امیرحسین احمدی ۹۷۵۲۲۲۹۲ – تمرین دهم

۱- در ابتدا به قسمتی از تابع train که مربوط به پیداکردن best_acc بود، اضافه کردم که همیشه وزن بهترین مدل را درون drive ذخیره کند.

الف) برای پارت اول همان نمونه داده شده را اجرا کردم با این تفاوت که pretrained را فالس کردم تا وزن ها به صورت رندوم باشند در ابتدا. همچنین مقداری data augmentation در ابتدا انجام دادم تا احتمال اورفیت کمتر شود. سپس مدل را ۲۰ epoch آموزش دادم که با توجه به طولانی بودن این ایپوک ها همه در نوت بوک موجود نیست (همانطور که در نوتبوک دیده میشود، دو تیکه کد برای لود کردن وزن ها و اوپتیمایزر از مدل وجود دارد که در صورت استفاده نکردن از آن ها میتوان آن ها را کامنت کرد.) و فقط ۱۰ ایپوک آخر که با هم ران کردم موجود است و نتیجه نهایی و ایپوک آخر آن را در زیر نیز میبینید.

```
Epoch 9/9
----------
Iterating through data...
train Loss: 143.0047 Acc: 1.9686
Iterating through data...
val Loss: 149.6712 Acc: 1.4365
=> saving checkpoint

Training complete in 49m 26s
Best val Acc: 1.436508
```

با توجه به دقیق نبودن فرمول دقت در تابع ترین نتیجه ی نهایی اعتبار چندانی ندارد ولی با برسی ۱۰ ایپوک آخر میتوان دید که دقت رفته رفته زیاد میشود و لاس کاهش میابد.

ب) برای این قسمت ابتدا مدل ***pretrained*** را گرفته و آن را به همراه مدل کلسیفایر ساده ای که ساخته ایم به کلاس ***ResnetModel*** داده تا یک مدل کلسیفایر ساده و مدل رسنت برای ***feature extraction*** داشته باشیم. کلاس ***ResnetModel*** را نیز به گونه ای تغییر دادیم که درصورت وجود ***Classifier*** آن را به جای حالت دیفالت قرار دهد. همچنین ***requires_grad*** پارامتر های مدل ***resnet*** را فالس کرده تا آموزش ندیده و ***freeze*** شوند. در آخر نیز بقیه پارامتر های مدل جز پارامترهای ***resnet*** را به ***optimizer*** داده تا آن ها را ***optimize*** کند و مدل را ران میکنیم. این مدل نیز ۲۰ ایپوک ران شده که به علت محدودیت های موجود فقط ۱۴ ایپوک آخر آن قابل مشاهده است. نتیجه:

```
Epoch 14/14
----------
Iterating through data...
train Loss: 141.8839 Acc: 1.3647
Iterating through data...
val Loss: 138.0302 Acc: 1.4881

Training complete in 64m 32s
Best val Acc: 1.496032
```

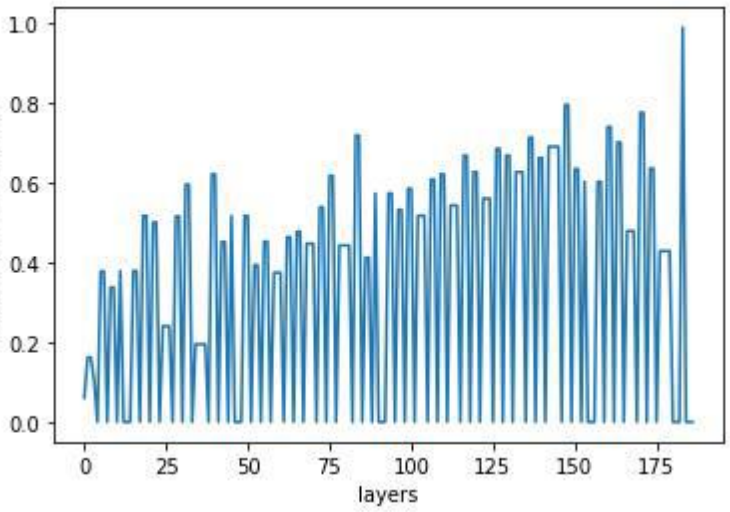
پ) برای این قسمت به جای ***classifier*** ساده از ***svm*** استفاده کردیم تا بتوانیم با مدل قبلی مقایسه کنیم. فقط به علت اضافه شدن ***svc*** مجمور شدم بخشی از تابع ترین را تغییر دهدم و به صورت دستی ***svc*** را ***optimize*** کنم، به این صورت که خروجی ***resnet*** را گرفته و به ***svc*** داده و مقادیر خطای آن را حساب کرده و اگه در فاز ***train*** بودیم آن را با ***label*** های داده شده فیت میکنیم. من سعی کردم که تا حد امکان بقیه تابع را تغییر ندهم، ولی نتایج خیلی با نتایج مدل قبلی تفاوت دارد و همچنین میتوان گفت که اورفیت شده زیرا از یک جایی به بعد دقت ولیدیشن کاهش هم میابد. برای این سوال نیز ۱۶ ایپوک آخر از ۲۰ ایپوک در نوتبوک موجود است.

```
Epoch 16/16
----------
Iterating through data...
train Acc: 0.1647
Iterating through data...
val Acc: 0.1667

Training complete in 77m 57s
Best val Acc: 0.301587
```

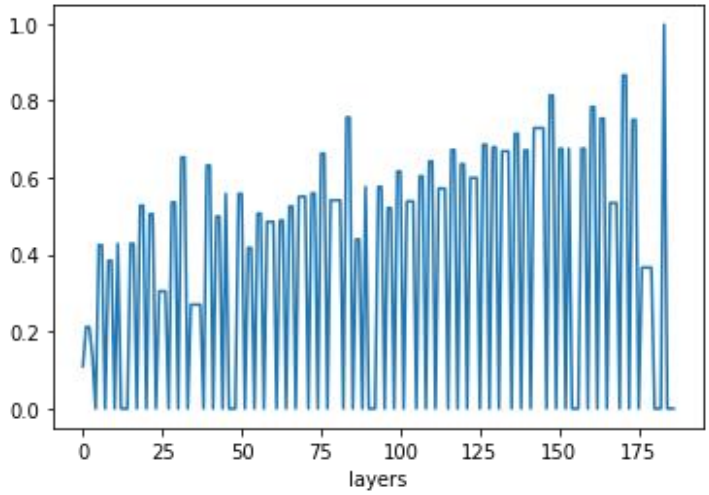
ت) برای این قسمت، همان مدل ***pretrained*** شده قسمت ب را در نظر گرفته و این بار همه وزن ها را با هدف ***fine-tune*** آموزش میدهیم و همانطور که مشاهده میشود نتایج بعد از ۵ ایپوک بسیار بهتر شده است.

```
Epoch 4/4
----------
Iterating through data...
train Loss: 133.2228 Acc: 1.3882
Iterating through data...
val Loss: 131.0654 Acc: 1.5992
=> saving checkpoint

Training complete in 26m 2s
Best val Acc: 1.599206
```

ث) برای این قسمت ابتدا یک کلاس هوک (مانند ***callback*** در ***keras***) تعریف کرده که بتوانیم خروجی هر لایه را بگیریم. آن را به انتهای همه لایه ها اضافه میکنیم. سپس برای سمپلی که برای این سوال درست کردیم خروجی را بدست آورده و به ازای هر لایه درصد مقادیر صفرش را بدست میاوریم و در نهایت پلات میکنیم. قبل از ***fine tune*** :



بعد از ***fine tune*** :



همانطور که مشاهده میشود برخی لایه ها درصد صفر بودنشان بعد از ***fine tune*** کاهش یافته است.