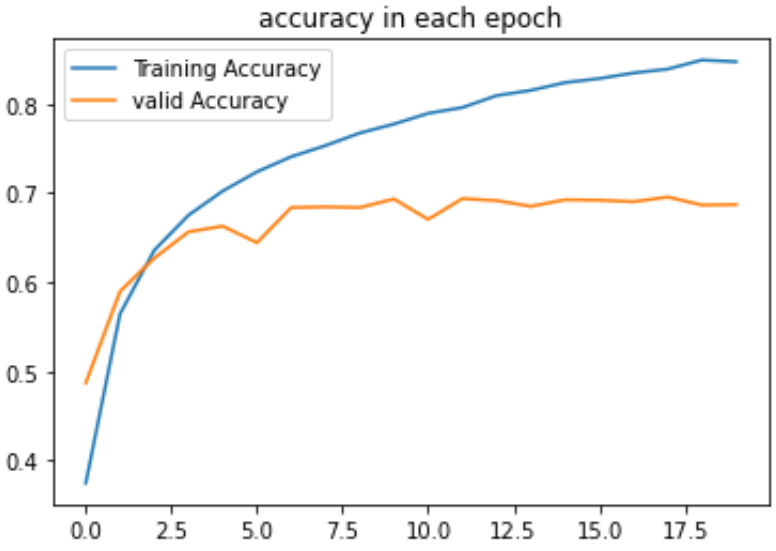
## سوال ۲ – CNN

**1)** مشخصات شبکه در ادامه آورده شده است.

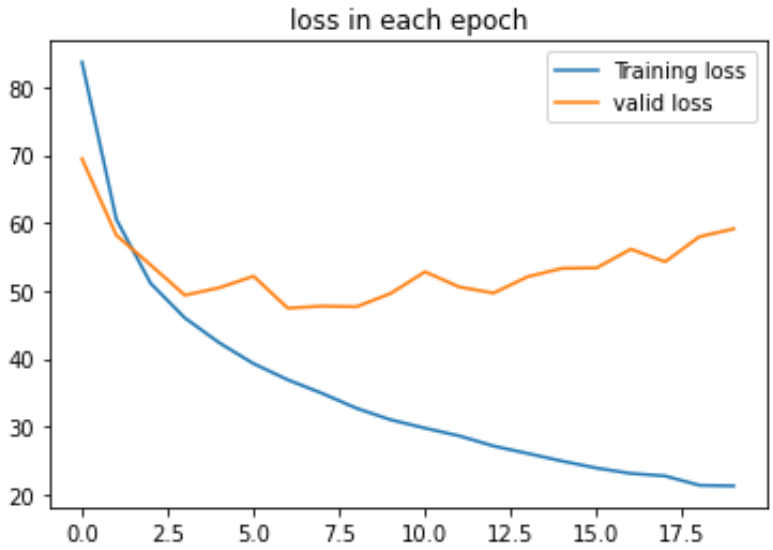
در این شبکه هر لایه شامل دو لایه کانوولوشنی و یک max polling می باشد بصورتیکه لایه های کانوولوشنی دارای kernel_size=3 و stride=1 و padding=1 که معادل همان padding='same' در نظر گرفته شده به وسیله لایه های کانوولوشنی سعی بر افزایش output channel در طی لایه ها داریم ( input channel=3 ).برای لایه های pooling نیز از فیلترهای 2در2 با stride=2 استفاده شده است که در هر مرحله سایز را در هر دو جهت نصف می کند.در نهایت نیز از سه لایه ی fully connected برای انجام طبقه بندی استفاده شده است.نکته مهم در انتخاب تعداد نورون های لایه ی اول fully connected با توجه به flatten کردن خروجی لایه ی کانوولوشنی قبلی بصورت 4*4*64 می باش. در این قسمت از تابع فعالساز RELU بعد از هر کاننولوشن استفاده شده است.

تابع loss در این بخش Cross_entropy بوده و برای optimzer از ADAM با learning rate=0.002 استفاده شده است.هم چنین batch_size=50 در نظر گرفتیم.نتایج مرتبط با آموزش دادن این شبکه در ۲۰ ایپاک بر روی داده ی ترین Cifar10 در بخش دوم این سوال آورده شده است.

**2)** شبکه طراحی شده بر روی داده های آموزش در 20 ایپاک یادگیری شد و نمودار های دقت در هر ایپاک و loss در هر ایپاک به ترتیب بصورت شکل های 1-2 و 2-2 شد.



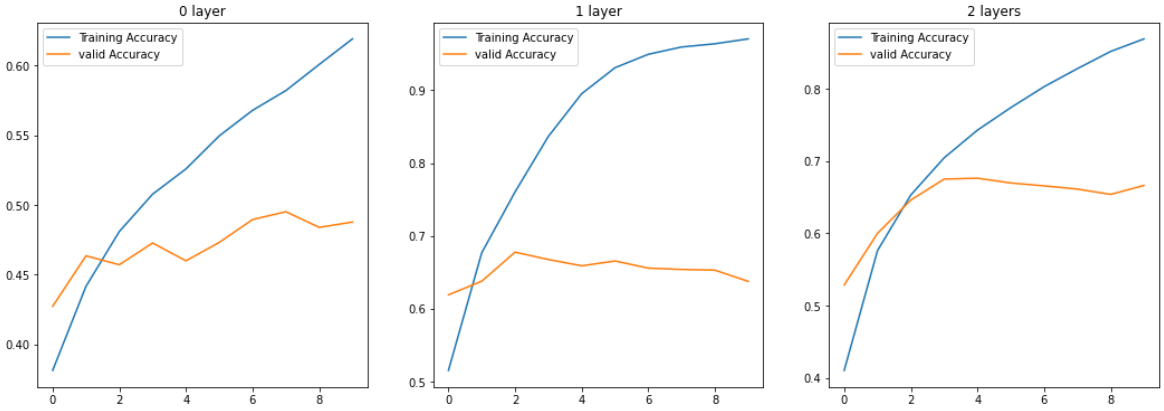

شکل1-2 نمودار دقت در هر ایپاک برای شبکه اولیه




شکل 2-2 نمودار loss در هر ایپاک برای شبکه اولیه

با توجه به اینکه تقریبا در هفتمین ایپاک over fit شروع می شود بهترین مدل را براساس آن در نظر میگیریم. که در این حالت دقت برای داده های آموزش 74 درصد و برای داده های تست 69 درصد بدست آمد. در ایپاک های بیشتر دقت روی داده آموزش افزایش می یافت ولی بخاطر over fitting دقت روی داده های تست کاهش می یافت.

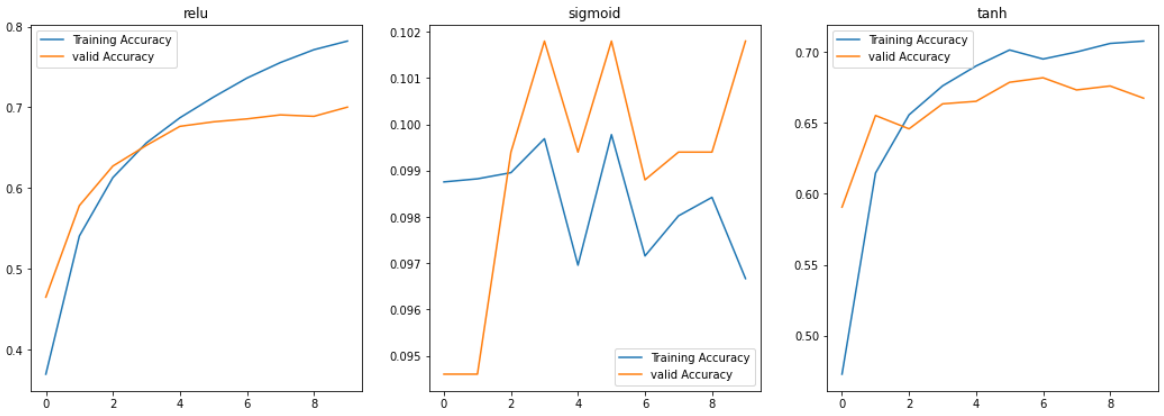
**3)** شبکه با 0، 1 و 2 لایه نیز طراحی شد. سرعت آموزش بر روی این شبکه ها سریعتر از حالت 3 لایه بود. نمودار های دقت در هر ایپاک برای این سه حالت در شکل 3-2 آورده شده است.




شکل 3-2 نمودار های دقت در هر ایپاک برای حالت های 0، 1 و 2 لایه مخفی

اولین نکته قابل توجه در هر 3 حالت رخ دادن سریعتر over fit به نسبت 3 لایه مخفی است. همچنین مشاهده می شود که در حالت های 1 و 2 لایه دقت آموزش بسیار زیاد می شود در حالیکه روی داده های تست اصلا پاسخ مناسبی ندارند. در 0 لایه دقت آموزش بسیار زیاد نیست که بخاطر تعداد پارامتر و ویژگی کمتر ممکن است رخ داده باشد. با مقایسه این سه حالت با هم و با حالت 3 لایه به نظر می رسد حالت 3 لایه عملکرد مناسبتری در این مساله دارد.

**4)** در این مرحله 3 شبکه مشابه طراحی شد و فقط تمام توابع فعالساز را متفاوت انتخاب کردیم. نتایج دقت در هر ایپاک برای این 3 تابع فعالساز در شکل 4-2 آورده شده است.

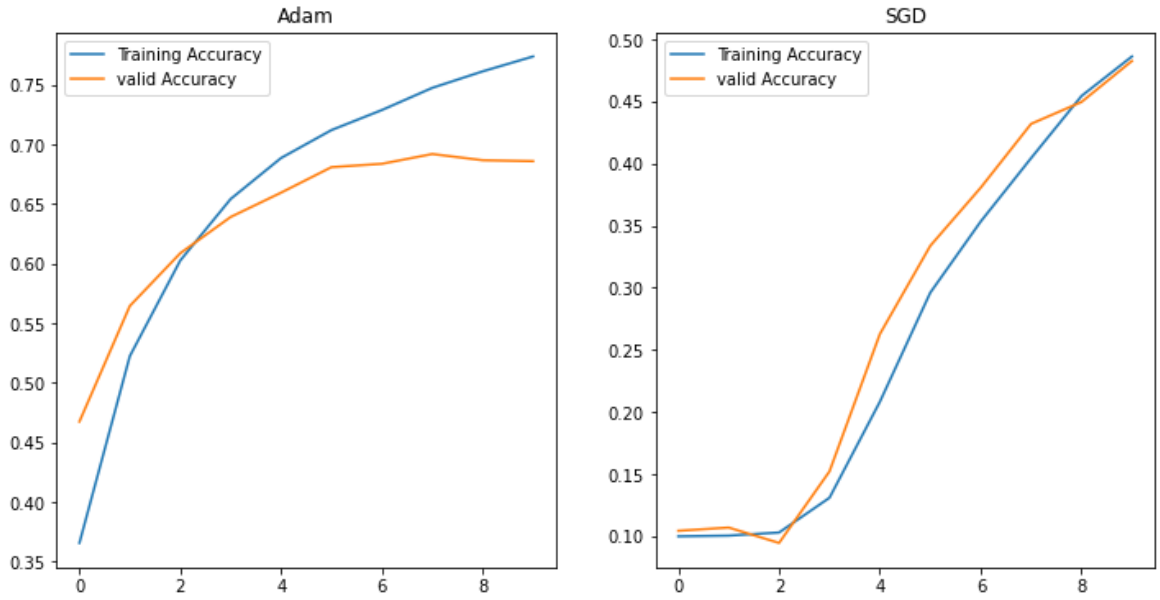



شکل 4-2 نمودار دقت در هر ایپاک برای توابع فعالساز relu، sigmoid و tanh

مشاهده می شود بین این 3، relu بهترین پاسخ را می دهد چون هم over fitness دیرتر رخ می دهد هم اینکه دقت بالاتری به نسبت دو تابع دیگر دارد.

پس با توجه به این قسمت برای این مساله تابع فعالساز relu انتخاب مناسبتری می باشد.

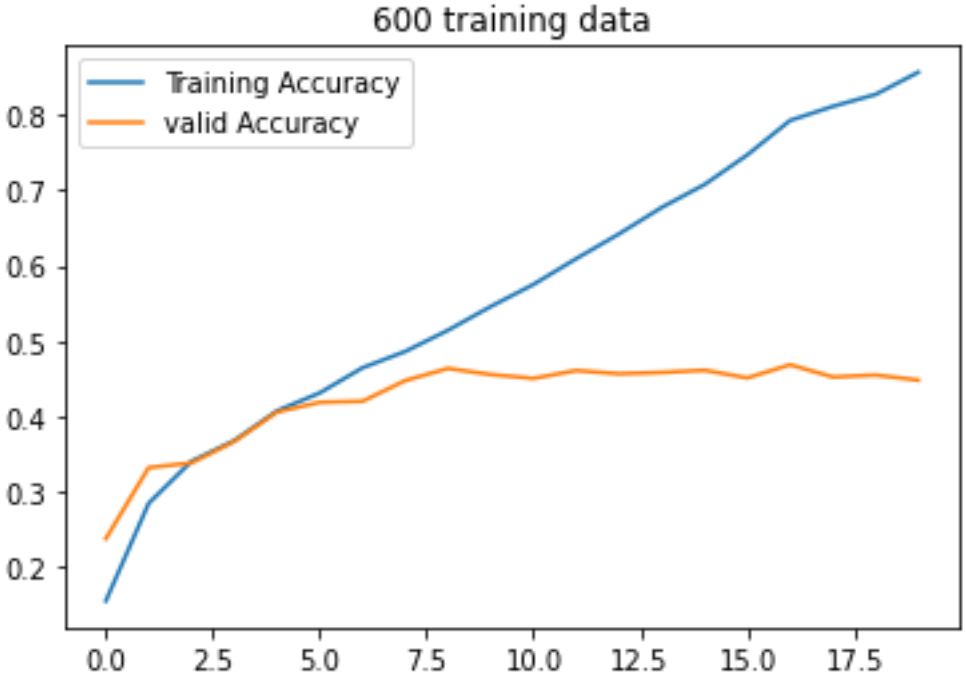
**5)** در این مرحله دو روش بهینه سازی SGD و Adam تست شد و نتایج آن بصورت شکل 5-2 شد.




شکل 5-2 دقت در هر ایپاک برای روش های بهینه سازی مختلف

مشاهده می شود روش SGD در ابتدا همگرایی بسیار کندی دارد و تقریا ثابت است و سپس شیب زیادی میگیرد که علت آن پدیده ی vanishing gradient بررسی شده در سوال ۱ می باشد، یعنی با توجه به تعداد زیاد لایه ها و مشتق زنجیره ای تغییرات در لایه های اولیه برای sgd ناچیز بوده و از جایی به بعد عملا آپدیت شدن وزن ها صورت نمی گیرد همچنین بخاطر این سرعت کم دقتی که بعد از 10 ایپاک بدست آمده نیز حدود 50 درصد است در حالی که برای روش Adam بالای 70 درصد رسیده است. با توجه به اینکه سرعت همگرایی SGD پایین است تعداد ایپاک زیادی لازم دارد و زمان زیادی صرف آموزش می شود که ممکن است مناسب نباشد ولی Adam پاسخی مناسب در عین حال سریع می دهد پس برای این مساله مناسبتر می باشد.

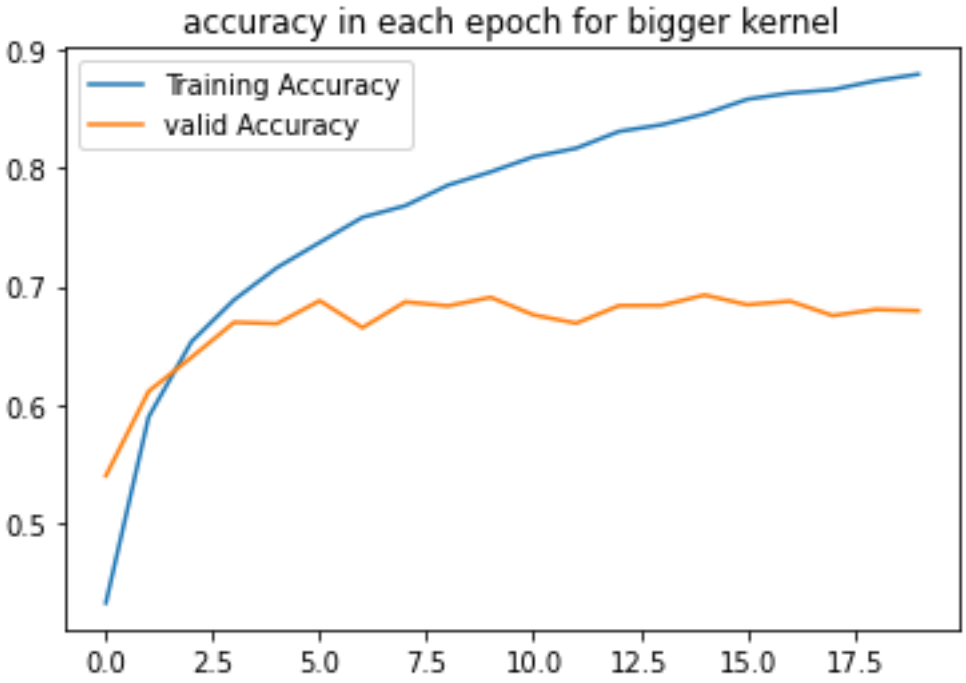
**6)** در این مرحله تعداد داده های آموزش را در هر کلاس به 600 عدد رساندیم و نمودار دقت در هر ایپاک بصورت شکل 6-2 شد.



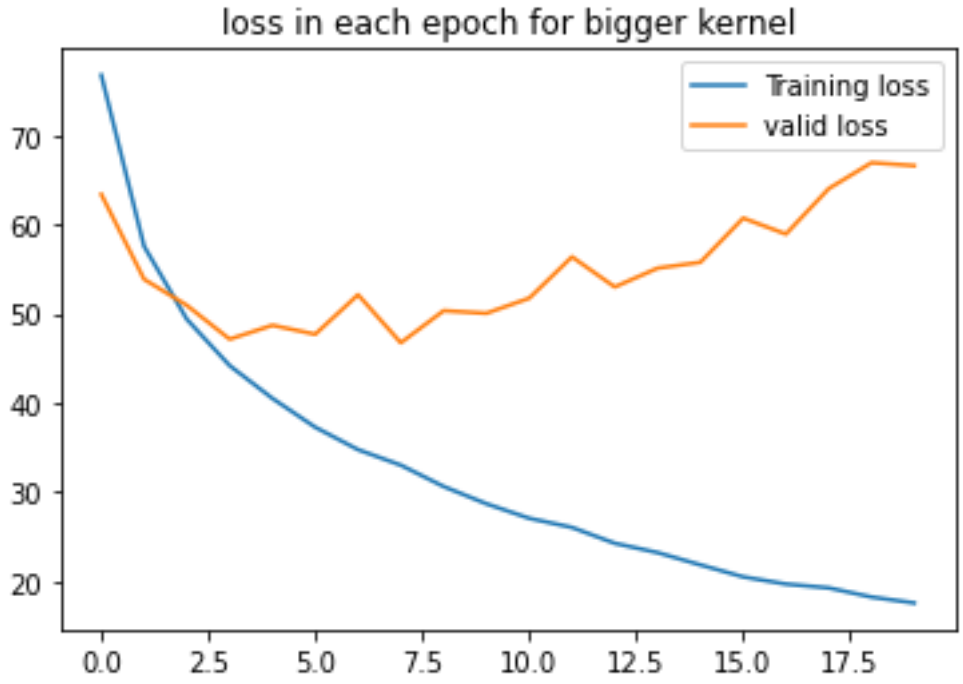

شکل 6-2 نمودار دقت در هر ایپاک برای 600 داده آموزش در هر کلاس

مشاهده می شود در این حالت هم over fit سریع رخ می دهد و دقت بسیار کمتری نسبت به حالت های قبل روی داده های تست داریم. دلیل این اتفاق این است که زمانی که داده های آموزش کم باشند یادگیری بسیار سریع تر بر روی آن ها فقط انجام میشود و احتمال اینکه بر روی آنها over fit شود زیاد است و در نتیجه generalization مدل کم می شود و نتایج مناسبی نمی دهد.

7) در این مرحله یک عمل کانولوشن از هر لایه حذف شد و به نسبت حالت اولیه که سایز kernel ها تماما 3 انتخاب شده بد در اینجا در لایه اول سایز kernel 5 انتخاب شد. در لایه دوم 6 و در لایه سوم 7 انتخاب شد. نتیجه دقت در هر ایپاک برای این شبکه در شکل 7-2 آورده شده است و loss در هر ایپاک نیز در شکل 8-2 آورده شده است.



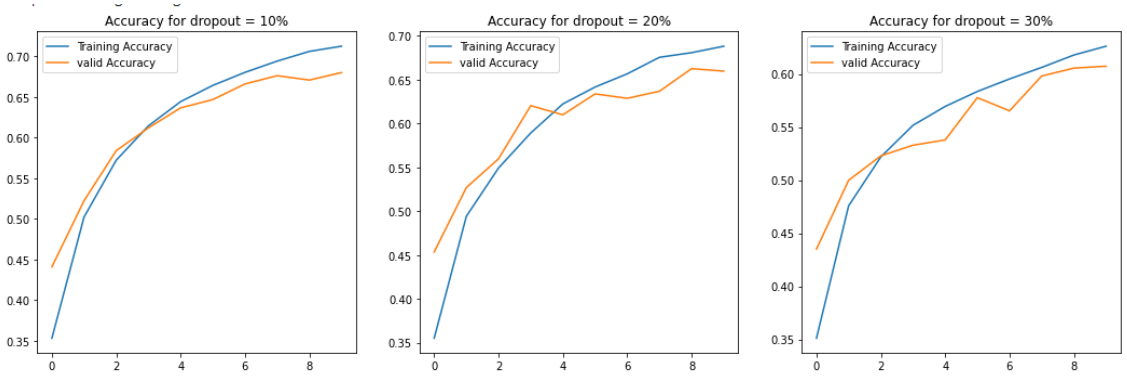

شکل 2-7 نمودار دقت در هر ایپاک برای کرنل بزرگتر




شکل 2-8 نمودار loss در هر ایپاک برای کرنل بزرگتر

می توان گفت در این مساله تفاوت زیادی بین دو حالت نبود و هم دقت و هم loss چه در حالت کرنل بزرگ تر با لایه کمتر یا کرنل کوچکتر با لایه بیشتر تقریبا مشابه بود.

**8)** با توجه به اینکه در این مرحله مقدار مناسب dropout هم خواسته شده بود، 3 شبکه با dropout های 10٪ ، 20٪ و 30٪ طراحی شدند و آموزش دیدند. نتایج دقت در هر ایپاک و همچنین loss برای این 3 حالت در شکل 9-2 آورده شده است.




شکل 9-2 دقت در هر ایپاک برای dropout های مختلف

مشاهده می شود افزودن dropout باعث شده شبکه دیرتر over fit شود و دقت بهتری پیدا کند و generalization بیشتر. در مقایسه مقدار dropout ها باید توجه کرد که زیاد کردن آن قدرت شبکه را کاهش می دهد. همچنین مشاده می شود که در 20 و 30 درصد دقت پایینتری میگیریم پس برای این شبکه و مساله به نظر انتخاب 10 درصد مناسب می باشد.

## سوال 3 – Data Augmentation

**1)**در data augmentation، زمانی که داده های آموزش کمی در اختیار داریم و بخاطر این کم بودن آموزش به خوبی انجام نمیپذیرد با روش های مختلف داده آموزش جدید تولید میکنیم براساس داده های قبلی و این داده ها را در آموزش شبکه دخیل میکنیم تا generalization بهتری شبکه داشته باشد.

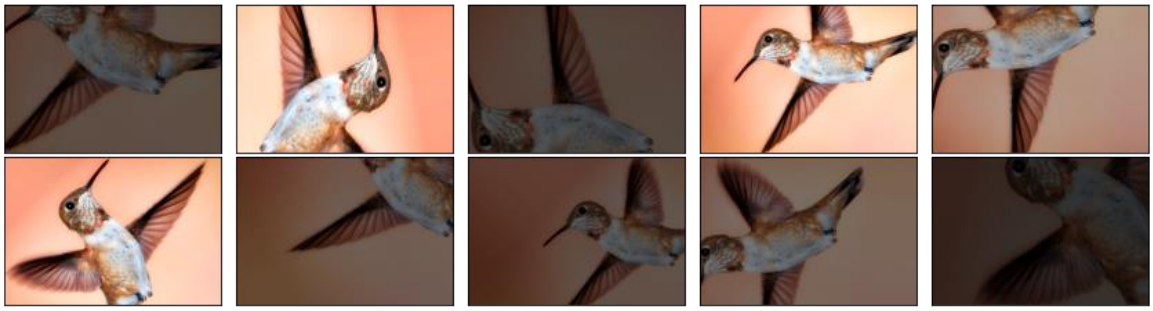
برای تصویر چند روش برای این کار وجود دارد که می توان به تغییر zoom، brightness، شیفت عکس به چپ یا راست، شیفت عکس به بالا یا پایین، چرخش(rotate) کردن عکس با زوایای مختلف و برگرداندن عکس (flip) حول محور افقی یا عمودی اشاره کرد. همچنین می توان ترکیبی از این روش ها را به کار برد و داده های جدید را تولید کرد.

هر چند که استفاده بیشتر این روش برای داده های آموزش است اما این روش برای داده های تست نیز کاربرد خاصی دارد. Test-Time Augmentation کاربرد این روش بر روی داده تست است که در آن داده های تولید شده همراه تصویر اصلی به شبکه داده می شوند و برآیندی از تخمین های شبکه برای این تصاویر را به عنوان برچسب تصویر اصلی در نظر می گیرند. این کار به این دلیل است که احتمال خطا را در طبقه بندی کاهش دهند. معمولا تعداد این تصاویر زیاد نیست و در حدود 10-20 عدد تصویر از روی تصویر اصلی که می خواهیم شبکه برچسب بزند تولید می شود و بررسی می شوند.

**2)** در این قسمت از ترکیب تمام روش های گفته شده برای تولید داده که در قسمت 1 سوال آورده شده بود بصورت هم زمان استفاده کردیم. عکس اصلی در شکل 1-3 آورده شده است.

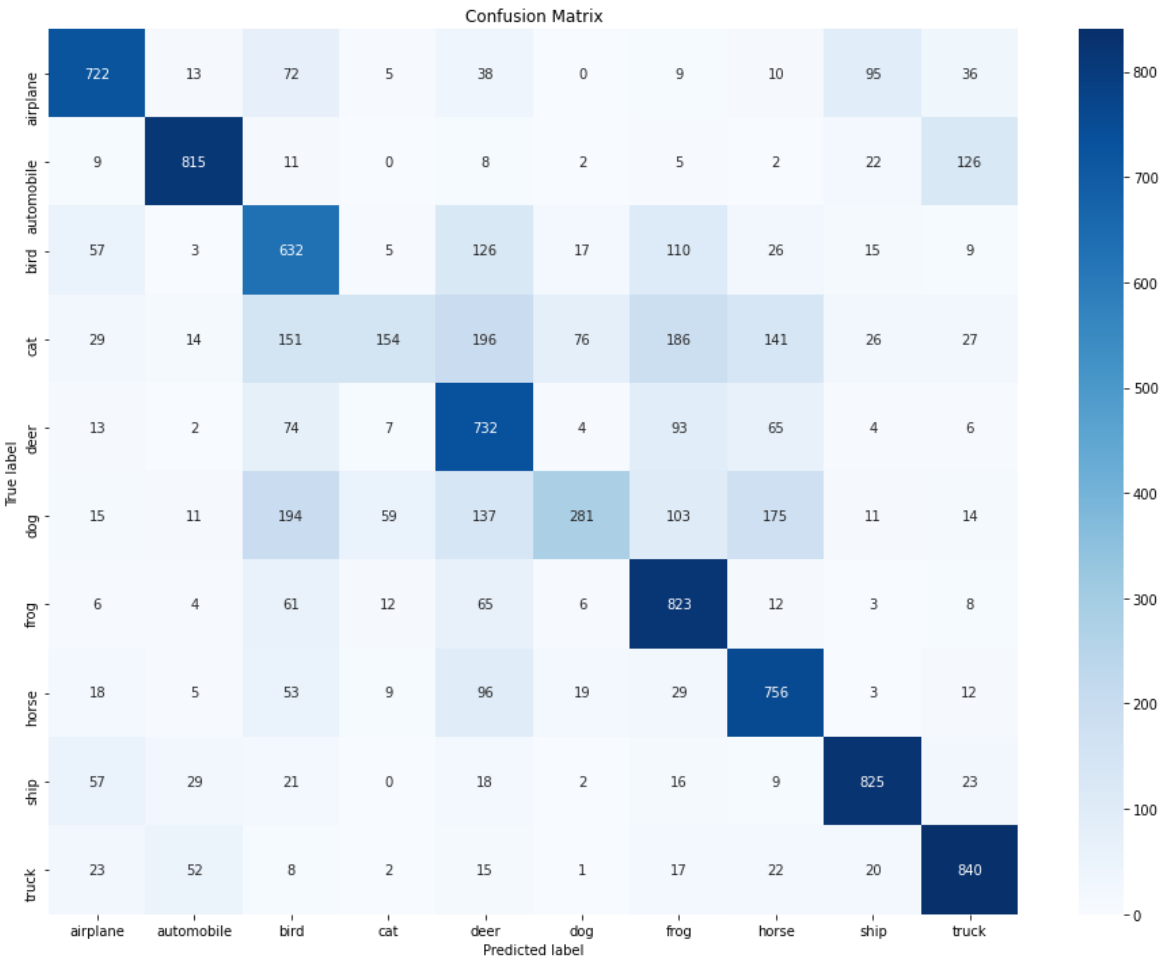
شکل 3-1 عکس اصلی داده شده

همچنین 10 تصویر ساخته شده از روی تصویر اصلی در شکل 3-2 آورده شده اند.



شکل 3-2 10 تصویر ساخته شده از روی تصویر اصلی

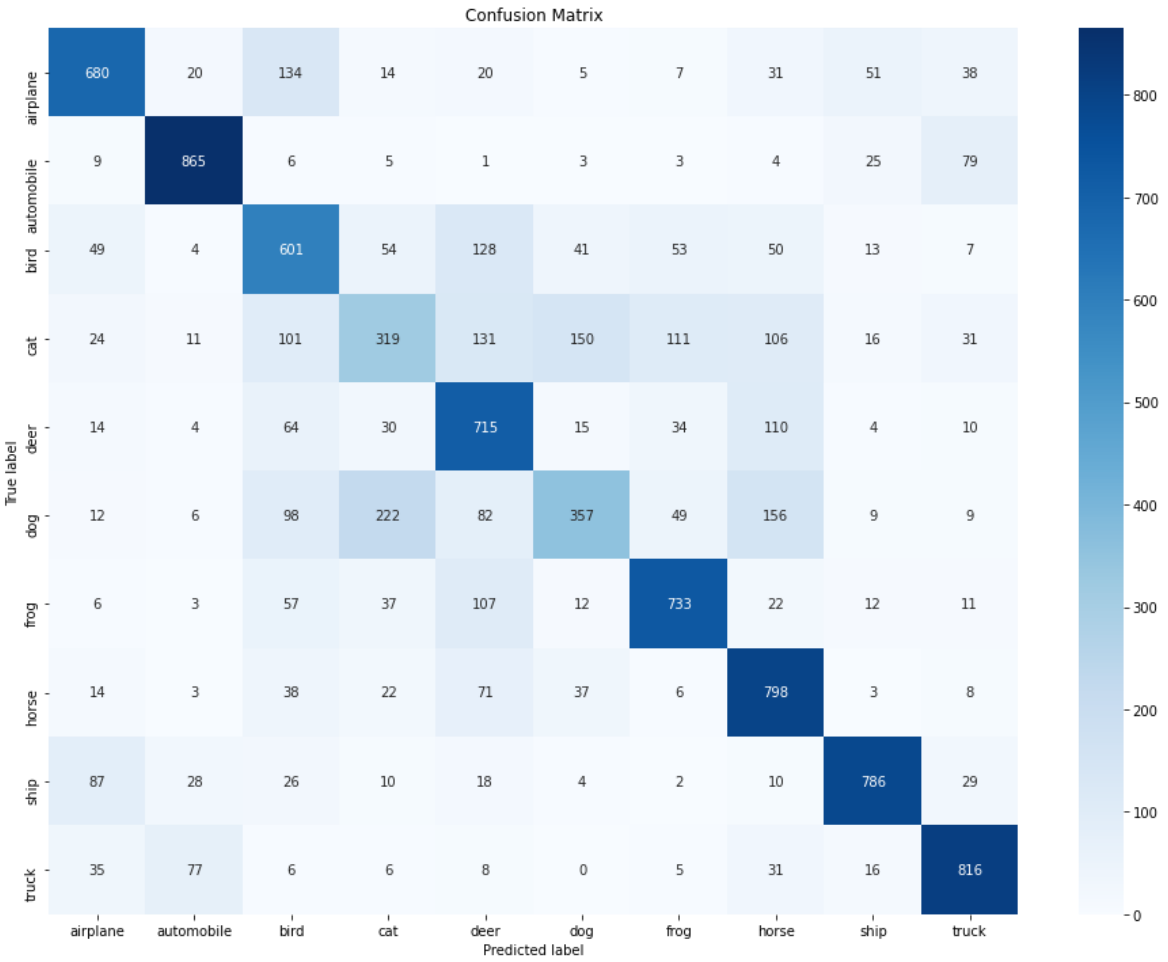
**3)** با حذف داده های گفته شده و آموزش شبکه براساس بهترین شبکه سوال 2 ماتریس آشفتگی بصورت شکل 3-3 شد.




شکل 3-3 ماتریس آشفتگی با حذف 4500 داده از هر کلاس سگ و گربه

مشاهده می شود در این حالت بخاطر کم بودن داده این دو کلاس آموزش روی آن ها به خوبی انجام نشده و کلاس گربه تقریبا 15 درصد دقت و کلاس سگ 28 درصد دقت دارد در حالی که دقت برای کلاس هایی که داده بیشتر داشتند بسیار بالاتر است. شبکه تقریبا حتی به اشتباه تصویری را با برچسب این دو کلاس شناسایی نمیکند.

**4)** در این بخش با استفاده از data augmentation تعداد داده دو کلاس سگ و گربه را دوباره به 5000 عدد رساندیم و شبکه را با این داده های جدید آموزش دادیم. ماتریس آشفتگی برای این حالت به صورت شکل 4-3 شد.




شکل 4-3 ماتریس آشفتگی برای شبکه با داده ی تولید شده

مشاهده می شود که دقت هر دو کلاس افزایش داشته است و دقت کلاس سگ به تقریبا 36 درصد و کلاس گربه به 32 درصد رسیده است. در عین حال دقت بقیه کلاس ها نیز تغییری نداشته است. همچنین نکته دیگر این است که هرچند به اشتباه ولی به تعداد بیشتری تصویر برچسب سگ یا گربه داده شده است

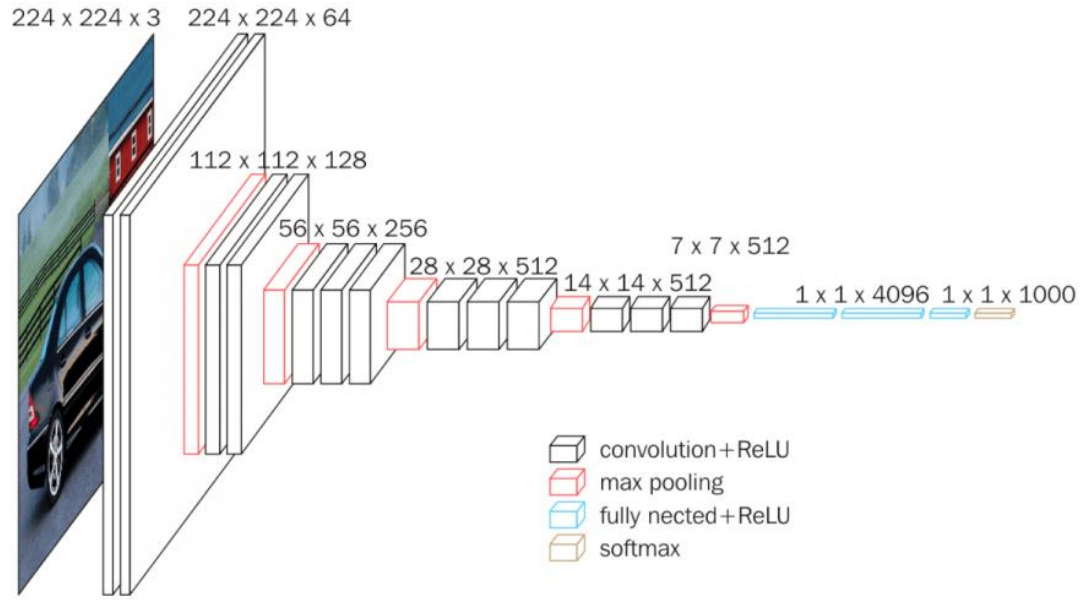
که در حالت اولیه اینگونه نبود. این نشان می دهد که روش data augmentation می تواند تاثیر خوبی در یادگیری و قدرت تعمیم شبکه داشته باشد و در زمان هایی که داده یادگیری کم است روش مناسبی است.

## سوال 4 – Transfer Learning

**در این بخش با توجه به شماره دانشجویی اعضای گروه مدل ۷ یعنی VGG Net مورد بررسی قرار گرفت و استفاده شد.

### 1) VGG Net

معماری شبکه : شکل ۴ـ۱ نشان دهنده ی معماری این شبکه می باشد که در ادامه توضیح داده می شود.




شکل 4ـ1) معماری VGG16

در این شبکه ابتدا تصویری با سایز 224 در 224 که در سه کانال rgb به عنوان ورودی به شبکه داده می شود.در ابتدا دو لایه ی کانوولوشنی با stride =1 و padding ='same' داریم که تنها سایز output channel =64 دارد.در ادامه به وسیله ی یک لایه ی Max pooling با سایز 2*2 و stride=2 سایز آن نصف شده و به (128*112*112) می رسد. در ادامه باز دو لایه ی کانوولوشنی با همان stride و padding مشخص شده سپس یک لایه ی pooling که باز هم سایز تصویر را نصف کرده در ادامه باز سه لایه ی کانوولوشنی ، یک لایه ی pooling ، سه لایه کانوولوشنی، یک لایه pooling و سه لایه ی کانوولشنی، یک لایه ی pooling و در نهایت با flatten کردن خروجی آن به

یک شبکه fully connected با سه لایه متصل شده و در نهایت سایز خروجی 1000 می باشد که به کمک softmax می توان بیشترین احتمال کلاس مربوط به تصویر را مشخص نمود.

برای پیش پردازش تصویر نیز از یک preprocess_input از کتابخانه keras استفاده می گردد که در واقع نقش نرمالایز کردن بردار ویژگی ها را دارد.

در ارتباط با خروجی مدل نیز با توجه به ۱۰۰۰ کلاس موجود ، خروجی را می توان liklihood محاسبه شده برای هر کلاس در نظر گرفت.

**2)** transfer Learning: مفهومی است که از آن به کمک وزن های بدست آمده در یک شبکه از قبل pre_trained شده ، می خواهیم برای یک شبکه جدید ( مثلا با سایز ورودی و خروجی متفاوت ) استفاده کنیم .علاوه بر آن مطابق چیزی که در این مسئله نیز خواسته شده می توان از شبکه و وزن های بدست آمده برای تست کردن یک ورودی(عکس جدید ) نیز بهره برد.

**3)** پیاده سازی شبکه آماده vgg net در کد و به کمک کتابخانه keras صورت گرفته است.

**4)**این شبکه ۱۰۰۰ کلاس متفاوت که در مجموعه داده Imagenet آورده شده را می تواند کلاسه بندی کند. این دسته بندی ها شامل ماشین ها انواع حیوانات لیوان و هواپیما و ... می باشند که.در لینک زیر قابل مشاهده می باشند.

https://gist.github.com/yrevar/942d3a0ac09ec9e5eb3a

5 ) عکس ۴-۲ به عنوان ورودی به شبکه داده شده است.


شکل ۴-۲ ) تصویر یک تفنگ اسباب بازی

خروجی کد به صورت زیر می باشد.

```
three top categories and their liklihoods are:
rifle (31.27%)
revolver (24.51%)
holster (21.11%)
```

که نشان از دسته بندی درست این عکس می باشد.