سوال اول)

در این سوال ابتدا دیتاست cifar۱۰ را بر اساس دو کلاس هواپیما و اتومبیل فیلتر میکنیم و ۰.۲ از داده‌های آموزش را برای validation در نظر می‌گیریم. سپس یک مدل googlenet را با وزن‌های اولیه بکار میگیریم و فقط در لایه fc آن بجای ۱۰کلاس، ۲کلاس قرار می‌دهیم. پس از آن، از مدل اپتیمایز sgd و تابع هزینه cross entropy loss استفاده میکنیم.

قسمت اول) این تابع با وزن‌های قبلی آموزش داده می‌شود اما وزن‌ها فریز نیستند و در طول ایپاک‌ها تغییر خواهند کرد. می‌بینیم که عدد گزارش شده برای تست ۹۵.۵٪ است که علت اصلی تغییر کم در طول این ۱۰ ایپاک مساله وجود وزن‌های مدل googlenet است. تمامی نتایج در داخل کد موجود است.

قسمت دوم) کلاس triplet loss را تعریف می‌کنیم و اینبار fc را به طور کامل از شبکه حذف می‌کنیم. حال شبکه را آموزش می‌دهیم و پس از آموزش داده شدن، لایه fc را اضافه کرده و وزن قسمت‌های قبل را ثابت نگه می‌داریم. در این حالت عدد گزارش شده برای تست ٪۹۶.۲۷ است که کمی بهتر شده و نشان می‌دهد اپتیموم کردن بر اساس triplet loss یک feature extractor خوب به ما می‌دهد. البته چون وزن‌های googlenet بود این تغییر کمتر احساس شد.

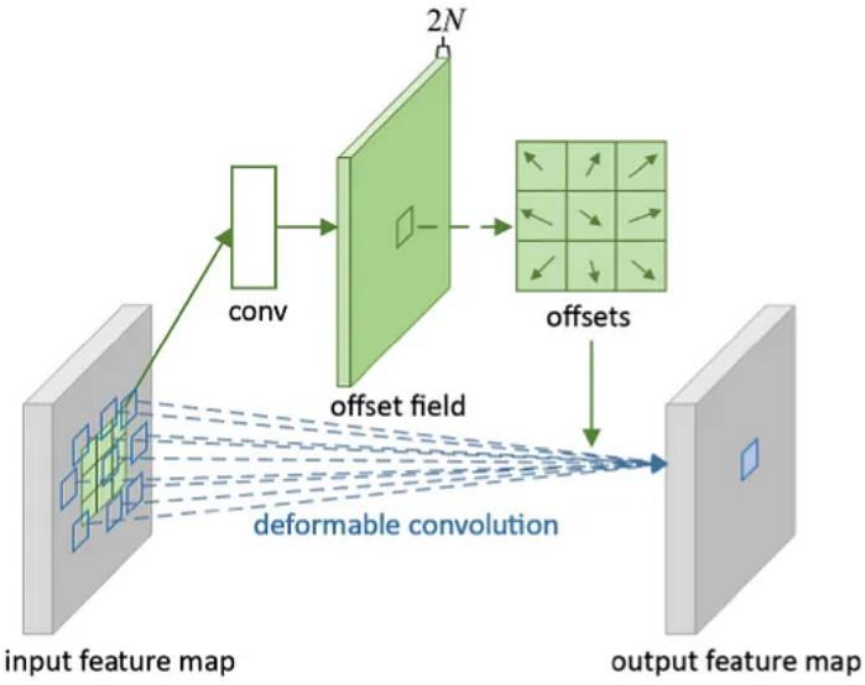
قسمت سوم) در این قسمت نیز مطابق خواسته سوال هم تابع هزینه triplet loss و هم cross entropy loss را همزمان در آموزش دخیل می‌کنیم. سپس میبینیم که دقت تست در حدود ٪۹۶.۵۰ خواهد بود که پیشرفت بسیار نامحسوسی داشتیم.

سوال دوم)

بخش اول – سوالات تئوری مربوط به تمرین عملی

۱. همانطور که در کلاس اشاره شد و می‌دانیم، کانولوشن‌های عادی به صورت منظم و با یک الگوی خاص که توسط پارامترهای لایه مشخص می‌شود دیتا را فیلتر کرده و خروجی می‌دهند. برای داده‌هایی که نظم داشته‌باشند و یا به عبارتی اتفاقی خاصی برای آنها رخ نداده باشد، این روش پاسخگوی خوبی است و شبکه نمونه برداری ثابت است؛ اما در deformable convolutional networks (DCNs)، شبکه نمونه برداری دیگر ثابت نیست و بر اساس محتوای ورودی می‌تواند تغییر کند. بدین شکل ما می‌توانیم از اشکال دیفرم در هنگام فیلتر کردن، خروجی بهتری بگیریم.

۲. همانطور که ذکر شد فیلتر در این شبکه‌ها منظم نیست و بر اساس محتوای ورودی انجام می‌گیرد. لذا یک offset در پارامترهای این شبکه وجود دارد که با توجه به شبکه یادگیری می‌شود. این offset باعث می‌شود که ما بتوانیم receptive field بیشتری پوشش دهیم و به صورت non-uniform فیلتر کنیم.



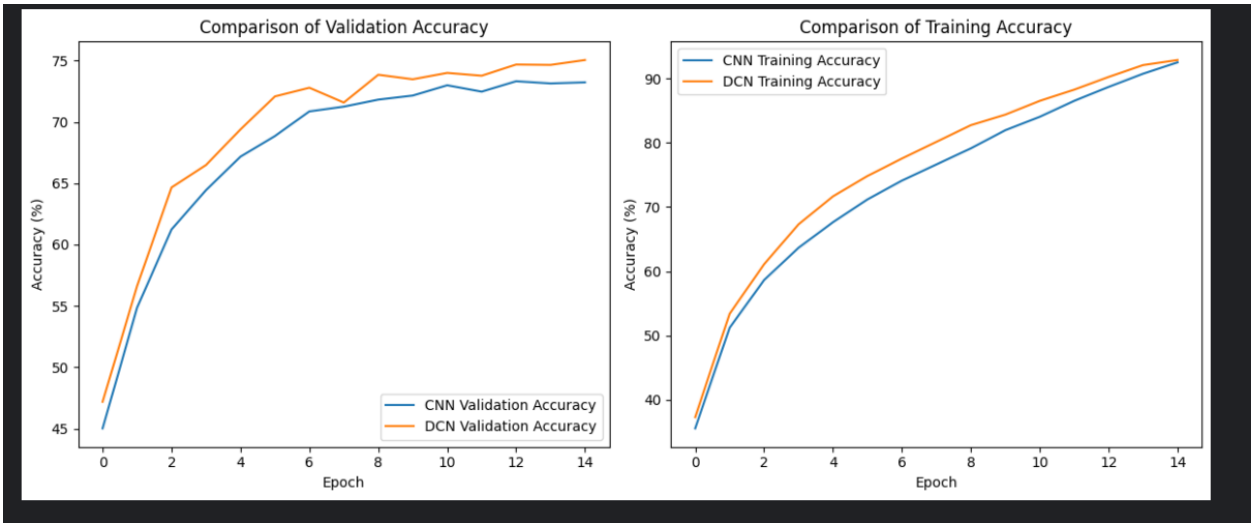
۳. CNN ها بر روی یک ساختار شبکه fixed کار می کنند و کرنل‌های کانولوشنال دارای یک receptive field ثابت هستند. هنگامی که یک تصویر می چرخد، شبکه ثابت ممکن است به خوبی با ساختارهای تبدیل شده در تصویر هماهنگ نباشد. اما در DCNها این اتفاقات نمی‌افتد زیرا در آنجا ثابت نیست.

٤. به طور کلی که این پارامتر قابل یادگیری است و از طریق loss , EBP محاسبه می‌گردد. اما در مقاله اشاره شده که مقادیر افست توسط یک لایه شبکه اضافی، معمولاً یک لایه کانولوشنال کوچک یا مجموعه ای از لایه های کانولوشن پیش بینی می شود. مقادیر افست به طور مستقل برای هر کانال از feature map ورودی محاسبه می شود.

فرض کنید که ما یک کرنل ۳*۳ در نظر داریم. در این صورت لایه اضافی ۱۸ خروجی (به خاطر دوبعدی بودن) دارد. این لایه همان تعیین کننده افست است. این افست می‌تواند مقادیر مثبت، منفی و صفر اختیار کند.

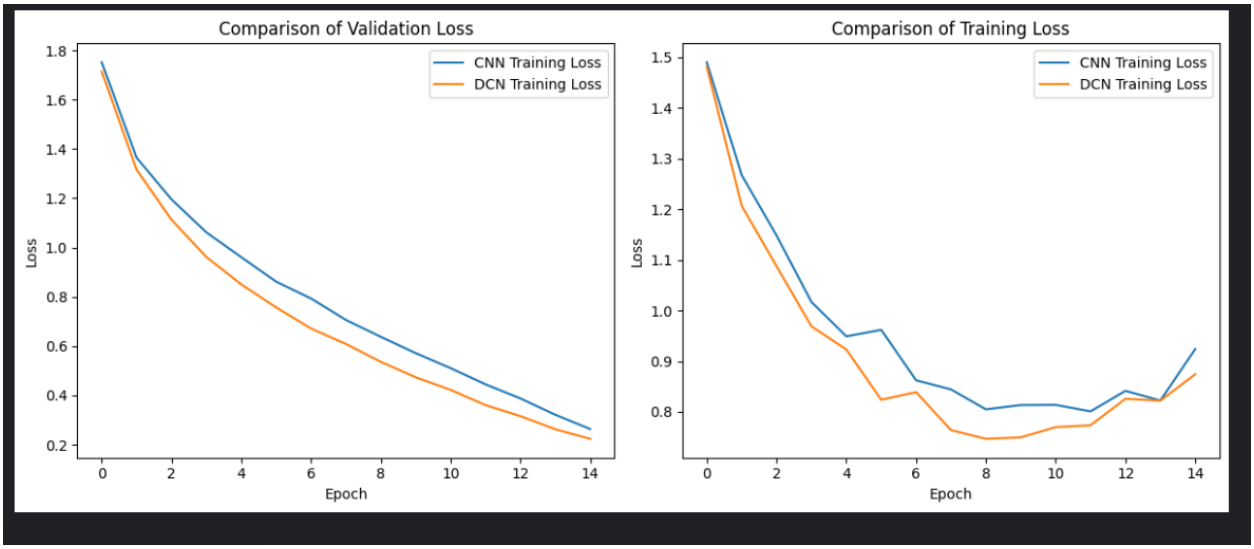
بخش سوم – گزاراشات مربوط به پیاده‌سازی:

- دقت داده‌های آموزشی و آزمایشی




```
Accuracy on the test set by CNN: 73.21%
Accuracy on the test set by DCN: 75.0%
```

- خطای مدل داده‌های آموزشی و آزمایشی



- مدت زمان اجرا برای هرکدام از موارد خواسته شده

```
Total time taken by CNN: 127.56 seconds
Total time taken by DCN: 196.62 seconds
```

سوال سوم)

در این سوال ابتدا باید یک کلاس برای تولید دیتاست بوجود آوریم.

در این کلاس ابتدا مشخص میکنیم که چند تغییر نیاز داریم و لیبل‌ها را صفر فرض میکنیم.

```
transform_num = np.random.randint(0, 3) # how many changes do we need?
```

سپس اگر تغییر ما displacement باشد عدد یک را به لیبل اضافه میکنیم، اگر تغییر ما scaling باشد عدد دو را به لیبل اضافه میکنیم و اگر تغییر ما rotation باشد عدد چهار را به لیبل اضافه میکنیم. بدین صورت اگر لیبل را یک عدد بیتی در نظر بگیریم، هر بیت نشان دهنده یک تغییر خواهد بود. پس از همه این تغییرات از ایده اصلی **cropping** استفاده می‌کنیم.

ایده اصلی: این ایده به این صورت است که ما ۵۰٪ از تصویر ورودی را کراپ کرده و به عنوان تصویر اورجینال در نظر می‌گیریم و ۵۰٪ از تصویر تغییر داده شده را بریده و به عنوان تصویر دوم یا تصویری که باید لیبل آن را حدس بزنیم در نظر می‌گیریم.

سپس به کمک loaderها دیتاست را لود کرده و تصاویر را نمایش می‌دهیم. (در کد موجود است)

در مراحل بعدی کلاس‌ها را مطابق شکل پیاده‌سازی می‌کنیم و مدل را آموزش می‌دهیم. ما در این مدل از Xavier initialization استفاده کردیم. البته از initialization های دیگر نیز استفاده کرده بودیم که دیدیم Xavier بهترین نتایج را می‌دهد. همچنین پارامتر lr = ۰,۰۰۰۱ و از cross entropy loss به عنوان تابع هزینه استفاده کردیم.

نتایج همگی در کد وجود دارد.

سوالات تئوری در کد:

۱. مقادیر زیر را برای لایه های کانولوشن چه قدر در نظر گرفته اید؟

channel out size (conv(۱*۱))

padding(conv)

تعداد پدینگ برای کرنل‌های ۳در۳ برابر با ۱ است و برای کرنل ۱در۱ برابر با صفر. تعداد چنل خروجی نیز با توجه به تصویر بعدی برابر ٦٤ در نظر گرفتیم.

۲. دلیل استفاده از کانولوشن ۱در۱ چیست؟

دلیل استفاده کانوولشن ۱*۱ این است که با به کمک این فیلتر میتوانیم روابط بین کانال‌های تصویر را بدست آوریم. همانطور که مستحضرید در فیلتر ۱ در ۱ ما تمامی کانال‌ها را به یکی تبدیل میکنیم و این عمل سبب میشود که به عبارتی اطلاعات تمامی کانال و رابطه بین آنها در یک بعد بدست بیاید. طبیعتا با این عمل حجم داده نیز کاهش می‌یابد و این باعث نیاز کمتر به مموری و محاسبه است.

۳. بررسی کنید عکس ها با یک ،۲ یا ۳ تغییر کدام بهتر تشخیص داده میشوند؟

با توجه به confusion matrix برسم شده میبینیم که هرچه تعداد تغییرات کمتر باشد، مدل بهتر عمل می‌کند. همچنین تصاویری که چرخش و جابجایی داشته‌اند بدتر شناسایی می‌شوند. علت این امر این است که در چرخش با توجه به اینکه ما هم چرخش منفی و هم مثبت داریم، گاهی اوقات این چرخش بسیار نزدیک به تصویر اصلی بوده و نمی‌توان آن را به خوبی تشخیص داد. در مورد جابجایی نیز دقیقا همین نکته قابل ذکر است. مثلا اگر به قسمت dis & scl توجه کنید، میبینید که بسیاری از آنها صرفا تغییر مقیاس تلقی شده، زیرا جابجایی ما بسیار کم بوده است.