دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده مهندسی برق

پروژه کارشناسی

عنوان:

طبقه‌بندی نورون‌های مهاری و تحریکی با استفاده از شبکه‌های عصبی پیچشی

نگارش:

امیرحسین مشق‌دوست

استاد پروژه:

دکتر علی خادم

بهار ۱۴۰۱

تاسیس ۱۳۰۷

دانشگاه صنعتی خواجه نصیرالدین طوسی

تاییدیه هیأت داوران

هیأت داوران پس از مطالعه پروژه و شرکت در جلسه دفاع از پروژه تهیه شده تحت عنوان **طبقه‌بندی نورون‌های مهاری و تحریکی با استفاده از شبکه‌های عصبی پیچشی** توسط **امیرحسین مشق دوست** صحت و کفایت تحقیق انجام شده را برای اخذ درجه **کارشناسی** رشته **مهندسی برق** در تاریخ **۱۴۰۱/۰۴/۲۰** مورد تأیید قرار دادند.

استاد راهنما دکتر **علی خادم**

امضاء

تاسیس ۱۳۰۷

دانشگاه صنعتی خواجه نصیرالدین طوسی

اظهارنامه دانشجو

اینجانب **امیرحسین مشق‌دوست** دانشجوی مقطع کارشناسی رشته **مهندسی برق** گواهی می‌نمایم که تحقیقات ارائه شده در پروژه با عنوان:

طبقه بندی نورون‌های مهاری و تحریکی با استفاده از شبکه‌های عصبی پیچشی

با راهنمایی استاد/اساتید محترم **جناب آقای دکتر علی خادم** توسط شخص اینجانب انجام شده است. صحت و اصالت مطالب نگارش شده در این پروژه مورد تأیید می‌باشد و در تدوین متن پروژه چارچوب (فرمت) مصوب دانشگاه را به طور کامل رعایت کرده‌ام.

امضاء دانشجو:

تاریخ: ۱۴۰۱/۰۲/۳۱

حق طبع، نشر و مالکیت نتایج

۱- حق چاپ و تکثیر این پروژه متعلق به نویسنده و استاد راهنمای آن می‌باشد. هرگونه تصویربرداری از کل یا بخشی از پروژه تنها با موافقت نویسنده یا استاد/استادان راهنما یا کتاب‌خانه دانشکده مهندسی برق دانشگاه صنعتی خواجه نصیرالدین طوسی مجاز می‌باشد.

۲- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی خواجه نصیرالدین طوسی می‌باشد و بدون اجازه کتبی دانشگاه به شخص ثالث قابل واگذاری نیست.

۳- استفاده از اطلاعات و نتایج موجود پروژه بدون ذکر مرجع مجاز نمی‌باشد.

تشکر و قدردانی

با تشکر از استاد دکتر علی خادم بابت کمک‌های فراوان و زحماتشان در راستای تکمیل این پروژه.

چکیده

در این پروژه به طبقه‌بندی نورون‌های مهاری و تحریکی با استفاده از انواع طبقه‌بند پرداخته‌ایم. تشخیص نوع نورون ها از این جهت برای ما مهم است که می‌تواند نقش آن ها را در شبکه های نورونی برای ما مشخص کند و در جراحی های مغز به پزشکان کمک کند تا با استفاده از مداخلات دارویی یا الکتریکی به بهبود بیمار کمک کنند. در این میان مدل‌های SVM، LDA و CNN را برای ورودی های یک بعدی اسپایک‌ها و دو بعدی (تبدیل موجک) آن‌ها پیاده‌سازی کردیم. مدل های مختلف را آموزش دادیم و صحت هر یک را با روش k-fold با دیگری مقایسه کردیم و نهایتا مدل CNN را به دلیل داشتن صحت بالاتر و قابلیت اطمینان بیشتر در داده های بیتر، به عنوان برترین مدل انتخاب کردیم. نهایتا یک برنامه‌ی نرم‌افزار گرافیکی از کاری که کردیم ارایه دادیم تا برچسبگذاری را برای اسپایک های ناشناخته انجام دهد.

کلمات کلیدی: نورون مهاری و تحریکی، یادگیری ماشین، یادگیری عمیق، ماشین بردار پشتیبان، شبکه ی عصبی پیچشی، آنالیز افتراقی خطی، ، تبدیل موجک

فهرست

فصل ۱- مقدّمه

از سؤالات مطرح در علوم اعصاب این است که چگونه برای تشخیص خواص تک نورون‌ها و تفکیک کارکردی آن‌ها از خصوصیات الکتروفیزیولوژی[۱]نظیر شکل موج اسپایک، میانگین نرخ آتش، تغییرات نرخ آتش و ... استفاده کنیم. پاسخ به این سؤال مهم است چون پس از تشخیص خواص تک نورون‌ها و تفکیک کارکردی آن‌ها می‌توانیم نقش آن‌ها را در مدارها و شبکه‌های نورونی مورد بررسی قرار دهیم. به علاوه، این رویکرد پزشکان را قادر خواهد ساخت که در زمان جراحی‌های عملکردی با تشخیص نوع نورون‌ها به صورت برخط، امکان مداخلات دارویی و الکتریکی را متناسب با کارکرد بافت مورد نظر داشته باشند.

وقتی نورون‌ها با نورون دیگر سیناپس برقرار می‌کنند، اگر سبب تحریک و شروع فعالیت یاخته دیگر شود، سیناپس تحریکی[۲]می‌باشد و اگر سبب آرامش و غیر فعال بودن یاخته دیگر شود سیناپس مهاری[۳]است. اصولا تشخیص مهاری یا تحریکی بودن یک نورون توسط بررسی تغییر نرخ آتش نورون‌های پس سیناپسی آن حین تحریک نورون انجام می‌شود. اخیرا طبقه بندی نورون‌های تحریکی و مهاری بر پایه ویژگی‌های استخراج شده از شکل موج اسپایک آن‌ها و روش‌های ساده یادگیری ماشین مورد توجه قرار گرفته است و نتایج نویدبخشی حاصل شده است.

در این پروژه، قصد داریم با استفاده از شبکه‌های یادگیری عمیق،[۴] نورون‌های تحریکی و مهاری را بدون استخراج ویژگی از اسپایک آن‌ها و با داده‌های خام شکل موج اسپایک از هم تفکیک کرده و صحت نتایج را با روش‌های سنتی یادگیری ماشین مقایسه کنیم. تا جایی که می‌دانیم، پیش از این هیچ تحقیقی در زمینه‌ی طبقه‌بندی نورون‌های تحریکی و مهاری با استفاده از شبکه‌های یادگیری عمیق روی داده‌های واقعی انجام نشده است و از این بابت این تحقیق نو آورانه است.

در این پروژه داده‌هایی که در اختیار داریم شامل اسپایک‌های ۶۷۶ نورون با فرکانس نمونه برداری ۲۵ کیلوهرتز از ناحیه هیپوکمپ نوعی موش بودند که برچسب تحریکی یا مهاری آن‌ها مشخص است. به طور جزیی با استفاده از مدل های SVM، LDA و شبکه های پیچشی روی داده های خام یک بعدی و همینطور روی تبدیل موجک ورودی‌ها، نورون های مهاری و تحریکی را کلاسبندی کردیم. برای ارزشیابی آموزش مدل هایمان از روش K-fold استفاده کردیم و دیدیم که روش های یادگیری عمیق شبکه های پیچشی بیشترین صحت را بین مدل هایمان داشتند.

---

۱ Electrophysiology
۲ Excitatory
۳ Inhibitory
۴ Deep Neural Networks

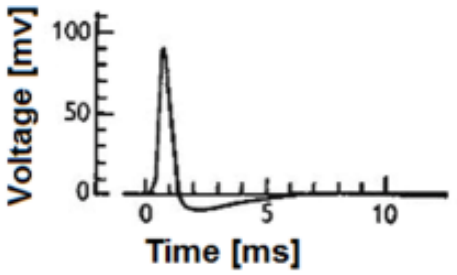
بعد از بررسی مدل هایمان آن ها را در محیط GUI پیاده‌سازی کردیم و برنامه ای طراحی کردیم که با گرفتن سیگنال خام اسپایک، طبقه‌بندی آن را به ازای هر مدل به ما می‌دهد. این برنامه با رای‌گیری از مدل های واحد، رای نهایی برای برچسب ورودی را به ما می‌دهد که همین امر باعث می‌شود صحت مدل نهایی ما احتمالا از صحت هر مدل به تنهایی بیشتر باشد. همچنین با استفاده از این برنامه می‌توانیم شکل اسپایک و شکل تبدیل موجک آن را هم ببینیم.

فصل ۲- مروری بر مفاهیم و ابزار مورد نیاز

۲-۱- پتانسیل عمل[۵]

[۱]پتانسیل عمل، عامل انتقال پیام عصبی در سیستم عصبی است. ایجاد محرک‌هایی مانند هرگونه پیام از محیط خارجی، فعالیت نورون‌ها جهت انتقال پیام عصبی را تقویت، مهار یا تعدیل می‌کنند. یون‌ها در جهت غلظت خود از کانال‌های یونی عبور می‌کنند که در نهایت این فرایند سبب ایجاد اختلاف بار الکتریکی در دو سمت غشا و خروج سلول از حالت آرامش می‌شود. نفوذپذیری غشا نسبت به یون پتاسیم بیشتر از یون سدیم است بنابراین پتانسیل آرامش، نزدیک به پتانسیل محیطی پتاسیم خواهد بود (پتانسیل حاصل از وجود یون پتاسیم به تنهایی). انتقال پیام عصبی در نورون‌ها با همین مکانیسم انجام می‌شود. فقط نورون‌ها و سلول‌های ماهیچه قادر به دریافت محرک و تولید پتانسیل عمل در پاسخ به آن هستند. تصویر اسپایک یک نورون را در شکل ۲-۱-۱ می‌بینیم.

دامنه پتانسیل عمل مستقل از جریان تولید شده عمل می‌کند یعنی بزرگ بودن جریان، ارتباطی با ایجاد پتانسیل عمل بزرگتر ندارد. بنابراین پتانسیل عمل از قانون همه یا هیچ پیروی می‌کند به این معنی که یا به صورت کامل ایجاد می‌شود یا ایجاد نمی‌شود. این مسئله با پتانسیل گیرنده‌ها (با دامنه وابسته به غلظت محرک) فرق می‌کند. اما تداوم هر دو پتانسیل گیرنده و پتانسیل عمل، با غلظت محرک در ارتباط هستند.



۲-۱-۱ نمونه ی شکل اسپایک یک نورون [۲]

۲-۲- نورون‌های مهاری و تحریکی

[۳]به طور کلی به محل اتصال یک نورون با نورون دیگر، سیناپس اطلاق می‌گردد. سیناپس‌ها به دو نوع شیمیایی و الکتریکی تقسیم می‌شوند. تقریبا تمام سیناپس‌های دستگاه عصبی مرکزی از نوع سیناپس‌های شیمیایی هستند. در این سیناپس‌ها، اولین نورون یک ماده شیمیایی در سیناپس انتهای عصبی (پایانه پیش سیناپسی) ترشح می‌کند که ناقل عصبی[۶] نام دارد. این ناقل به نوبه‌ی خود بد پروتئین‌های گیرنده‌ی موجود در غشا نورون بعدی (پایانه پس سیناپسی) اثر می‌کند و سبب تحریک یا مهار نورون یا تغییر حساسیت آن می‌شود. سیناپس‌های

---

۵ Spike
۶ Neurotransmitter

الکتریکی که بیش‌تر در سلول‌های عضله‌ی صاف و قلبی وجود دارند، امکان انتشار جریان‌های الکتریکی از یک سلول به سلول دیگر را از طریق اتصالات شکافی فراهم می‌آورند. یک تفاوت عمده سیناپس الکتریکی با شیمیایی این است که سیناپس الکتریکی می‌تواند سیگنال-ها را در دو جهت هدایت کند اما سیناپس شیمیایی، سیگنال‌ها را فقط در یک جهت از نورون پیش سیناپسی به نورون پس سیناپسی ارسال می‌کند.

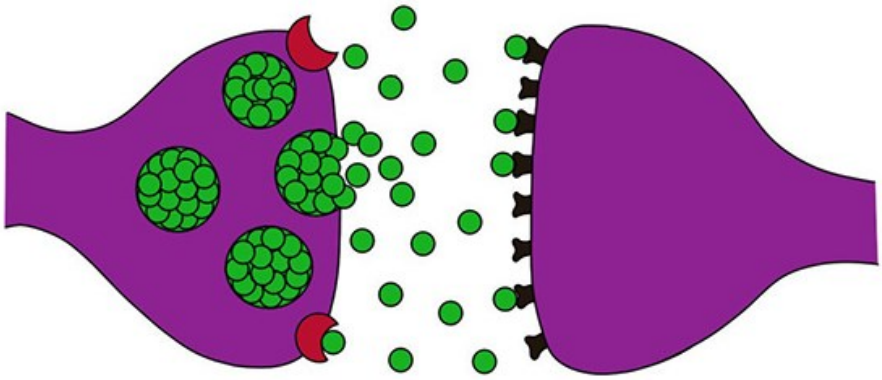
به پایانه پیش سیناپسی که معمولا روی دندریت و تا حدی روی جسم سلولی نورون پس سیناپسی، سیناپس ایجاد می‌کند، گره پایانه ای، دکمه یا پایک انتهایی با گره سیناپسی اطلاق می‌شود. بین این پایانه و پایانه پس سیناپسی یک شکاف سیناپسی قرار دارد که ماده ناقل عصبی به داخل این فضا آزاد می‌شود. در داخل پایانه پیش سیناپسی، وزیکول‌های ماده ناقل و میتوکندری وجود دارد وزیکول‌ها حاوی ماده ناقل عصبی بوده و میتوکندری‌ها، آدنوزین تری فسفات[۷] را می‌سازند که انرژی لازم برای سنتز مواد ناقل جدید را فراهم می‌کند.

حال ببینیم که چه عاملی سبب می‌شود تا وزیکول‌های حاوی ماده میانجی، محتویات خود را به داخل شکاف سیناپسی رها کنند. زمانی که پتانسیل عمل ایجاد شده در یک نورون به انتهای آکسون آن نورون (پایانه پیش سیناپسی) می‌رسد، موجب می‌شود تا کانال‌های کلسیمی وابسته به ولتاژ در آن قسمت فعال شوند. در این حالت مقادیر زیادی کلسیم وارد پایانه پیش سیناپسی می‌گردد. این یون‌های کلسیم به مولکول‌های پروتئینی خاصی در سطح داخلی غشای پیش سیناپسی متصل می‌شوند که محل‌های آزادسازی نام دارند اتصال کلسیم به این محل‌ها لازمه آزاد شدن وزیکول‌های حاوی ماده میانجی به داخل شکاف سیناپسی می‌باشد به طوری که مقدار ماده ناقل رها شده به طور مستقیم با تعداد یون‌های کلسیمی که وارد پایانه پیش سیناپسی می‌شوند، ارتباط دارد.

ماده ناقل پس از آزاد شدن از پایانه پیش سینایسی به گیرنده‌های پروتئینی اختصاصی خود در غشا پایانه پس سیناپسی متصل می‌شود. این گیرنده‌ها یک بخش متصل شونده به ناقل عصبی دارند که در سطح بیرونی غشاء نورون پس سیناپسی قرار گرفته و یک بخش میان غشایی دارند که کل عرض غشاء را طی می‌کند. این می‌تواند یک کانال یونی باشد با یک فعال کننده‌ی پیام رسان ثانویه داخل سلولی باشد. برخی از موارد گیرنده خاص خود به طور مستقیم سبب باز شدن یک کانال یونی می‌گردند. اگر کانالی که باز شده است. کانال سدیمی‌باشد، مقادیر زیادی یون سدیم وارد سلول شده و سبب تحریک سلول یا دپلاریزاسیون می‌شوند. اگر کانال‌های پتاسیم و یا کانال‌های آنیونی باز شوند، خروج یون پتاسیم از سلول و ورود آنیون‌ها (مانند کلر) به داخل سلول سبب هیپرپلاریزاسیون (افزایش بار منفی) سلول و مهار نورون پس سیناپسی خواهند شد. مواد ناقلی که با اتصال به گیرنده‌های خود بر غشا نورون پس سیناپسی، سیستم پیام رسان ثانویه داخل سلولی را فعال می‌کنند، برخلاف

[۷] ATP

موادی که بر کانال‌های یونی اثر دارند، سبب تغییرات طولانی مدت از چند ثانیه تا چند ماه پس از حذف ماده ناقل اولیه در نورون می‌کردند. یکی از معمول ترین انواع پیام رسان ثانویه، پروتئین‌های G هستند پروتئین‌های G معمولا در سطح داخلی غشاء به گیرنده پروتئینی ماده ناقل متصل می‌شوند. این پروتئین‌ها از سه جزء تشکیل شده‌اند: یک جز الفا که قسمت فعال کننده پروتئین می‌باشد و اجزای بتا و گاما که به جزء آلفا و به درون غشای سلولی مجاور گیرنده پروتئینی متصل هستند. اتصال ماده ناقل عصبی به گیرنده پروتئینی سبب جداشدن زیر واحدآلفا از زیر واحدهای بتا و گاما می‌گردد. جزء القای فعال شده می‌تواند موجب باز شدن کانال‌های یونی، فعال کردن آنزیم آدنیلیل سیکلاز[۸]و گوانیلیل سیکلاز[۹]و تولید آدنوزین مونو فوسفات حلقه‌ای[۱۰]و گوانوزین مونو فوسفات حلقه‌ای[۱۱]فعال کردن یک یا چند آنزیم درون سلولی و فعال کردن رونویسی از برخی ژن‌ها گردد. در شکل ۲-۱-۲ نمایی ساده شده از این توضیحات معلوم است.



۲-۱-۲ گیرنده‌های نورونی عصبی در محل سیناپس[۴]

گیرنده‌های تحریکی و مهاری غشاء پس سیناپسی اگر اتصال یک ماده ناقل عصبی به گیرنده اختصاصی خود سبب شود تا کانال‌های سدیمی باز شوند، هدایت از کانال‌های پتاسیمی یا کلری متوقف شود که این امر سبب دپلاریزاسیون نورون می‌گردد و با تغییراتی را در متابولیسم داخل نورون پس سیناپسی ایجاد کند که سبب تحریک فعالیت سلول گشته، تعداد گیرنده‌های غشایی تحریکی را افزایش داده یا تعداد گیرنده‌های مهاری را در غشاء کاهش دهد. این ماده یک ماده تحریکی می‌باشد. برعکس، یک ماده مهاری با اتصال به گیرنده خود می‌تواند سبب باز کردن کانال‌های کلری و ورود یون کلر به داخل سلول (هایپرپلاریزاسیون)[۱۲]و یا افزایش هدایت یون‌های

[۸] Adenylyl Cyclase
[۹] Guanylyl Cyclase
[۱۰] Cyclic adenosine monophosphate
[۱۱] Cyclic guanosine monophosphate
[۱۲] Hyperpolarization

پتاسیم به خارج از نورون گردد و همچنین می‌تواند سبب فعال شدن گیرنده‌های آنزیمی ای گردد که فعالیت سلول را مهار می‌کنند به طور کلی هر عاملی که سبب شود پتانسیل استراحت غشاء به صفر نزدیک تر شود، یک عامل تحریکی بوده و اگر پتانسیل استراحت غشاء را به سمت منفی سوق دهد، یک عامل مهاری می‌باشد.

اگر گیرنده نوروترانسمیتر، خود یک کانال یونی باشد، به آن گیرنده آینوتروپیک[۱۳]گفته می‌شود، در صورتی که گیرنده‌هایی که از طریق سیستم‌های پیام رسان ثانویه عمل می‌کنند، گیرندههای متابوتروپیک نامیده می‌شوند.

۲-۳- طبقه‌بندی

طبقه‌بندی[۱۴] یکی از روش‌های یادگیری ماشین است و برای یادگیری چگونگی تخصیص برچسب کلاس به یک نمونه ورودی، استفاده می شود. برای مثال، با طبقه بندی می توان مشخص کرد که یک ایمیل اسپم است یا خیر. برچسب‌های کلاس در اینجا اسپم و غیر اسپم هستند که باید به مقادیر عددی تبدیل شوند، یعنی اسپم را برابر صفر و غیر اسپم را برابر یک قرار می دهیم. مثال دیگر از طبقه بندی، دسته بندی کاراکترهای دست نویس به کاراکترهای موجود می باشد.

در این پژوهش از طبقه بندهای ماشین بردار پشتیبان[۱۵]، جداکننده‌ی خطی[۱۶]شبکه‌های عصبی پیچشی[۱۷]یک بعدی و دو بعدی استفاده کردیم که آن‌هارا با اختصار در ادامه توضیح داده ایم.

۲-۴- ماشین بردار پشتیبان

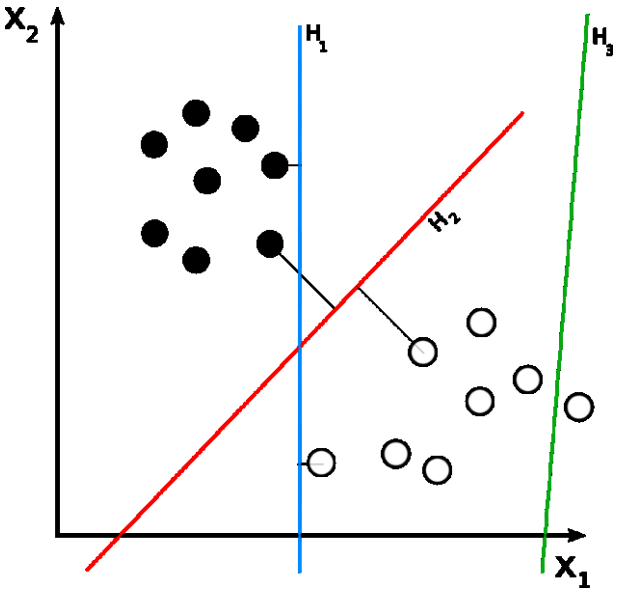
ماشین بردار پشتیبان یا SVM یکی از روش‌های یادگیری نظارت‌شده است که از آن برای طبقه‌بندی داده‌ها استفاده می‌شود[۵].

---

۱۳ Inotropic Receptor
۱۴ Classification
۱۵ Support Vector Machine (SVM)
۱۶ Linear Discriminator Analysis (LDA)
۱۷ Convolutional Neural Networks

در این روش ما به دنبال یک خط (در فضای دو بعدی) یا یک آبَرصفحه (در فضاهای با بعد بیش‌تر) هستیم که داده‌ها را با بیش‌ترین حاشیه از هم جدا کند.

شکل ۲-۴-۱ را در نظر بگیریم:



۲-۴-۱ نمونه ای از SVM خطی [۶]

خط H1 دو دسته را از هم جدا می‌کند اما حاشیه آن کم است.

خط H2 دو دسته را از هم جدا می‌کند و بیش‌ترین حاشیه را دارد.

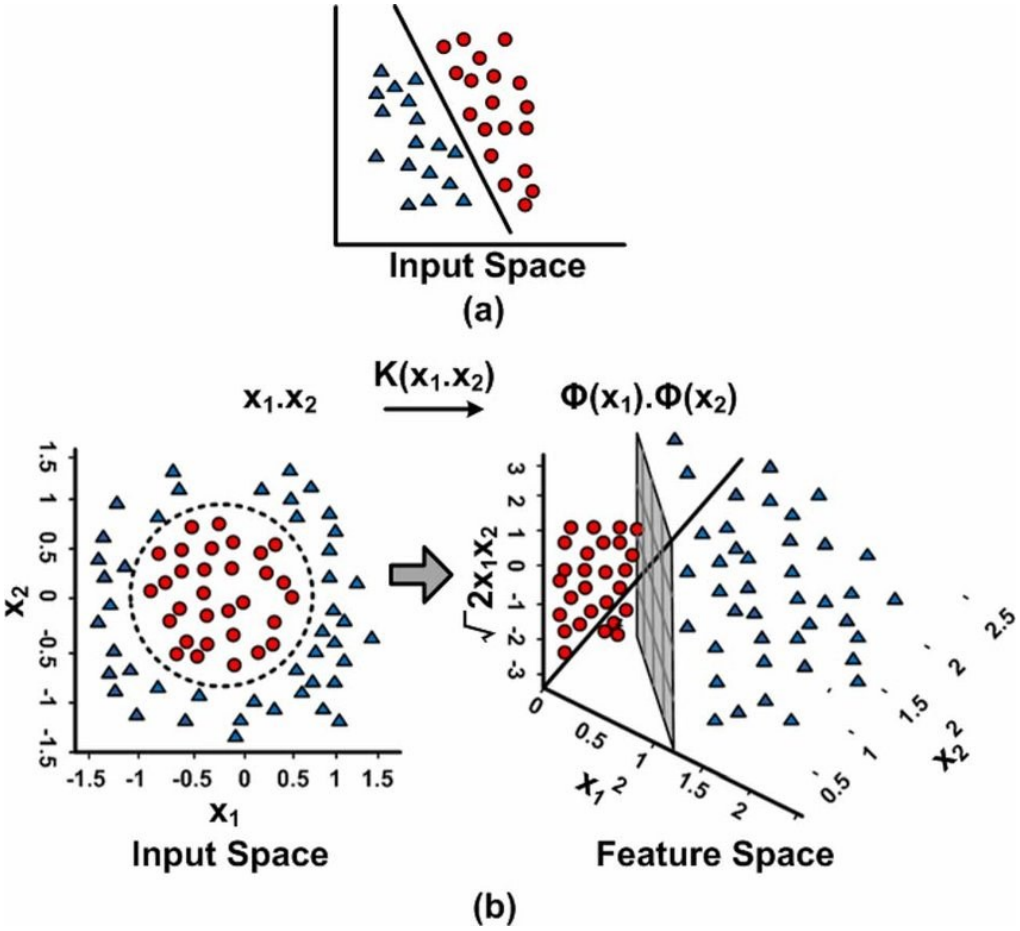
خط H3 نمی‌تواند دو دسته را از هم جدا کند.

الگوریتم SVM به دنبال یافتن خط H2 می‌باشد.

برای داده‌هایی که به طور خطی قابل تمیز نباشند می‌توان از SVM با کرنل گاوسی[۱۸] استفاده کرد. در این روش به طور کلی، مدل با ترکیب ویژگی‌های قبلی، ویژگی‌های جدید درست می‌کند و با این کار ابعاد ماتریس ویژگی‌ها

[۱۸] Gaussian Kernel

را افزایش می‌دهد. حال در این فضای با ابعاد بیش‌تر طبقه‌بندی داده‌ها راحت تر می‌شود. به این کرنل، کرنل RBF هم گفته می‌شود. شکل ۲-۴-۲ نمونه از کرنل غیر خطی را نشان می‌دهد.

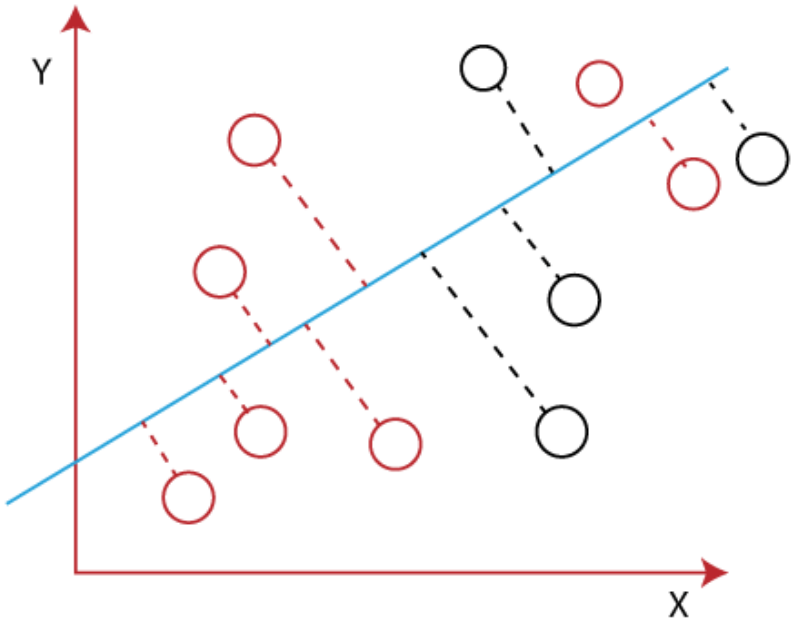


۲-۴-۲ نحوه‌ی عملکرد کرنل گاوسی [۷]

۲-۵- آنالیز افتراقی خطی[۱۹]

[۱۹] Linear Discriminant Analysis (LDA)

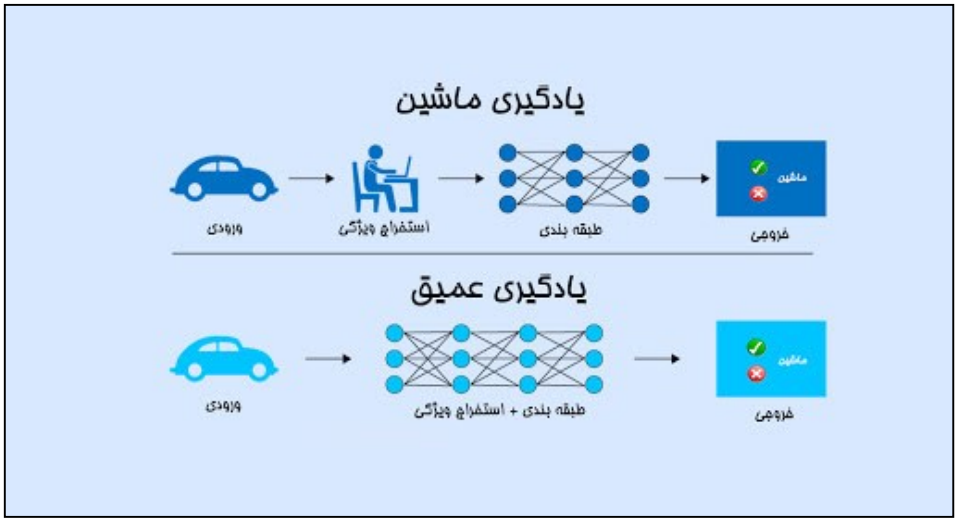
این روش به طور خلاصه، سعی می‌کند با یافتن برداری مناسب در فضای ویژگی‌ها، ویژگی‌ها را طوری روی آن بردار تصویر کند که ضمن خطی شدن داده‌ها، ویژگی‌های کلاس‌های مختلف، بیش‌ترین فاصله (بیش‌ترین واریانس) را از هم داشته باشند[۸]. شکل ۲-۵-۱ به درک این مساله کمک خواهد کرد:



۲-۵-۱ عملکرد شهودی از جدا سازی داده ها توسط LDA

۲-۶- یادگیری عمیق

یادگیری عمیق زیرشاخه‌ای از یادگیری ماشین است که از لایه‌های تبدیلات خطی برای پردازش داده‌ها استفاده می‌کند. در الگوریتم‌های یادگیری عمیق ویژگی‌ها توسط خود شبکه استخراج می‌شوند و برخلاف روش‌های کلاسیک نیازی به استخراج ویژگی به شکل دستی نیست. تفاوت عمده یادگیری عمیق با روش‌های کلاسیک در شکل ۲-۶-۱ مشهود است:



۲-۶-۱ *مقایسه ی یادگیری ماشین و یادگیری عمیق* [۹]

شبکه‌های یادگیری عمیق بر اساس نوع کاربرد، معماری‌های مختلفی دارند. در این پژوهش از شبکه‌ی عصبی پیچشی استفاده شده و در زیرفصل بعد آن را به اختصار توضیح می‌دهیم.

۲-۷- شبکه‌ی عصبی عمیق پیچشی

شبکه‌های عصبی عمیق پیچشی یکی از الگوریتم‌های یادگیری عمیق هستند که از تعداد زیادی لایه کانولوشنی برای پردازش داده‌ها استفاده می‌کنند. این لایه‌ها (فیلترها) در واقع وزن‌های پنجره‌ای هستند که در هر مرحله بر روی ورودی می‌لغزند و عمل کانولوشن را انجام می‌دهند[۱۰].

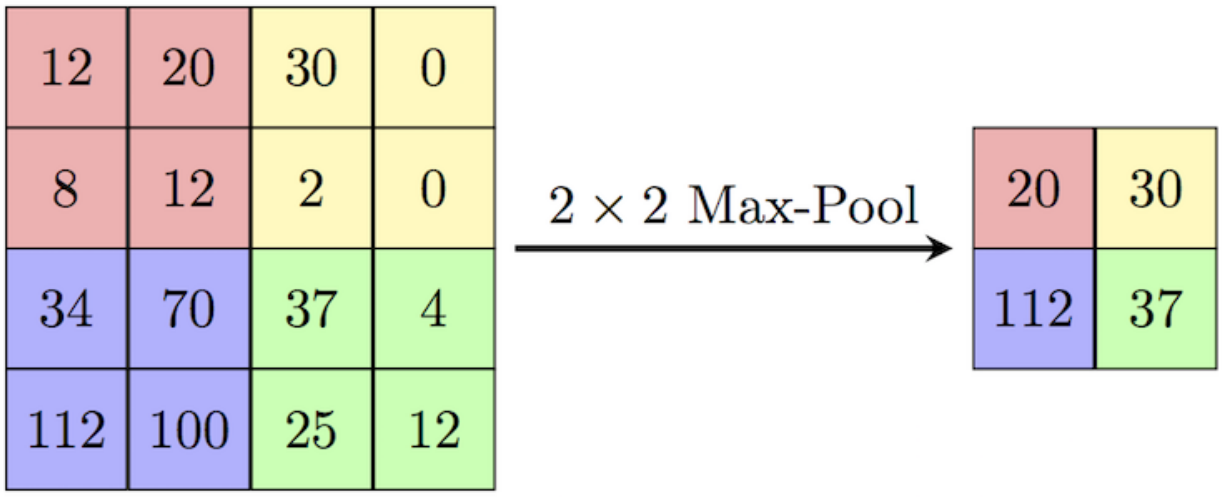
در هر مرحله می‌توان به جای یک فیلتر، چند فیلتر قرار داد و تعداد خروجی را بیش‌تر کرد.

این لایه‌های کانولوشنی در واقع همان کار استخراج ویژگی را انجام می‌دهند و در هر مرحله ویژگی‌های سطح بالاتری را محاسبه می‌کنند.

معمولا بعد از انجام عمل کانولوشن، بر روی خروجی، تابع فعالسازی[۲۰] ReLU اعمال می‌شود و سپس به لایه کاهش‌اندازه[۲۱] داده می‌شود. لایه کاهش‌اندازه با هدف کاهش‌اندازه داده‌ها و حجم محاسبات به کار می‌رود. دو نوع

[۲۰] Activation Function
[۲۱] Pooling

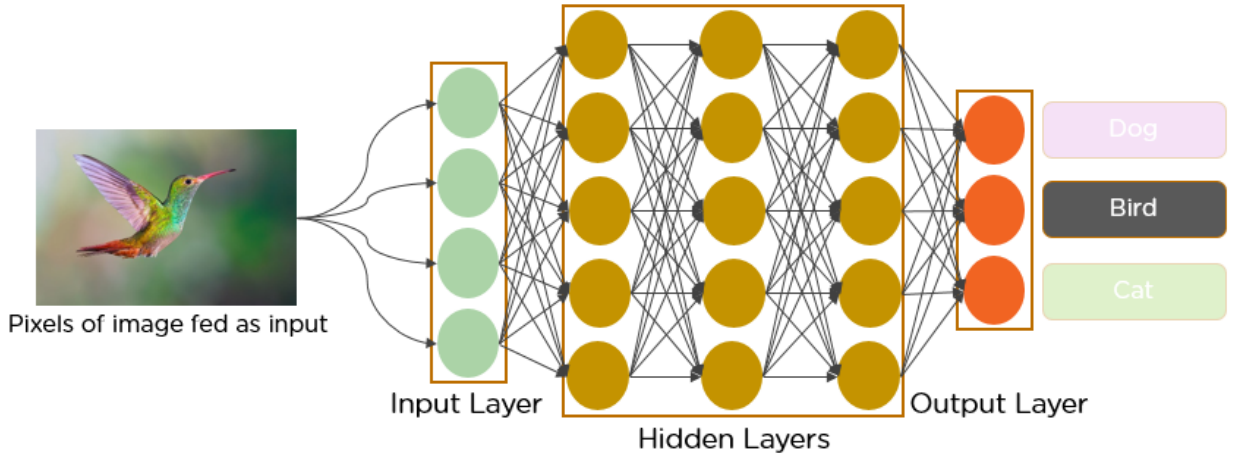
از لایه‌های کاهش‌اندازه پرکاربرد عبارتند از Max Pooling و Average Pooling. در شکل ۲-۷-۱ مفهوم لایه کاهش‌اندازه نشان داده شده است:



۲-۷-۱ مثالی از روش max pooling [*۱۱*]

همانطور که می‌بینیم بر روی داده اولیه یک عملیات Max Pooling با سایز ۲در۲ و گام ۲ انجام داده‌ایم و سایز داده از ۴در۴ به ۲در۲ کاهش یافته است. لایه Average Pooling هم مشابه همین است فقط به جای مقدار بیشینه، میانگین آنها را قرار می‌دهیم.

بعد از چند مرحله عملیات کانولوشن و کاهش‌اندازه، داده‌ها را به یک بردار تبدیل می‌کنیم و بعد مانند یک شبکه عصبی معمولی و با استفاده از لایه‌های تمام متصل عملیات طبقه‌بندی داده‌ها انجام می‌شود.به طور شهودی ساختار یک شبکه عصبی پیچشی عمیق به شکل ۲-۷-۱ است.



۲-۷-۱ شبکه ی عمیق پیچشی [۱۲]

شبکه‌های CNN برای داده‌های تصویری بسیار مناسب هستند.

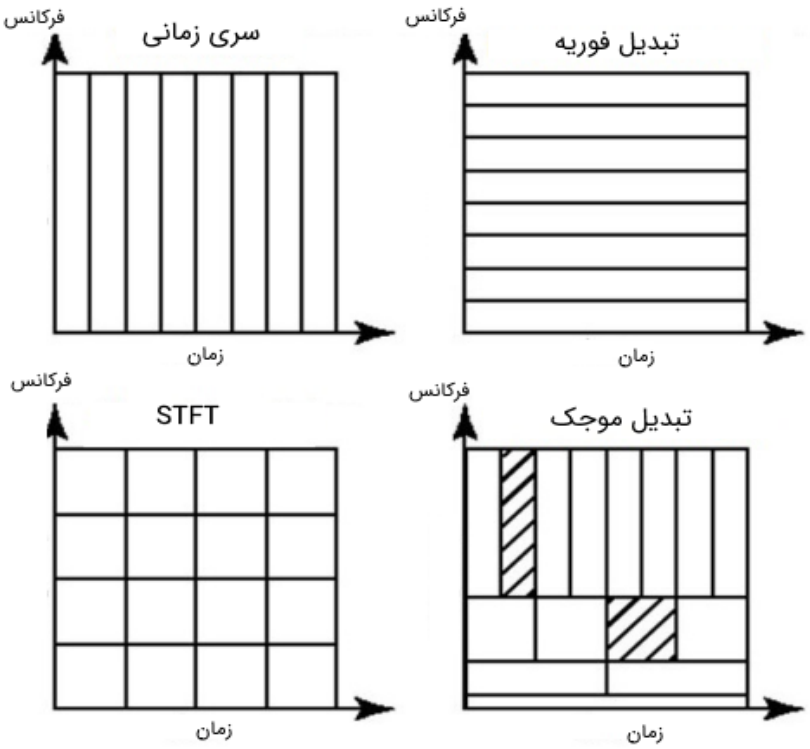
شبکه‌های CNN یک بعدی دقیقا مانند دو بعدی قابل تعریف‌اند. تنها کافیست به جای کرنل‌های دو بعدی، از کرنل‌های یک بعدی روی سیگنال‌های یک بعدی استفاده کنیم.

۲-۸- تبدیل موجک

تبدیل موجک[۲۲] یکی از تبدیلات مهم ریاضی است که در حوزه‌های مختلف علوم کاربرد دارد. ایده اصلی تبدیل موجک این است که بر ضعف‌ها و محدودیت‌های موجود در تبدیل فوریه غلبه کند. این تبدیل را بر خلاف تبدیل فوریه، می‌توان در مورد سیگنال‌های غیر ایستا و سیستم‌های دینامیک نیز مورد استفاده قرار داد [۱۳].

برای درک شهودی و مناسب تبدیل موجک و مقایسه آن با تبدیل فوریه شکل ۲-۸-۱ را در نظر می‌گیریم.

[۲۲] Wavelet Transform



۲-۸-۱ تبدیل های زمان فرکانس مختلف [۱۴]

همانطور که ملاحظه می‌شود در سری زمانی، اطلاعات در تکه‌های زمانی وجود دارد اما اطلاعات فرکانسی نداریم. همینطور بر عکس این داستان برای تبدیل فوریه صادق است.

در تبدیل فوریه‌ی کوتاه که نوع دیگر و ساده‌ی تبدیل‌های زمان-فرکانس هست، زمان ما به بازه‌های برابر تقسیم شده و وزن برابری به بازه‌های فرکانسی داده شده.

در تبدیل موجک اما بازه‌های فرکانسی به گستره‌های دلخواه ما تقسیم شده‌اند و اگر دقیق تر بگوییم هر بازه‌ی فرکانسی دو برابر بازه‌ی فرکانسی قبلی خودش دقیق تر است.

۲-۹- اعتبارسنجی متقابل[۲۳]

[۱۵]هدف در اعتبارسنجی متقابل، دستیابی به مدلی است که تعداد پارامترهای آن بهینه باشد. یعنی پیدا کردن مدلی است که دچار بیش‌برازش نباشد. برای برای دستیابی به این هدف در آموزش ماشین معمولا داده‌ها را به دو قسمت تفکیک می‌کنند.

- **قسمت داده‌های آموزشی:** از این بخش از داده‌ها به منظور ایجاد مدل و برآورد پارامترهای آن استفاده می‌شود.
- **قسمت داده‌های آزمایشی :** این قسمت از داده‌ها برای بررسی کارایی مدل استفاده می‌شود. اهمیت این بخش از داده‌ها در این نکته است که این مشاهدات شامل مقدارهای متغیرهای مستقل ها و پاسخی هستند که در مدل به کار نرفته ولی امکان مقایسه مقدار پیش‌بینی شده را با مقدار واقعی به ما می‌دهند. البته توجه داریم که این داده‌ها مدل را تحت تاثیر قرار نداده‌اند، پس در تعیین پارامترهای مدل نقشی نداشته و فقط برای ارزیابی مدل به کار می‌روند.

با توجه به تفکیکی که برای این دو گروه داده در نظر گرفته شد، مدل‌سازی فقط براساس بخش داده‌های آموزشی خواهد بود. ولی در روش اعتبارسنجی متقابل که از این به بعد آن را به اختصار «CV» می‌نامیم، طی یک فرآیند تکرار شونده، قسمت داده‌های آموزش که به منظور مدل‌سازی به کار می رود، خود به دو بخش تفکیک می‌شود. در هر بار تکرار فرآیندCV ، بخشی از داده‌ها برای آموزش و بخشی دیگر برای آزمایش مدل به کار می‌رود. به این ترتیب این فرآیند یک روش بازنمونه‌گیری به منظور برآورد خطای مدل محسوب می‌شود. شکل ۲-۹-۱ به درک این مفهوم کمک خواهد کرد.

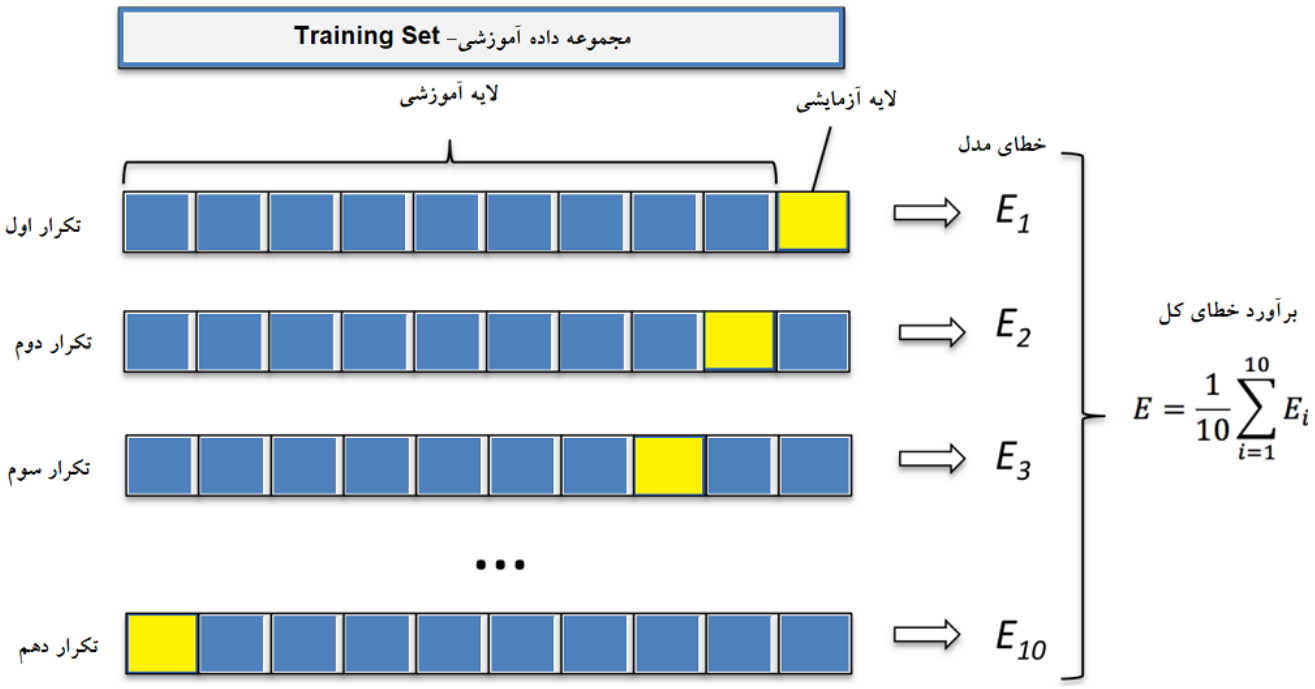
[۲۳] Cross Validation

۲-۹-۱ تصویری از مراحل تکرار فرایند CV [۱۶]

۲-۱۰- روش ارزشیابی K-fold

اگر مجموعه داده‌های آموزشی را به طور تصادفی به k زیرنمونه یا لایه[۲۴] با حجم یکسان تفکیک کنیم، می‌توان در هر مرحله از فرایندCV ، تعداد k-1 از این لایه‌ها را به عنوان مجموعه داده آموزشی و یکی را به عنوان مجموعه داده اعتبارسنجی در نظر گرفت. تصویر زیر، مراحل روش k-Fold را به خوبی نشان می‌دهد. مشخص است که با انتخابk=10 ، تعداد تکرارهای فرآیند CV برابر با ۱۰ خواهد بود و دستیابی به مدل مناسب به سرعت امکان‌پذیر می‌شود. برای درک بهتر شکل ۲-۱-۱۰ را می‌بینیم.

[۲۴] Fold



۲-۱۰-۱ شمایی کلی از k-fold [۱۷]

با برابر قرار دادن k با داده‌های مساله به یک ارزشیابی‌ای دقیق در داده ی کم ولی زمانبر به اسم Leave One Out می‌رسیم که در این پروژه از آن استفاده شده.

۲-۱۱- جمع بندی

در این فصل به توضیح بعضی از مفاهیم و ابزار مورد استفاده در این پژوهش پرداختیم و انواع نورون، مدل‌های یادگیری ماشین و برخی دیگر مفاهیم ریاضی مورد استفاده در پروژه را با به اختصار توضیح دادیم. تفاوت های یادگیری ماشین و یادگیری عمیق را مورد بررسی قرار دادیم و از ابزاری که در مسیر این پروژه به ما کمک می‌کنند پرده برداری کردیم.

حال با این دانش و ابزاری که داریم، آماده‌ایم تا نگاه مختصری به مقاله ها و کارهای مشابهی که پیش تر انجام شده بیندازیم.

فصل ۳- مروری بر پیشینه‌ی پژوهش

همانطور که پیش تر گفته شد، موضوع طبقه بندی نورون‌های مهاری و تحریکی موضوعی است که کم‌تر به آن پرداخته شده است. هرچند تعداد انگشت شماری مقاله با موضوع‌های مشابه وجود دارد.

۳-۱- پژوهش ها

برای مثال در پژوهشی که پیش تر توسط خادم و اوغازیان انجام شده [۱۸] نورون‌های برچسب خورده‌ی نوعی میمون و موش با استفاده از استخراج ویژگی از اسپایک نورون‌ها و روش‌های یادگیری ماشین غیر عمیق مورد بررسی قرار گرفته‌اند و به صحت بالای ۹۰ درصد دست پیدا کرده‌اند. در مدل های یادگیری ماشین این‌چنینی نیاز به استخراج ویژگی از داده وجود دارد. در نتیجه اولا نیاز به پیش‌پردازش بیشتر روی داده های اولیه و حذف نویز است و دوما پردازش و انتخاب ویژگی درست می‌تواند فرایندی پیچیده و زمانبر باشد. در اسپایک ها ویژگی های طول اسپایک و ارتفاع پیک می‌توانند ویژگی های مناسبی باشند.

همچنین در مقاله ای دیگر که در سال ۲۰۱۸ در IEEE منتشر شد[۱۹]، با استفاده از شبکه‌های عمیق پیچشی به طبقه بندی انواع نورون پرداخته شد و به صحتی نزدیک به ۹۳ درصد هم رسیدند. اما ضعف کار آن‌ها آن جا است که داده‌های مورد استفاده‌ی آن‌ها در مقاله، داده‌های شبیه‌سازی شده با پایتون هستند و نه داده‌های برچسب گذاری شده‌ی واقعی.

ما در این پژوهش قصد داریم طبقه بندی را بدون استخراج ویژگی و روی داده‌های واقعی انجام دهیم. مزیت استفاده از یادگیری عمیق نسبت به یادگیری ماشین این است که دیگر لازم نیست زمان صرف پردازش سیگنال داده ها کنیم و همچنین در داده های زیاد احتمال اینکه صحت مدل یادگیری عمیق بیشتر از مدل های کلاسیک باشد بیشتر است. در را بطه با داده های شبیهسازی شده هم می‌توان گفت که بدیهی است که داده های واقعی ارزش بیشتری دارند و تمرکز ما هم روی داده های واقعی است و از این نظر نسبت به کارهای پیشین نوآوری داریم.

۳-۲- جمع بندی

در این فصل توضیح مختصری راجع به کارهای پیشین انجام شده در راستای طبقه بندی نورون های مهاری و تحریکی پرداختیم. با مقایسه‌ای مناسب استدلال کردیم پژوهش های انجام شده ی پیشین چه ضعف‌های احتمالی‌ای داشتند و استدلال کردیم که این نقاط ضعف ممکن است به چه مشکل هایی منجر شوند.

در فصل آینده قصد داریم تا با ارایه‌ی روش خود، این نقاط ضعف را جبران کنیم و نتایج بدست آمده را مقایسه کنیم.

فصل ۴- رویکرد پیشنهادی

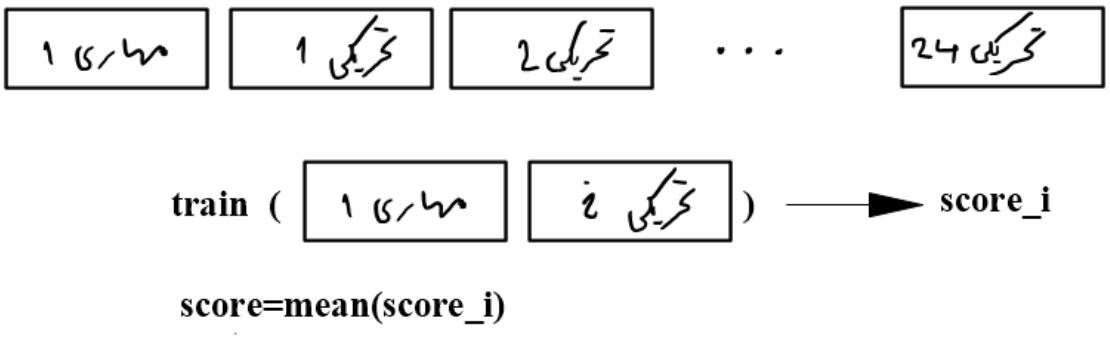
۴-۱- داده‌ی مورد استفاده

داده‌های ما از سایت آزمایشگاه آقای بوزاکی برداشته شده‌اند. داده‌هایی که در اختیار داریم شامل اسپایک‌های ۶۲۵ نورون با فرکانس نمونه برداری ۲۵ کیلوهرتز از ناحیه هیپوکمپ نوعی موش هستند که برچسب تحریکی یا مهاری آن‌ها مشخص است [۲۰]. هر اسپایک شامل ۳۲ نمونه زمانی است. ۲۶ تا از اسپایک‌ها مهاری و باقی آن‌ها تحریکی می‌باشد.

۴-۲- روش پیشنهادی

۴-۲-۱- متعادل کردن داده‌ها

همان طور که در بالا اشاره شد داده‌های دو دسته متعادل نیستند و این یک ضعف است برای ترین کردن مدل، بنابر این باید روشی ارایه دهیم تا ابتدا این مشکل حل شود. برای حل این مشکل توجه شما را به شکل ۴-۲-۱ جلب میکنم:



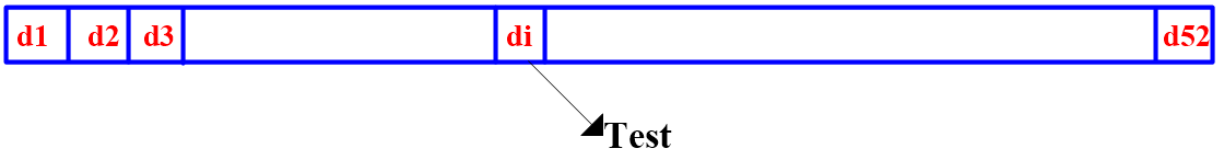
۴-۲-۱ شمایی از کلیت دسته بندی داده ها برای آموزش

داده‌ها را مطابق شکل بالا در بسته‌های ۲۶ داده ای دسته بندی می‌کنیم. برای مثال بسته‌ی مهاری۱ شامل ۲۶ اسپایک مهاری می‌باشد. باقی ۶۰۰ داده‌ی تحریکی هم در ۲۴ بسته ی۲۶ تایی دسته بندی می‌شوند. حال هر یک از دسته‌های تحریکی را با دسته‌ی مهاری ۲۶ تایی می‌گیریم و به مدل یادگیری ماشین می‌دهیم تا آموزش ببینند. علت کارمان این است که داده‌های دو کلاس به طور متوازن در فرایند آموزش شرکت کنند. بعد از این که داده‌های

هر دسته را ترین کدیم و آن را ارزشیابی کردیم، ۲۴ امتیاز به دست آمده را میانگین می‌گیریم و به عنوان امتیاز نهایی نمایش می‌دهیم.

۴-۲-۲- روش ارزشیابی داده‌ها

همانطور که در فصل دوم اشاره کردیم، برای ارزشیابی داده‌ها از روش leave one out استفاده کردیم، که دقیق ترین روش برای داده‌های کم است. بی این شکل که در هر دسته بندی ۵۲ تایی (۲۶ تا مهاری و ۲۶ تا تحریکی) هر بار یک داده به عنوان تست و ۵۱ داده‌ی دیگر برای آموزش کنار گذاشته می‌شوند. پس هر دسته‌ی ۵۲ تایی بعد از ۵۲ بار آموزش و تست، صحتی به ما می‌دهد که میانگین صحت‌های پیشین است و آن را به عنوان صحت مدل در نظر می‌گیریم. حال طبق توضیحات قسمت بالا از ۲۴ صحت بدست آمده میانگین می‌گیریم و به عنوان صحت نهایی مدل ارایه می‌دهیم. این مساله در شکل ۴-۲-۱ به نمایش گذاشته شده است.



۴-۲-۱ نمایی از روش leave one out - در این روش هر بار یک داده ی di از بین ۵۲ داده برای تست انتخاب می‌شود و باقی برای آموزش. پس از ۵۲ بار به ۵۲ عدد ۰ و ۱ میرسیم که هر یک صحت هر داده ی تست است. پس از میانگین گرفتن از این ۵۲ مقدار به صحت نهایی میرسیم.

۴-۲-۳- آموزش مدل‌ها و تصمیمگیری نهایی

برای آموزش مدل‌ها و پیاده سازی این پروژه تماما از پایتون استفاده شده است. در ادامه به بررسی بعضی از پارامترهای استفاده شده در این تابع می‌پردازیم و نهایتا کد برنامه را بررسی می‌کنیم.

برای مدل یک بعدی LDA از پارامترهای دیفالت تابع استفاده شده و چیزی را تغییر ندادیم. عملکرد روش هم در فصل دوم توضیح داده شده است که روشی خطی است و انتظار صحت بالایی از آن نداریم چون داده‌ها لزوما خطی نیستند. در این مدل ورودی‌های برچسبگذاری شدهبه صورت سیگنال اسپایک وارد می‌شوند و مدل آموزش می‌بیند.

در مدل یک بعدی SVM بر خلاف مدل قبل به جای کرنل دیفالت تابع که خطی است، از کرنل گاوسی برای داده‌های غیر خطی استفاده کردیم که انتظار میرود صحت بالایی دهد. در این مدل هم همانند مدل قبل اسپایک‌های برچسب خورده‌ی یک بعدی به صورت خام به تابع داده می‌شوند.

برای مدل CNN یک بعدی اما، نیاز به توضیحات بیش‌تری است. در این مدل از ۳ لایه‌ی میانی به ترتیب پیچشی[۲۵]،پیچشی،دنس [۲۶]استفاده شده است و و نهایتا به یک لایه دنس ۲ نورونی ختم می‌شود تا برچسب گذاری به درستی انجام شود. لایه‌ی پیچشی او شامل ۲۵۶ لایه و یک کرنل ۱در۳ می‌باشد و تابع relu که پیش تر توضیح داده شده بود روی آن اعمال می‌شود. سپس یک max pooling به‌اندازه‌ی ۲ روی آن اعمال می‌شود که کارش این است تا از هر دو داده‌ی خروجی شبکه پیچشی، ماکسیمم را انتخاب و دیگری را حذف کند. این کار علاوه بر کم کردن عملیات، رابطه‌های غیر خطی را هم با صحت بیش‌تری پیدا می‌کند. پس از یک لایه پیچشی دیگر با ۱۲۸ لایه و انازه‌ی ۳ و تابع relu، داده‌ها را به یک لایه‌ی دنس با ۶۴ راس می‌دهیم و نهایتا برای جلوگیری از overfitting از تابع dropout با نرخ ۰,۳ استفاده می‌کنیم که کارش این است به صورت رندوم یکسری راس را حذف می‌کند تا overfitting نداشته باشیم. در نهایت هم یک لایه دنس با دو راس و تابع softmax اعدادی بین ۰ و ۱ به ما می‌دهند که در واقع امتیاز هر کلاس را مشخص می‌کند. همچنین از ۱۰ epoch برای آموزش استفاده کردیم که با این معناست ۳ بار فرایند آموزش روی داده‌ی ثابت انجام شده. تفاوت دیگر مدل‌های پیچشی، برچسبگذاری آنهاست. هنگام آموزش به جای برچسب‌های ساده‌ی ۰ و ۱ آن‌ها را با تابع to_categorical به

[۲۵] Convolutional
[۲۶] Dense

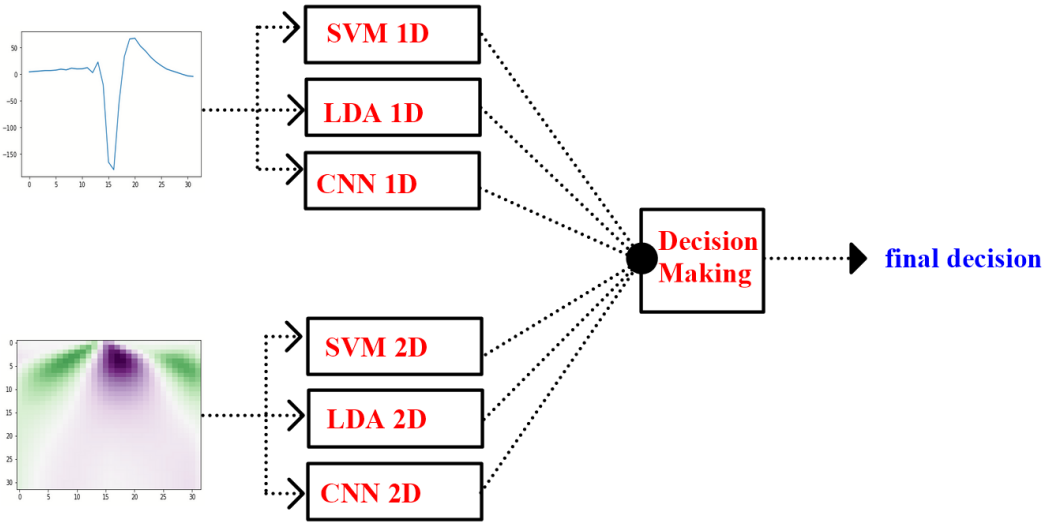
برچسب‌هایی با شمایل دیگر تبدیل می‌کنیم تا در خروجی دنس ۲ تایی بتوانیم امتیاز هر کلاس را راحت تر ببینیم.

برای مدل‌های ۲ بعدی از ورودی‌های اسپایک، تبدیل موجک با پارامترهای اولیه‌ی خود تابع گرفتیم. طول تبدیل موجک را هم همانند طول اسپایک ۳۲ گرفتیم و در نتیجه به یک ماتریس ۳۲در۳۲ رسیدیم.

برای مدل LDA و SVM ماتریس ۳۲در۳۲ را باز کردیم و به یک ماتریس ۱در۱۰۲۴ تبدیل کردیم و به مدل‌ها دادیم و پارامترها دقیقا مانند قسمت قبل بودند.

برای CNN دو بعدی هم دقیقا مانند قسمت یک بعدی عمل کردیم با این تفاوت که به جای کرنل‌های ۱در۳ از کرنل‌های ۳در۳ استفاده کردیم.

ما در این پژوهش از ۶ مدل استفاده کردیم که ۳ تا از آن‌ها ورودی خام اسپایک و ۳ تای دیگر تبدیل موجک اسپایک را به عنوان ورودی می‌گیرند. برای تست کردن داده ای نا شناخته، بعد از اظهار نتیجه توسط هر ۶ مدل، با توجه به وزندهی به تصمیم هر یک (که بر اساس صحت آن‌ها بوده) تصمیم نهایی اعلام می‌شود. برای درک بهتر موضوع فرایند را در شکل ۴-۳-۱ به نمایش میگذاریم.



۴-۳-۱ شمایی از طرز تصمیم گیری مدل بر مبنای مدل های سازنده‌ی آن

صحت هر مدل و وزن دهی آن‌ها در فصل نتایج آمده است.

در ادامه کدهای نوشته شده در پایتون را می‌بینیم.

۴-۳- کدهای مورد استفاده در برنامه

۴-۳-۱- کدهای مربوط مدل‌های یک بعدی

در این قسمت مدل های CNN، LDA و SVM برای داده‌ی یک بعدی مورد آموزش و تست قرار گرفته‌اند.

```
for i in range(0,24): #(0,24)
    print(i,"************ \n ")
    X1D=np.append(inh[0:26],exc[i*26:i*26+26],axis=0)
    y1D=np.append(np.zeros(26),np.ones(26))
    X1D, y1D = sklearn.utils.shuffle(X1D, y1D, random_state=4)

    y1D_n = tf.keras.utils.to_categorical(y1D)

    ## K-fold
    kfold = KFold(n_splits=52) #k=52
    fold_no=1

    acc_1D_fold=[]
    acc_LDA1_fold=[]
    acc_SVM1_fold=[]

    for train,test in kfold.split(X1D, y1D_n):

        model=tf.keras.models.Sequential([
        tf.keras.layers.Conv1D(256,3,activation = "relu" , input_shape = (32,1)) ,
        tf.keras.layers.MaxPooling1D(pool_size=2) ,
        tf.keras.layers.Conv1D(128,3,activation = "relu") ,
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(64,activation= "relu") ,
        tf.keras.layers.Dropout(0.3,seed = 43),
        tf.keras.layers.Dense(2,activation= "softmax")
        ])

        model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

        # Fit data to model
        history = model.fit(X1D[train], y1D_n[train],
                  epochs=10, verbose=0)

        # Generate generalization metrics
        scores = model.evaluate(X1D[test], y1D_n[test], verbose=0)
        acc_1D_fold.append(scores[1]*100)

        clf = LinearDiscriminantAnalysis()
        clf.fit(X1D[train], y1D[train])
        scores=clf.score(X1D[test], y1D[test])
        acc_LDA1_fold.append(scores*100)

        regr = svm.SVC(C=0.2,kernel='rbf')
        regr.fit(X1D[train], y1D[train])
        scores=regr.score(X1D[test], y1D[test])
        acc_SVM1_fold.append(scores*100)

        # Increase fold number
        fold_no = fold_no + 1

    acc_1D.append(sum(acc_1D_fold)/len(acc_1D_fold))
    acc_LDA1.append(sum(acc_LDA1_fold)/len(acc_LDA1_fold))
    acc_SVM1.append(sum(acc_SVM1_fold)/len(acc_SVM1_fold))

print("acc_1D= ", sum(acc_1D)/len(acc_1D), "\n acc_LDA1= ",
      sum(acc_LDA1)/len(acc_LDA1), "\n acc_SVM1= ", sum(acc_SVM1)/len(acc_SVM1))
```

۴-۳-۲- کدهای مربوط به مدل‌های دو بعدی

در این قسمت مدل های CNN، LDA و SVM برای داده‌ی دو بعدی مورد آموزش و تست قرار گرفته‌اند.

```
for i in range(0,24):
    print(i,"********** \n ")
    X2D=np.append(inh2[0:25],exc2[i*26:i*26+25],axis=0)
    y2D=np.append(np.zeros(25),np.ones(25))
    X2D, y2D = sklearn.utils.shuffle(X2D, y2D, random_state=4)

    y2D_n = tf.keras.utils.to_categorical(y2D)

    ## K-fold
    kfold = KFold(n_splits=52) #k=52
    fold_no=1

    acc_2D_fold=[]
    acc_LDA2_fold=[]
    acc_SVM2_fold=[]



    for train,test in kfold.split(X2D, y2D_n):

        model=tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(256,(3,3),activation = "relu" , input_shape = (32,32,1)) ,
        tf.keras.layers.MaxPooling2D(pool_size=2) ,
        tf.keras.layers.Conv2D(128,(3,3),activation = "relu") ,
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(64,activation= "relu") ,
        tf.keras.layers.Dense(2,activation= "softmax")
        ])

        model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

        # Fit data to model
        history = model.fit(X2D[train], y2D_n[train],
                  epochs=10, verbose=0)

        # Generate generalization metrics
        scores = model.evaluate(X2D[test], y2D_n[test], verbose=0)
        acc_2D_fold.append(scores[1]*100)

        clf = LinearDiscriminantAnalysis()
        clf.fit(X2D.reshape(50,1024)[train], y2D[train])
        scores=clf.score(X2D.reshape(50,1024)[test], y2D[test])
        acc_LDA2_fold.append(scores*100)

        regr = svm.SVC(C=0.99,kernel='rbf')
        regr.fit(X2D.reshape(50,1024)[train], y2D[train])
        scores=regr.score(X2D.reshape(50,1024)[test], y2D[test])
        acc_SVM2_fold.append(scores*100)

        # Increase fold number
        fold_no = fold_no + 1

    acc_2D.append(sum(acc_2D_fold)/len(acc_2D_fold))
    acc_LDA2.append(sum(acc_LDA2_fold)/len(acc_LDA2_fold))
    acc_SVM2.append(sum(acc_SVM2_fold)/len(acc_SVM2_fold))

print("acc_2D= ", sum(acc_2D)/len(acc_2D), "\n acc_LDA2= ",
      sum(acc_LDA2)/len(acc_LDA2), "\n acc_SVM2= ", sum(acc_SVM2)/len(acc_SVM2))
```

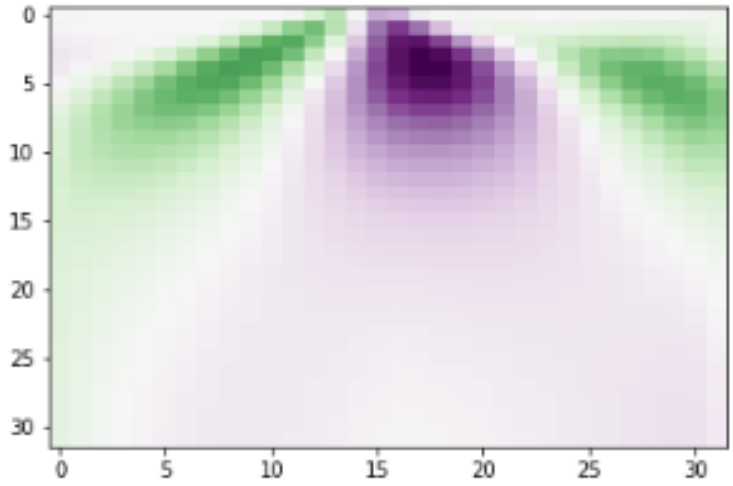
۴-۳-۳- کدهای مربوط به تبدیل موجک

در این بخش از کد با استفاده از اسپایک های در دسترس، تبدیدل موجک آن‌ها را در قالب تصویری ۳۲ در ۳۲ در آورده‌ایم تا به عنوان ورودی مدل های دو بعدیملن استفاده کنیم. همچنین نمونه از تصویر تبدیل ویولت نورونی تحریکی قابل مشاهده است.

In [40]:
```
widths = np.arange(1, 33)
cwtmatr = signal.cwt(exc[18], signal.ricker, widths)
plt.imshow(cwtmatr, cmap='PRGn', aspect='auto',
           vmax=abs(cwtmatr).max(), vmin=-abs(cwtmatr).max())
plt.show()

inh2=np.zeros((26,32,32))
for i in range(0,26):
    inh2[i,:,:]= signal.cwt(inh[i], signal.ricker, widths)

exc2=np.zeros((24*26,32,32))
for i in range(0,24*26):
    exc2[i,:,:]= signal.cwt(exc[i], signal.ricker, widths)
```



۴-۳-۴- کدهای مربوط به بخش GUI

در ادامه کدهایی که مربوط به راه‌اندازی بخش نرم افزاری است را می‌بینیم.

تمام بخش‌هایی از کد که با def آغاز شده‌اند برای تعریف عملگرهای دکمه های برنامه می‌باشند که هر یک کاری می‌کند.

```
def findName(filename):
    name=''
    for i in filename[::-1]:
        if i=='/':
            return name[:-4]
        name=i+name


def plot1D():

    # the figure that will contain the plot
    fig = Figure(figsize = (7,7),
                 dpi = 60)


    # adding the subplot
    plot1 = fig.add_subplot(111)

    # plotting the graph
    plot1.plot(mat)

    # creating the Tkinter canvas
    # containing the Matplotlib figure
    pw=tk.Tk()
    pw.geometry('450x500')
    pw.title('spike figure')
    pw.iconbitmap('./kntu.ico')
    canvas = FigureCanvasTkAgg(fig,
                               master = pw)
    canvas.draw()

    # placing the canvas on the Tkinter window
    canvas.get_tk_widget().pack()

    # creating the Matplotlib toolbar
    toolbar = NavigationToolbar2Tk(canvas,
                                   pw)
    toolbar.update()

    # placing the toolbar on the Tkinter window
    canvas.get_tk_widget().pack()
```

```
def plot2D():
    # the figure that will contain the plot
    fig = Figure(figsize = (7,7),
                 dpi = 60)


    # adding the subplot
    plot2 = fig.add_subplot(111)

    # plotting the graph
    widths = np.arange(1, 33)
    global cwtmatr
    cwtmatr = signal.cwt(mat, signal.ricker, widths)
    plot2.imshow(cwtmatr, cmap='PRGn', aspect='auto',
           vmax=abs(cwtmatr).max(), vmin=-abs(cwtmatr).max())

    # creating the Tkinter canvas
    # containing the Matplotlib figure
    pw=tk.Tk()
    pw.geometry('450x500')
    pw.title('spike figure')
    pw.iconbitmap('./kntu.ico')
    canvas = FigureCanvasTkAgg(fig,
                               master = pw)
    canvas.draw()

    # placing the canvas on the Tkinter window
    canvas.get_tk_widget().pack()

    # creating the Matplotlib toolbar
    toolbar = NavigationToolbar2Tk(canvas,
                                   pw)
    toolbar.update()

    # placing the toolbar on the Tkinter window
    canvas.get_tk_widget().pack()


def ImportAction():
    global mat
    filename = filedialog.askopenfilename()
    mat = sio.loadmat(filename)
    mat=mat[findName(filename)].ravel()

def LDA1():
    global model
    model = pickle.load(open("LDA1", 'rb'))
    a=model.predict(mat.reshape(1,32))
    if a[0]==1:
        tk.messagebox.showinfo(title="یک بعدی LDA", message="نورون تحریکی است")
    else:
        tk.messagebox.showinfo(title="یک بعدی LDA", message="نورون مهاری است")


def SVM1():
    global model
    model = pickle.load(open("SVM1", 'rb'))
    a=model.predict(mat.reshape(1,32))
    if a[0]==1:
        tk.messagebox.showinfo(title="یک بعدی SVM", message="نورون تحریکی است")
    else:
        tk.messagebox.showinfo(title="یک بعدی SVM", message="نورون مهاری است")
```

```
def CNN1():
    global model
    model=load_model("CNN1.h5")
    a=model.predict(mat.reshape(1,32))
    if a[0,0]>a[0,1]:
        tk.messagebox.showinfo(title="یک بعدی CNN", message="نورون مهاری است")
    else:
        tk.messagebox.showinfo(title="یک بعدی CNN", message="نورون تحریکی است")

def LDA2():
    global model
    model = pickle.load(open("LDA2", 'rb'))
    a=model.predict(cwtmatr.reshape(1,32*32))
    if a[0]==1:
        tk.messagebox.showinfo(title="دو بعدی LDA", message="نورون تحریکی است")
    else:
        tk.messagebox.showinfo(title="دو بعدی LDA", message="نورون مهاری است")


def SVM2():
    global model
    model = pickle.load(open("SVM2", 'rb'))
    a=model.predict(cwtmatr.reshape(1,32*32))
    if a[0]==1:
        tk.messagebox.showinfo(title="دو بعدی SVM", message="نورون تحریکی است")
    else:
        tk.messagebox.showinfo(title="دو بعدی SVM", message="نورون مهاری است")


def CNN2():
    global model
    model=load_model("CNN2.h5")
    a=model.predict(cwtmatr.reshape(1,32,32))
    if a[0,0]>a[0,1]:
        tk.messagebox.showinfo(title="دو بعدی CNN", message="نورون مهاری است")
    else:
        tk.messagebox.showinfo(title="دو بعدی CNN", message="نورون تحریکی است")
```

```
def dec():
    global model

    model = pickle.load(open("LDA1", 'rb'))
    a=model.predict(mat.reshape(1,32))
    if a[0]==1:
        a1=87
    else:
        a1=-87

    model = pickle.load(open("SVM1", 'rb'))
    a=model.predict(mat.reshape(1,32))
    if a[0]==1:
        a2=92
    else:
        a2=-92

    model=load_model("CNN1.h5")
    a=model.predict(mat.reshape(1,32))
    if a[0,0]>a[0,1]:
        a3=-94
    else:
        a3=94

    model = pickle.load(open("LDA2", 'rb'))
    a=model.predict(cwtmatr.reshape(1,32*32))
    if a[0]==1:
        b1=87
    else:
        b1=-87

    model = pickle.load(open("SVM2", 'rb'))
    a=model.predict(cwtmatr.reshape(1,32*32))
    if a[0]==1:
        b2=93
    else:
        b2=-93

    model=load_model("CNN2.h5")
    a=model.predict(cwtmatr.reshape(1,32,32))
    if a[0,0]>a[0,1]:
        b3=-93
    else:
        b3=93

    if a1+a2+a3+b1+b2+b3>0:
        tk.messagebox.showinfo(title="نوع نورون", message="نورون تحریکی است")
    else:
        tk.messagebox.showinfo(title="نوع نورون", message="نورون مهاری است")
```

بخش نهایی کد مربوط می‌شود به جایگاه دکمه‌ها و جلوه‌های بصری نرم‌افزار.

```
root = tk.Tk()
root.geometry('700x700')
root.title(' برنامه ی تشخیص اسپایک')
root.iconbitmap('./kntu.ico')

photo = PhotoImage(file = "logo-kntu.PNG")
photoimage = photo.subsample(5,5)
Button(root, image = photoimage).place(x=150,y=0)

button1 = tk.Button(master=root, text='انتخاب فایل سیگنال', width =30, height =2, command=ImportAction)
button1.place(x=250,y=400)

button2 = tk.Button(master=root, text='سیگنال اسپایک', width =30, height =2, command=plot1D)
button2.place(x=100,y=450)

button3 = tk.Button(master=root, text='تبدیل ویولت اسپایک', width =30, height =2, command=plot2D)
button3.place(x=400,y=450)


button4 = tk.Button(master=root, text="یک بعدی LDA", width=10, height=1, command=LDA1)
button4.place(x=200,y=500)

button5 = tk.Button(master=root, text= "یک بعدی SVM", width=10, height=1, command=SVM1)
button5.place(x=320,y=500)

button6 = tk.Button(master=root, text="یک بعدی CNN", width=10, height=1, command=CNN1)
button6.place(x=440,y=500)

button7 = tk.Button(master=root, text="دو بعدی LDA", width=10, height=1, command=LDA2)
button7.place(x=200,y=535)

button8 = tk.Button(master=root, text="دو بعدی SVM", width=10, height=1, command=SVM2)
button8.place(x=320,y=535)

button9 = tk.Button(master=root, text="دو بعدی CNN", width=10, height=1, command=CNN2)
button9.place(x=440,y=535)

button10 = tk.Button(master=root, text="نوع اسپایک", width=30, height=2, command=dec)
button10.place(x=250,y=585)

root.mainloop()
```

۴-۴- جمع‌بندی

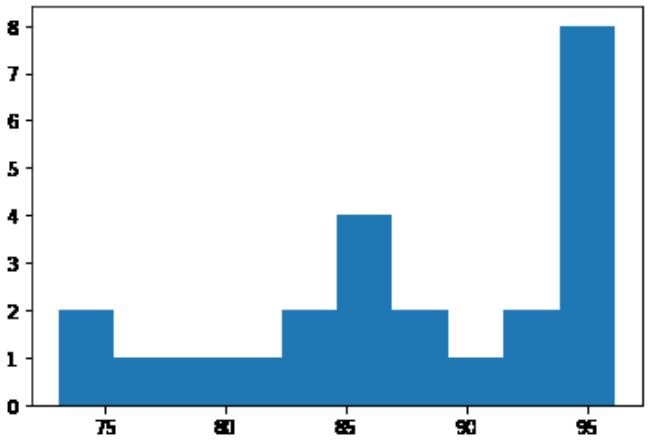
در این بخش توضیح دادیم که داده‌ها را چه طور برای آموزش آماده کردیم. توضیح مختصری راجب روش leave one out که در این کد استفاده شده دادیم و جزییات مدل‌های خود را شرح دادیم. پارامتر های ورودی مدل های CNN,SVM,LDA را برای ورودی یک بعدی و دو بعدی توضیح دادیم و همچنین توضیح دادیم که با چند مدلی که داریم چه طور به تصمیم نهایی برسیم. و نهایتا کدهای مربوط به پروژه را بارگذاری کردیم و توضیحات مربوط به هر بخش را به اختصار آوردیم.

فصل ۵- نتایج و بحث

۵-۱-صحت آموزش

همانطور که پیش تر بحث کردیم از روش leave one out برای ارزشیابی آموزش استفاده کرده بودیم، و به خاطر متعادل نبودن داده‌های دو کلاس، مجبور شدیم ۲۴ بار این عمل را تکرار کنیم و از آن‌ها میانگین بگیریم. در ادامه میخواهیم صحت نهایی و صحت هر یک از این ۲۴ بار ارزشیابی برای هر مدل را مورد بررسی قرار دهیم و پراکندگی صحت‌ها را در نمودار هیستوگرام آن مشاهده کنیم.

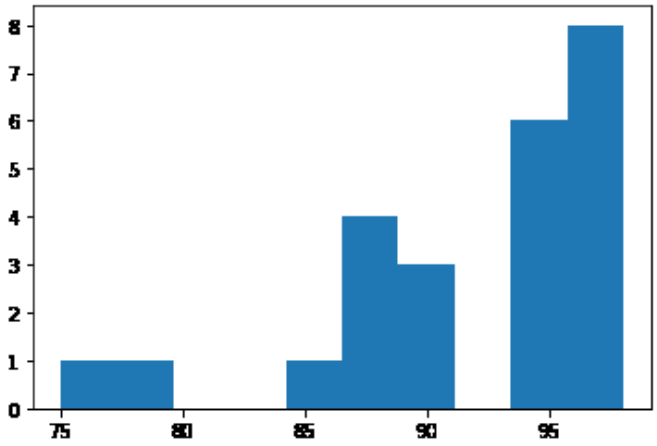
LDA یک بعدی:



صحت نهایی= **٪۸۷,۶۶**

با وجود صحت قابل قبولی که مدل LDA روی اسژایک های یک بعدی به ما می‌دهد، نمودار هیستوگرام آن شاید می‌توانست به حالت گاوسی نزدیک تر باشد. این نمودار از واریانس بالای صحت های بدست آمده خبر می‌دهد که مطلوب نیست. زیرا همواره ترجیح ما با آن است که صحت اطراف عدد ثابتی باشد و اینگونه می‌توان ادعا کرد که صحت بدست آمده ارزش بیش‌تری دارد.

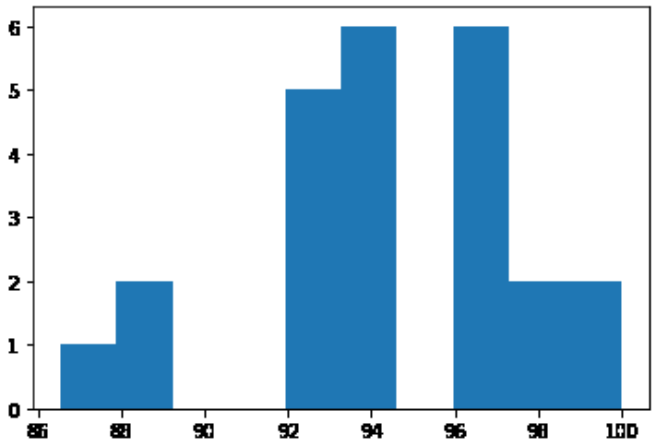
SVM **یک بعدی**



صحت نهایی= **۹۲,۰۶٪**

صحت SVM بدست آمده و نمودار هیستوگرام صحت‌های آن برتری واضحی نسبت مدل پیشین دارد. صحت این مدل از صحت LDA بیش‌تر است و نمودار آن هم به نمودار گاوسی نزدیک تر است(که حول ۱۰۰٪ تغییر دارد). پس واریانس کم‌تری هم دارد و مناسب‌تر است.

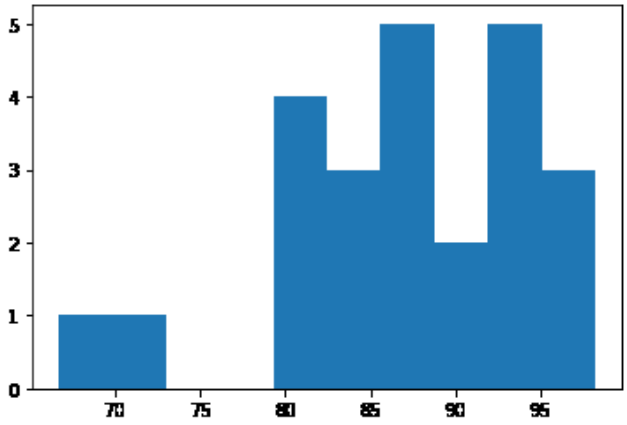
CNN **یک بعدی**



صحت نهایی= **۹۴,۳۱ ٪**

مدل آخر ما در ورودی‌های یک بعدی مدل CNN است که از اظ صحت و واریانس از هر دو مدل پیشین خود بهتر است. صحت‌های بدست آمده حول ۹۴٪ در بازه‌ی کوچکی تغییر می‌کنند.

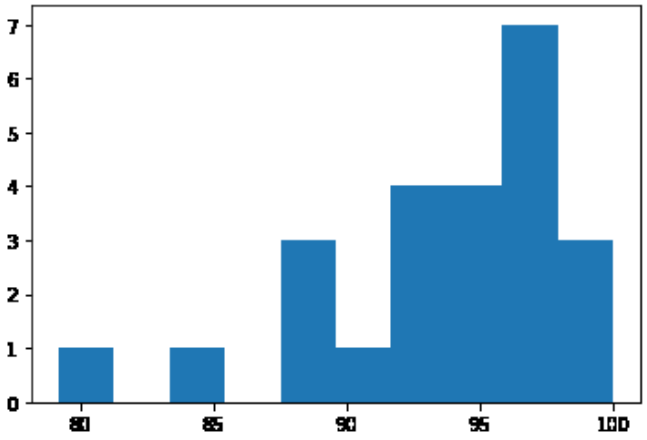
LDA **دو بعدی**



صحت نهایی= **۸۷,۲۵**

صحت بدست آمده ار مدل LDA دو بعدی تقریبا ۸۷ درصد است و مقدار مناسبی است. اما شاید واریانس صحت ها به خوبی واریانس بدست آمده در CNN یک بعدی نباشد و از این جهت ضعف دارد.

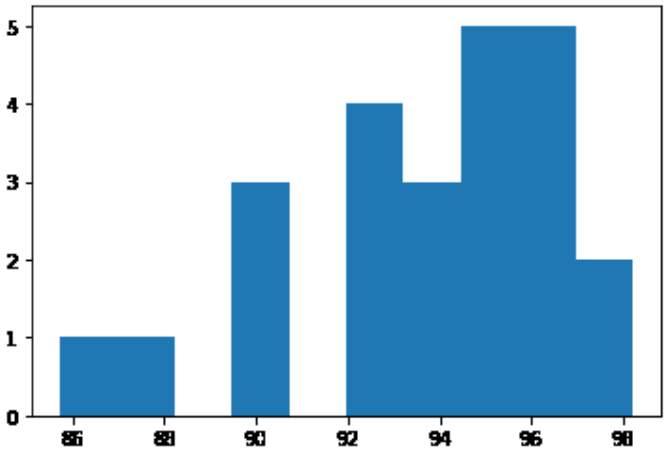
SVM **دو بعدی**



صحت نهایی= **٪۹۳,۵۰**

مدل SVM دو بعدی با داشتن صحت ۹۳,۵ درصد و واریانس تقریبا مناسب از بهترین مدل‌های موجود است.

CNN **دو بعدی**



صحت نهایی= **۹۳,۶۷ ٪**

مدل CNN دو بعدی واریانسی شبیه به مدل SVM دو بعدی دارد اما صحت آن کمی بیش‌تر است.

جدول صحت‌ها

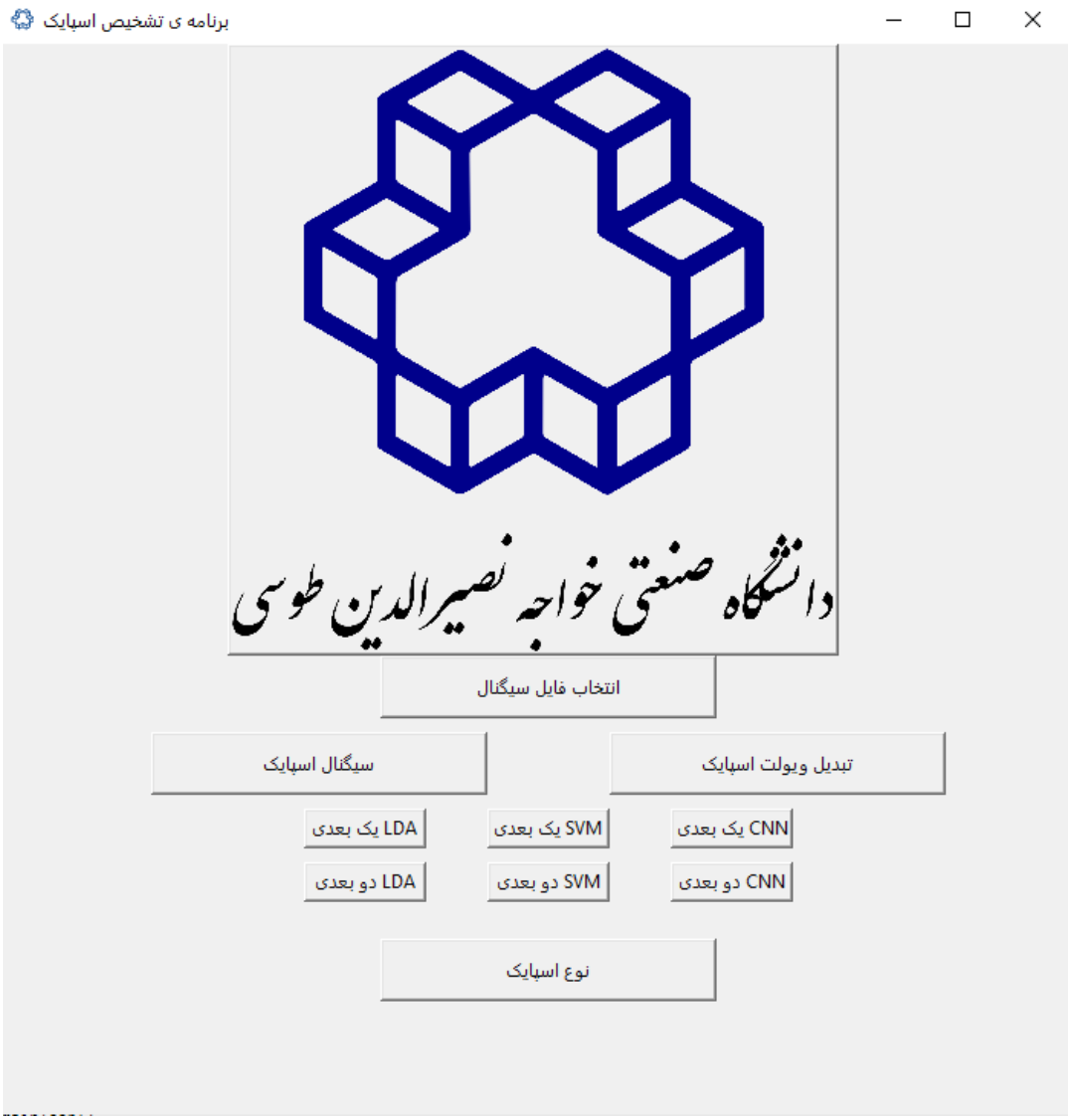
۵-۱-۱ جدول صحت های بدست آمده از ارزشیابی مدل‌های آموزش دیده

	CNN	LDA	SVM
یک بعدی	۹۴,۳۱	۸۷,۶۶	۹۲,۰۱
دو بعدی	۹۳,۶۷	۸۷,۲۵	۹۳,۵۰

همانطور که مشاهده می‌کنیم شبکه‌های پیچشی یک بعدی و دو بعدی از دیگر مدل‌ها بهتر عمل کرده‌اند.

۵-۲- GUI نرم افزار

خروجی نرم افزار به شکل ۵-۲-۱ می‌باشد.



۵-۲-۱ تصویر برنامه ی خروجی

در این برنامه با کلیک بر روی <انتخاب فایل>، اسپایک مورد نظر را بارگزاری می‌کنیم. سپس با کلیک بر روی سیگنال اسپایک و تبدیل ویولت اسپایک می‌توانیم شکل اسپایک را در دو فضای زمان و زمان-فرکانس ببینیم.

با زدن بر روی هر یک از مدل ها تصمیم مودل مذکور نمایش داده می‌شود. توجه شود برای مدل های دو بعدی باید از قبل روی تبدیل ویولت اسپایک کلیک شده باشد. نهایتا با کلیک بر روی نوع اسپایک، تصمیم نهایی برنامه با توجه به تصمیم مدل های پیشین بیان می‌شود.

۵-۳- جمع بندی

در این فصل نتایج حاصل از ارزشیابی مدل‌ها را به نمایش گذاشتیم و آن ها را با توجه به پارامتر های صحت و پراکندگی مقایسه کردیم. مدل های CNN یک بعدی و دو بعدی بهترین عملکرد را داشتند. بعد از آن‌ها SVM ها بودند و نهایتا LDA های یک بعدی و دو بعدی. در نهایت تصویری کلی از نرم افزار خروجی را دیدیم و توضیح مختصری راجع به نحوه‌ی کارکرد آن دادیم.

فصل ۶- نتیجه گیری و پیشنهاد ها

با توجه به نتایج بدست آمده می‌توان اظهار کرد که تمام مدل‌ها به خوبی کارشان را انجام می‌دهند. همان طور که پیشبینی میشد LDA کم ترین صحت را داشت زیرا طبقه بندی خطی است و ساده. تنها مزیت آن سریع بودن آن است. مدل SVM هم با کرنل گاوسی عمکرد قابل قبولی داشته و کارامد است. مساله ای که راجع به این دو مدل وجود دارد این است که هر دو روی داده‌های نه چندان زیاد خروجی خوبی می‌دهند و با زیاد شدن داده حتی ممکن است صحت کاسته شود. اینجا ارزش کار ما مشخص می‌شود. مدل‌های یادگیری عمیق در داده‌های زیاد بهترین عملکرد را دارند. در اینجا مدل شبکه عصبی ما تنها ۵۲ داده‌ی آموزشی داشت که بسیار کم حساب می‌شود. اما می‌بینیم با وجود کم بودن داده‌های آموزشی شبکه‌ی عصبی پیچشی یک بعدی و دو بعدی ما بهترین خروجی‌ها را می‌دهند. این مساله بیانگر این است که اگر داده‌های آموزشی بیش‌تری پیدا کنیم، این مدل‌ها می‌توانند به صحت‌های بالاتر هم برسند. این در حالی است که مدل‌های یادگیری ماشین سنتی، بعید است بهبود چندانی داشته باشند. همینطور از این جهت که مدل ما نیاز به استخراج ویژگی از داده‌ها ندارد نسبت به کارهای قبلی برتری داریم.

مراجع

[۱] Kodama, Takashi et al. “Neuronal classification and marker gene identification via single-cell expression profiling of brainstem vestibular neurons subserving cerebellar learning.” The Journal of neuroscience : the official journal of the Society for Neuroscience vol. 32,23 (2012): 7819-31. doi:10.1523/JNEUROSCI.0543-12.2012

[۲]https://blog.faradars.org/%D9%BE%DB%8C%D8%A7%D9%85%D8%B9%D8%B5%D8%A8%DB%8C, 10/2022

[۳] UCHIZONO, K. Characteristics of Excitatory and Inhibitory Synapses in the Central Nervous System of the Cat. Nature 207, 642–643 (1965).

[۴] https://neuroscientificallychallenged.com/glossary/synaptic-vesicles, 10/2022

[۵] Shmilovici, A. (2005). Support Vector Machines. In: Maimon, O., Rokach, L. (eds) Data Mining and Knowledge Discovery Handbook. Springer, Boston, MA.

[۶] https://blog.faradars.org/support-vector-machine-simplified/ 10/2022

[۷]https://www.researchgate.net/figure/Motivation-behind-non-linear-SVM-classifier_fig2_272520997 , 10/2022

[۸] Boedeker P, Kearns NT. Linear Discriminant Analysis for Prediction of Group Membership: A User-Friendly Primer. Advances in Methods and Practices in Psychological Science. September 2019:250-263.

[۹] https://blog.faradars.org/deep-learning-with-python/, 10/2022

[۱۰] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021).

[۱۱] https://paperswithcode.com/method/max-pooling , 10/2022

[۱۲] https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ ,10/2022

[۱۳] Nanavati, S.P., Panigrahi, P.K. Wavelet transform. Reson 9, 50–64 (2004).

[۱۴] https://blog.faradars.org/%D8%AA%D8%A8%D8%AF%DB%8C%D9%84-%D9%85%D9%88%D8%AC%DA%A9/ ,10/2022

[۱۵] Michael W Browne,Cross-Validation Methods,Journal of Mathematical Psychology,Volume 44, Issue 1, 2000

[۱۶] https://blog.faradars.org/cross-validation/ , 10/2022

[۱۷] https://blog.faradars.org/cross-validation/ , 10/2022

[۱۸] M. Oghazian, F. Saffari, and A. Khadem, “Discrimination between Inhibitory and Excitatory Neurons of Mouse Hippocampus Based on the Shape of Extracellular Spike Waveforms,” Frontiers Biomed Technol., Vol. 8, No. 3, pp.161-169, 2021.

[۱۹] Buccino AP, Ness TV, Einevoll GT, Cauwenberghs G, Hafliger PD. A Deep Learning Approach for the Classification of Neuronal Cell Types. Annu Int Conf IEEE Eng Med Biol Soc. 2018 Jul.

[۲۰] https://buzsakilab.nyumc.org/datasets/McKenzieS/PV2/20160210/revisions_cell_metrics

Abstract

In this paper, we study the classification of excitatory and inhibitory neurons using multiple classification methods. It is important to identify the neuron type since it can help doctors decide how to intervene in brain surgery. SVM, LDA, and CNN models are used for 1-dimensional and 2-dimensional (Wavelet transform) spike inputs in this project. The mentioned models were trained and their accuracies were compared to reach the conclusion of CNN's superiority over the other models. Finally, A GUI was presented using this work to label unlabeled spikes.

Keywords: Inhibitory Neuron and Excitatory Neurons, Machine learning, Deep Neural Networks, SVM ,LDA, CNN, Wavelet Transform

K. N. Toosi University of Technology
Faculty of Electrical Engineering

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of Bachelor of Science (B.Sc)
in Electrical Engineering.

Classifying Excitatory and Inhibitory Cells Using Different ML models

By:
Amirhossein Mashghdoust

Supervisor:
Dr. Ali Khadem

Spring 2022