



به نام خدا



دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

استاد

دکتر کلهر

مینی پروژه اول

شقايق طربخواه- سحر رجبى	نام و نام خانوادگى
۸۱۰۱۹۹۱۶۵-۸۱۰۶۹۸۱۹۳	شماره دانشجویی
۱۴۰۰/۲/۲۸	تاریخ ارسال گزارش

فهرست گزارش سوالات

3.....	سوال 1 – مفاهیم تئوری
9.....	CNN – 2
19	سوال 3 Data Augmentation – 3
24	سوال 4 Transfer Learning – 4

سوال ۱ – مفاهیم تئوری

• قسمت ۱

همانطور که می دانیم به منظور بهینه سازی شبکه های عصبی از روش گرادیان کاهاشی استفاده می شود.

الف) در مورد مشکلاتی که استفاده از این روش می تواند به وجود بیاورد توضیح دهید.

ب) در مورد عملکرد هر یک از روش های زیر توضیح دهید و بیان نمایید که آیا می توانند مشکلات روش گرادیان کاهاشی را حل نمایند یا خیر؟ (AdaDelta و adam و momentum برای گرادیان کاهاشی)

الف) به طور کلی، سه مشکل اساسی در طول به روزرسانی وزن ها با استفاده از gradientها ممکن است وجود داشته باشد: ۱- vanishing gradient: که در واقع به خاطر مقادیر بسیار کوچک گرادیان حاصل می شود. اگر گرادیان در یک مرحله بسیار کوچک باشد، با ضرب در مقادیر کوچک دیگر در سایر لایه های یک شبکه ای عصبی این مقدار تقریبا صفر خواهد شد؛ در نتیجه آپدیت وزن ها متوقف می شود. ۲- exploding gradient: این مشکل دقیقا عکس قسمت قبل است. ضرب مقادیر بزرگ در یکدیگر می تواند باعث رشد خیلی زیاد گرادیان و نوسانات مدل بدون همگرایی به نقطه ای بهینه شود. ۳- local minimum: در مینیمم های محلی، این روش امکان تشخیص ندارد و در نقاط عطف هم با توجه به مشتق نزدیک به صفر در هر دو سمت، مشکل خواهیم داشت.

در ادامه این موارد در قالب الگوریتم gradient descent بیشتر شکافته می شود.

یکی از مواردی که در استفاده از این روش حائز اهمیت است ضریب یادگیری است. این پارامتر باید به گونه ای انتخاب شود که هم دارای دقت لازم بوده و هم گام های بلندی نداشته باشد؛ تا در مواردی از روی اکسترمم ها عبور نکند (مشکل exploding gradient): چرا که در این صورت در همگرایی به مشکل خواهد خورد. اما از طرف دیگر، کوچک کردن این ضریب، سرعت همگرایی را تا حد زیادی کاهش داده و زمان زیادی برای آموزش صرف خواهد شد و شاید حتی دیگر به نتیجه نرسیم (در مشکل vanishing gradient، در صورتی که مقدار این پارامتر هم عددی کوچک باشد، باعث تشدید شدید موضوع و حتی متوقف شدن یادگیری خواهد شد). در نتیجه در استفاده از این روش نیاز است تا با تکنیک هایی مقدار این پارامتر تاثیرگذار را تعیین کنیم. گاه نیاز است تا این پارامتر در حین فرآیند یافتن پاسخ، تغییر کند (بر مبنای دیتابست) که ممکن است با توجه به ویژگی های داده های مسئله، توابع مربوطه در این حوزه دچار مشکل شوند و سازگاری نداشته باشد. همچنین با توجه به شرایط مسئله ممکن است تا در مورد داده های با حجم بالا، فرآیند به اندازه ای کند باشد که عملا فایده ای نداشته باشد. از طرفی دیگر ممکن است که ویژگی های مختلف موجود در مسئله، بسیار پراکنده بوده و فرکانس تکرار بالایی داشته باشند؛ که در این موارد نمی توان همه آن ها را به یک اندازه به روز رسانی کرد و نیاز است که با ضرایب مختلفی به روز رسانی صورت گیرد که در بسیاری از موارد امکان پذیر نمی باشد.

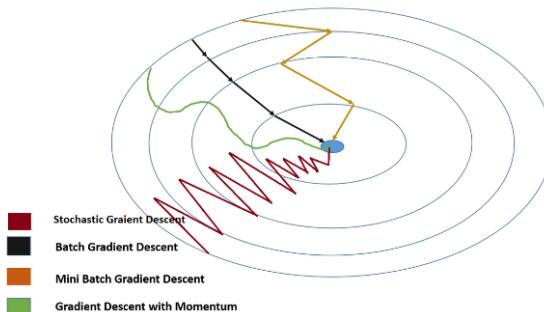
همچنین همانطور که در قسمت سوم گفته شد؛ باید توجه داشت که مقداردهی های اولیه، در فرآیند یافتن نقاط اکسترمم، بسیار تاثیرگذار است و ممکن است با مقداردهی های اولیه نامناسب، گرفتار اکسترمم های محلی شده و به مقادیر مطلق همگرا نشونیم که نیاز است تا این شرایط اولیه را تغییر داده و مجددا تلاش کنیم. چالش دیگر، بحث کاهش توابع خطای بسیار غیر محدب است. مشکل ایجاد شده در این موارد، بیشتر در ارتباط با نقاط در نواحی زینی شکل می باشد چرا که شیب در تمامی جهات نزدیک به صفر است.

ب) هر کدام از این روش‌ها به اختصار بررسی شده‌اند:

Momentum with gradient descent •

این روش در واقع به بهبود روش SGD کمک می‌کند و نوسانات مربوطه در فرآیند رسیدن به نقطه اکسترم را کاهش می‌دهد. این عمل را از طریق افزودن یک پارامتر γ به بردار بهروزرسانی انجام می‌دهیم؛ به این صورت که علاوه بر گردیان این مرحله، تغییرات مرحله‌ی قبل را هم در این لحظه دخیل می‌کنیم (ضریب γ در این مقدار که می‌توان آن را به عنوان حافظه‌ی گذشته در نظر گرفت؛ ضرب می‌شود و جهت تغییرات این مرحله را تا حدی هم‌راستای تغییرات پیشین می‌کند). در نتیجه این فرآیند در مسیر رسیدن به مسیر حرکت به سمت نقطه اکسترم هموار‌تر شده که منجر به افزایش سرعت همگرایی می‌شود (در مسیر رسیدن به مینیمم، کمتر شاهد انحرافات به چپ و راست خواهیم بود که در نتیجه، به روزرسانی‌های کمتری خواهیم داشت). استفاده از momentum به نفع رفع یکی از نقصان‌های روش گرادیان کاهشی می‌باشد.

به عبارت دیگر همانند توپی که از یک بلندی سرازیر می‌شود که هر چه به سمت پایین حرکت می‌کند سرعت می‌یابد (در جهتی که تغییرات ارتفاع وجود دارد) در این روش نیز پارامتر گفته شده در راستاهایی که تغییرات تابع بیشتر است، مقدار بیشتری دارد و به طور طبیعی سرعت همگرایی را افزایش می‌دهد. همچنین در مواردی کاهش نوسانات منجر می‌شود تا نوسان حول نقطه اکسترم نیز کاهش یابد (اما همچنان ممکن است در مواردی حول نقطه اکسترم نوسان کند). شکل ۱ می‌تواند برای مقایسه‌ی روش momentum با سایر الگوریتم‌های مشتق‌شده از gradient descent مفید باشد.



شکل ۱- مقایسه‌ی روش gradient descent with momentum با سایر روش‌های پایه‌ای

Adam •

این روش یکی از الگوریتم‌هایی است که در حوزه یادگیری ماشین در بسیاری از موارد به جای روش sgd استفاده می‌شود. این روش در واقع ترکیبی از روش‌های AdaGrad و RMSprop می‌باشد که به اختصار توضیح می‌دهیم.

در learning rate AdaGrad یا adaptive gradient algorithm در نظر گرفته می‌شود که می‌تواند باعث بهبود الگوریتم یادگیری، مخصوصاً در مسائلی با گرادیان‌های sparse شود.

روش RMSProp یا root mean square propagation هم ویژگی بالا دارد به علاوه اینکه نرخ یادگیری را با توجه به میانگین تغییرات گرادیان آن‌ها در دوره‌ای گذشته، تنظیم می‌کند.

بهطور خلاصه Adam نرخ یادگیری را به تدریج بر حسب میانگینی از momentum اول و دوم تعییر می‌دهد و بهینه سازهای مرتبه دوم سریعتر به نقطه بهینه با توجه به انحنا منحنی دست میابند و ضربه یادگیری هه در این روش به صورت Adaptive Gradient تغییر خواهد کرد؛ در نتیجه سریعتر از بهینه ساز descent همگرا خواهد شد.

AdaDelta •

این روش در واقع توسعه یافته AdaGrad می‌باشد. این روش سعی می‌کند مشکل diminishing learning rate را که در AdaGrad اتفاق می‌افتد، به این صورت حل کند که به جای اینکه از لحظه‌ی صفر t را در نظر بگیرد؛ تنها به اندازه‌ی پنجره‌ای ثابت در نظر بگیرد و به این ترتیب، تنها تعییرات اخیر در گرادیان‌ها، بر گرادیان‌های بعدی اثر خواهد گذاشت.

این روش مزایای روش AdaGrad را دارد خواهد بود؛ یعنی برای استفاده از این روش، نیازی نیست که مقدار نرخ یادگیری را تعیین کنیم؛ چرا که به تدریج، خود الگوریتم آن را بهینه می‌کند و همچنین نرخ یادگیری را برای هر پارامتر متغیر در نظر می‌گیرد پس ما یادگیری سریع‌تر و بهینه‌تری خواهیم داشت و همچنین مشکلات تعیین مقدار مناسب نرخ یادگیری هم دیگر وجود ندارد. همچنین مشکل diminishing learning rate هم، با در نظر گرفتن یک پنجره‌ی ثابت، رفع خواهد شد.

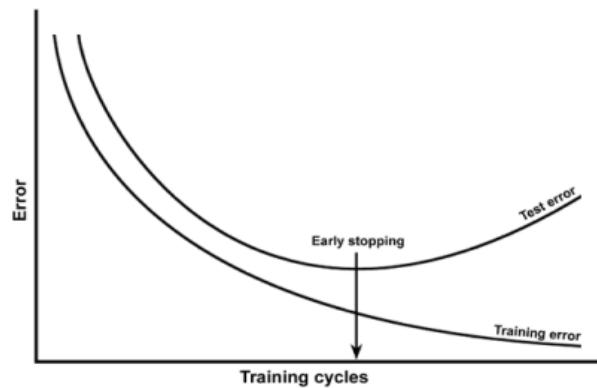
• قسمت ۲

بیش برآش در طبقه بندی را توضیح دهید برای حل این مشکل از سه روش norm ، drop out و early stopping penalty و بیان کنید که روش‌های مذکور چگونه می‌توانند این مشکل را حل نمایند.

مسئله Overfitting در شبکه‌هایی به وجود خواهد آمد که عملکرد مدل آموزش‌دیده، برای داده‌های train و validation بسیار بالاتر از داده‌های تست است. و برای داده‌های جدیدی که شبکه تا به حال آن‌ها را ندیده و در آموزش استفاده نشده، بسیار عملکرد ضعیفی دارد. به عبارتی شبکه برای داده‌های جدید قابل تعمیم نیست. این پدیده هنگامی به وجود خواهد آمد که مدل مربوطه داده‌های آموزش را با جزئیات زیاد و حتی اثربخشی از نویزها یاد بگیرد که تاثیر منفی بر عملکرد شبکه برای داده‌های جدید خواهد داشت. در این مدل‌ها، نویز و اغتشاشات یا اعوجاج‌های درون ویژگی‌های ورودی به عنوان اصل و ویژگی‌هایی از شبکه شناخته خواهد شد و یادگیری بر این اساس انجام خواهد گرفت در حالی که این مقادیر نویز و اغتشاشات به صورت رندوم و اتفاقی می‌باشد و به صورت یکسان روی داده‌های جدید موجود نخواهد بود یا به شکل دیگری وجود خواهد داشت، در نتیجه شبکه قادر به پیش‌بینی خوبی برای داده‌های جدید نخواهد شد.

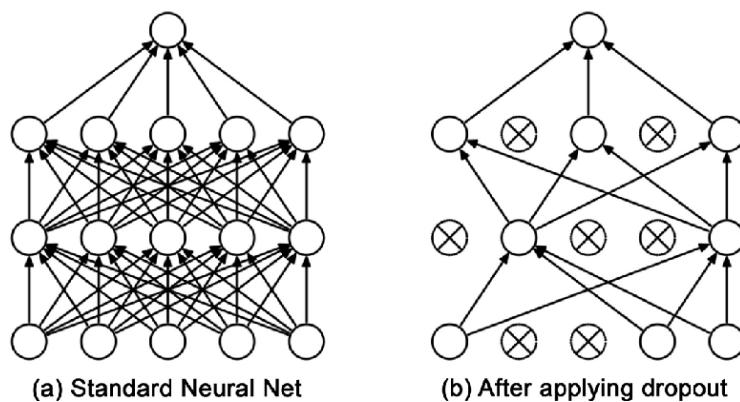
• اعتبارسنجی شبکه هنگام آموزش (early stop) : استفاده از داده‌های validation برای مشاهده چگونگی تعییرات عملکرد شبکه برای داده‌های آموزش و آن‌هایی که در آموزش استفاده نشده است. اگر هنگام ترین شبکه منحنی خطرا برای هر دو دسته داده رسم کنیم، جایی که هر دو نمودار کاهش یابند، شبکه در بهترین حالت ترین شده است. در واقع طی این فرآیند در هر مرحله پیش‌بینی بهتری از داده‌ها صورت می‌گیرد و روند کلی خطرا برای داده‌های ارزیابی نزولی می‌باشد. اما در برخی از موارد، نقطه‌ای فرا می‌رسد که از این نقطه به بعد اگر آموزش شبکه ادامه یابد، خطای مربوط به داده‌های تست و اعتبارسنجی افزایش خواهد یافت در حالی که برای داده‌های آموزش کاهش خواهد یافت. به عبارت دیگر قدرت تعمیم شبکه برای داده‌های جدید کاهش خواهد یافت. در این روش، با

تشخیص این اتفاق در طول یادگیری، به محض رخداد آن، آموزش شبکه متوقف خواهد شد. این فرآیند بیشتر در شبکه‌های با مقادیر پیوسته (Regression) به کار گرفته می‌شود.



شکل ۲- توضیح نحوه عملکرد early stopping؛ همانطور که مشخص است با شروع افزایش خطا در داده‌های تست، آموزش

Drop out •
با خاموش کردن تصادفی خروجی برخی از نورون‌ها حین فرایند آموزش، قدرت یادگیری شبکه با اطلاعات ورودی کمتر را بالا خواهد برد و باعث می‌شود با اطلاعات مفید کمتری، بتواند خروجی مطلوب را در اختیار ما قرار دهد. به عبارت دیگر در این روش با این عمل شبکه‌های متفاوتی را در مراحل مختلف آموزش می‌دهیم که این عمل سبب عدمیت بخشیدن به مسئله و نیز جلوگیری از بیش برآش خواهد شد. نمونه‌ای از این روش در زیر مشاهده می‌شود.



شکل ۳- بعد از اعمال dropout بخشی از شبکه به صورت تصادفی حذف می‌شود تا از overfitting جلوگیری کند.

Norm Penalty •
یکی دیگر از روش‌هایی که برای جلوگیری از بیش برآش استفاده می‌شود افرادن یک ترم به تابع خطا می‌باشد تا از زیاد شدن برخی از پارامترهای مسئله در فرآیند آموزش شبکه جلوگیری شود. به این ترتیب تابع خطا به صورت زیر بازنویسی می‌شود:

$$J'(\theta) = J(\theta) + \alpha \sigma(\theta)$$

که در رابطه بالا مقدار $\alpha > 0$ بوده و به عنوان یکی از هایپرپارامترهای مسئله محسوب می‌شود و باید بر اساس شرایط مسئله تعیین گردد و در واقع میزان مشارکت $(\theta) \sigma$ را تعیین می‌کند. به طور معمول در حوزه شبکه‌های عصبی، از این روش برای محدود کردن ضرایب تحول نورون‌ها استفاده

شده و در مورد مقادیر بایاس اعمال نمی‌گردد چرا که در اکثر موارد مقادیر بایاس با داده‌های کمتری در مقایسه با مقادیر وزن‌ها تعیین می‌گردد و چه بسا در برخی موارد با این عمل سبب under fit شدن مدل گردد. در نهایت با استفاده از این روش، با محدود کردن مقادیر وزن‌ها می‌توان از حساسیت بیش از اندازه‌ی شبکه در مقابل تغییرات ورودی کاست؛ که سبب می‌شود مدل در مواجهه با داده‌های جدید، بهتر عمل کند.

• قسمت ۳

می‌دانیم که شبکه‌های عصبی تنها با دولایه مخفی خاصیت general function approximator بودن خود را حفظ می‌کنند. با این وجود چرا گاهی بیش از دولایه مخفی برای شبکه‌ها استفاده می‌شود؟

همانطور که در داخل درس مشاهده کردیم با استفاده از شبکه‌های با دو لایه مخفی می‌توان به مدلسازی سیستم‌های پیشرفت‌نمای (non convex) پرداخت و نیز قدرت تعمیم شبکه را افزایش داد (نسبت به شبکه‌های با یک لایه مخفی که قدرت برونویابی کمتری دارند). اما افزودن لایه‌های بیشتر به شبکه می‌تواند فوایدی را به ارمغان بیاورد:

• Dimension Reduction

در واقع ابعاد افزوده شده این قدرت را دارند که correlation بین ابعاد را بشکند و در نتیجه این امر، می‌توان ابعاد ویژگی‌های مسئله را کاهش داد که منجر به کاهش حجم و هزینه محاسبات شده و در بسیاری از موارد نتایج حاصل از مدلسازی را به میزان چشمگیری بهبود بخشد.

• Data Filtering and Scaling

افزودن تعداد لایه‌های مخفی، علاوه بر آنچه در بالا گفته شد، می‌تواند منبع disturbance و نیز distortion را در داده‌ها شناسایی و رفع نماید که طبیعتاً انتظار می‌رود نتایج بهتری از شبکه حاصل شود. در واقع این لایه‌ها به نحوی نقش Filter کردن داده‌ها را بر عهده می‌گیرند. از طرفی یکی از مواردی که می‌تواند به نتایج بهتری برسد هم Scale کردن داده‌های مختلف است تا تاثیر این پارامتر را در نتیجه از بین ببریم.

در نهایت می‌توان گفت که در استخراج ویژگی‌های مرتبط و مفید بسیار کارساز است.

همچنین می‌توان گفت اضافه کردن یک لایه به شبکه، در واقع افزایش یک لایه non-linearity دیگر است؛ که در نتیجه‌ی آن، مدل ممکن است بتواند به سرعت بیشتری الگوهای پیچیده را یاد بگیرد؛ همچنین اگر ما سه لایه با ۱۰ نورون داشته باشیم، ۲۰۰ پارامتر برای یادگیری داریم در حالی که اگر دو لایه با ۱۵ نورون داشته باشیم، ۲۲۵ پارامتر برای یادگیری خواهیم داشت. در نتیجه‌ی این دو مورد می‌توان برای مدل کردن یک الگو، با تعداد پارامترهای کمتر هم از افزایش لایه استفاده کرد. هرچند که این کار می‌تواند با افزایش خیلی زیاد non-linearity در برخی کاربردها، باعث overfit شدن شبکه و گرفتن نتیجه‌ی معکوس باشد.

• قسمت ۴

در طراحی شبکه‌های عصبی عمیق چرا در انتخاب تعداد لایه‌ها یا تعداد پارامتر هر لایه محدود هستیم؟

همانطور که واضح است هنگامی که تعداد نورون‌ها و یا تعداد لایه‌های مخفی افزایش پیدا می‌کند محاسبات لازم برای به دست آوردن مقادیر صحیح هم با سرعت افزایش پیدا می‌کند و این بدان معناست که

ممکن است از نظر زمان اختصاص داده شده برای آموزش شبکه دچار مشکل شویم و شبکه مورد نظر در عمل کابردی نخواهد داشت.

از طرفی با افزایش هر چه بیشتر تعداد لایه‌ها و تعداد نورون‌ها، آموزش شبکه با مشکل vanishing gradient مواجه خواهد شد. همانگونه که میدانیم، به روز رسانی وزن‌ها و بایاس‌ها به روش Error back propagation انجام می‌شود و شبکه آموزش می‌بیند. حال هر چقدر تعداد لایه‌ها و نورون‌ها بیشتر باشد، تاثیر خطا (اختلاف بین مقدار پیش‌بینی شده و مقدار واقعی) بر بهروزرسانی وزن و بایاس لایه‌های اخر کمتر می‌شود؛ چرا که گرادیان خطا در لایه‌های ابتدایی به خاطر ضرب‌های زیاد در مقادیر کوچک، ممکن است ناچیز باشد و این امر باعث می‌شود آموزش لایه‌های اول شبکه با مشکل مواجه شده و یا حتی متوقف شود. در نتیجه کیفیت مدل آموزش دیده، کیفیت مطلوبی نخواهد داشت.

مشکل بعدی که بر اثر افزودن نورون و لایه، ممکن است رخ دهد؛ این است که همانطور که می‌دانیم در برخی موارد افزودن پارامترهای بیشتر به مسئله نه تنها پیچیدگی مدل و حجم محاسبات مربوطه را افزایش می‌دهد بلکه منجر به over parametrization یا overfitting می‌شود که به این معناست، که مدل بیش از اندازه داده‌های آموزش را دنبال کرده و به نحوی آنها را حفظ می‌کند و در نتیجه در مواجهه با داده‌های جدید، عملکرد مطلوبی از خود نشان نخواهد داد و قابلیت عمومیت‌بخشی آن کاهش می‌یابد.

• قسمت ۵

در یک شبکه عصبی اگر تمامی وزن‌های اولیه یکسان تعیین شوند چه مشکلی ایجاد می‌شود؟

فرض کنید که از تابع sigmoid به عنوان فعالساز در مسئله استفاده شود و برای تمامی این پارامترها از مقدار X استفاده شود. در این صورت تمامی نورون‌های یک لایه، سیگنال برابر را دریافت می‌کنند ضرب خطی وزن‌های آن نورون در خروجی لایه‌ی قبل و یا ورودی شبکه) که این امر سبب می‌شود تا مشتق‌های موجود در تابع هزینه مشابه شوند. این اتفاق در نهایت منجر می‌شود تا تمامی واحدهای لایه‌های مخفی یکسان با یکدیگر عمل کرده و در ایتریشن‌های بعدی نیز همین روند ادامه می‌یابد و یادگیری شبکه دچار اختلال می‌شود. به همین دلیل، انتخاب رندم وزن‌ها، منجر به این خواهد شد که هر نورون، بخش خاصی را مدل کند و نه اینکه همگی مشابه یکدیگر عمل کنند.

• قسمت ۶

تفاوت exploding gradient و vanishing gradient را بیان نمایید.

در الگوریتم back propagation، گرادیان‌ها در یکدیگر ضرب می‌شوند. مسئله‌ی این دو مشکل دقیقاً از ضرب این گرادیان‌ها سرچشمه می‌گیرد:

در صورتی که اندازه‌ی گرادیان‌ها بسیار بزرگ باشد، ضرب تعدادی از آن‌ها در یکدیگر می‌تواند باعث رشد خیلی زیاد آن مقدار، و در نتیجه خروجی بسیار بزرگ شود. استفاده از این خروجی می‌تواند مدل را به سمت نقطه‌ای بسیار دور از نقطه‌ی بهینه بکشاند. این مشکل، exploding gradient نام دارد.

در مقابل اگر این مقادیر خیلی کوچک باشند، ضرب تعدادی از آن‌ها باعث می‌شود که در لایه‌ی آخر این مقدار برابر با صفر شود! در نتیجه عملاً گرادیان ما باعث تغییر وزن‌ها نخواهد شد و عملاً یادگیری صورت نمی‌گیرد. این مشکل vanishing gradient نام دارد.

سوال 2 - CNN

• قسمت ۱

مشخصات شبکه عصبی خود را بنویسید.

(الف) اندازه پنجره های convolution و اندازه stride و تعداد filter ها در هر لایه

(ب) توابع فعالسازی مورد استفاده

(ج) اندازه لایه های fully connected

(د) تابع loss و روش بهینه سازی مورد استفاده

(ه) اندازه mini-batch مورد استفاده .

در اینجا معماری شبکه طراحی شده برای طبقه بندی دیتابست موجود را می آوریم:

جدول ۱: پارامترهای مربوط به معماری شبکه

	No. filters	Filter window	Stride	Activation Function	Pooling size
Conv. 1	16	(3*3)	1	relu	2
	32	(3*3)	1	relu	
Conv. 2	32	(3*3)	1	relu	2
	64	(3*3)	1	relu	
Conv. 3	64	(3*3)	1	relu	2
	128	(3*3)	1	relu	

در جدول زیر اطلاعات تابع بهینه ساز و تابع هزینه آورده شده است:

جدول ۲: توابع هزینه و بهینه ساز

Optimizer	Loss Function	Metrics
Compiler	Adam	Categorical crossentropy

تعداد لایه مخفی fully connected برابر با ۲ قرار داده شده است (به غیر از لایهی خروجی که ۱۰ نورون دارد) و در جدول ۳، تعداد نورون و توابع فعالساز مربوطه آورده شده است.

جدول ۳: هایپرپارامترهای مربوط به لایه‌های مخفی

	No. neurons	Activation Function
Fully connected. 1	300	relu
Fully connected. 2	100	relu

همچنین در این بخش برای گرفتن نتیجه بهتر از drop out در میان لایه‌های Fully connected استفاده شده است؛ که اطلاعات آن‌ها بدین ترتیب است.

جدول ۴: اطلاعات مربوط به لایه‌های drop out شبکه‌ی استفاده شده

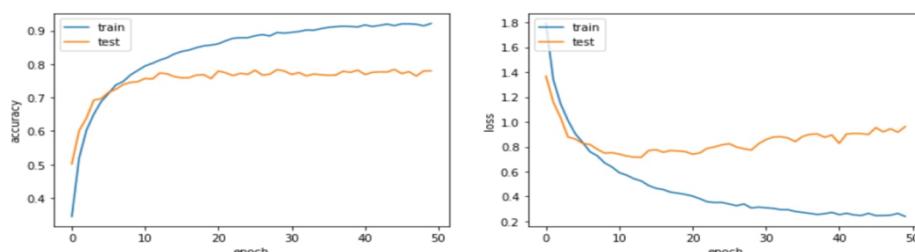
	size	position
Drop Out 1	0.2	After conv. layers
Drop Out 2	0.3	After first FC layer
Drop Out 3	0.3	After second FC layer

و سایز mini batch هم برابر با ۱۲۸ قرارداده شده است.

• قسمت ۲

شبکه عصبی را طراحی کرده اید را اجرا و نمودار دقت بر حسب ایپاک برای داده‌های یادگیری و تست رسم کنید.

در ادامه نمودار دقت و خطای آموزش شبکه مشاهده می‌شود:



شکل ۴- نمودار دقت و خطای آموزش برای داده‌های آموزش و ارزیابی

و دقت ما در بر روی داده‌ها آموزش و ارزیابی، در جدول زیر گزارش شده است:

جدول ۵: بیشترین دقت بدست‌آمده بر روی داده‌های آموزش و ارزیابی

	Accuracy	epoch
Train	92.52	46
Validation	78.44	45

مشاهده می شود که بعد از 50 مرحله دیگر تغییر مشخصی در فرایند آموزش شبکه صورت نمی‌گیرد پس نیازی نیست که بعد ازین مرحله فرآیند آموزش را ادامه دهیم. همچنین در ایپاک ۵۰، دقت ما بر روی دادگان validation برابر ۷۸ درصد است.

تصویر زیر خلاصه‌ی عملکرد مدل، با معیارهای مختلف، بر روی مجموعه‌ی دادگان تست ماست:

	precision	recall	f1-score	support
0	0.81	0.76	0.78	1000
1	0.91	0.86	0.88	1000
2	0.74	0.60	0.66	1000
3	0.55	0.60	0.57	1000
4	0.68	0.72	0.70	1000
5	0.59	0.77	0.67	1000
6	0.82	0.77	0.79	1000
7	0.81	0.79	0.80	1000
8	0.90	0.83	0.86	1000
9	0.85	0.86	0.86	1000
accuracy			0.76	10000
macro avg	0.77	0.76	0.76	10000
weighted avg	0.77	0.76	0.76	10000

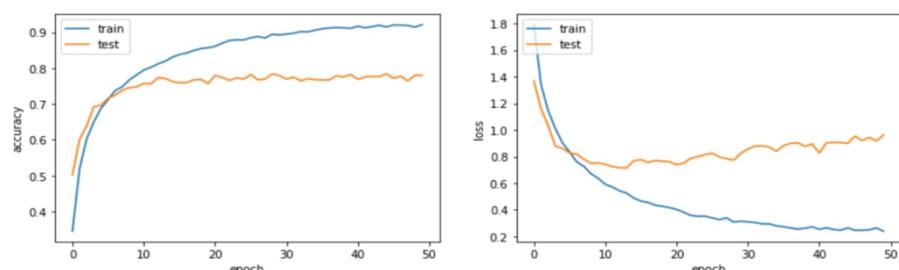
شکل ۵- عملکرد شبکه در معیارهای مختلف، بر روی داده‌های آزمون

• قسمت ۳

کارایی شبکه طراحی شده را با ۰ و ۱ و ۲ لایه مخفی بررسی نمایید و نمودار دقت را برای داده‌های یادگیری و تست رسم کنید.

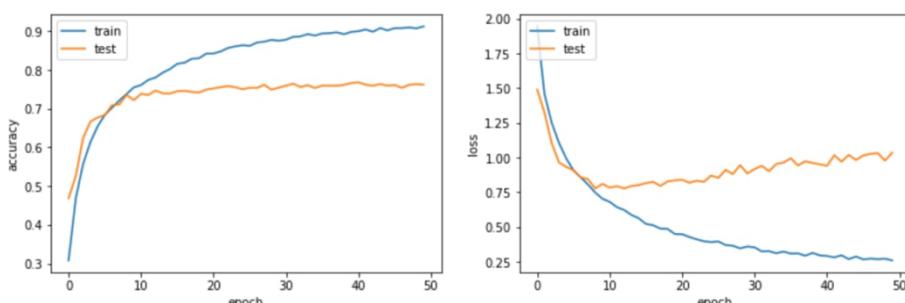
در این بخش خواسته شده است که با تغییر در تعداد لایه‌های پنهان نتیجه را مشاهده کنیم. که نتایج به شرح زیر است:

(الف) با استفاده از ۲ لایه مخفی داریم:



شکل ۶- نمودار مقدار خطأ و دقت برای داده‌های آموزش و آزمون در شبکه‌ای با ۲ لایه‌ی پنهان

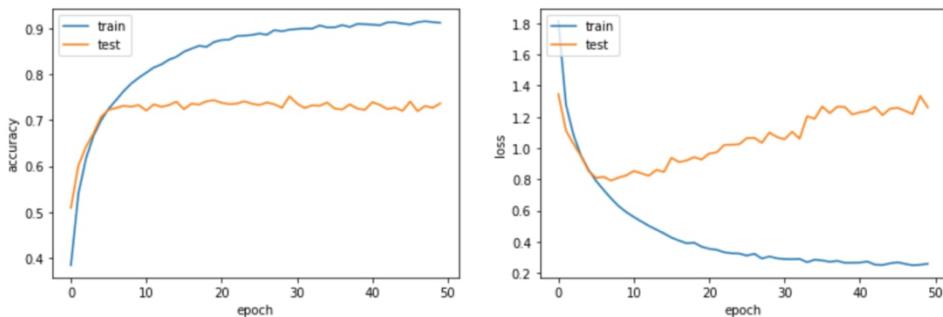
(ب) با استفاده از ۱ لایه مخفی داریم:



شکل ۷- نمودار مقدار خطأ و دقت برای داده‌های آموزش و آزمون در شبکه‌ای با ۱ لایه‌ی پنهان

با استفاده از یک لایه، دقت مدل بر روی دادگان validation مقدار کمی کاهش یافته.

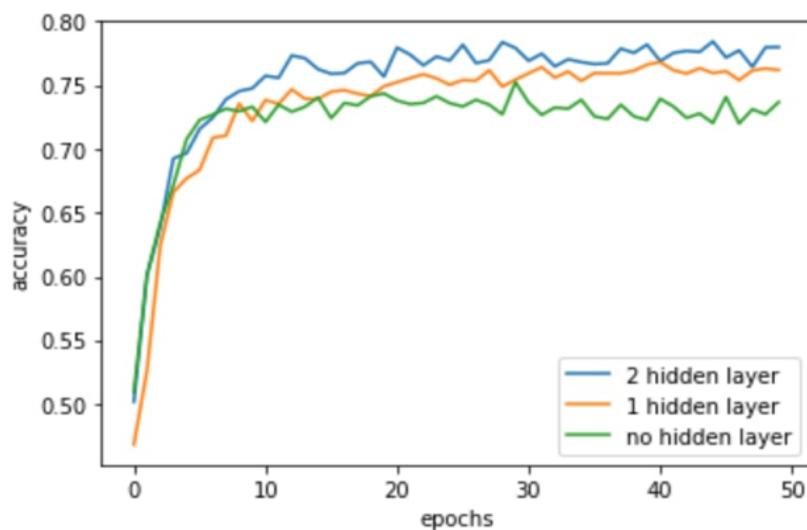
(ج) با استفاده از ۰ لایه مخفی داریم:



شکل ۸- نمودار مقدار خطا و دقت بر روی داده‌های آموزش و آزمون در شبکه‌ای با صفر لایه‌ی پنهان

مشاهده می شود که در مورد صفر لایه مخفی علاوه بر اینکه قدرت تعمیم شبکه کاهش یافته است؛ نمودار خطای دادگان ارزیابی از حدود مرحله 10 ام به بعد در حال افزایش چشمگیر است.

تصویر زیر، مقدار خطا بر روی دادگان ارزیابی را برای این سه مدل، در کنار هم رسم کرده و مقایسه می کند:



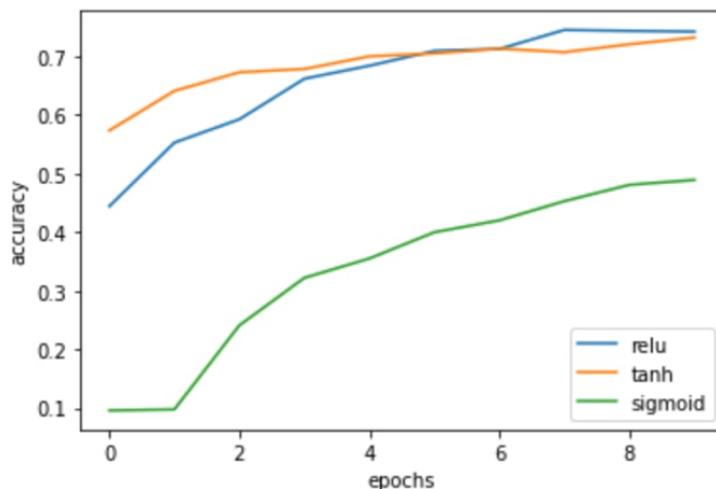
شکل ۹- مقایسه دقت عملکرد بر روی داده‌ها ارزیابی برای سه مدل با صفر، یک و دو لایه‌ی پنهان

همانطور که مشخص است، با حذف لایه‌های مخفی، دقت مدل هم کاهش پیدا کرده است.

• قسمت 4

از توابع فعالساز مختلف (tanh ، sigmoid و RELU) استفاده کنید و نتایج حاصل از این 3 حالت را بر روی یک نمودار رسم نمایید. (تا 10 ایپاک) نتیجه را تحلیل نمایید.

در این بخش به دنبال آن هستیم که تاثیر تغییر تابع فعالساز را بر روی نتایج حاصل از طبقه مشاهده کنیم. در ادامه نمودار دقت در فرآیند آموزش بر روی داده‌های ارزیابی بهازای توابع مختلف رسم شده است.



شکل ۱۰- مقایسه عملکرد شبکه در آموزش، با سه تابع relu , tanh و sigmoid در ۱۰ ایپاک ابتدایی بر روی داده‌های ارزیابی

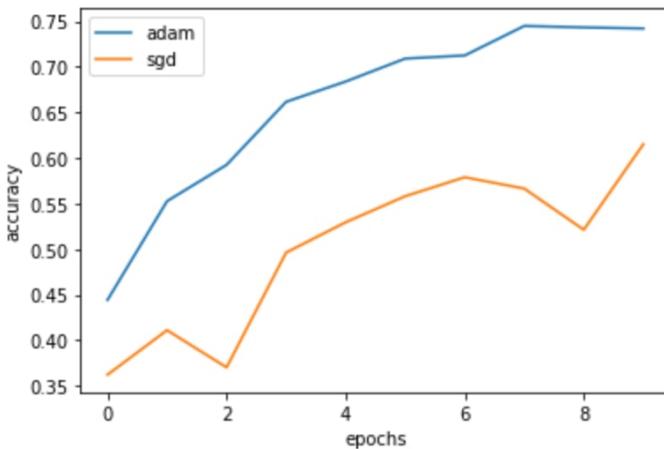
همانگونه که انتظار میرفت شبکه ترین شده با تابع فعالساز relu نسبت به دو تابع دیگر بهتر عمل کرده است البته در این مورد هر دو تابع فعالساز نتایج تقریباً یکسانی را ارائه می‌دهد. همچنین در مقایسه بین relu و tanh به این نتیجه خواهیم رسید که شبکه با تابع فعالساز Sigmoid نیاز به زمان بیشتری برای ترین دارد زیرا بازه خروجی محدودتر (صفر تا یک) است و عمل Error Back Propagation بیشتر طول خواهد کشید زیرا که پدیده vanishing gradient problem رخ خواهد داد.

به دلیل همان علت بیان شده در حالت اول، شبکه با تابع فعالساز Sigmoid عملکرد خوبی نشان نمی‌دهد و زمان بیشتری برای ترین نیاز دارد علاوه بر موارد ذکر شده در شبکه‌هایی که تعداد لایه‌ها زیاد است و با شبکه عمیق سروکار داریم مشکل از بین رفتن تاثیر گرادیان لایه‌های اخر در لایه‌های ابتدایی به مراتب بیشتر می‌شود زیرا که در یکدیگر ضرب خواهند شد و به صورت نمایی تاثیر آن در یادگیری کاهش خواهد یافت و اگر شبکه زمان بیشتری صرف شود شاید دقت بالاتری در اختیار ما بگذارد. با توجه به اینکه دو لایه مخفی در شبکه عمیق بعد کانولوشن به کار گرفته شده است این پدیده با قوت بیشتری رخ خواهد داد. اما تابع tanh خروجی‌ای بین ۱ و -۱ را دارد که به طور میانگین حول صفر قرار دارند و نسبت به مرکز متقارنند؛ و این ویژگی یک برتری آماری است که منجر به یادگیری سریع‌تر این تابع نسبت به sigmoid می‌شود.

• قسمت ۵

از روش‌های بهینه سازی gradient descent و adam استفاده نموده و نمودار دقت بر حسب ایپاک را تا ۱۰ ایپاک بر روی یک نمودار رسم کنید.

با تغییر در تابع بهینه ساز نتایج زیر حاصل می‌شود:



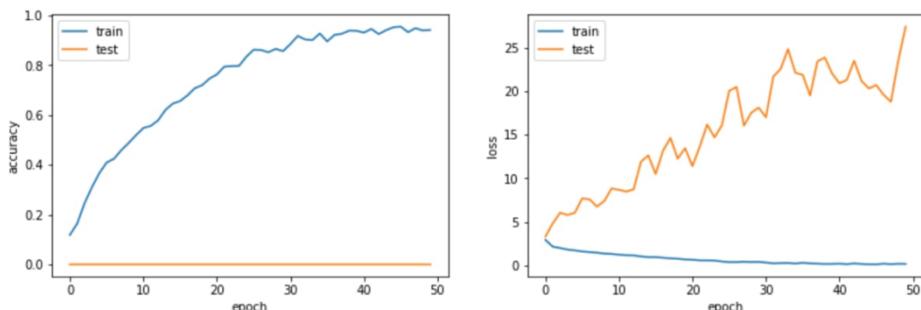
شکل ۱۱- مقایسه روند یادگیری در ۰۰ ایپاک اول، بر روی داده‌های آموزش و آزمون با دو بهینه‌ساز adam و sgd

به علت اینکه روش Adam ترکیبی از Rmsprop و Momentum می‌باشد، و بهینه‌سازهای مرتبه دوم سریعتر به نقطه بهینه با توجه به اینها منحنی دست می‌یابند و همچنین کاهش نوسانات به خاطر استفاده از momentum و اینکه ضریب یادگیری در این روش به صورت Adaptive Gradient descent تغییر خواهد کرد، سریعتر و کم‌نوسان‌تر از بهینه‌ساز Gradient descent همگرا خواهد شد در حالی که زمان مورد نیاز برای بهینه‌ساز Gradient descent با توجه به اینکه بهینه‌ساز مرتبه اول است، به مراتب بیشتر می‌باشد.

• قسمت ۶

حجم داده‌ها را کاهش دهید تا در هر کلاس تنها 600 نمونه وجود داشته باشد و با اجرای شبکه نمودار دقت و رسم کرده و نتیجه را تحلیل نمایید.

می‌توان تغییرات دقت و خطای را بعد از کاهش حجم دادگان، در تصویر زیر مشاهده کرد.



شکل ۱۲- نمودار تغییرات خطای دقت برای داده‌های آموزش و آزمون بعد از کاهش تعداد نمونه‌های هر کلاس به ۶۰۰ عدد

همانطور که پیداست، داده‌های آموزش با دقت نزدیک به ۱۰۰ درصد! آموزش یافته‌اند؛ اما خطای دادگان ارزیابی، پیوسته افزایش یافته و دقتی حاصل نشده. این مشاهده بیانگر آن است که مدل ما توانسته دادگان ورودی را حفظ کند! و ما در اینجا شاهد overfitting با بیشترین شدت خود هستیم.

تعداد دادگان آموزش کم، باعث شده است که مدل ما قابلیت generalization را دارا نباشد و نتواند تصاویر ندیده را شناسایی کند.

• قسمت 7

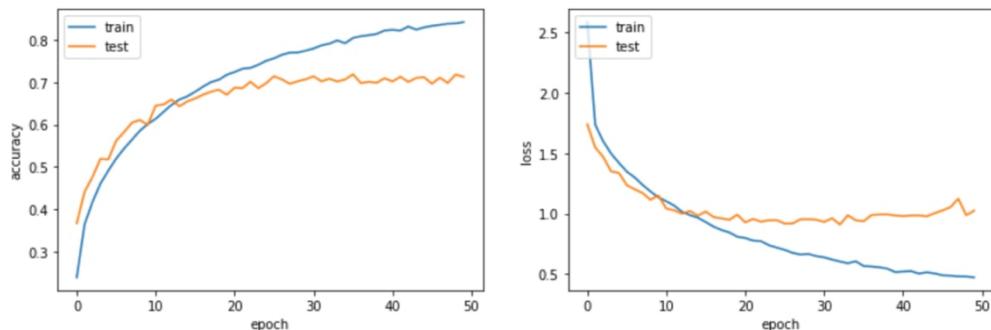
به جای دو عمل کانولوشن در هر لایه یک عمل کانولوشن با کرنل بزرگتر قرار دهید و عملکرد را بررسی نموده و تحلیل کنی.

همانطور که در مشخصات شبکه در بخش های قبلی مشاهده کردیم در هر یک از لایه ها از ترکیب دو فیلتر استفاده کردیم حال در این بخش مطابق خواسته سوال یکی از این فیلتر ها را در هر مرحله حذف کرده و در عوض یک کانولوشن با کرنل بزرگتر جایگزین می کنیم. معماری شبکه فوق در زیر نوشته شده است.

جدول ۶: معماری لایه های convolutional بعد از استفاده از کرنل بزرگتر

	No. filters	Filter window	Stride	Activation Function	Pooling size
Conv. 1	16	(5*5)	1	relu	2
Conv. 2	32	(5*5)	1	relu	2
Conv. 3	64	(5*5)	1	relu	2

نمودار دقت و خطأ در فرآیند آموزش شبکه در شکل مشاهده می شود.



شکل ۱۳- نمودار تغییرات دقت و خطأ بعد از جایگزینی لایه های با دو کانولوشن و سایز کرنل ۳ با لایه های تک کانولوشن و اندازه کرنل ۵

در نمودار بالا مشاهده می شود که از مرحله حدود 20 به بعد مقدار خطأ در داده های ارزیابی افزایش می یابد. که نشان دهنده آن است که مدل دجار overfit شده است. همچنین به نظر می رسد از مرحله 10 به بعد تغییر چندانی در نمودار های دقت برای داده های آموزش و ارزیابی افزایش نمی یابد.

در ادامه نتایج حاصل از این طبقه بند برای کلاس های مختلف مشاهده می کنیم.

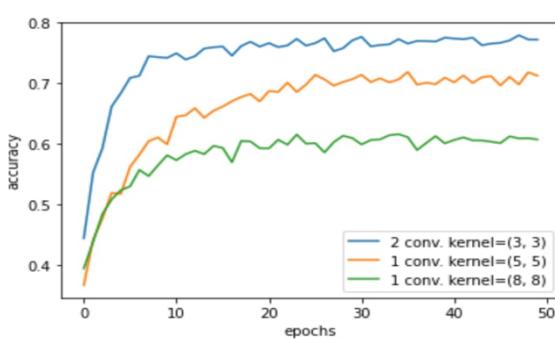
	precision	recall	f1-score	support
0	0.75	0.74	0.75	1000
1	0.83	0.85	0.84	1000
2	0.63	0.58	0.60	1000
3	0.49	0.50	0.49	1000
4	0.62	0.68	0.65	1000
5	0.62	0.56	0.59	1000
6	0.68	0.83	0.75	1000
7	0.84	0.68	0.75	1000
8	0.84	0.80	0.82	1000
9	0.76	0.80	0.78	1000
accuracy			0.70	10000
macro avg	0.71	0.70	0.70	10000
weighted avg	0.71	0.70	0.70	10000

شکل ۱۴- عملکرد طبقه بند بر داده های آزمون بعد از افزایش اندازه کرنل و تبدیل دو کانولوشن به یک.

با مقایسه این دو روش متوجه می شویم از طرفی زمان اجرای این دو حالت بسیار با یکدیگر متفاوت است و با استفاده از پنجه ای به اندازه (50) زمان و هزینه محاسباتی با اندازه قابل توجهی افزایش می یابد. در واقع یکی از دلایلی که محققان را ترغیب می کند تا از کرنل های با سایز پایین استفاده کنند حجم پایین محاسباتی است.

از طرفی به نظر می رسد که در مورد کرنل سایز های کوچکتر مدل بهتر می تواند پترن های ریز تر را تشخیص دهد. این مسئله سبب می شود که جزئیات بیشتری را در داخل داده ها تشخیص داده و طبیعتاً می تواند ویژگی های بیشتری را استخراج کند. البته که این مسئله بستگی به سایز داده های موجود دارد و باید نسبی در نظر گرفته شود. در اینجا مشاهده می شود که (3x3) نه تنها زمان بسیار کمتری را برای آموزش شبکه نیاز دارد بلکه نتایج بهتری به دست می آید.

تصویر زیر، مقایسه ای بین مدل های با 2 کانولوشن در هر لایه و اندازه کرنل 3، 1 کانولوشن در هر لایه با اندازه کرنل 5 و یک کانولوشن در هر لایه با اندازه کرنل 8 انجام داده است. مطالب گفته شده در بالا هم قابل تایید توسط نمودارهای زیر است:



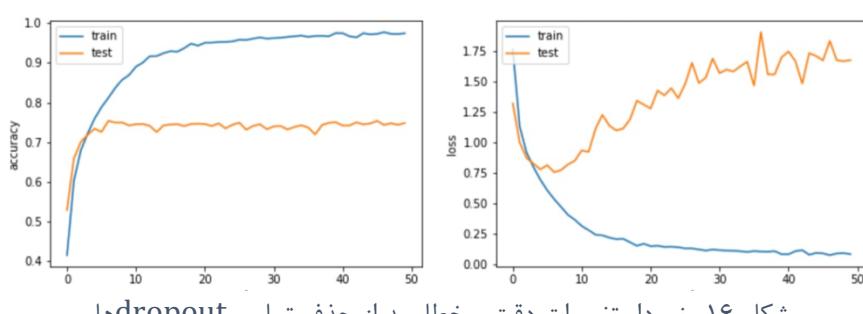
شکل ۱۵- مقایسه عملکرد اندازه کرنل و تعداد کانولوشن در هر لایه بر روی داده های ارزیابی

• قسمت 8

تأثیر drop out را بعد از هر تابع فعالساز در شبکه بررسی نموده و درصد بهینه را بیابید.

در شبکه اصلی، ما یک drop out بعد از لایه های convolution داشتیم و بعد از هر یک از لایه های هم یک لایه fully-connected قرار دادیم.

از آنجا در بخش های قبلی از drop out استفاده شده است برای مشاهده تاثیر این لایه، در ابتدا تمامی drop out های استفاده شده در بخش های قبلی را حذف کردیم تا نتایج را مشاهده کنیم. نمودار دقت و خطا مطابق زیر خواهد بود:



شکل ۱۶- نمودار تغییرات دقت و خطا بعد از حذف تمامی dropouts

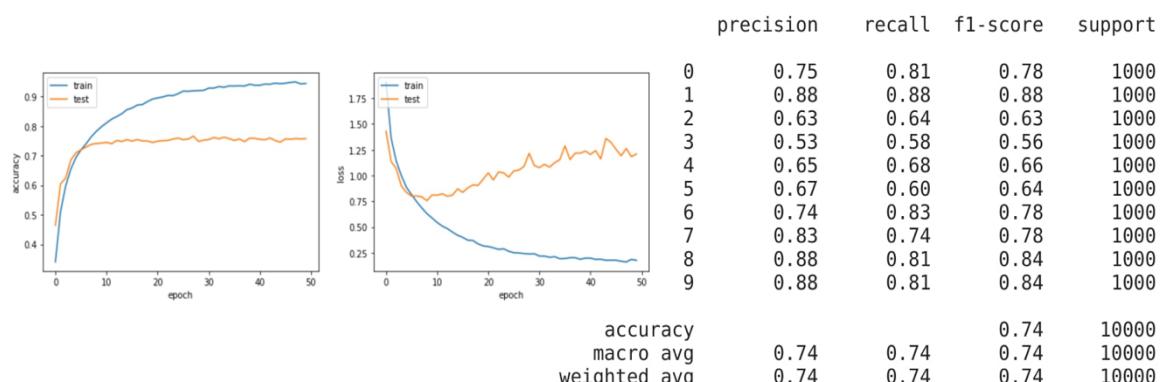
مشاهده می شود که مطابق انتظار با حذف drop out های موجود در شبکه مقدار خطا در مورد داده های ارزیابی از حدود مرحله 10 به بعد به شکل چشمگیری شروع به افزایش می کند و شبکه دچار نوعی overfitting می شود. همچنین نتایج طبقه بندی بر روی داده های تست، در تصویر زیر خلاصه شده است.

	precision	recall	f1-score	support
0	0.77	0.77	0.77	1000
1	0.83	0.88	0.85	1000
2	0.61	0.62	0.61	1000
3	0.57	0.52	0.54	1000
4	0.73	0.60	0.66	1000
5	0.67	0.60	0.63	1000
6	0.71	0.85	0.77	1000
7	0.75	0.78	0.77	1000
8	0.80	0.86	0.83	1000
9	0.85	0.82	0.83	1000
accuracy			0.73	10000
macro avg	0.73	0.73	0.73	10000
weighted avg	0.73	0.73	0.73	10000

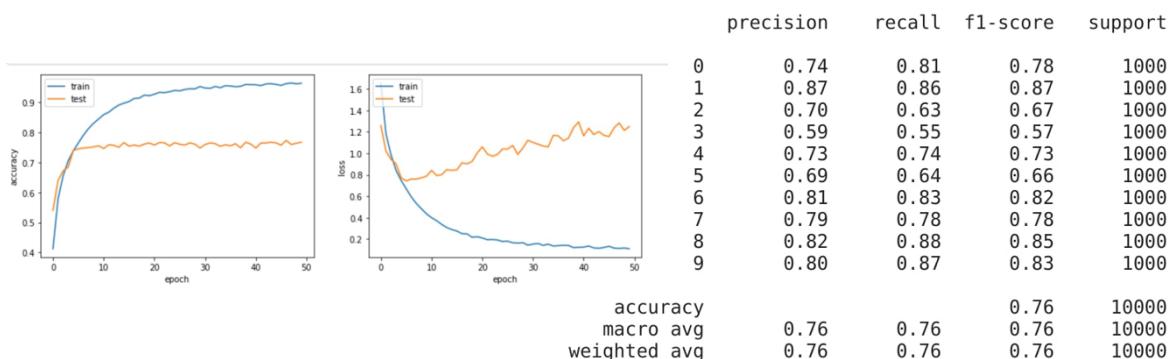
شکل ۱۷- عملکرد طبقه بندی بر روی داده های آزمون بعد از حذف تمامی dropout

مشاهده می شود که مقدار دقت طبقه بند نسبت به مدل اصلی ۳ درصد کاهش پیدا می کند که مطابق با انتظار است. همانگونه که انتظار میرفت با به کار بردن لایه dropout، عملکرد شبکه برای داده های validation به مراتب کم دقت تر خواهد بود؛ زیرا که در صورت وجود لایه های drop put برخی از اطلاعات لایه ها به صورت تصادفی حذف خواهند شد و این حذف تصادفی اطلاعات، باعث می شود که شبکه مجبور به یادگیری generalتر برای کاهش تابع هزینه باشد و احتمال overfitting شبکه، کاهش پیدا کند.

اما در تصاویر بالا ما تمامی لایه های drop out را از شبکه حذف کردیم؛ در اینجا یک بار تنها drop out مربوط به لایه های convolutional و یک بار drop out مربوط به بخش انتهایی شبکه و قسمت لایه های fully connected را حذف می کنیم که می توانید نتایج هر کدام را ببینید:

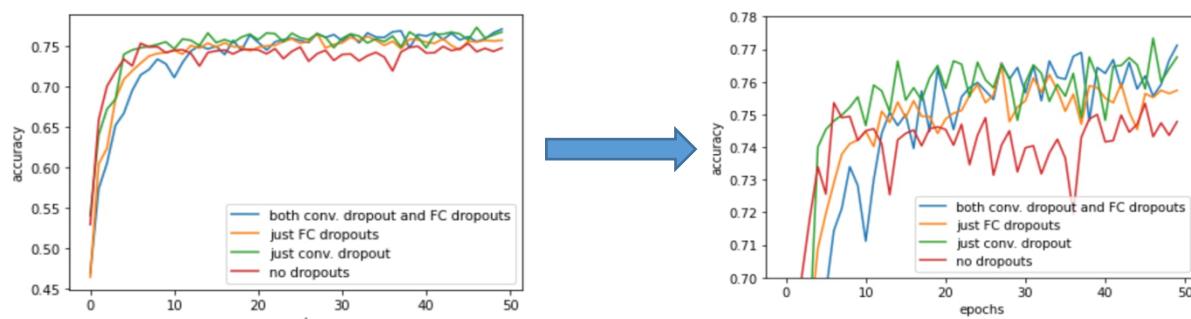


شکل ۱۸- نمودار تغییرات دقت و خطای بر روی داده های آموزش و ارزیابی در حین آموزش؛ و عملکرد مدل بر روی داده های آزمون بعد از حذف dropout مربوط به لایه convolution



شکل ۱۹- نمودار تغییرات دقت و خطا بر روی داده‌های آموزش و ارزیابی در حین آموزش؛ و عملکرد مدل بر روی داده‌های آزمون بعد از حذف لایه‌های dropout مربوط به لایه‌های fully-connected

همانطور که از نتایج مشخص است، هر کدام این حذف‌ها باعث افت عملکرد شبکه شده‌اند اما تاثیر حذف لایه‌ی convolution drop out بیشتر بوده. به نظر می‌رسد که این قسمت از شبکه، بیشتر مستعد overfitting می‌باشد. تصویر زیر مقایسه‌ای از این سه مدل، با مدل اصلی ارائه می‌دهد.



شکل ۲۰- مقایسه‌ی مدل‌ها با حذف لایه‌های dropout مختلف

بررسی تصویر فوق هم اثر بیشتر convolution drop out در لایه‌ی convolution را نشان می‌دهد. (نمودار نارنجی رنگ که پایین‌تر از نمودار سبز رنگ قرار دارد). همچنین با بررسی مقادیر مختلف، درصدهای گزارش شده در جدول زیر، بهترین عملکرد را ارائه دادند.

جدول ۷: بهترین اعداد برای هر کدام از لایه‌های dropout استفاده شده

	size	position
Drop Out 1	0.2	After conv. layers
Drop Out 2	0.3	After first FC layer
Drop Out 3	0.3	After second FC layer

سوال 3 – Data Augmentation

• قسمت ۱

در مورد **Data Augmentation** تحقیق کرده و نحوه عملکرد این روش را توضیح داده و تبدیل های مربوطه را شرح دهید. آیا برای داده های تست استفاده می شود.

شامل فرآیندی است که داده های جدید با دستکاری داده های اصلی بدست خواهد آمد. برای مثال، تصاویر جدید با چرخاندن، تغییر سایز، تغییر رنگ، ترکیب چند عکس، بردین و یا تغییر محل قسمت اصلی عکس بدست خواهد آمد. این فرآیند به ما کمک خواهد کرد تا داده های جدید بدون نیاز به انجام ازمایش یا شبیه سازی بدست آید و همچنین نسبت به نویزها و اختشاشات احتمالی موجود در داده ها مقاوم تر عمل کند. برای مثال با حذف یک ناحیه مستطیلی از یک عکس یکی از روش های Data Augmentation احتساب می شود به گونه ای که تصاویر دیگر نسبت به حذف ناحیه ای از عکس مقاوم می شوند و همچنین از پدیده Overfitting شبکه نیز جلوگیری خواهد شد و دقت شبکه در پیش بینی خروجی برای داده های validation افزایش خواهد یافت. در نهایت این عمل باعث بهتر شدن عملکرد شبکه و افزایش داده های ترین خواهد شد و در زمینه هایی که با داده های تجربی کم رو برو هستیم و یا حتی جمع آوری داده های بیشتر هزینه برو زمان گیر خواهد بود برای مثال داده هایی که در زمینه پزشکی استفاده خواهد شد را شامل می شود. تبدیل هایی که به این منظور استفاده می شود به شرح زیر است:

• Rotation

عکس های موجود در دیتابست را می توان با زوایای دلخواه چرخاند که سبب می شود شبکه طراحی شده در مقابل تغییر زاویه مقاوم باشد.

• Width Shifting

با استفاده از این قابلیت عکس مورد نظر را می توان در راستای محور طول ها جابه جا کرد که در نهایت نسبت به این تغییر مقاوم خواهد شد.

• Height Shifting

با استفاده از این قابلیت عکس مورد نظر را می توان در راستای محور عرض ها جابه جا کرد که در نهایت نسبت به این تغییر مقاوم خواهد شد.

• Brightness

با استفاده از این آیتم می توان نور و روشنایی عکس مورد نظر را تغییر داده که عددی بین صفر و یک خواهد بود و انتظار می رود که نسبت به تغییرات نور مقاومت ایجاد می کند.

• Shear Intensity

عکس مورد نظر را هم در راستای جهت های مختلف کشیده و هم تحت زاویه ای مشخص برش داده می شود.

• Zoom

با استفاده از این بخش می توان عکس را بزرگ کرده و نمای جدیدی از آن ساخت تا نسبت به تغییرات فاصله و اندازه مقاوم گردد.

• Channel Shift

یک پارامتری است که بر مبنای یک مقدار تصادفی در راستای کانال رنگی تصادفی تغییر ایجاد می کند.

• Horizontal Flip

این فیلتر همانند آینه عمل کرده و جهت عکس را در راستای افق وارونه نشان می دهد.

• Vertical Flip

این فیلتر همانند آینه عمل کرده و جهت عکس را در راستای عمود وارونه نشان می دهد.

به طور معمول از این روش بر روی داده های تست انجام نمی شود چرا که به طور طبیعی از این روش به منظور افزایش قدرت تعمیم شبکه بهره گرفته می شود تا در صورت وجود نقصان در داده ها در آموزش شبکه با مشکل مواجه نشویم. بر این مبنای نیازی نیست که داده های ساختگی برای مجموعه تست ساخته شود.

• قسمت ۲

با استفاده از چند تبدیل تصویر test را تبدیل به ۱۰ عکس مصنوعی کرده و با هم نشان دهید.

در این بخش به دنبال این هستیم که با استفاده از روش های گفته شده در بخش قبل عکس داده شده برای پروژه را استفاده کرده و بر مبنای آن ۱۰ عکس جدید به دست بیاوریم. در زیر عکس اصلی مشاهده می شود:

و می توانیم در ادامه، عکس های تولید شده از این عکس را مشاهده کنیم:



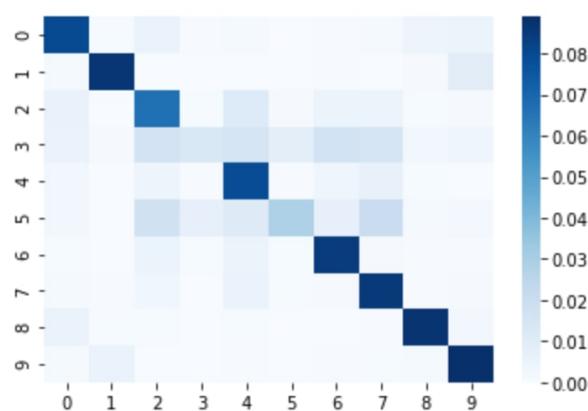
شکل ۲۱- تصویر اصلی به همراه ۱۰ تصویر تولید شده از طریق اعمال تغییرات

• قسمت ۳

با استفاده از داده های cifar 10 و با حذف ۹۰٪ از داده های موجود در بخش آموزش دو کلاس سگ و گربه تعداد تصاویر آموزش را به ۴۱۰۰۰ برسانید. پس از این مرحله با استفاده از بهترین مدل طراحی شده در سوال ۲ شبکه را آموزش داده و ماتریس آشفتگی در داده های تست را رسم نمایید. همچنین اختلال ایجاد شده در شبکه را تحلیل کرده و شرح دهید.

در این بخش از سوال خواسته شده است که با استفاده از داده های سوال ۲ و با حذف ۹۰ درصد از داده های آموزش از دو گروه گربه و سگ، نتایج حاصل از شبکه منتخب بخش قبل را به دست آورده و ماتریس آشفتگی برای داده های تست را نمایش دهیم.

در شکل زیر ماتریس آشفتگی حاصل از این مجموعه مشاهده می شود:



شکل ۲۲- ماتریس آشفتگی دادگان آزمون، بعد از حذف ۹۰ درصد
داده های دو کلاس گربه (۳) و سگ (۵)

در این شکل به خوبی مشاهده می شود که میزان تشخیص های درست در کلاس ۳ (گربه) و نیز کلاس ۵ (سگ) میباشد به شکل چشم گیری کاهش یافته است که به وضوح به دلیل کمبود صورت گرفته در داده های آموزش مسئله می باشد. همانطور که انتظار می رود آموزش شبکه با ۱۰ درصد داده های اصلی برای دو کلاس ذکر شده در بالا، شبکه را در تشخیص گربه ها و سگ های مجموعه تست دچار مشکل ساخته است. در زیر نتایج حاصل از این طبقه بند را به صورت عددی به تفکیک معیارها می توان مشاهده کرد.

	precision	recall	f1-score	support
0	0.73	0.80	0.76	1000
1	0.89	0.87	0.88	1000
2	0.55	0.67	0.60	1000
3	0.56	0.13	0.22	1000
4	0.60	0.80	0.69	1000
5	0.70	0.29	0.41	1000
6	0.70	0.85	0.76	1000
7	0.62	0.85	0.72	1000
8	0.86	0.87	0.87	1000
9	0.77	0.89	0.83	1000
accuracy			0.70	10000
macro avg	0.70	0.70	0.67	10000
weighted avg	0.70	0.70	0.67	10000

شکل ۲۳- عملکرد طبقه بند طبق معیارهای مختلف، بر روی دادگان آزمون پس از
حذف ۹۰ درصد داده های کلاس های گربه (۳) و سگ (۵)

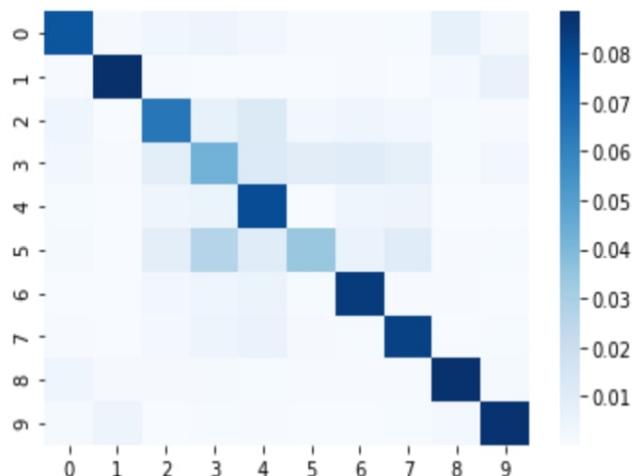
نتایج گفته شده در مورد ماتریس آشفتگی به خوبی در گزارش مربوطه نیز قابل مشاهده می باشد. مقدار معیار recall در کلاس های 3 و 5 به ترتیب 13 و 29 درصد می باشد که به وضوح نسبت به سایر کلاس ها پایین تر می باشد. همچنین با وجود اینکه تنها داده های این دو کلاس تغییر کرده و 8 کلاس دیگر بدون تغییر مورد استفاده قرار گرفته اند اما دقت کل شبکه بر روی داده های تست به میزان ۶٪ کاهش یافته است که با توجه به آموزش شبکه با داده های کمتر قابل انتظار است.

• قسمت 4

برای حل مشکل پیش آمده در بخش قبلی با استفاده از Data Augmentation تعداد تصاویر گربه و سگ را تا 5000 رسانده و مجموع داده های آموزش را به 50000 برسید. سپس داده های جدید را با بهترین شبکه به دست آمده در سوال 2 آموزش داده و ماتریس آشفتگی برای داده های تست را رسم نمایید. سپس بررسی نمایید که چه بهبودی نسبت به حالت قبلی دارد و تفاوت ها را شرح دهید.

در این بخش نیز به دنبال آن هستیم که با استفاده از روش های data augmentation نقصان ایجاد شده در داده ها را رفع کرده و تعداد را به همان میزان اولیه برسانیم. در ادامه نتایج این عمل مشاهده می شود.

در زیر ماتریس آشفتگی برای داده های تست رویت می شود.



شکل ۲۴- ماتریس آشفتگی داده های آزمون بعد از اضافه کردن داده با روش های data augmentation در دو کلاس گربه(۳) و سگ(۵)

با دقت اندکی در شکل متوجه می شویم که میزان تخصیص های درست شبکه برای دو کلاس گربه و سگ که بر روی قطر اصلی قرار گرفته است تا حد خوبی بهبود یافته و رنگ این دو خانه نسبت به حالت قبلی بیشتر شده است که مطابق انتظار است. به خوبی می توان مشاهده نمود که با استفاده از داده های تولید شده بر مبنای این روش نسبت به حالت قبل قدرت پیشビینی شبکه در مواجهه با داده های دیده نشده افزایش پیدا کرده است.

در ادامه نیز جدول مربوط به معیارهای طبقه بند حاصل مشاهده می شود.

	precision	recall	f1-score	support
0	0.81	0.76	0.78	1000
1	0.90	0.89	0.89	1000
2	0.66	0.64	0.65	1000
3	0.43	0.43	0.43	1000
4	0.61	0.79	0.69	1000
5	0.68	0.34	0.46	1000
6	0.75	0.85	0.79	1000
7	0.74	0.82	0.78	1000
8	0.85	0.88	0.86	1000
9	0.84	0.87	0.86	1000
accuracy			0.73	10000
macro avg	0.73	0.73	0.72	10000
weighted avg	0.73	0.73	0.72	10000

شکل ۲۵- نتایج عملکرد طبقه بند بر روی داده های آزمون بعد از اضافه کردن داده به کلاس های گربه (۳) و سگ (۵)

با استفاده از این جدول می توان فهمید که مقدار recall در این دو کلاس مور نظر به ترتیب به ۰.۷۳ و ۰.۴۳ درصد افزایش پیدا کرده است که با توجه به افزایش داده ها با استفاده از روش های data augmentation مطابق پیشビینی می باشد.

همچنین مقدار دقت کلی برای طبقه بند نیز تا ۰.۷۳٪ افزایش یافته که بیشتر از زمانی است که با استفاده از داده های ناقص عمل طبقه بندی را انجام دادیم. اما همانطور که انتظار می رود دقت حاصل شده از این روش به میزان ۰.۳٪ از زمانی که طبقه بندی لا داده های کامل صورت می گیرد کمتر است که به نظر می رسد این امر بدیهی است چرا که داده های اصلی شامل ۴۵۰۰ عکس متمایز در هر یک از کلاس ها می باشد و طبیعتا نتایج بهتری را نسبت به زمانی که با استفاده از ۱۰٪ عکس ها داده های جدیدی تولید می کنیم را به دست می دهد.

سوال 4 – Transfer Learning

شبکه‌ای که ما باید در این بخش بررسی کنیم؛ شبکه‌ی Xception است که در واقع از شبکه‌ی Inception الهام گرفته شده و نام آن هم از xtreme inception می‌آید.

• قسمت 1

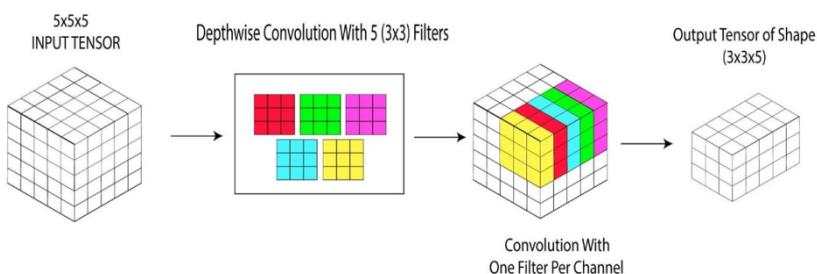
مختصراً موارد زیر را در مورد مدل خود توضیح دهید.

- معماری شبکه
- کاربرد و مزایا
- سایز تصویر ورودی
- آیا تصویر ورودی نیاز به پیش‌پردازش اولیه دارد؟ چرا؟ اگر بله چه پیش‌پردازشی؟
- سایز خروجی و معنای آن

برای درک بهتر معماری این شبکه، باید مفهوم depth-wise separable convolution را توضیح دهیم. این مفهوم از دو بخش تشکیل شده است:

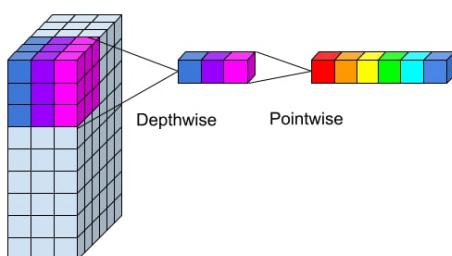
:Depthwise convolution - 1

در یک کانولوشن عادی، کرنل تمامی کانال‌ها (در تصویر RGB این عدد ۳ است) را در هر مرحله شامل می‌شود. اما در این نوع از convolution، به ازای هر یک از کانال‌ها، یک عمل کانولوشن انجام می‌شود. این به این معناست که برای یک تصویر ۳ کاناله، برای انجام این کانولوشن ما ۳ فیلتر با اندازه‌ی کرنل مشخص داریم که هر کدام بر روی یک کانال اعمال شده‌اند. تصویر زیر می‌تواند دید بهتری از این توضیح ارائه کند:

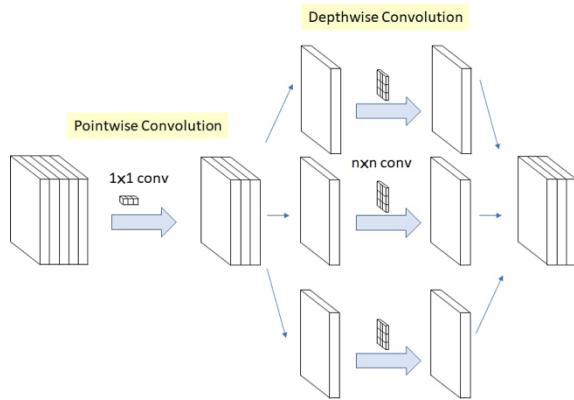


:Pointwise convolution - 2

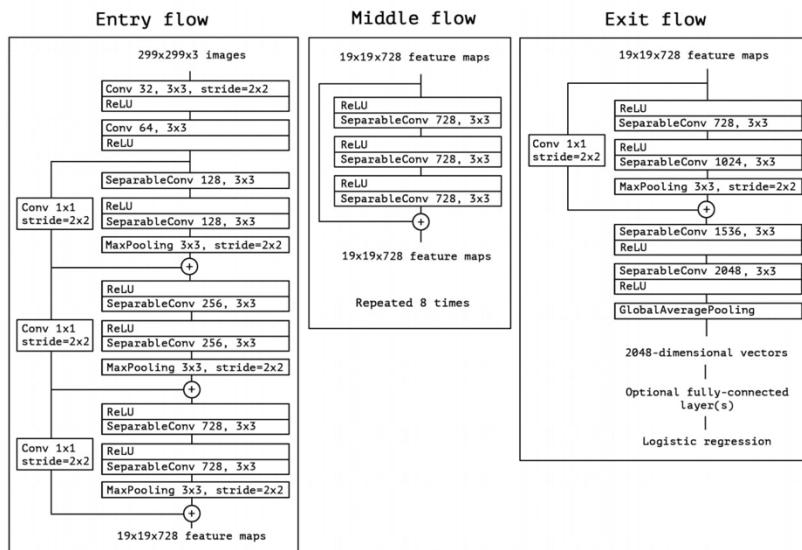
این نوع کانولوشن در واقع یک کانولوشن 1×1 است که برای تغییر عمق، یا تعداد کانال‌ها استفاده می‌شود. این کاربرد هم برای کم‌کردن و هم برای زیاد کردن تعداد کانال می‌تواند موثر باشد. تصویر زید نحوه‌ی عملکرد یک depthwise convolution، و به دنبال آن یک pointwise convolution را نشان می‌دهد:



بعد از آشنایی با این دو مفهوم، می‌توانیم معماری Xception را بررسی کنیم. این شبکه از مفهومی به نام depthwise separable convolution استفاده می‌کند؛ به این ترتیب که ابتدا depthwise convolution استفاده می‌شوند. البته به صورت عادی depthwise convolution و سپس depthwise convolution استفاده می‌شوند. این دو نوع، برعکس این شبکه است. تصویر زیر می‌تواند معماری یک depthwise convolution از این دو نوع را در یک شبکه‌ی Xception نشان بدهد.



معماری‌ای که در مقاله‌ی معرف Xception بیان شده، به این صورت است:



این شبکه از ۳۶ لایه‌ی کانولوشن ساخته شده‌است. داده‌های ورودی ابتدا به entry flow می‌روند؛ بعد از آن به middle flow می‌رسند که این معماری ۸ بار تکرار می‌شود و نهایتاً وارد exit flow می‌شوند. با توجه به ساختار هر flow متوجه می‌شویم که این ۳۶ لایه‌ی کانولوشن در ۱۴ مازلول مختلف قرار گرفته‌اند. این ساختار در واقع feature extraction مربوط به شبکه را انجام می‌دهد و بعد از آن می‌توانیم بسته به کاربرد، به شبکه ساختارهای دیگری اضافه کنیم. در اینجا که ما کار image classification بر روی داده‌های image net را داریم، در انتهای شبکه نیاز به یک ساختار برای logistic regression classification داریم.

استفاده از این شبکه، دقت بسیار بالایی بر روی دیتابست imagenet که ۱۰۰۰ کلاسه است نشان داده و این معماری ظاهرا قادر است ویژگی‌های بسیار مناسبی برای تشخیص تصاویر استخراج کند که بتوان در کلاس‌بندی‌های گستردگر از آن استفاده کرد. همچنین از معماری Inception V3 که در واقع مادر معماری Xception است و حتی تعداد پارامترهای آن‌ها برابر است؛ دقت بالاتری در این دیتابست دریافت کرده است.

سایز تصویر ورودی به این دیتابست باید $299 \times 299 \times 3$ باشد.

همچنین تصاویر ورودی به این شبکه نیاز به پیش‌پردازش دارند؛ چرا که این شبکه از قبیل اموزش دیده و این وزن‌های بدست‌آمده برای فرمت خاصی از ورودی کارایی دارند. با استفاده از تابع preprocess_input تصویر RGB به BGR تبدیل می‌شود و هر کانال رنگی با توجه به دیتاست imagenet نرمالایز می‌شود. اما این تابع scaling را انجام نمی‌دهد و ما نهایتاً باید با تقسیم بر ۲۵۵ اعداد را اسکیل کنیم.

سایز خروجی این شبکه (بخش تولید ویژگی) یک بردار با بعد 20×48 است که در واقع توصیفی از تصویر ورودی است که بعد از عبور از این 36×36 لایه‌ی کانولوشن به منظور استخراج ویژگی، بدست می‌آید و می‌توانیم آن را برای کلاس‌بندی عکس‌ها استفاده کنیم. اما لایه‌ی نهایی که برای تشخیص کلاس بر حسب داده‌های imagenet است هزار نورون خروجی دارد که با استفاده از softmax نزدیک‌ترین کلاس‌ها را پیدا می‌کند.

• قسمت ۲

به چه معناست و چه زمانی استفاده می‌شود؟ Transfer learning

Transfer learning در واقع این است که ما از دانش کسب شده در یک تسک، در تسک دیگر هم استفاده کنیم و همیشه از ابتدا اقدام به ساخت مدل از صفر نکنیم. برای مثال برای تشخیص تصاویر، یک بار لایه‌های کانولوشن در یک تسک آموزش دیده‌اند و می‌توانند استخراج ویژگی را به خوبی انجام دهند؛ در تسک بعدی ما می‌توانیم دقیقاً از بخش feature extraction قبلی استفاده کنیم؛ و با استخراج ویژگی‌های مشابه، تنها بخش classification یا همان لایه‌های انتهایی را با توجه به دیتاست خودمان آموزش دهیم.

اصولاً اصلی‌ترین مزایای transfer learning این است که زمان آموزش کمتری نیاز دارد، اکثراً به دقت بالاتری می‌رسد و در صورتی که داده‌ی کافی نداشته باشیم، از آنچایی که نیاز به داده‌ی کمتری دارد؛ کارآمدتر است. چرا که بخشی از آموزش قبلاً با داده‌های دیگری انجام شده است. پس می‌توانیم در صورت محدودیت زمانی یا داده‌ها از این روش استفاده کنیم. همچنین برای مثال اگر شرکت google مدلی را آموزش داده باشد قطعاً general از آموزش ما خواهد بود و به این صورت می‌توانیم دسترسی به مدلی داشته باشیم که با حجم بسیار زیادی از دادگان آموزش دیده است.

• قسمت ۳

شبکه‌ی انتخابی را به کمک transfer learning پیاده‌سازی نمایید.

این شبکه در keras با دیتاست imagenet موجود است و مدل لود شده است و در فایل ارسالی موجودی است. نتایج در قسمت بعد گزارش خواهند شد.

• قسمت ۴

لیست اشیایی که شبکه‌ی شما می‌تواند شناسایی کند چیست؟

دیتاست imagenet مجموعه‌ای عکس رنگی در هزار کلاس مختلف است که بعضی از کلاس‌های آن window screen، wallet، cup، pizza و ... می‌باشد. آوردن نام هر هزارتای این کلاس‌ها در اینجا ممکن نیست؛ اما این [لینک](https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a) (<https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a>) لیست کاملی از آن‌ها را دارد.

• قسمت ۵

ابتدا یک عکس رنگی با کیفیت مناسب از محیط اطراف خود بگیرید که منطقاً این عکس باید شامل اشیای قابل تشخیص توسط شبکه‌ی شما باشد. سپس در صورت لزوم عکس را پیش‌پردازش

کنید و به شبکه‌ای که در بخش ۳ پیاده‌سازی کرده‌اید داده و ۳ شی را به ترتیب بیشترین احتمال و همراه با مقدار احتمالی‌شان در خروجی نشان دهید.

برای این بخش از عکس یک فنجان استفاده شد که در زیر مشاهده می‌کنید.

و سه پیش‌بینی با بیشترین احتمال در این شبکه در جدول

زیر گزارش شده‌اند.



جدول ۸: ۳ شی با بیشترین احتمال

class	probability
cup	0.90
espresso	0.02
Soup-bowl	0.01