداد ۱۰ است کاهش می‌دهیم و طی یک فرایند آموزش نظارت شده دقت شبکه را می‌سنجیم.

۳.۵ پیاده سازی الگوریتم نزدیک ترین همسایه

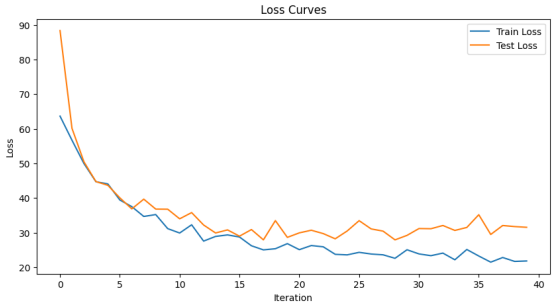
از کتابخانه sklearn برای پیاده کردن الگوریتم استفاده کرده و بر روی دو مجموعه داده آزمایش کرده ایم به این صورت که چند نمایش خروجی از شبکه را به صورت تصادفی انتخاب کرده و ده همسایه نزدیک به آن را مشاهده کرده ایم . با توجه به شکل ۴.۵ مشاهده می‌شود که الگوریتم نزدیک ترین همسایه نمایش‌های مشابه را برای نمونه‌های تصادفی از خروجی شبکه به درستی تشخیص داده است.



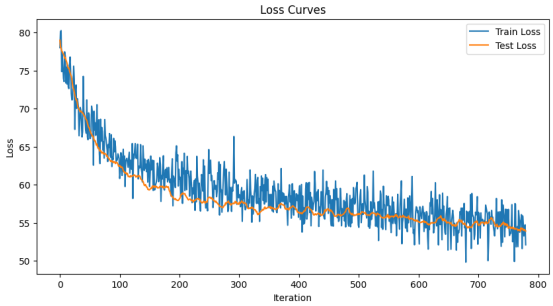
شکل ۴.۵: نمونه خروجی الگوریتم نزدیک ترین همسایه بر روی نمایش های تولید شده توسط شبکه عصبی اسپایکی در طول یادگیری خودنظارتی

۴.۵ پیاده سازی دسته بندی کننده و نتایج

برای اضافه کردن دسته‌بندی‌کننده از یک لایه خطی کاملا متصل استفاده می‌کنیم که بعد تانسور خروجی را به تعداد دسته‌ها در داده کاهش می‌دهد و با جمع بستن خروجی در طول زمان هر درایه‌ای در تانسور خروجی اسپایک که بیشترین فعالیت را داشته باشد به عنوان برچسب مربوط به آن داده در نظر گرفته می‌شود. لازم به ذکر است چون یکی از اهداف مهم یادگیری خودنظارتی کم کردن نیاز به داده برچسب دار است در این مرحله از فرآیند آموزش شبکه با حدود ٪۲۰ از حجم داده اصلی آموزش می‌بینند. در صورتی که ٪۸۰ از داده برای آموزش خودنظارتی شبکه استفاده شده است. در شکل‌های ۵.۵ و ۶.۵ تغییر تابع هزینه در طول فرآیند یادگیری نظارت شده را مشاهده می‌کنیم.



شکل ۵.۵: نمودار تابع هزینه شبکه برای مجموعه داده MNIST فرایند یادگیری نظارت شده با ٪۲۰ حجم داده



شکل ۶.۵: نمودار تابع هزینه شبکه برای مجموعه داده CIFAR فرایند یادگیری نظارت شده با ٪۲۰ حجم داده

پارامترهایی همچون اندازه دسته ، انتخاب روش افزودن داده و تعداد گام زمانی در عملکرد نهایی شبکه تاثیرگذارند. در جدول‌های زیر آزمایش‌های انجام شده برای مشاهده تاثیر هر یک از پارامترها نشان داده شده است.

اندازه دسته	MNIST	CIFAR۱۰
۶۴	۷۷.۴۵	۵۲.۷۶
۱۲۸	۷۹.۸۷	۵۵.۲۸
۲۵۶	۸۲.۱۳	۵۸.۷۹

جدول ۱.۵: درصد دقت عملکرد شبکه با توجه به اندازه دسته‌های مختلف

از طرفی با توجه به اینکه هدف اصلی اعمال یادگیری خودنظارتی بر روی شبکه، بی‌نیاز کردن یادگیری

گام زمانی	MNIST	CIFAR۱۰
۱۰	۷۸.۲۱	۵۳.۶۳
۲۰	۸۱.۵۴	۵۷.۸۴
۳۰	۸۲.۳۴	۵۸.۱۵

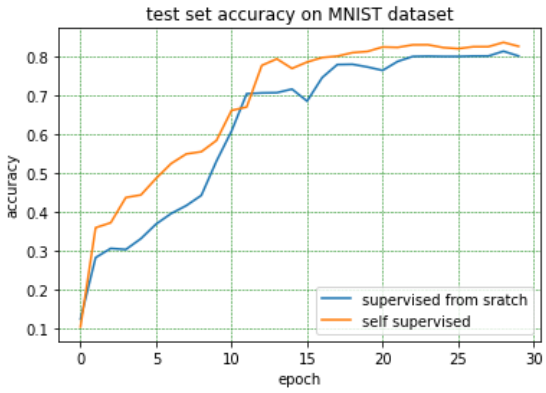
جدول ۲.۵: درصد دقت عملکرد شبکه با توجه به گام های زمانی مختلف

روش افزودن داده	CIFAR۱۰	MNIST
برش - چرخش	۵۵.۲۱	۸۱.۴۳
برش - نویز گاوسی	۵۶.۱۲	۸۲.۰۶
چرخش - نویز گاوسی	۵۸.۵۴	۸۲.۲۳

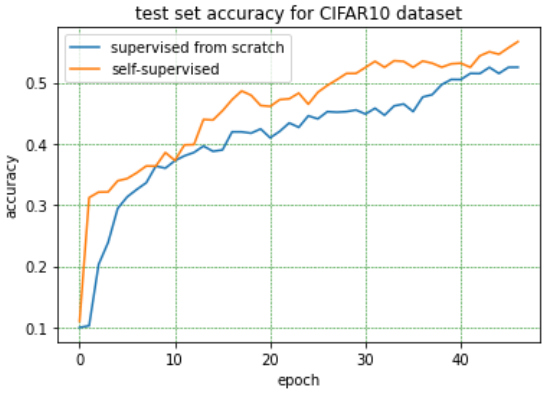
جدول ۳.۵: درصد دقت عملکرد شبکه با توجه به روش‌های مختلف افزودن داده

از داده برچسب‌دار است انتظار می‌رود پس از اعمال این فرآیند، شبکه از فرآیند یادگیری نظارت شده روی داده برچسب دار با حجم کم دقت بهتری داشته باشد.در نتیجه یک بار نیز شبکه با همان معماری را به صورت نظارت‌شده بر روی داده با حجم کم(۲۰٪ از حجم کل داده) آموزش می‌دهیم و نتایج را مقایسه می‌کنیم . پس از آزمایش‌های انجام شده نتیجه گرفتیم که شبکه عصبی اسپایکی با فرآیند یادگیری خودنظارتی عملکرد بهتری از آموزش نظارت شده برای داده‌های برچسب‌دار با حجم کم دارد.

در نمودارهای ۷.۵ و ۸.۵دقت هریک از روش‌های خودنظارتی و نظارت شده بر روی مجموعه داده‌های MNIST ، CIFAR۱۰ نشان داده شده است.



شکل ۷.۵: مقایسه دقت عملکرد روش‌های یادگیری خودنظارتی و نظارت‌شده بر روی مجموعه داده MNIST



شکل ۸.۵: مقایسه دقت عملکرد روش‌های یادگیری خودنظارتی و نظارت شده بر روی مجموعه داده CIFAR۱۰

همانطور که مشاهده می‌شود پیاده سازی روش یادگیری خودنظارتی موجب افزایش دقت عملکرد شبکه شده‌است. و همچنان نیاز به داده برچسب‌دار را بسیار کاهش داده و با توجه به اینکه روش یادگیری خودنظارتی بر روی شبکه‌های اسپایکی اعمال شده به علت ساختار این شبکه‌ها، نیاز به مصرف انرژی برای یادگیری نیز کاهش پیدا کرده و فرایند آموزش بهینه تر انجام شده است.

۵.۵ چشم انداز پژوهشی

با توجه به اینکه پیاده‌سازی روش‌های یادگیری خودنظارتی بر روی شبکه‌های عصبی اسپایکی یک حوزه پژوهشی نوآورانه و جذاب محسوب می‌شود، امکانات بی‌نظیری را برای بهبود کاربردها و پیشرفت در علم و فناوری فراهم می‌کند. این پیشرفت در ارتقاء روش‌های یادگیری و استفاده از داده‌های پیچیده‌تر، می‌تواند اثرات مثبتی بر تحقیقات مختلف داشته باشد. با توجه به چالش کمبود داده برچسب‌دار و هزینه و زمان‌بر بودن فرآیند تولید برچسب برای داده، این چالش ممکن است مانع از استفاده کامل از روش‌های یادگیری تقویتی و عمقی شود.

از طرفی با توجه به ماهیت بیولوژیکی شبکه‌های عصبی اسپایکی و بهینه بودن آن‌ها در مصرف انرژی زمان نسبت به شبکه‌های عصبی مصنوعی، استفاده از این شبکه‌ها منجر به پیشرفت‌هایی در حوزه‌های مختلف یادگیری و علوم اعصاب خواهد شد. با پیاده‌سازی روش یادگیری خودنظارتی بر روی شبکه‌های عصبی اسپایکی، می‌توان به صورت تخمینی و بدون نیاز به داده‌های برچسب‌دار، توانایی‌های شبکه‌ها

در تشخیص الگوها و اطلاعات مهم را بهره‌برداری کرد. این روش، باعث تولید ابزاری مؤثر در ایجاد نمایندگان با کیفیت از داده‌ها می‌شود که می‌تواند در کاربردهای متعددی در زمینه‌های مختلف مانند تشخیص بیماری‌ها، پیش‌بینی تغییرات و... مفید واقع شود. امید است این ابزار محاسباتی در حوزه‌های مختلف یادگیری مفید واقع شود و بتوان به این وسیله در مسائل مختلف یادگیری ماشین و پردازش داده تصویری موجب پیشرفت شد.

در ادامه روند این پژوهش بنا داریم با پیچیده‌تر کردن شبکه و بررسی پارامترها عملکرد شبکه را بهبود بخشیم و با افزایش دقت شبکه یک ابزار قوی و قابل اعتماد محاسباتی تولید کنیم که نه تنها نیاز به داده‌برچسب‌دار را کاهش دهد بلکه در مصرف انرژی و حافظه نیز بهینه باشد.

واژه‌نامه فارسی به انگلیسی

نورون	Neuron
شبکه عصبی	Neural Network
یادگیری ماشینی	Machine Learning
اطلاعات	Information
انتقال	Transfer
بینایی ماشین	Computer Vision
تصویر	Image
داده‌ها	Data
افزایش داده	Data Augmentation
طبقه‌بندی	Classification
آموزش با نظارت	Supervised Learning
آموزش بدون نظارت	Unsupervised Learning
آموزش نیمه‌نظارتی	Semi-Supervised Learning
یادگیری انتقالی	Transfer Learning
تشخیص نمونه‌های خارجی	Outlier Detection
یادگیری نظارت‌نشده	Unsupervised Learning
شباهت	Similarity
تصویرسازی داده‌ها	Data Visualization
روش‌های مقایسه‌ای	Comparative Methods
الگوریتم‌های یادگیری	Learning Algorithms
مدل‌های شبکه عصبی	Neural Network Models
آموزش خودنظارتی	Self-Supervised Learning

الگوریتم k نزدیک‌ترین همسایه (k-NN) ..... k-Nearest Neighbor Algorithm (k-NN)
شبکه عصبی اسپایکی (SNN) ..... Spiking Neural Network (SNN)
نورون عصبی اسپایکی ..... Spiking Neuron
انتقال اطلاعات با پالس‌ها ..... Spike-based Information Transmission
تاخیر انتشار پالس ..... Spike Propagation Delay
ساختار لایه نورونی ..... Neural Layer Structure
سیناپس‌های ورودی ..... Input Synapses
سیناپس‌های خروجی ..... Output Synapses
مدل هادجکین ـ هاکسلی ..... Hodgkin-Huxley Model
نقطه توازن دینامیکی ..... Dynamic Equilibrium Point
الگوریتم آموزش نورون‌ها ..... Neuron Training Algorithms
نورون مصنوعی ..... Artificial Neuron
شبکه عصبی پالسی ..... Spike-based Neural Network
پالس الکتریکی ..... Electric Pulse (Spike)
پتانسیل غشایی ..... Membrane Potential
سقف آستانه ..... Threshold Ceiling
آستانه پتانسیل غشایی ..... Membrane Potential Threshold
تغییر اندازه تصویر ..... Image Resizing
چرخش تصویر ..... Image Rotation
برش تصویر ..... Image Cropping
افزودن نویز ..... Adding Noise
تغییر روشنایی تصویر ..... Image Brightness Adjustment
انعکاس تصویر ..... Image Flipping
تغییر زاویه تصویر ..... Image Perspective Transformation
اعمال افکت‌های تصویری ..... Applying Image Effects
تغییر ترتیب کانال‌های رنگی ..... Changing Color Channel Order
تغییر شدت تصویر ..... Image Intensity Adjustment

واژه‌نامه انگلیسی به فارسی

نورون	neuron
تیزه	spike
سیناپس	synapse
یادگیری خودنظارتی	self-supervised learning
بینایی کامپیوتری	computer vision
ماژول	module
پرسپترون	perceptron
نویز	noise
متضاد	contrastive
فضای نمونه	embedding space
فاصله اقلیدسی	Euclidean Distance
شباهت کسینوسی	Cosine Similarity
اطلاعات	Information
انتقال	Transfer
تصویر	Image
داده‌ها	Data
افزایش داده	Data Augmentation
طبقه‌بندی	Classification
آموزش با نظارت	Supervised Learning
آموزش بدون نظارت	Unsupervised Learning
آموزش نیمه‌نظارتی	Semi-Supervised Learning
یادگیری انتقالی	Transfer Learning

تشخیص نمونه‌های خارجی ..... Outlier Detection
شباهت ..... Similarity
تصویرسازی داده‌ها ..... Data Visualization
روش‌های مقایسه‌ای ..... Comparative Methods
الگوریتم‌های یادگیری ..... Learning Algorithms
مدل‌های شبکه عصبی ..... Neural Network Models
شبکه عصبی اسپایکی ..... Spiking Neural Network (SNN)
نورون عصبی اسپایکی ..... Spiking Neuron
انتقال اطلاعات با پالس‌ها ..... Spike-based Information Transmission
تاخیر انتشار پالس ..... Spike Propagation Delay
ساختار لایه نورونی ..... Neural Layer Structure
سیناپس‌های ورودی ..... Input Synapses
سیناپس‌های خروجی ..... Output Synapses
مدل هادجکین ـ هاکسلی ..... Hodgkin-Huxley Model
نقطه توازن دینامیکی ..... Dynamic Equilibrium Point
الگوریتم آموزش نورون‌ها ..... Neuron Training Algorithms
نورون مصنوعی ..... Artificial Neuron
استفاده از پالس‌ها برای پردازش اطلاعات ..... Spike-based Information Processing
شبکه عصبی پالسی ..... Spike-based Neural Network
پالس الکتریکی ..... Electric Pulse (Spike)
پتانسیل غشایی ..... Membrane Potential
سقف آستانه ..... Threshold Ceiling
آستانه پتانسیل غشایی ..... Membrane Potential Threshold
تغییر اندازه تصویر ..... Image Resizing
چرخش تصویر ..... Image Rotation
برش تصویر ..... Image Cropping
افزودن نویز ..... Adding Noise
تغییر روشنایی تصویر ..... Image Brightness Adjustment
تغییر کنتراست تصویر ..... Image Contrast Adjustment
انعکاس تصویر ..... Image Flipping
تغییر زاویه تصویر ..... Image Perspective Transformation
اعمال افکت‌های تصویری ..... Applying Image Effects

تغییر ترتیب کانال‌های رنگی ..... Changing Color Channel Order

تغییر شدت تصویر..... Image Intensity Adjustment

کتاب‌نامه

[1] Assran, M., Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas (2023). Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629. 23

[2] Bing, Z., C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll (2018). A survey of robotics control based on learning-inspired spiking neural networks. *Frontiers in neurorobotics 12*, 35. 1

[3] Chamand, B. and P. Joly (2022). Self-supervised spiking neural networks applied to digit classification. In *Proceedings of the 19th International Conference on Content-based Multimedia Indexing*, pp. 196–200.

[4] Chaney, K., A. Panagopoulou, C. Lee, K. Roy, and K. Daniilidis (2021). Self-supervised optical flow with spiking neural networks and event based cameras. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5892–5899. IEEE.

[5] Eshraghian, J. K., M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu (2021). Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:2109.12894*. 46

[6] Falcon, W. and K. Cho (2020). A framework for contrastive self-supervised learning and designing a new approach. *arXiv preprint arXiv:2009.00104*. 24, 25, 31, 32

[7] Gidaris, S. and N. Komodakis (2018). Unsupervised representation learning by predicting image rotations. *International Conference on Learning Representations (ICLR)*. 1

[8] Guo, G., H. Wang, D. Bell, Y. Bi, and K. Greer (2003). Knn model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*, pp. 986–996. Springer. 51

[9] Hagenaars, J., F. Paredes-Vallés, and G. De Croon (2021). Self-supervised learning of event-based optical flow with spiking neural networks. *Advances in Neural Information Processing Systems 34*, 7167–7179.

[10] Hodgkin, A. and A. Huxley (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology* 117(4), 500–544. 8

[11] Hsu, C.-Y., A. Zeitoun, G.-H. Lee, D. Katabi, and T. Jaakkola (2019). Self-supervised learning of appliance usage. In *International Conference on Learning Representations*. 22

[12] Huynh, P. K., M. L. Varshika, A. Paul, M. Isik, A. Balaji, and A. Das (2022). Implementing spiking neural networks on neuromorphic architectures: A review. *arXiv preprint arXiv:2202.08897*. 2

[13] Jaiswal, A., A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon (2020). A survey on contrastive self-supervised learning. *Technologies* 9(1), 2. 2, 19

[14] Jiao, Y., K. Yang, D. Song, and D. Tao (2022). Timeautoad: Autonomous anomaly detection with self-supervised contrastive loss for multivariate time series. *IEEE Transactions on Network Science and Engineering* 9(3), 1604–1619. 32

[15] Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 51

[16] Lein, E., T. C. Südhof, R. C. Malenka, and R. A. Nicoll (2016). Synaptic transmission and plasticity. *Cold Spring Harbor Perspectives in Biology* 4(5). 4

[17] Li, G., R. Togo, T. Ogawa, and M. Haseyama (2022). Tribyol: Triplet byol for self-supervised representation learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3458–3462. IEEE. 33

[18] Liu, X., F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang (2021). Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering* 35(1), 857–876. 24

[19] Mao, H. H. (2020). A survey on self-supervised pre-training for sequential transfer learning in neural networks. *arXiv preprint arXiv:2007.00800*. 30

[20] Medina, C., A. Devos, and M. Grossglauser (2020). Self-supervised prototypical transfer learning for few-shot classification. *arXiv preprint arXiv:2006.11325*. 30

[21] Misra, I. and L. v. d. Maaten (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6707–6717. 20

[22] Moreno-Bote, R. and N. Parga (2006). Auto-and crosscorrelograms for the spike response of leaky integrate-and-fire neurons with slow synapses. *Physical review letters* 96(2), 028101. 10

[23] Ng, N., K. Cho, and M. Ghassemi (2020). Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness. *arXiv preprint arXiv:2009.10195*. 27

[24] O'Shea, K. and R. Nash (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*. 46

[25] Ponulak, F. and A. Kasiński (2011). Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis 71*, 409. 3

[26] Rafi, T. H. (2021). A brief review on spiking neural network-a biological inspiration. 1

[27] Rathi, N., I. Chakraborty, A. Kosta, A. Sengupta, A. Ankit, P. Panda, and K. Roy (2023). Exploring neuromorphic computing based on spiking neural networks: Algorithms to hardware. *ACM Computing Surveys* 55(12), 1–49. 2

[28] Shurrab, S. and R. Duwairi (2022). Self-supervised learning methods and applications in medical imaging analysis: A survey. *PeerJ Computer Science 8*, e1045. 2

[29] Taher, M. R. H., F. Haghighi, M. B. Gotway, and J. Liang (2022). Caid: Context-aware instance discrimination for self-supervised learning in medical imaging. In *International Conference on Medical Imaging with Deep Learning*, pp. 535–551. PMLR. 25

[30] Taherkhani, A., A. Belatreche, Y. Li, G. Cosma, L. P. Maguire, and T. M. McGinnity (2020). A review of learning in biologically plausible spiking neural networks. *Neural Networks 122*, 253–272. 1

[31] Wu, Y., L. Deng, M. Chen, Y. Liu, and J. Li (2020). Training spiking neural networks using lessons from deep learning. *Nature* 584(7822), E8–E16. 5

[32] Xie, Y., Z. Xu, J. Zhang, Z. Wang, and S. Ji (2022). Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence* 45(2), 2412–2429. 21, 22

[33] Zhai, X., A. Oliver, A. Kolesnikov, and L. Beyer (2019). S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1476–1485. 20

Surname: Bahari Asl Name: Yeganeh

Title: self-supervised learning on spiking neural network

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

Degree: Master of Science Subject: Department of Data & Computer Science
Field: Computer Sciences - Data Mining

Shahid Beheshti University Faculty of Mathematical Sciences
Date: September 2023 Number of pages: 76

Keywords: Spiking neural networks, self-supervised learning, data classification

Abstract

In this thesis, we examine and introduce spiking neural networks as a new generation of neural networks. Inspired by the structure of the brain, these networks encode and process information through precisely timed spike trains. Unlike the rest of the neural networks, in this type of network, the neurons are resting except when generating spikes, and this property makes the structure of these networks optimal and uses less memory. In this research, we investigate self-supervised learning methods in spiking neural networks. This method has the ability to use techniques such as data augmentation to perform learning in a redundant network of labeled data. Finally, the aim of this research is to build an optimal spiking neural network with the capability of self-supervised learning methods to generate representations from the MNIST and CIFAR10 datasets without the need for data labels, and to use these representations for various tasks such as supervised classification. This research tries to finally create more accuracy on the classification methods on the data set by making these representations.

Shahid Beheshti University

Faculty of Mathematical Sciences

Department of Data & Computer Science

Master Thesis

self-supervised learning on spiking neural network

by

Yeganeh Bahari Asl

Supervisors

Dr.Hadi Farahani and Dr. Saeed Reza Kheradpisheh

September 2023