
بسم الله الرحمن الرحيم

عنوان

تمرین دوم درس یادگیری ماشین (شبکه عصبی)

دانشجو

امیرحسین جراره – ۴۰۰۶۱۶۰۰۴

استاد درس

دکتر عبدی هجراندوست

الف) بخش تئوری

سوال ۱ : در شبکه های عصبی مفهومی تحت عنوان توابع فعال ساز وجود دارند. ابتدا به صورت کامل توضیح دهید که توابع فعال ساز چیست و دلیل استفاده از آنها چیست و همچنین مکانیزم فعالیت این توابع را توضیح داده و انواع توابع فعال ساز را نام برده و به اختصار توضیح دهید.

در تابع فعالسازی (Activation Function) تابعی است که ابتدا مقادیر ورودی گره با یکدیگر ترکیب می‌شوند تا خروجی حاصل به تابع فعالسازی شبکه های عصبی منتقل شود. به عبارتی، مقادیر حاصل از ضرب وزن‌ها با ویژگی‌های ورودی با یکدیگر جمع می‌شوند و مقدار حاصل به تابع فعالسازی منتقل می‌شود. نقش تابع فعالسازی در شبکه های عصبی این است که با استفاده از مقادیر ورودی گره، مقداری در خروجی تولید کند. به بیان جزئی‌تر، تابع فعالسازی مقدار مجموع ورودی وزن‌دار گره را به مقادیری بین ۰ تا ۱ یا ۱- تا ۱ (بسته به تابع فعالسازی) نگاشت می‌کند. سپس، این تابع، مقدار نهایی خود را به لایه بعد منتقل می‌کند. به همین دلیل، به این تابع، تابع انتقال نیز گفته می‌شود. توابع فعالسازی در شبکه های عصبی را می‌توان به سه دسته کلی تقسیم‌بندی کرد که در زیر فهرست شده‌اند:

- تابع پله دودویی: تابع پله دودویی حد آستانه‌ای را در نظر می‌گیرد که براساس آن تصمیم بگیرد آیا گره باید فعال باشد یا ورودی گره برای شبکه اهمیت ندارد. به عبارتی، ورودی این تابع فعالسازی با مقدار حد آستانه مقایسه می‌شود.
- تابع فعالسازی خطی: این تابع، بر روی مجموع وزن‌دار ورودی، محاسبات انجام نمی‌دهد و این مقدار را بدون هیچ تغییری به لایه بعد منتقل می‌کند.
- تابع فعالسازی غیرخطی: این نوع توابع بیشترین استفاده را در شبکه های عصبی دارند. تعمیم‌پذیری و تطبیق‌پذیری مدل با انواع مختلف داده با استفاده از توابع فعالسازی غیرخطی آسان می‌شود.

توابع فعالسازی غیرخطی در شبکه های عصبی را می‌توان به چندین نوع تقسیم کرد

- تابع لوجستیک یا سیگموئید
- تابع ReLU

- تابع SoftMax

سوال ۲: در شبکه عصبی سه مفهوم بهینه ساز ، تـابع هزینـه (lost and cost func) و ابـر پارامتر ها را توضیح دهید.

بهینه سازها روش هایی هستند که برای به حداقل رساندن یک تابع loss یا به حداکثر رسـاندن مـی شوند. توابع ریاضی هستند که به پارامترهای قابل یادگیری مدل مانند وزن و تعصب وابسـته هسـتند. آن ها به دانستن نحوه تغییر وزن و نرخ یادگیری شبکه عصبی برا ی کاهش loss کمک می کنند.

Loss function تابعی با معیـار اخـتلاف بـین خروجـی و تخمـین مـی باشـد. Cost function معیاری از مجموع loss ها برای همه‌ی داده هاست.

ابرپارمترها پارمتر های آزاد سیستم هستند که می توانند در سیستم تنظیم شوند و روند یادگیری مدل را کنترل کنند.برای مثال تعداد نورون ها و لایه ها پارمتر آزاد در شبکه عصبی عمیق هستند.

سوال ۳: الگوریتم های یادگیر ی را نام برده و توضیح دهید و معایب و محدودیتهای هر کدام را بیان کنید .

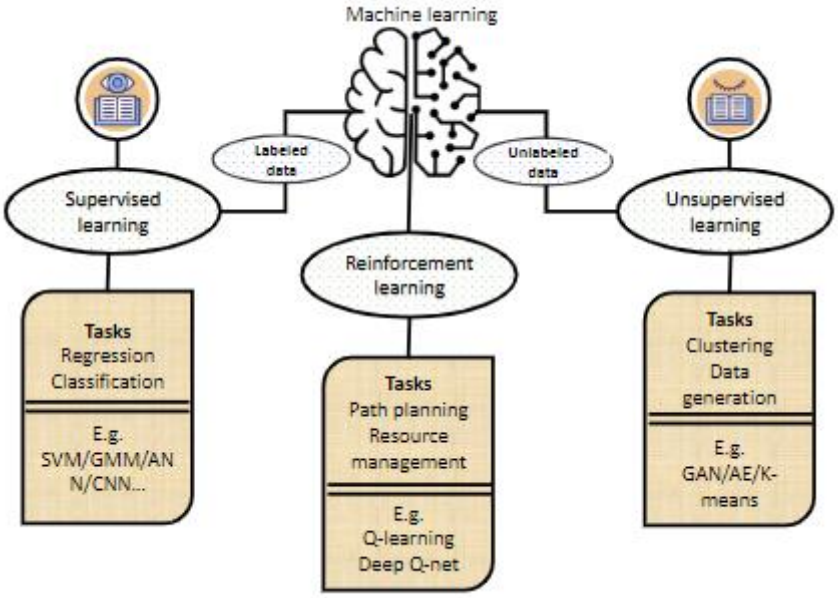
الگوریتم‌های یادگیری به ۳ دسته عمده تقسیم می‌شوند.

۱. یادگیری با نظارت[1]
۲. یادگیری بدون نظارت[2]
۳. یادگیری تقویتی[3]

[1] Supervised Learning

[2] Unsupervised Learning

[3] Reinforcement Learning



شکل (۱-۱) نمایی سراسری از یادگیری ماشین

در یادگیری نظارت شده، داده‌های ارائه‌شده برچسب‌گذاری می‌شوند، به عبارت دیگر، ما برای هـر ورودی داده، ارزش پایه را ارائه می‌کنیم تا الگوریتم از این مقادیر برای یادگیری نحوه تصمیم‌گیری برای ورودی بدون برچسب جدید استفاده کند. به عنوان مثال، پیش‌بینی قیمت پهپاد از روی ویژگی‌های آن در ایـن مثال، شما باید مجموعه‌ای از داده‌های آموزشی را برای الگوریتم ارائه دهید که حاوی هر ویژگی پهپـاد و برچسب مرتبط با آن (قیمت) باشد. مجموعه داده معمـولاً بـه یـک مجموعـه آموزشـی و یـک مجموعـه آزمایشی تقسیم می شود. مجموعه آموزشی برای یادگیری رابطه بین ورودی و خروجی و مجموعه تست برای اعتبارسنجی مدل با اندازه‌گیری دقت آن استفاده می‌شود. مشکلات نظارت شده اغلب به دو دسـته مشکلات رگرسیونی یا طبقه بندی تقسیم می شوند. مشکلات رگرسـیون مقـادیر خروجـی مسـتمری را فراهم می کند (مثلاً پیش بینی قیمت). با این حال، مشکلات طبقه بندی مقادیر گسسته ای را ارائه می دهد که نشان می دهد ورودی به کدام کلاس تعلق دارد (مثلاً طبقه بندی بیماری سرطان خوش خیم یا بدخیم). در ادامه، شناخته شده ترین الگوریتم های یادگیری ماسین را بـرای یـادگیری تحـت نظـارت و بدون نظارت ارائه شده‌است. برخی از الگوریتم های نظارت شده و معماری‌های شبکه عصبی عبارتند از:

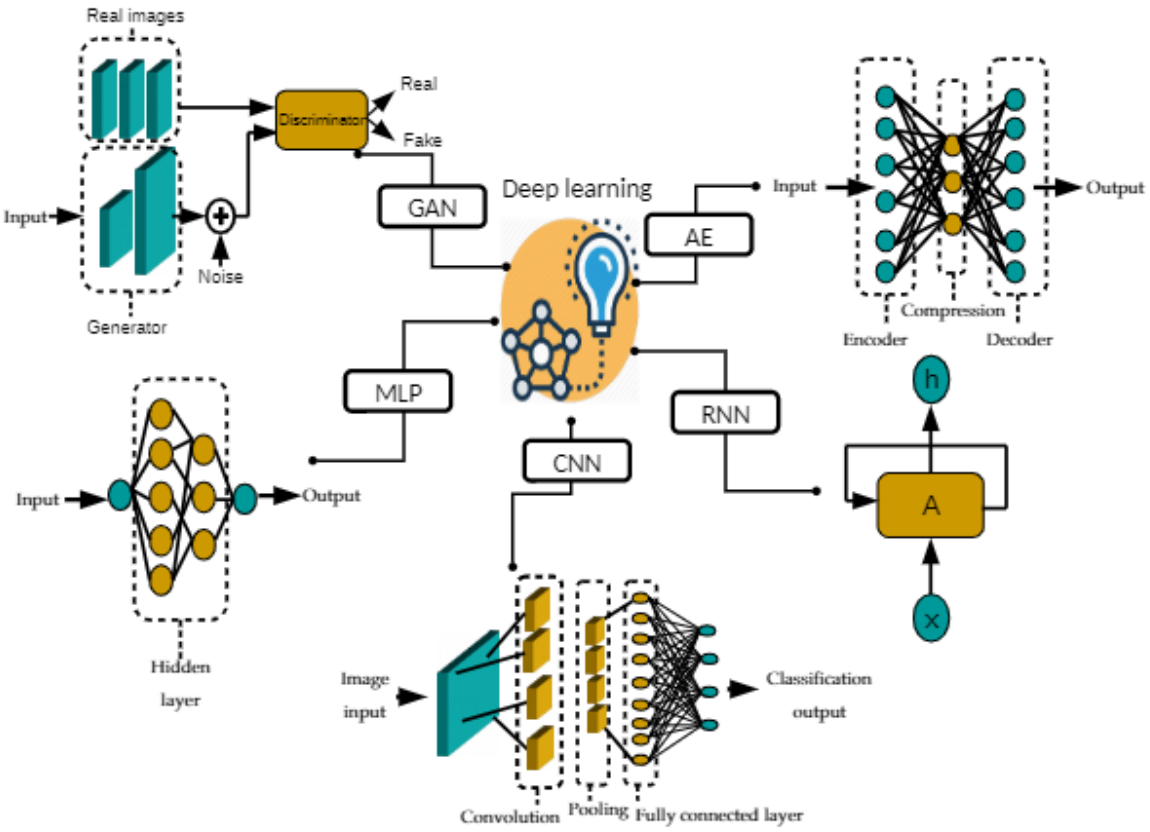
۱) الگوریتم های ترکیبی طبقه بندی و رگرسیون چندین الگوریتم نظارت شده وجود دارد که می توان از آنها برای طبقه بندی یا رگرسیون استفاده کرد. برای مثال ماشین بردار پشتیبان (SVM) می‌تواند هر دو

کار را انجام دهد، درخت‌های تصمیم نیز می‌توانند بسته به مورد استفاده برای حل رگرسیون یا طبقه‌بندی فرموله شوند.

۲) الگوریتم های رگرسیون الگوریتم هایی وجود دارند که وظیفه رگرسیون خالص را با پیش بینی خروجی مقدار پیوسته انجام می دهند. به عنوان مثال، می توان به دو الگوریتم کلاسیک در ML اشاره کرد که رگرسیون خطی و رگرسیون لجستیک هستند.

۳) الگوریتم های طبقه بندی منطقی است که در مورد طبقه بندی کننده های خالص در ML صحبت شده‌است. اگرچه در برخی منابع ذکر شده است که طبقه‌بندی‌کننده ساده بیز با "مقداری اصلاح" می‌تواند برای رگرسیون استفاده شود، اما آن را به عنوان یک نمونه طبقه‌بندی کننده خالص ارائه شده‌است زیرا در ابتدا برای طبقه‌بندی بر اساس قضیه احتمالی بیز مشتق شده بود. ۴) پرسپترون چند لایه (MLP) برای تقلید از شبکه های عصبی بیولوژیکی انسان، ANN ها به صورت ریاضی برای ML فرموله شده اند. شبکه های عصبی مصنوعی با تعدادی گره نیمه متصل ساخته شده اند که با پرسپترون ها مشخص شده و در لایه‌های مختلف گروه بندی می شوند. هر پرسپترون مسئول پردازش اطلاعات از ورودی خود و ارائه خروجی است. همانطور که در شکل نشان داده شده است، MLP ساده ترین شکل یک ANN است که از یک لایه ورودی، یک یا چند لایه پنهان و یک لایه خروجی تشکیل شده است که در آن یک کار طبقه بندی یا رگرسیون انجام می شود. ۵) شبکه های عصبی کانولوشنال (CNN (CNN نوع دیگری از ANN است که در ابتدا برای وظایف بینایی کامپیوتری طراحی شده است. یک CNN معمولاً یک تصویر را به عنوان ورودی می گیرد، وزن ها و سوگیری های قابل یادگیری را اختصاص می دهد که طبق یک الگوریتم خاص به روز می شوند. معماری CNN با لایه‌های کانولوشن مشخص می‌شود که ویژگی‌های سطح بالا را از تصویر استخراج می‌کند که بعداً استفاده خواهد شد. جزئیات فنی مانند توابع فعال سازی، لایه های ادغام و عملیات padding خارج از محدوده این نظرسنجی است. شکل یک معماری معمولی CNN را نشان می دهد که در آن استخراج ویژگی در اولین لایه های کانولوشن انجام می شود و طبقه بندی از طریق یک لایه کاملاً متصل انجام می شود. ۶) شبکه های عصبی مکرر (RNN) هنگامی که داده ها ماهیت ترتیبی دارند، RNN ها برای حل مشکل انجام می شوند. به عنوان مثال، می توان به یک سخنرانی متنی، یک ویدیو یا یک ضبط صدا اشاره کرد. RNN ها به طور گسترده در پردازش زبان طبیعی (NLP)، در تشخیص گفتار، و برای تولید توصیف تصویر به طور خودکار استفاده می شوند. معماری RNN شبیه یک شبکه عصبی معمولی است، فقط حاوی یک حلقه است که به مدل

اجازه می دهد تا نتایج نورون های قبلی را به جلو منتقل کند. RNN در ساده ترین شکل خود از یک خروجی حاوی پیش بینی تشکیل شده است که در شکل با h نشان داده شده است و یک حالت پنهان که نشان دهنده حافظه کوتاه مدت سیستم است.



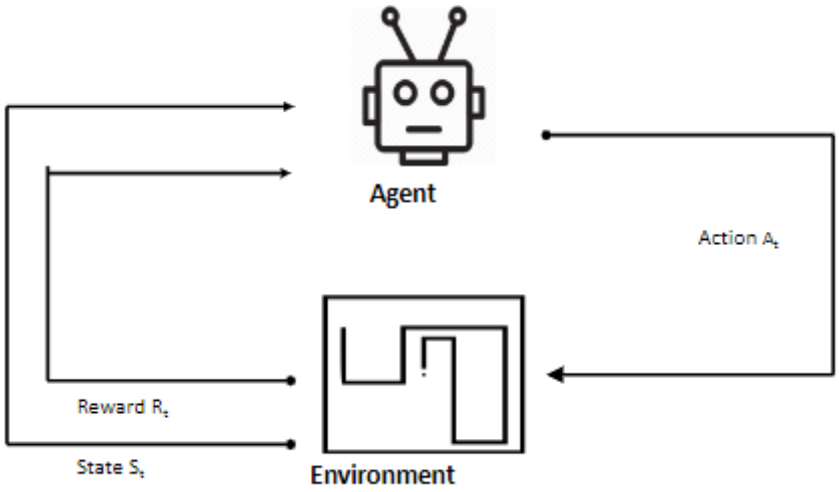
شکل (۲-۱) معماری یادگیری با نظارت

برخلاف یادگیری نظارت شده، یادگیری بدون نظارت از داده های برچسب گذاری شده استفاده نمی کند، در عوض، به دنبال ساختار زیربنایی یا الگوی پنهان در داده ها می گردد و آن را آشکار می کند. برای مثال، خوشه‌بندی داده‌ها، کاهش ابعاد داده‌ها و تولید داده‌ها به عنوان کارهای معمولی برای یادگیری بدون نظارت در نظر گرفته می‌شوند. در ادامه، برخی از الگوریتم‌های کلاسیک بدون نظارت را ارائه می‌کنیم.

الگوریتم های بدون نظارت و معماری‌های شبکه عصبی عبارتند از:

K-means یک الگوریتم بسیار محبوب برای خوشه بندی در ML است. تعـدادی از clustersKas یـک ورودی می‌گیرد و هر نقطه داده را به نزدیکترین خوشه اختصاص می دهد، در حالی که مرکزها را تا حد امکان کوچک نگـه مـی دارد. ۲) مدل‌سـازی مخلـوط گاوسـی یـا GMM یکـی دیگـر از الگوریتم‌هـای خوشه‌بندی در ML است، اما برخلاف الگوریتم K-means، GMM یک مدل احتمالی اسـت. همـانطور که از نام آن مشخص است، خوشه ها از توزیع گاوسی مشتق شده‌اند و ارتباط نرم است. به عبارت دیگـر، هر نقطه داده احتمال ارتباط با هر مرکز خوشه را دارد، با این حال، در الگوریتم K-mean، مـا سیاسـت تداعی سختی داریم. ۳) رمزگذارهای خودکار نوعی شبکه عصبی است که برای یادگیری نمایش داده هـا و در نتیجه رمزگذاری آن استفاده می شود. این تکنیک اغلب برای کاهش ابعـاد اسـتفاده مـی شـود. بـا کمال تعجب، معماری یک AE بسیار ساده است. این شبکه معمولاً توسط یک لایـه ورودی و یـک لایـه پنهان به نام "گلوگاه" تشکیل می‌شود که یک نمایش دانش فشرده از ورودی اصلی را مجبور مـی کنـد. ۴) شبکه های متخاصم مولد GAN معماری‌های الگوریتمی هستند که از دو شـبکه عصـبی بـه منظـور تولید نمونه های مصنوعی جدید از داده ها استفاده می کنند که می توانند برای داده های واقعی ارسـال شوند. آنها به طور گسترده در تولید تصویر، تولید ویدئو و تولید صدا استفاده می شوند.

مانند حوزه های یادگیری تحت نظارت و بدون نظارت ML، RL نیز حوزه دیگری از ML است کـه بـه تصمیم گیری در یک محیط کاملاً تعریف شده اختصاص داده شده است.



شکل (۳-۱) المان‌های یادگیری تقویتی

الگوریتم های یادگیری مبتنی بر مسئله عبارتند از:

۱) RL مبتنی بر مدل همانطور که از نامش مشخص است، مسئله RL مبتنی بر مدل از یک مدل به عنوان عنصر ششم برای تقلید رفتار محیط برای عامل استفاده می کند. در نتیجه، عامل قادر به پیش بینی وضعیت و عمل برای زمان می شود. T+1 با توجه به وضعیت و عمل در زمان T. در این سطح، یادگیری تحت نظارت می تواند ابزار قدرتمندی برای انجام کار پیش بینی باشد. بنابراین، بر خلاف RL بدون مدل، در RL مبتنی بر مدل، به روز رسانی تابع مقدار بر اساس مدل است و نه بر اساس تجربه.

۲) RL بدون مدل در مسائل RL بدون مدل، عامل نمی تواند آینده را پیش بینی کند و این تفاوت اصلی با چارچوب RL مبتنی بر مدل است که قبلا توضیح داده شد. در عوض، اقدامات بر اساس روش به اصطلاح "آزمایش و خطا" است که در آن عامل، برای مثال، می تواند در فضای سیاست جستجو کند، پاداش های مختلف را ارزیابی کند، و در نهایت یک پاداش بهینه را انتخاب کند. یک مثال کلاسیک شناخته شده برای RL بدون مدل، روش یادگیری Q است که در آن مقادیر Q بهینه هر عمل و پاداش را تخمین می زند و اقدامی را که بالاترین مقدار Q را برای وضعیت فعلی دارد انتخاب می کند. به طور خلاصه، تمایز بین مسائل RL مبتنی بر مدل و بدون مدل کار آسانی است. فقط سوال زیر را از خود بپرسید: آیا عامل قادر به پیش بینی وضعیت و عمل بعدی است؟ اگر پاسخ مثبت است، شما با یک RL مبتنی بر مدل سر و کار دارید، در غیر این صورت، به احتمال زیاد مشکل RL بدون مدل است.

۳) یادگیری تقویت عمیق (DRL) بررسی اجمالی در حالی که RL کلاسیک یک راه حل کارآمد برای بسیاری از انواع مسائل تصمیم گیری گسسته پیشنهاد می کرد، راه حل های واقعی تری را می توان با استفاده از DRL ارائه کرد که کارایی خود را با رسیدن به کنترل سطح فوق انسانی ثابت کرده است. DRL مبتنی بر استفاده از ANN برای ارزیابی مقادیر عمل با استفاده از تجربیات قبلی عامل است. از میان الگوریتم‌های متعددی که در ادبیات ارائه شده‌اند.

سوال ۴: مشکل عمده در عمیقتر کردن شبکه های عصبی چیست؟ نام برده و توضیح دهید.

مشکل عمده عمیق تر کردن شبکه های عصبی افزایش زمان برای آموزش یا سخت افزار پرهزینه تر می‌باشد.

از مشکلات دیگر با وجود تابع فعالساز نامناسب محو شدگی و انفجار گرادیان می باشد.

سوال ۵: قانون دلتا و تعمیم یافته این قانون را توضیح دهید.

با قاعده Delta که توسط Widrow و Hoff تنظیم شده یکی از رایج ترین قواعد یادگیری است و جزء یادگیری نظارت شده می باشد. این قانون بیان می‌کند که اصلاح در وزن یک گره برابر با ضرب خطا و ورودی است. فرم ریاضی قانون Delta بصورت زیر بیان می شود :

$$w = h\,(t - y)\,Xi\Delta$$

برای یک بردار ورودی معین ، پاسخ صحیح ازمقایسه بردار خروجی بدست می آید. اگر اختلاف صفر باشد ، هیچ یادگیری انجام نمی گیرد. در غیر این صورت ، وزن ها تنظیم می شوند تا این اختلاف کاهش یابد. تغییر وزن از ui به uj بصورت زیر می باشد :

$$dwij = r * ai * ej$$

که در آن r نرخ یادگیری ، ai نشان دهنده فعالساز ui است و ej اختلاف بین خروجی مورد انتظار و خروجی واقعی uj است. اگر مجموعه الگو های ورودی ، یک مجموعه مستقل را تشکیل دهند، آنگاه وابستگی های اختیاری را با استفاده از قانون Delta یاد می گیرند.

مشاهده شده است که برای شبکه هایی با توابع فعالساز خطی و فاقد واحد مخفی ، مربع خطا نسبت به وزن در فضای n بعدی یک سهمی گون (Paraboloid) است. از آنجا که ثابت تناسب منفی است ، نمودار چنین تابعی مقعر به سمت بالاست و دارای کمترین مقدار است. راس این سهمی گون نقطه ای است که خطا را کاهش می دهد. بردار وزن متناظر با این نقطه، بردار وزن ایده آل است.

ما می‌توانیم از قاعده یادگیری Delta با یک واحد خروجی یا چندین واحد خروجی استفاده کنیم.

در حالی که اعمال قاعده Delta فرض می کند که می توان خطا را مستقیماً اندازه گیری کرد.

هدف از اعمال قاعده Delta کاهش اختلاف بین خروجی واقعی و مورد انتظار است که برابر با خطا است.

عمل انتقال تابع دلتا برای پخش عادلانه وزن ها به سایر نورن ها نیز توسعه تابع دلتا می باشد.

سوال ۶: الگوریتم پروپکیشن چیست، به طور کامل شرح دهید.

انتشار رو به عقب خطاها که به اختصار پس انتشار یا Backpropagation نامیده می‌شود، الگوریتمی برای یادگیری نظارتی شبکه عصبی با استفاده از گرادیان کاهشی است. در این روش، برای یک شبکه عصبی مصنوعی و تابع خطای مشخص، گرادیان تابع خطا نسبت به وزن‌های شبکه عصبی محاسبه می‌شود. الگوریتم پس انتشار تعمیمی از قانون دلتا برای پرسپترون‌ها به شبکه‌های عصبی پیشخور چندلایه است. واژه پس‌رو یا رو به عقب (Backward) که بخشی از اصطلاح پس انتشار است، در واقع از این موضوع می‌آید که محاسبه گرادیان به صورت رو به عقب در شبکه انجام می‌شود و گرادیان لایه خروجی وزن‌ها در ابتدا و گرادیان لایه ورودی در آخر انجام می‌شود؛ بدین صورت که از محاسبات مشتق جزئی گرادیان یک لایه برای گرادیان لایه قبلی استفاده می‌شود. این حرکت رو به عقب اطلاعات خطا، منجر به محاسبه کارآمد گرادیان در هر لایه نسبت به حالتی می‌شود که در آن گرادیان لایه‌ها به صورت جداگانه به دست می‌آید.

یکی از مشکلات اساسی آموزش شبکه‌های عصبی پیشخور چندلایه تصمیم‌گیری در مورد چگونگی یادگیری نمایش داخلی مناسب است (یعنی وزن‌ها و بایاس‌های گره‌های لایه پنهان چه باید باشد). برخلاف پرسپترون که از قانون دلتا برای تقریب یک خروجی هدف تعریف شده است، گره‌های لایه پنهان خروجی هدف ندارند، زیرا از آن‌ها به عنوان گام‌های میانی در محاسبه استفاده می‌شود.

از آنجا که گره‌های لایه پنهان خروجی هدف ندارند، نمی‌توان یک تابع خطا را که مخصوص آن گره باشد، تعریف کرد. در عوض، هر تابع خطایی برای آن گره به مقادیر پارامترهای موجود در لایه‌های قبلی (زیرا لایه‌های قبلی، ورودی آن گره را تعیین می‌کنند) و لایه‌های بعدی (زیرا خروجی آن گره بر محاسبه تابع خطای $E(X,\theta)E(X,\theta)$تأثیر می‌گذارد) بستگی دارد. این اتصال پارامترها بین لایه‌ها می‌تواند ریاضیات مسئله را بسیار دشوار کند (در درجه اول در نتیجه استفاده از قانون ضرب که در ادامه مورد بحث قرار می‌گیرد)، و در صورت عدم اجرای هوشمندانه، موجب کندی محاسبات گرادیان نزولی نهایی شود. پس‌انتشار با ساده کردن ریاضیات گرادیان نزولی، هر دو مشکل را برطرف می‌کند، ضمن اینکه محاسبه کارآمد آن را نیز تسهیل می‌کند.

سوال ۷: دو ساختار شبکه رو به جلو و بازگشتی را توضیح دهید و موارد استفاده از هرکدام را توضیح دهید.

در شبکه های عصبی مصنوعی پیشخور ، جریان اطلاعات فقط از یک جهت است. یعنی جریان اطلاعـات از لایه ورودی به لایه پنهان و در نهایت به خروجی می رسد. هیچ حلقه پسخوری در این شـبکه عصـبی وجود ندارد. این نوع شبکه های عصبی عمدتاً در یادگیری نظارت شده برای مواردی مانند طبقه بنـدی ، شناسایی تصویر و غیره مورد استفاده قرار می گیرند. ما از آن ها در مواردی که داده هـا ذاتـاً دنبالـه ای نیستند استفاده می کنیم.

شبکه های عصبی مصنوعی پسخور ، دارای حلقه های پسخور هستند. این نوع شبکه های عصـبی مثـل شبکه های عصبی بازگشتی ، عمدتاً برای نگهداری حافظه مورد استفاده قرار می گیرند. این نوع شبکه ها برای حوزه هایی که داده ها دنباله ای یا وابسته به زمان هستند ، مناسب ترند.از جمله موارد اسـتفاده از آن پردازش زبان طبیعی می باشد.

سوال ۸: محدودیت پرسپترون را بیان کنید.

خروجی شبکه پرسپترون به صورت باینری می باشد و از آن برای سیستم هایی می توان استفاده کرد که با جداساز خطی قابل جداسازی باشند.

سوال ۹: مشکل overfitting و underfitting و عوامل ناشـی از آن را شـرح داده و راه‌هـای مقابله با آنها را به صورت کامل و منحصرا در شبکه های عصبی عمیق شرح دهید.

مشکل overfitting به حفظ کردن تابع مدل بر روی داده های ورودی می باشد. این مشکل بـه عوامـل متعددی بستگی دارد از جمله :

- عمق زیاد و پارمترهای زیاد شبکه
- آموزش زیاد شبکه

و راه های رفع آن

- ساده کردن مدل با کاهش لایه ها یا نورون ها
- افزیش تعداد المان های دیتاست
- استفاده از cross validation

underfitting یا bias شدن به سبب سادگی مدل می باشد.از جمله موارد آن تعداد کم لایه ها یا نورون ها می باشد و راه حل آن نیز افزایش تعداد نورون ها و یا لایه هاست.

سوال ۱۰: چهار نوع حوزه اصلی در زمینه پردازش تصویر را نام ببرید و هر کدام را به اختصار توضیح دهید.

۱. تشخیص کاراکتر دست نویس : ANN ها در کنار پردازش تصویر برای تشخیص کاراکتر دست‌نویس مورد استفاده قرار می گیرند. برای این کار شبکه های عصبی برای شناسایی کاراکتر های دست نویس که می‌توانند به شکل حروف یا ارقام باشند، آموزش داده می شوند.

۲. طبقه بندی امضا : به منظور تشخیص امضاء ها و دسته بندی آن ها بر اساس افراد ، از شبکه های عصبی مصنوعی و پردازش تصویر برای ساخت این سیستم ها و اعتبارسنجی استفاده می کنیم.

۳. شناسایی شی یا object detection : برای شناسایی یک شی یا چند شی و تعیین کلاس آن ها به کار می رود. ازجمله موارد استفاده از آن در راهنمایی رانندگی می باشد.

۴. ردیابی یا Tracking : دنبال کردن یک المان تصویر یا با ناظر انسانی یا با ویژگی معلوم در تصویر یا فیلم

ب) بخش عملی

سوال ۱ : دیتاست mnist بوسیله PyTorch

این فایل در سکشـــن Mnist ANN در فایـــل HW_2_ANN_Minst_and_Acustic.ipynb قرار دارد

پیاده سازی با ANN :

با سرچ عبارت صورت سوال در گوگل متوجه می شویم این یک تمپلیت معروف از کتابخانه پایتورچ که از کتابخانه بسیار معروف در زمینه شبکه های عصبی می باشد.ابتدا ساختار مطرح شده در سیستم را پیاده سازی می کنیم. سپس دیتاست را از گیتهاب اینجانب از لینـک کلـون مـی کنـیم و آن را درقالـب یـک تنسور در می آوریم. مدل را ساخته قسمت بکپروپگیشن را پیاده سازی و solver را Adam در نظر می گیریم و خروجی نهایی را برای کلاس بندی softmax در نظر می گیـریم.batch_size را بـا توجـه بـه محدودیت کلب ۱۶ در نظر می گیریم. شبکه طراحی شده با سعی خطا و تست های مکرر بـه صـور زیـر طراحی می کنیم.

```
DeepNeuralNetwork(
  (linear_block): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=784, out_features=1024, bias=True)
    (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU()
    (4): Dropout(p=0.5, inplace=False)
    (5): Linear(in_features=1024, out_features=2048, bias=True)
    (6): BatchNorm1d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): ReLU()
    (8): Dropout(p=0.5, inplace=False)
    (9): Linear(in_features=2048, out_features=1024, bias=True)
    (10): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU()
    (12): Dropout(p=0.5, inplace=False)
    (13): Linear(in_features=1024, out_features=512, bias=True)
    (14): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): ReLU()
    (16): Dropout(p=0.5, inplace=False)
    (17): Linear(in_features=512, out_features=256, bias=True)
    (18): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): ReLU()
    (20): Dropout(p=0.5, inplace=False)
    (21): Linear(in_features=256, out_features=64, bias=True)
    (22): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (23): ReLU()
    (24): Dropout(p=0.2, inplace=False)
    (25): Linear(in_features=64, out_features=10, bias=True)
  )
)
```

این شبکه از ۲۵ لایه تشکیل شده است. ورودی آن ۲۸*۲۸ پیکسل یعنی ۷۸۴ ویژگی می باشد و خروجی آن ۱۰ لایه کلاس بندی می باشد. در حل این مسئله learning_rate را برابر ۰,۰۱ قرار می دهیم. در گوگل کلب runtime را نیز روی GPU برای سرعت بیشتر قرار می دهیم. خروجی ANN برای این شبکه به صورت زیر می باشد.

	precision	recall	f1-score	support
0	0.94	0.99	0.96	980
1	0.97	0.98	0.98	1135
2	0.95	0.92	0.93	1032
3	0.92	0.93	0.93	1010
4	0.95	0.86	0.90	982
5	0.97	0.85	0.91	892
6	0.97	0.94	0.95	958
7	0.95	0.91	0.93	1028
8	0.90	0.92	0.91	974
9	0.80	0.95	0.87	1009
accuracy			0.93	10000
macro avg	0.93	0.93	0.93	10000
weighted avg	0.93	0.93	0.93	10000

همچنین ماتریس درهمریختگی آن نیز به صورت زیر می باشد.

```
[ 966,    0,    1,    1,    0,    2,    4,    2,    4,    0]
[   0, 1115,    4,    1,    1,    0,    4,    0,   10,    0]
[  16,    3,  950,    2,   16,    0,    6,   24,   13,    2]
[   1,    3,   20,  939,    0,    3,    1,   12,   17,   14]
[   2,    0,    3,    0,  841,    0,    7,    0,    5,  124]
[  11,    4,    0,   54,    4,  762,    9,    3,   32,   13]
[  18,    5,    1,    1,   11,    7,  902,    0,   13,    0]
[   1,    9,   14,    2,    2,    0,    0,  940,    0,   60]
[   9,    4,   10,    9,    4,    9,    1,    4,  898,   26]
[   8,    7,    1,   10,    9,    0,    0,    3,   11,  960]
```

پیاده سازی با CNN :

این فایل در سکشن CNN Mnist در فایل HW_2_ANN_Minst_and_Acustic.ipynb قرار دارد

اگر در شبکه بالا ورودی کانولوشنی و فولی کانکتد را به صورت زیر طراحی کنیم.

```
DeepNeuaralNetwork(
  (conv_block): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (linear_block): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=6272, out_features=128, bias=True)
    (2): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU(inplace=True)
    (4): Dropout(p=0.5, inplace=False)
    (5): Linear(in_features=128, out_features=64, bias=True)
    (6): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): ReLU(inplace=True)
    (8): Dropout(p=0.5, inplace=False)
    (9): Linear(in_features=64, out_features=10, bias=True)
  )
)
```

علاوه بر با کاهش لایه به ۱۹ لایه و کاهش پارامتر های سیستم دقت خروجی تقریبا ۹۹ می باشد. که در زیر می بینید.

0	0.99	1.00	1.00	980
1	1.00	0.99	1.00	1135
2	0.98	1.00	0.99	1032
3	0.99	0.99	0.99	1010
4	1.00	0.98	0.99	982
5	1.00	0.99	0.99	892
6	0.99	0.99	0.99	958
7	0.99	0.98	0.99	1028
8	0.98	0.99	0.99	974
9	0.99	0.98	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

مارتیس درهمریختگی آن به صورت زیر می باشد.

```
[ 977,    0,    0,    0,    0,    0,    1,    1,    1,    0]
[   0, 1129,    3,    2,    0,    0,    0,    1,    0,    0]
[   0,    0, 1029,    0,    0,    0,    0,    1,    2,    0]
[   0,    0,    0, 1004,    0,    1,    0,    2,    3,    0]
[   0,    0,    2,    0,  966,    0,    5,    0,    4,    5]
[   0,    0,    0,    6,    0,  880,    2,    0,    1,    3]
[   4,    3,    0,    0,    1,    1,  948,    0,    1,    0]
[   0,    2,   17,    1,    1,    0,    0, 1005,    0,    2]
[   1,    0,    2,    2,    0,    0,    0,    0,  968,    1]
[   1,    0,    1,    0,    2,    2,    0,    2,   10,  991]
```

مقایسه با درخت تصمیم:

در مقایسه با شبکه درخت تصمیم ، CNN بادقت ۹۹ درصدی بر روی داده تست به سبب در نظر گرفتن ورودی ماتریسی عملکرد بهتری دارد. شبکه ANN فولی کانکتد نیز اگر درست طراحی شـود دقـت ۹۳ درصدی داردکه از عملکرد درخت تصمیم با دقت ۸۹ درصد بهتر است ولی سرعت آن کمتر می باشد.

چالش ها:

در مجموع در حل این مسئله چالش های زیر مطرح بودند.

- مشکل ارور های تایپ داده در ورودی و پردازش خروجـی در کتابخانـه Pytorch واقعـا چـالش بزرگی بود.
- مشکل تیدیل داده ماتریسی به تنسور و درک تنسور و تئوری آن چالش بود.
- مشکل بسته شدن Runtime GPU در گوگل کلب

سوال ۲: استفاده از شبکه کانولوشنی عمیق توسـط MobileNetV2 و تغییـر Head بـرای کلاس بندی

این فایل در HW2_MobileNetV2_CHANGED_HEAD.ipynb قرار دارد.

در ابتدا به دانلود دیتاست CIFAR100 از دیتاست های نمونه Keras می‌کنیم. و آن را به به بازه ۰ تا ۱ نرمالایز می کنیم.

حال باید بهترین شبکه را انتخاب نماییم. با توجه به ویژگی کم ولی تعداد کلاس بالا در CIFAR است برای دقت بالا به یک شبکه با لایه کم و پارامتر کم نیاز داریم بنابراین بهترین شبکه را MobileNetV2 انتخاب می کنیم. البته شبکه های inceptionV3 ، MobileNet ، Resnet50V2 تست شدند که عملکرد MobileNetV2 عملکرد بهتری داشت. البته بهترین عملکرد را Resnet18 داشت ولی در پکیج پیشنهادی تمرین نبود.

پس از تولید دو دسته داده ترین و تست اقدام به طراحی اپتیمایز می کنیم و آن را از نوع آدام انتخاب می کنیم.

حال به شبکه دولایه فولی کانکتد اضافه می کنیم تا feature انتهایی ما را ابتدا از ۱۲۸۰ به ۵۱۲ و سپس به ۲۵۶ برسانند.این کار در قسمت انتهایی برای تغییر Head سرعت شبکه را بسیار بالا می برد.

همچنین Trainable بودن شبکه MobileNetV2 را غیرفعال می کنیم تا آن ها را که روی شبکه PreTrain آن با دیتاست imagenet می باشد آپدیت نشود.البته ۵ لایه ی انتهایی آن را Trainable می کنیم تا بتوان لایه های انتهایی را برای کلاس بندی بهبود دهیم.

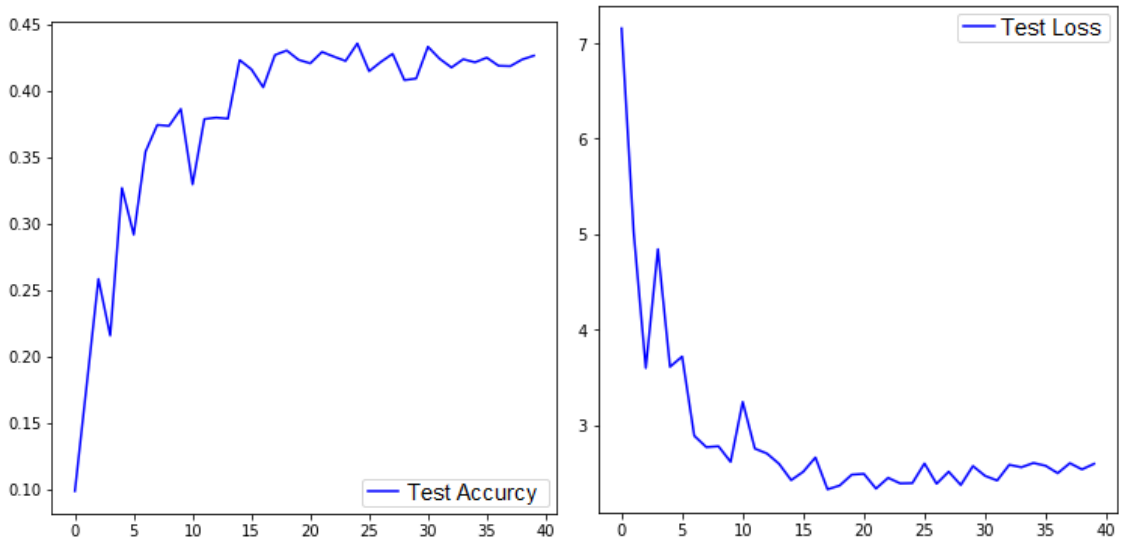
حال با یک لایه ی فولی کانکتد با ۱۰۰ نرون کلاس بندی می کنیم تا فرآیند ترین و آموزش را شروع کنیم.

آموزش MobileNetV2 با دو لایه اضافی fullyConnected به همراه یک لایه Predection :

این مدل را با ۶۵ اپوک آموزش می دهیم. دقت خروجی آن به صورت زیر می باشد.

```
- loss: 0.7865
- sparse_categorical_accuracy: 0.7712
- val_loss: 3.3020
- val_sparse_categorical_accuracy: 0.4275
```

با توجه به اینکه بیشتر از یک مقدار استفاده از خدمات گوگل شما را مشمول تحریم می کند بیشتر از ۶۵ اپوک گوگل اکانت شخصی را برای GPU محدود کرد و این مدل را ذخیره می کنیم. خروجی برای ۴۰ اپوک بروی داده ترین به صورت زیر می باشد.



شبکه MobileNetV2 با دو لایه اضافی با Head شبکه SVM :

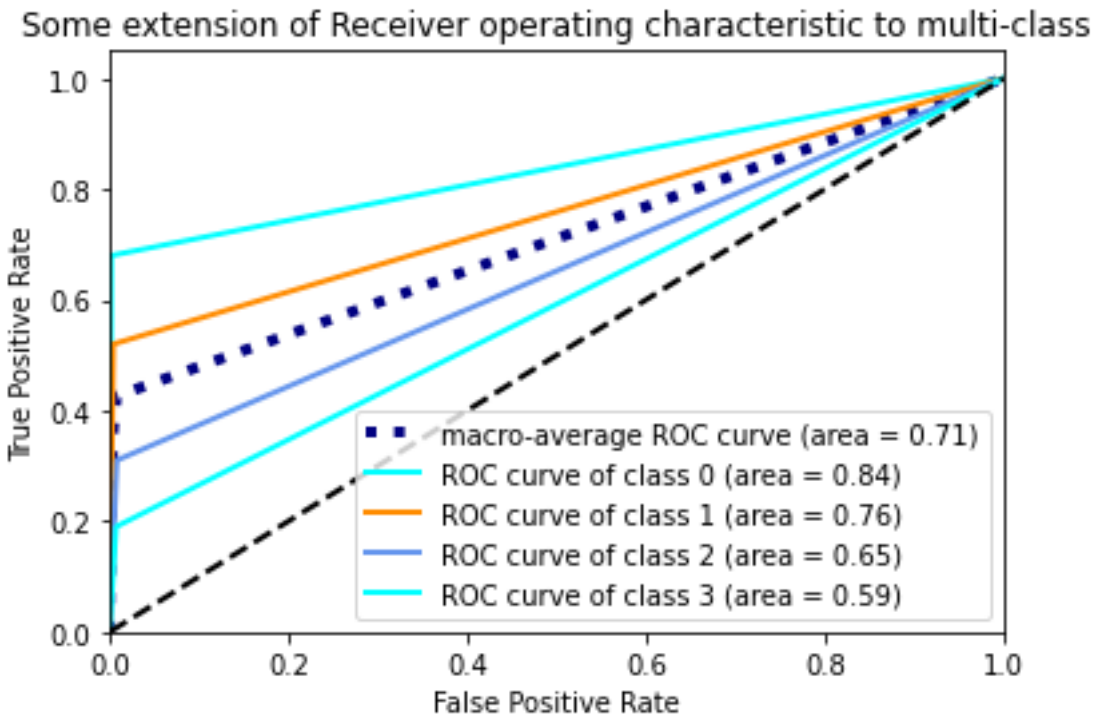
برای اینکار از شبکه آموزش دیده در تست قبل لایه Predection را بر می‌داریم. مدل به صورت زیر می باشد.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224_input  [(None, 32, 32, 3)]       0
 (InputLayer)

 mobilenetv2_1.00_224 (Funct  (None, 1, 1, 1280)       2257984
 ional)

 global_max_pooling2d_1 (Glo  (None, 1280)             0
 balMaxPooling2D)

 dense_1 (Dense)             (None, 512)               655872

 dense_2 (Dense)             (None, 256)               131328

=================================================================
Total params: 3,045,184
Trainable params: 2,648,640
Non-trainable params: 396,544
```

داده های ورودی را به شبکه می دهیم و خروجی آن را تبدیل به آرایه numpy کـرده و بـه SVM مـی دهیم و کلاس بندی می کنیم. دقت خروجی به صورت زیر می باشد.

دقت برابر می شود با :

```
accuracy_score = 0.4279
```



شبکه MobileNetV2 با دو لایه اضافی با Head شبکه درخت تصمیم :

برای اینکار از شبکه آموزش دیده در تست قبل لایه Predection را بر می‌داریم. مدل به صورت زیر می باشد.

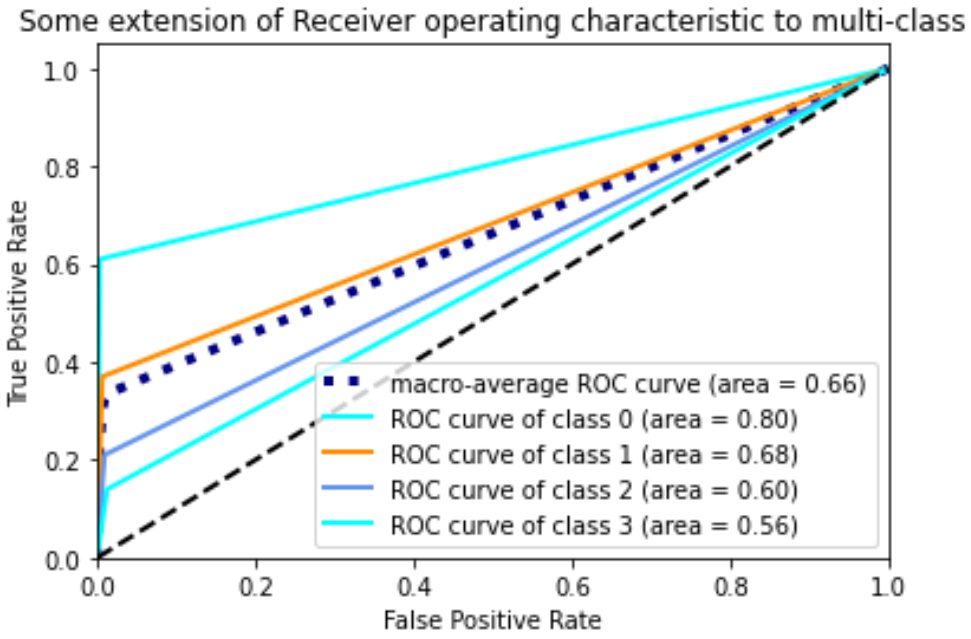
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224_input  [(None, 32, 32, 3)]       0
 (InputLayer)

 mobilenetv2_1.00_224 (Funct  (None, 1, 1, 1280)       2257984
 ional)

 global_max_pooling2d_1 (Glo  (None, 1280)             0
 balMaxPooling2D)

 dense_1 (Dense)             (None, 512)               655872

 dense_2 (Dense)             (None, 256)               131328

=================================================================
Total params: 3,045,184
```

```
Trainable params: 2,648,640
Non-trainable params: 396,544
```

داده های ورودی را به شبکه می دهیم و خروجی آن را تبدیل به آرایه numpy کرده و به درخت تصمیم می دهیم و کلاس بندی می کنیم. دقت خروجی به صورت زیر می باشد.

دقت برابر می شود با :

```
accuracy_score = 0.3174
```



شبکه MobileNetV2 با دو لایه اضافی با Head شبکه MLP :

برای اینکار از شبکه آموزش دیده در تست قبل لایه Predection را بر می‌داریم. مدل به صورت زیر می باشد.

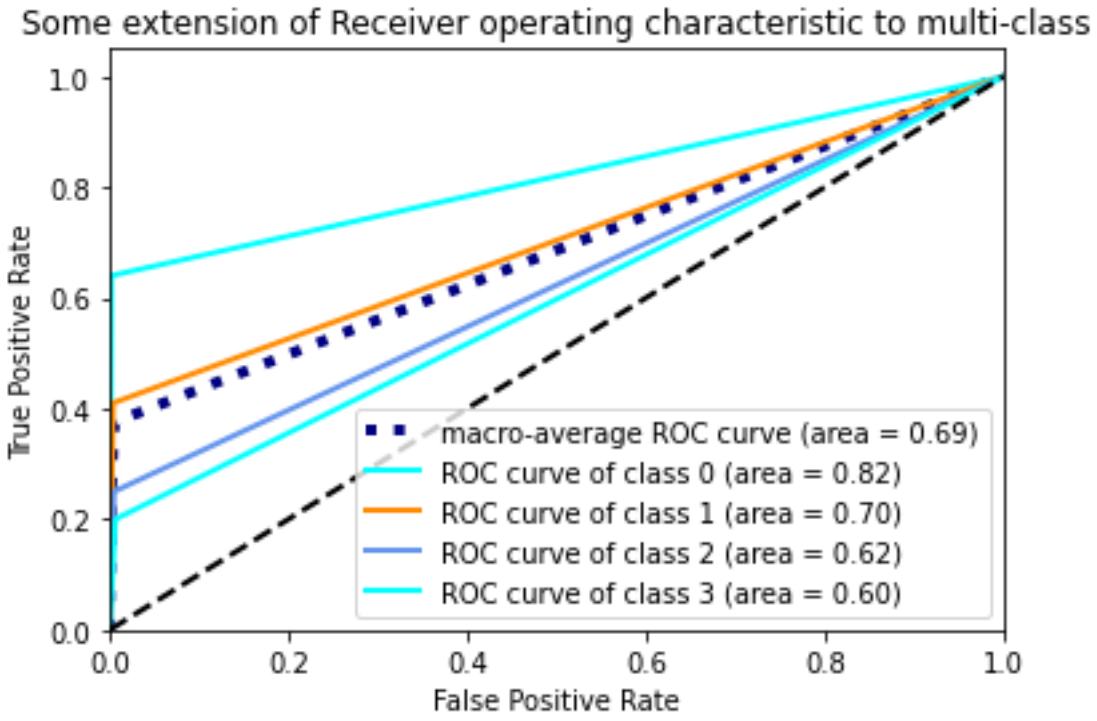
```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224_input  [(None, 32, 32, 3)]       0
 (InputLayer)

 mobilenetv2_1.00_224 (Funct  (None, 1, 1, 1280)       2257984
 ional)

 global_max_pooling2d_1 (Glo  (None, 1280)             0
 balMaxPooling2D)
```

```
 dense_1 (Dense)              (None, 512)               655872

 dense_2 (Dense)              (None, 256)               131328

=================================================================
Total params: 3,045,184
Trainable params: 2,648,640
Non-trainable params: 396,544
```

داده های ورودی را به شبکه می دهیم و خروجی آن را تبدیل به آرایه numpy کرده و به MLP بـا ۱۵ لایه ی مخفی و ۱۲۸ نورونه می‌دهیم و کلاس بندی می کنیم. دقت خروجی به صورت زیر می باشد.

دقت برابر می شود با :

```
accuracy_score = 0.4047
```



شبکه MobileNetV2 با دو لایه اضافی با Head شبکه Perceptron:

برای اینکار از شبکه آموزش دیده در تست قبل لایه Predection را بر می‌داریم. مدل به صورت زیر می باشد.

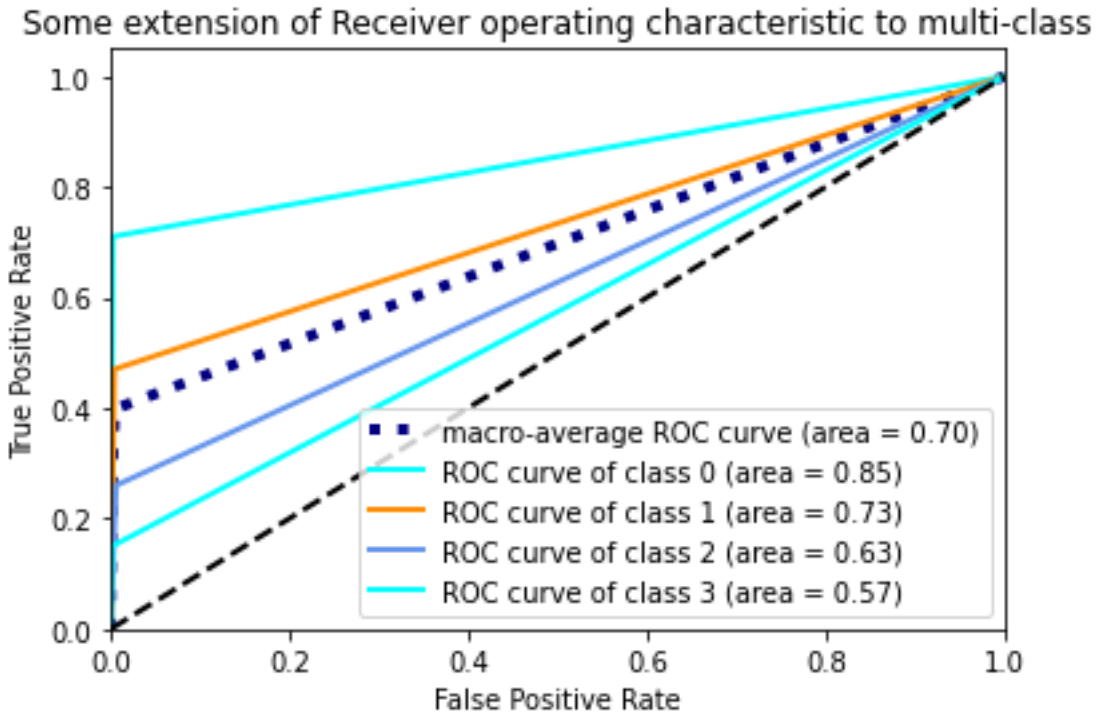
```
Model: "model"
_________________________________________________________________
```

```
 Layer (type)                 Output Shape              Param #
=================================================================
 mobilenetv2_1.00_224_input   [(None, 32, 32, 3)]      0
 (InputLayer)

 mobilenetv2_1.00_224 (Funct  (None, 1, 1, 1280)       2257984
 ional)

 global_max_pooling2d_1 (Glo  (None, 1280)             0
 balMaxPooling2D)

 dense_1 (Dense)              (None, 512)              655872

 dense_2 (Dense)              (None, 256)              131328

=================================================================
Total params: 3,045,184
Trainable params: 2,648,640
Non-trainable params: 396,544
```

داده های ورودی را به شبکه می دهیم و خروجی آن را تبدیل به آرایه numpy کرده و به Perceptron با نقطه خروج ۰. کلاس بندی می کنیم. دقت خروجی به صورت زیر می باشد.

دقت برابر می شود با :

```
accuracy_score = 0.4122
```



در مجموع در حل این مسئله چالش های زیر مطرح بودند.

- مشکل محروم و بسته شدن Runtime GPU در گوگل کلب برای Train قابل قبول شبکه MobileNetV2
- عدم آشنایی با اصطلاحات مانند Head شبکه و نوع و تایپ داده برای کلاس بندی
- عدم آشنایی ابتدایی با حذف لایه و fineTune کردن در ابتدای کار