تمرین دوم شبکه عصبی

محمدرضا صیدگر-401422215

Exercise 1)

Exercise 2)

شکست تقارن به نیاز اولیه مدل های یادگیری ماشین مانند شبکه های عصبی اشاره دارد.
هنگامی که برخی از مدل‌های یادگیری ماشین دارای وزن‌هایی هستند که همگی با مقدار یکسانی مقداردهی اولیه شده‌اند، ممکن است تفاوت وزن‌ها در حین آموزش مدل دشوار یا غیرممکن باشد.
کار هایی که میتوانیم انجام دهیم این است که در مرحله اول شبکه را با وزن صفر مقداردهی اولیه نکنیم، در برخی موارد ممکن است شبکه اصلاً یاد نگیرد.
در مرحله دوم شبکه را با وزن ثابت مقداردهی اولیه نکنیم، در نهایت به یک شبکه متقارن خواهیم رسید.
می توانیم با مقداردهی اولیه وزن به طور تصادفی و ثابت نگه داشتن bias، تقارن را بشکنید. اما کافی نیست، اگر وزن خیلی کوچک باشد،ما را به مشکل vanishing gradient می‌رساند، اگر وزن خیلی بزرگ باشد، ما را به مشکل exploding gradient می‌رساند. در عوض، وزن را با یک محدوده خاص مقداردهی کنیم.
یک تکنیک بهتر این است که از مقداردهی اولیه Xavier استفاده کنیم تا واریانس در هر لایه یکسان باشد.

Exercise 3)

لایه pooling معمولاً بعد از یک لایه convolution اعمال می شود تا اندازه فضایی ورودی کاهش یابد. به طور مستقل برای هر برش عمقی حجم ورودی اعمال می شود.
عمق حجم همیشه در عملیات pooling حفظ می شود.
از فواید لایه pooling این است که تعداد پارامترهای آموزشی و هزینه محاسبات را کاهش می دهد، بنابراین overfitting را کنترل می کند و مدل را نسبت به distortion معین تغییرناپذیر می کند.

لایه pooling در حین backpropagation گرادیان ها آموزش داده نمی شود زیرا حجم خروجی داده ها به مقادیر حجم ورودی داده ها بستگی دارد.
انواع مختلفی از لایه ها pooling داریم مانند:
Max Pooling: در این نوع pooling ، حداکثر مقدار هر کرنل در هر برش عمقی گرفته می شود و به لایه بعدی منتقل می شود.
Min Pooling: در این نوع حداقل مقدار هر کرنل در هر برش عمقی گرفته می شود و به لایه بعدی منتقل می شود.
L2 Pooling: در این نوع ،نرم Frobenius برای هر هسته اعمال می شود.
Average Pooling: در این نوع میانگین مقدار کرنل محاسبه می شود.
در بیشتر موارد، به خصوص در کامپیوتر ویژن، عملکرد pooling دقت مدل را افزایش می دهد. اما در برخی موارد که می‌خواهیم مدل ما نسبت به تغییرات کوچک بسیار حساس باشد، باید از pooling اجتناب کنیم.

Exercise 4)

اولاً، Cross-entropy معیار بهتری نسبت به MSE برای طبقه بندی است، زیرا مرز تصمیم گیری در یک کار طبقه بندی بزرگ است . MSE طبقه بندی های غلط را به اندازه کافی مجازات نمی کند، اما loss مناسبی برای رگرسیون است، جایی که فاصله بین دو مقدار قابل پیش بینی کم است.
دوم، از دیدگاه احتمالی، Cross-entropy به عنوان تابع هزینه طبیعی برای استفاده در صورتی که sigmoid یا softmax غیرخطی در لایه خروجی شبکه خود دارید و می‌خواهید احتمال طبقه‌بندی صحیح داده‌های ورودی را به حداکثر برسانید، ایجاد می‌شود. .
اگر در عوض فرض کنید هدف پیوسته و به طور معمول توزیع شده است، و احتمال خروجی شبکه را تحت این مفروضات به حداکثر برسانید، MSE (ترکیب با یک لایه خروجی خطی) را دریافت خواهید کرد. برای طبقه‌بندی، Cross-entropy نسبت به MSE مناسب‌تر است - مفروضات اساسی برای این تنظیم منطقی‌تر هستند. همانطور که گفته شد، می‌توانید یک طبقه‌بندی کننده را با از دست دادن MSE آموزش دهید و احتمالاً خوب کار خواهد کرد (اگرچه با غیرخطی‌های sigmoid/softmax خیلی خوب کار نمی‌کند، یک لایه خروجی خطی در این مورد انتخاب بهتری خواهد بود). برای مشکلات رگرسیون، تقریبا همیشه از MSE استفاده می کنید. جایگزین دیگری برای طبقه بندی

استفاده از margin loss است که اساساً به معنای قرار دادن یک SVM (خطی) در بالای شبکه شما است.

Exercise 5)

تابع فعال‌سازی ReLU پس از اینکه قابلیت‌های غلبه بر saturation که یک مدل از تابع فعال‌سازی سیگموئید استفاده می‌کرد، را نشان داد، به کار گرفته شد.
تابع فعال سازی ReLU از دو طریق به کاهش مشکل saturation کمک کرد.
مانند تابع فعالسازی سیگموئید (در امتداد محور y) با صفر و یک محدود نمی شود.
هر مقداری که منفی باشد به صفر ارسال می شود.

$$f(x) = max(0, x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

تابع فعال سازی ReLU حداکثر مقدار x را در هر نمونه می گیرد. علاوه بر این، تابع فعال‌سازی ReLU سریع‌تر همگرا می‌شود، زیرا در محاسباتش خاصیت تقسیم یا نمایی ندارد.
Leaky ReLU با یک تغییر بسیار شبیه به تابع فعال‌سازی ReLU است. به جای ارسال مقادیر منفی به صفر، از یک پارامتر شیب بسیار کوچک استفاده می شود که اطلاعاتی از مقادیر منفی را در خود جای می دهد.

$$f(x) = max(0, x) = \begin{cases} x & x > 0 \\ 0 & mx \leq 0, \text{where } m \text{ is a preselected slope value} \end{cases}$$

Leaky ReLU با این شیب غیر صفر به افزایش سرعت آموزش مدل کمک می کند. پس میتوان گفت که Leaky ReLU از ReLU سرعت بیشتری دارد.
در ادامه نتایج بررسی این دو تابع فعالیت را روی مجموعه داده های MNIST می بینیم:

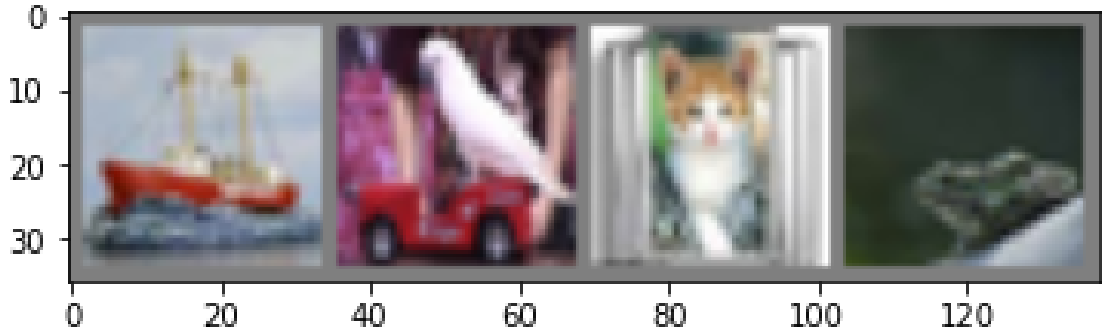
برای epoch 1 مدل ReLU در مجموعه آموزشی بهتر عمل کرد در حالی که مدل LReLU در مجموعه تست عملکرد بهتری داشت. نتایج حاصل از epoch 1 آموزش مدل روشن‌ترین نشانه‌ای را ارائه نمی‌دهند که کدام activation function در نهایت بهتر عمل می‌کند، اما به تجزیه و تحلیل نحوه تأثیرگذاری عملکرد مدل با افزایش تعداد دوره‌ها کمک می‌کند. برای مجموعه آموزشی، مدل ReLU دارای دقت 0.2789٪ بیشتر و مقدار ضرر 0.9952٪ کمتر از مدل LReLU بود. برای مجموعه اموزشی در epoch 1 مدل ReLU بهتر عمل کرد. برای مجموعه تست مدل ReLU دارای دقت 0.5230% کمتر و مقدار ضرر 1.0000% بیشتر از مدل LReLU بود. برای مجموعه تست در epoch 1 مدل LReLU بهتر عمل کرد.
با استفاده از epoch 1000 عملکرد فعال سازی LReLU در همه دسته ها بهتر عمل کرد. هنگام توسعه مدل با مجموعه آموزشی، مدل ReLU دارای دقت 0.0888٪ کمتر و مقدار ضرر که 0.3474٪ بیشتر از مدل LReLU بود. مدل LReLU با مجموعه آموزشی در 1000 دوره عملکرد بهتری داشت. با ارزیابی توسعه مدل با مجموعه تست، مدل ReLU دارای دقت 0.0900٪ کمتر و مقدار ضرر بود که 2.1124٪ بیشتر از مدل LReLU بود. این از همان الگوی قبلی پیروی می کند: مدل LReLU بهتر از مدل ReLU عمل می کند.

نتیجه گیری نهایی: مدل پشتیبانی شده توسط تابع فعال سازی LReLU از مدل پشتیبانی شده توسط تابع فعال سازی ReLU بهتر عمل می کند.

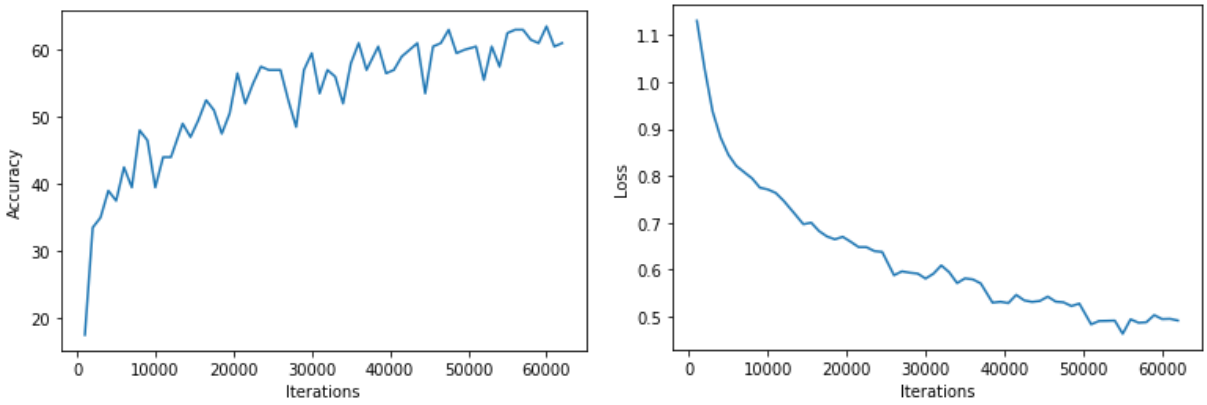
Exercise 6)

گزارش پروژه:

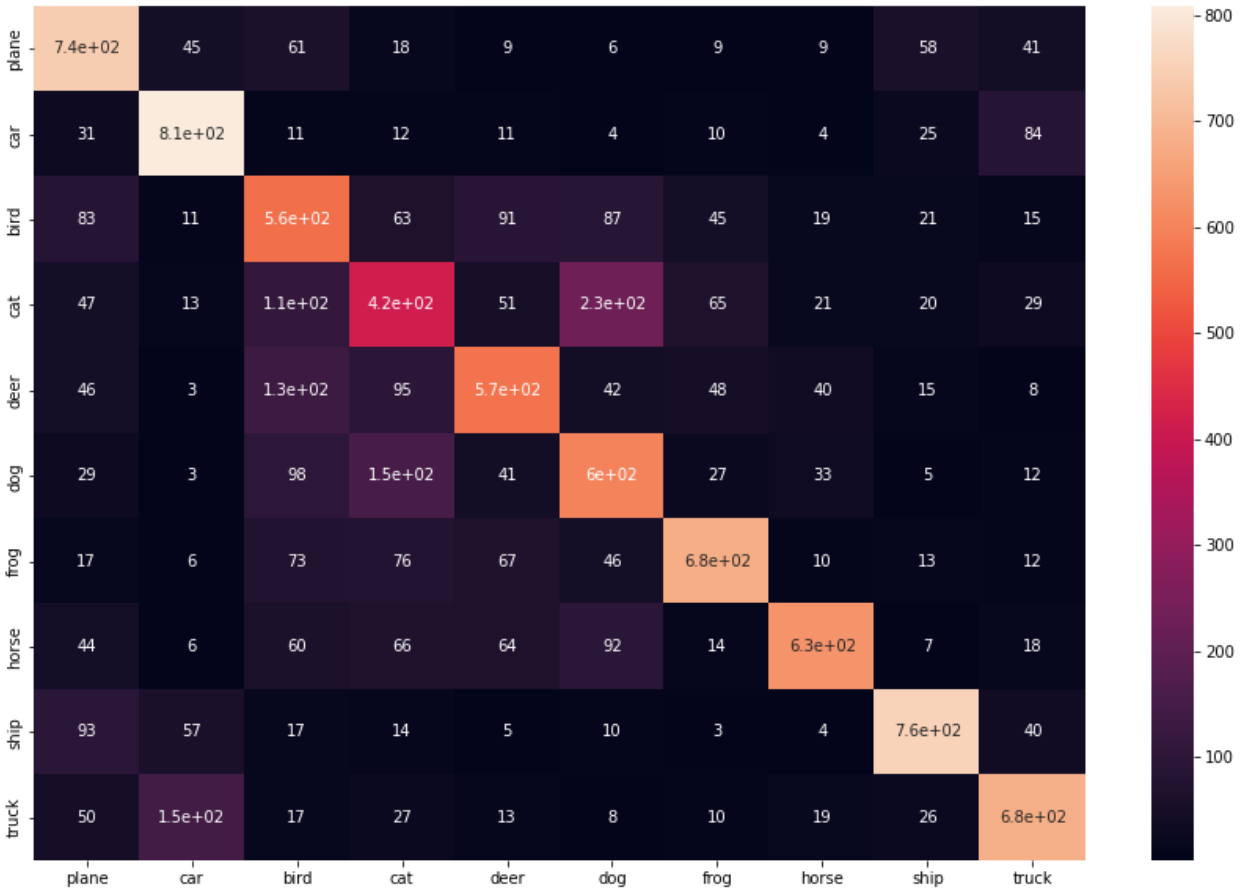
در مرحله اول داده ها را دریافت و خواندیم. داده هایمان داده های cifar-10 هستند که شامل عکس هستند که توی 10 کلاس مختلف هواپیما ،ماشین ،پرنده ،گربه و ... دسته بندی می شوند.
بعضی از عکس ها را چاپ کردیم تا ببینیم چه عکسایی هستند:



کلاس های عکسای بالا به ترتیب از راست به چپ ship , bird , cat , frog هستند.
برای پیشبینی کلاس های این مدل از داده ها که عکس هستند باید از شبکه های عصبی کانولوشن استفاده کنیم.
برای مدل اول یک لایه کانولوشن که ورودی 3 میگیرد بخاطر کانال های رنگی rgb و خروجی 6 میدهد و کرنل سایز لایه هم 5 است یعنی فیلتر ما یک window با سایز 5*5 است.لایه بعدی maxpool است که کرنل سایز آن 2 است یعنی فیلتر یک window با سایز 2 است و با stride 2 که در هر مرحله max میگیرد.لایه بعدی هم کانولوشن است با ورودی 6 و خروجی 16 و باز هم کرنل 5 دوباره لایه maxpool به مانند قبلی و در ادامه یک شبکه fully connected که ورودی که از قبل میگیرد 400 است و خروجی آن بخاطر تعداد کلاس ها 10 است و درکل این شبکه از 4 لایه نورون تشکیل شده است.
همچنین بین لایه ها از activation function relu استفاده شده است.
از CrossEntropyLoss برای محاسبه loss استفاده شده و از SGD به عنوان optimizer با پارامتر های lr=0.001, momentum=0.9.
5 ایپاک در نظر گرفته و مدل را ترین کرده و روی داده های تست ارزیابی کردیم که نتیجه در نهایت loss = 0.492 و accuracy = 62 و (loss بر عدد 2000 تقسیم شده) .
نمودار های زیر نتایج رو به خوبی در طول آموزش نشان می دهد:



کانفیشون ماتریس را برای این مدل روی داده های تست بدست آوردیم:



چیزیز که از این ماتریس میتوانیم بفهمیم این است که بیشتر از همه چیز مدل ما cat و dog را خیلی باهم اشتباه می گیرد و در تشخیص deer و bird هم خوب عمل نمی کند و در بین اشیا هم در تشخیص truck و car هم مشکل دارد.

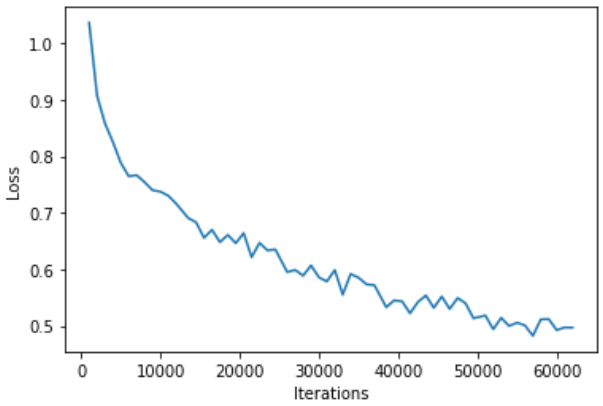
در آخر هم گزارش خلاصه ای از این کلسیفیکیشن ارائه می دهیم:

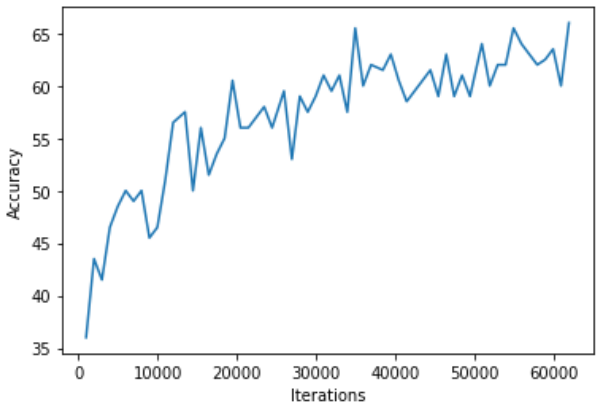
	precision	recall	f1-score	support
plane	0.64	0.71	0.67	1000
car	0.73	0.80	0.77	1000
bird	0.59	0.41	0.48	1000
cat	0.38	0.47	0.43	1000
deer	0.54	0.54	0.54	1000
dog	0.67	0.34	0.46	1000
frog	0.59	0.79	0.67	1000
horse	0.65	0.72	0.68	1000
ship	0.77	0.68	0.72	1000
truck	0.70	0.69	0.70	1000
accuracy			0.62	10000
macro avg	0.63	0.62	0.61	10000
weighted avg	0.63	0.62	0.61	10000

برای مدل دوم دقیقا همان مدل قبلی است فقط تنها فرق این است که از لایه های batch normalization هم استفاده کردیم.

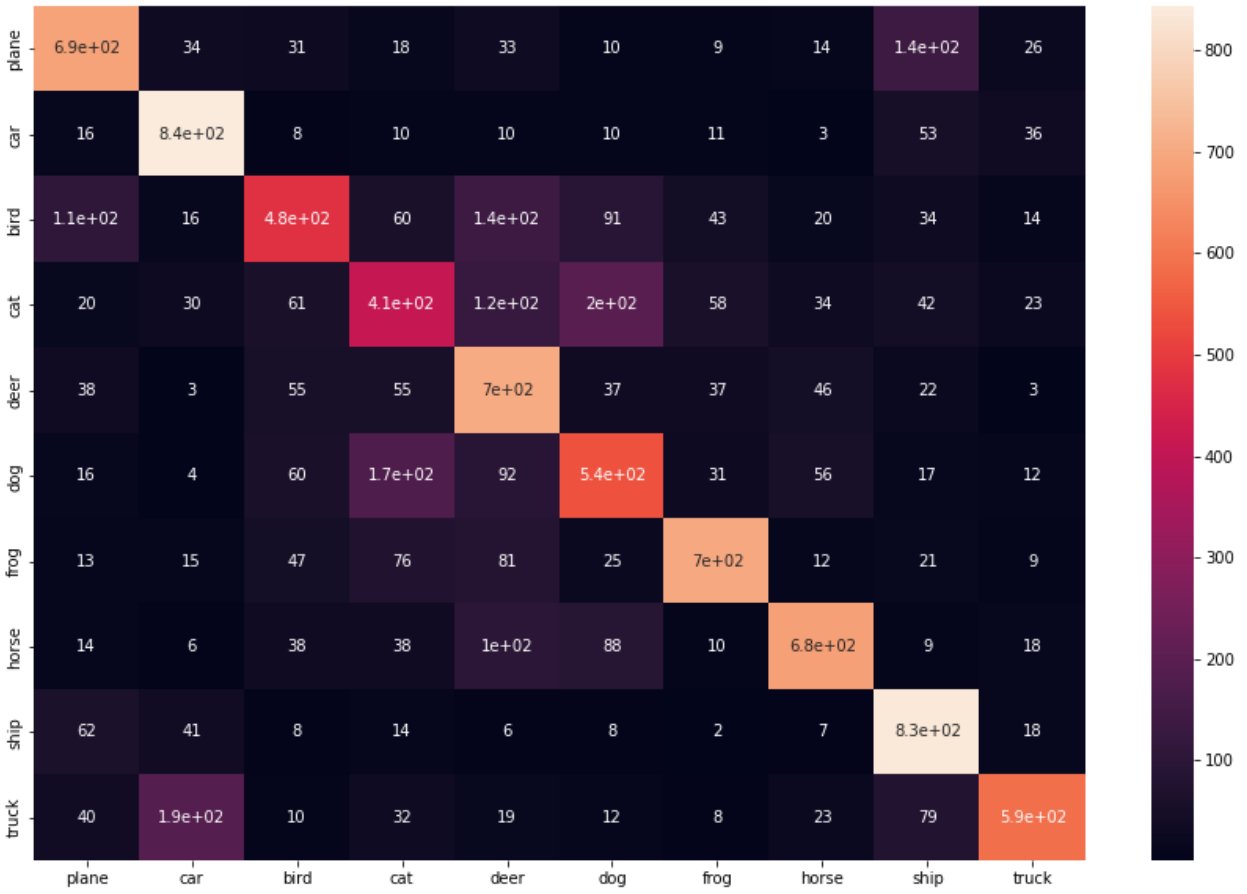
5 اپیاک در نظر گرفته و مدل را ترین کرده و روی داده های تست ارزیابی کردیم که نتیجه در نهایت loss = 0.497 و accuracy = 65 و (loss بر عدد 2000 تقسیم شده) .

نمودار های زیر نتایج رو به خوبی در طول آموزش نشان می دهد:





کانفیشون ماتریس را برای این مدل روی داده های تست بدست آوردیم:

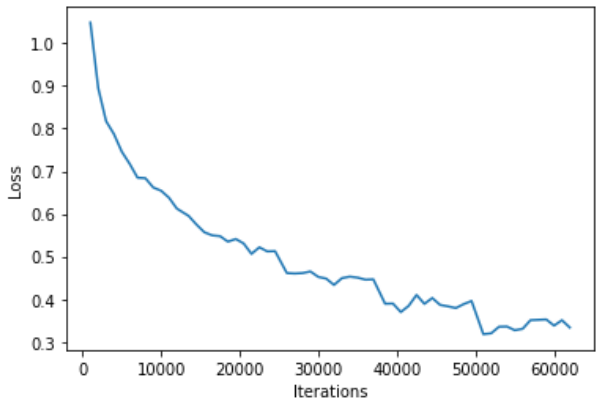


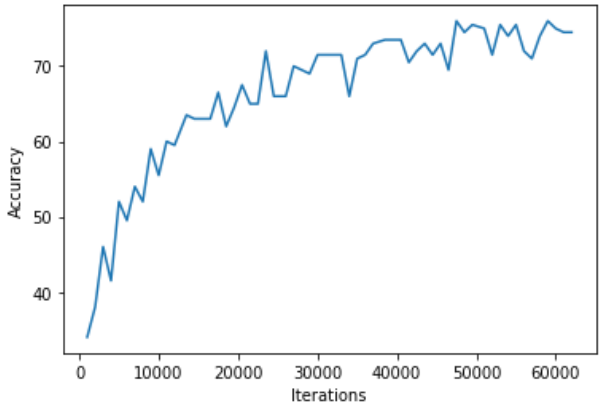
اینجا هم مشکلات قبلی همچنان هست و همچنین horse و deer را هم مشکل دارد.

در آخر هم گزارش خلاصه ای از این کلسیفیکیشن ارائه می دهیم:

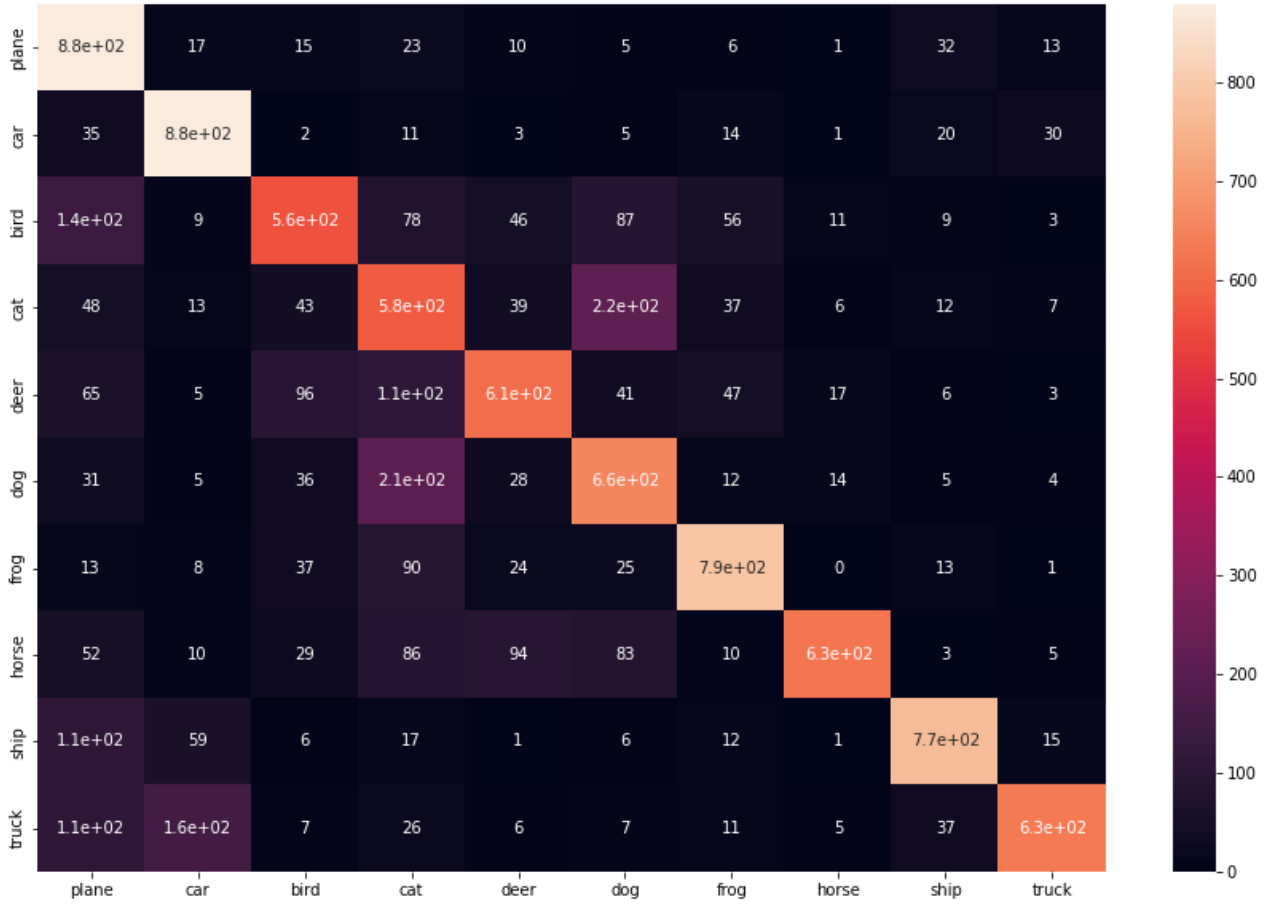
	precision	recall	f1-score	support
plane	0.68	0.69	0.68	1000
car	0.71	0.84	0.77	1000
bird	0.60	0.48	0.53	1000
cat	0.46	0.41	0.43	1000
deer	0.54	0.70	0.61	1000
dog	0.53	0.54	0.53	1000
frog	0.77	0.70	0.73	1000
horse	0.76	0.68	0.72	1000
ship	0.67	0.83	0.74	1000
truck	0.79	0.59	0.67	1000
accuracy			0.65	10000
macro avg	0.65	0.65	0.64	10000
weighted avg	0.65	0.65	0.64	10000

برای مدل سوم یک لایه کانولوشن که ورودی 3 میگیرد بخاطر کانال های رنگی rgb و خروجی 30 میدهد و کرنل سایز لایه هم 3 است یعنی فیلتر ما یک window با سایز 3*3 است.لایه بعدی maxpool است که کرنل سایز آن 2 است یعنی فیلتر یک window با سایز 2 است و با stride 2 که در هر مرحله max میگیرد.لایه بعدی هم کانولوشن است با ورودی 30 و خروجی 30 و باز هم کرنل 3 دوباره لایه maxpool به مانند قبلی و در ادامه یک شبکه fully connected که ورودی که از قبل میگیرد 1080 است و خروجی آن بخاطر تعداد کلاس ها 10 است و درکل این شبکه از 5 لایه نورون تشکیل شده است.
همچنین بین لایه ها از activation function relu استفاده شده است.
5 ایپاک در نظر گرفته و مدل را ترین کرده و روی داده های تست ارزیابی کردیم که نتیجه در نهایت loss = 0.334 و accuracy = 70 و (loss بر عدد 2000 تقسیم شده) .
نمودار های زیر نتایج رو به خوبی در طول آموزش نشان می دهد:





کانفیشون ماتریس را برای این مدل روی داده های تست بدست آوردیم:



اینجا هم مشکلات قبلی همچنان هست ولی خب کمتر شده است.

در آخر هم گزارش خلاصه ای از این کلسیفیکیشن ارائه می دهیم:

	precision	recall	f1-score	support
plane	0.59	0.88	0.71	1000
car	0.76	0.88	0.81	1000
bird	0.67	0.56	0.61	1000
cat	0.47	0.58	0.52	1000
deer	0.71	0.61	0.66	1000
dog	0.58	0.66	0.62	1000
frog	0.79	0.79	0.79	1000
horse	0.92	0.63	0.75	1000
ship	0.85	0.77	0.81	1000
truck	0.89	0.63	0.74	1000
accuracy			0.70	10000
macro avg	0.72	0.70	0.70	10000
weighted avg	0.72	0.70	0.70	10000