به نام خدا

تمرین سری هشتم درس یادگیری عمیق

دکتر محمدی

محمد یارمقدم

۹۶۴۶۲۱۰۴

سوال اول)

الف)

استفاده از dropout در مواقعی است که می خواهیم از overfitting جلوگیری کنیم. در این روش پس از فراخوانی این لایه بعد از هر لایه، به نسبتی(درصدی) که در آرگومان این لایه تعریف می شود، خروجی نورون های آن لایه نادیده گرفته میشود. پارامتر یا آرگومان نگهداری نورون ها در لایه dropout احتمال آموزش یک گره در خروجی است. در آن اگر مقدار پارامتر ۱ باشد هیچ نورونی حذف نمی شود و اگر صفر باد تمامی نورون ها در آموزش حذف شده و در نظر گرفته نمی شوند. در این صورت روند همگرایی شبکه به تاخیر افتاده و دیرتر overfit رخ می دهد. طبق مقاله معرفی این لایه و همچنین تحقیقات اخیر مشخص شده است که استفاده از این لایه با $p = 0.5$ پس از هر یک لایه های یک شبکه fully connected مفید است و استفاده می شود. همچنین استفاده از آن در خروجی لایه conv با مقدار $p = 0.1$ یا $p = 0.2$ پس از خروجی تابع فعالسازی می تواند نتیجه شبکه را بهبود دهد. مقدار خوب برای آن بین 0.5 تا 0.8 است که در لایه های ورودی از مقادیر بزرگ تر مثل 0.8 استفاده می کنیم.

اما استفاده از آن بلافاصله پس از لایه conv و با نرخ بالا توصیه نمی شود زیرا با حرکت فیلتر روی هر بخش از داده ورودی پاسخ خاص همان بخش داده می شود و استفاده از آن با نرخ بالاتر از 0.1 ممکن است ویژگی ها مهمی یادگیری نشود.

در قسمت دوم سوال اگر بخواهیم روند همگرایی زودتر اتفاق بیفتد استفاده از نرخ نگهداری پایین تر مناسب است ولی اگر بخواهیم شبکه را پیچیده تر کنیم باید آنرا بالا در نظر بگیریم. در کل عمدتا در اغلب موارد $p = 0.5$ در نظر گرفته می شود.

ب)

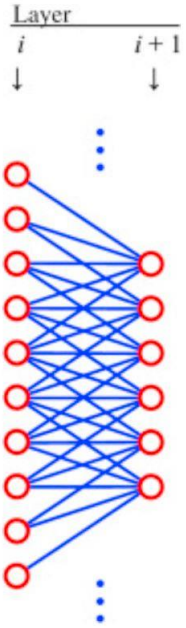
در این مورد نیز هر چه نرخ نگهداری بالاتر باشد ظرفیت شبکه بالاتر خواهد بود زیر داده و اطلاعات کمتری نادیده گرفته می شود و یادگیری بهتر رخ خواهد داد. بنابراین ظرفیت نیز بالاتر خواهد بود. به علاوه هر چه نرخ نگهداری کمتر باشد ظرفیت نیز کمتر می شود. چون داده ها و نورون های بیشتری نادیده گرفته می شوند و ممکن اطلاعات مهمی از شبکه آموزش نبینند.

سوال دوم)

در لایه fully connected هر نورون به تمامی نورون لایه قبل متصل است و هر نورون وزن خاص خود را دارد. در این مورد آموزش ترکیبی از کلیت شبکه تا قبل از این لایه است. بنابراین هدف عمومی در داده دارد و هزینه مموری و محاسبات زیادی دارد. زیرا هر نورون وزن خاص خو را دارد.

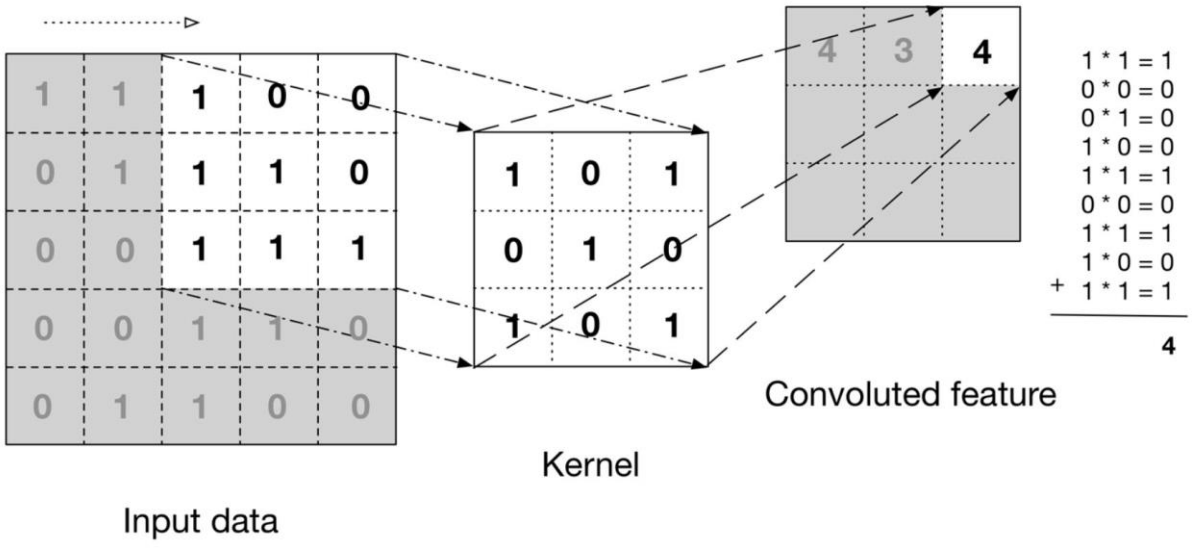
موارد استفاده:

- در شبکه مسائل classification در پشت feature extractor معمولا از این لایه استفاده می شود که باعث میشود ۲ بعد feature extractor را به بردار از feature ها تبدیل می کند.
- در شبکه مسائل segmentation کاربردی ندارد.
- عمدتا در لایه آخر شبکه های عصبی استفاده می شود.
- شبکه های feed forward



در لایه convolutional هر نورون فقط یه یک سری از نورون های لایه قبل متصل است و تمرکز به جای کلیت داده ورودی بر روی قسمت های محدود تر می رود و feature های محلی در قسمت های مختلف داده بررسی می شود. شرط استفاده از این لایه این است که داده ورودی در قسمت های مختلف وابستگی و ارتباط داشته باشد تا بتوان با بررسی هر بخش آن آموزش را پیش برد. در مواردی که هر بخش از داده مستقل است استفاده از این لایه بیهوده خواهد بود. هم چنین این لایه برای آموزش از یک فیلتر برای بررسی در همه قسمت ها استفاده می کند. هزینه محاسباتی به شدت در این لایه نسبت به مورد قبل کمتر می شود.

بنابراین این لایه در پردازش و آموزش تصاویر و صدا و پردازش سیگنال و speech recognition می تواند کاربرد گسترده ای داشته باشد.



در لایه locally connected همه چیز مانند لایه convolutional است و لایه به طور کامل به قبل متصل نیست. همانطور که در مورد قبل بیان شد یک فیلتر در کل برای همه قسمت ها و نورون ها وجود دارد. درحالیکه در این لایه هر نورون و هر قسمت فیلتر خاص خود را دارد. بنابراین تعداد پارامتر ها بسیار بیشتر خواهد شد. به طور دقیق تعداد پارامتر ها پارامتر های یک فیلتر در تعداد نورون های خروجی ضرب می شود. بنابراین به دلیل وجود بیشتر پارامتر اگر داده کافی در دسترس نباشد ممکن نتیجه کار با overfitting همراه باشد. اما مزیت این لایه این است که این امکان وجود دارد که یادگیری feature های متفاوت در قسمت های مختلف عکس اتفاق بیفتد.

مورد اصلی این لایه با توجه به موارد بیان شده در بالا در face verification است. از نمونه های این شبکه ها DeepFace و DeepID3 است. در کل اما کاربرد کمتری نسبت به لایه convolution دارد.

سوال سوم)

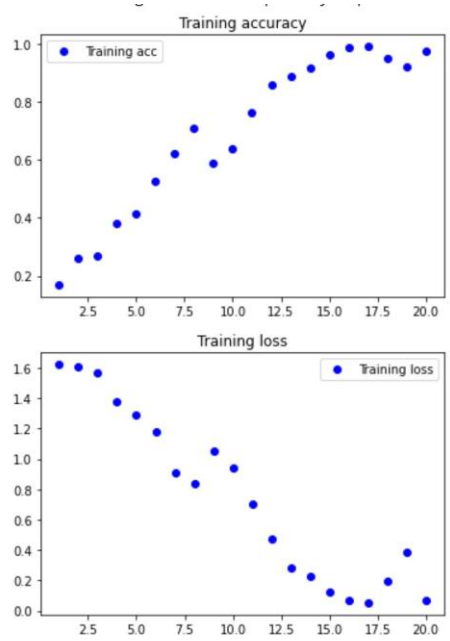
در این سوال ابتدا generator متناسب را به فرمت زیر نوشته تا عکس های جدید اضافه شود.

```
train_datagen =  ImageDataGenerator(rescale=1./255)

train_datagen_aug =  ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale=1./255)

test_datagen_aug =  ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

train_generator_aug  = train_datagen_aug.flow_from_directory('./train', batch_size=batch_size, class_mode='categorical', shuffle=True)

validation_generator_aug  = test_datagen_aug.flow_from_directory('./test', batch_size=batch_size, class_mode='categorical', shuffle=True)

train_generator = train_datagen.flow_from_directory('./train', batch_size=batch_size, class_mode='categorical', shuffle=True)

validation_generator = test_datagen.flow_from_directory('./test', batch_size=batch_size, class_mode='categorical', shuffle=True)
```

سپس برای استفاده از داده افزایی از کلاس ImageDataGenerator با پارامتر های متفاوت ۳ نمونه برای آموزش و تست تهیه کردم و شبکه را برای هر کدام جداگانه آموزش دادم که نتیجه مقایسه استفاده از داده افزایی و عدم استفاده از آن در زیر آورده شده است:

- در ابتدا شبکه را با داده های اصلی آموزش دادم.

```
loss: 3.2738 - accuracy: 0.4211
```

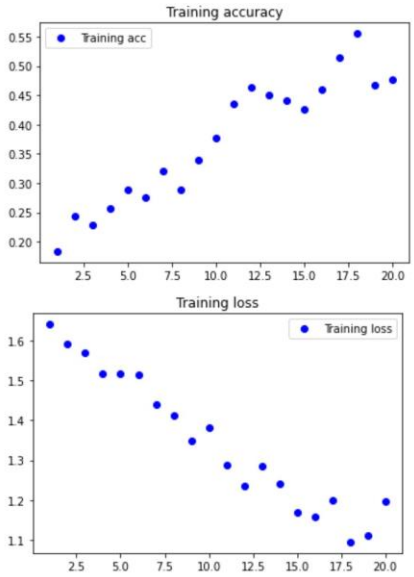


همانطور که نتایج گویاست اورفیت رخ داده است و نتایج مطلوبی بدست نیامده است.

- در قسمت بعد از داده افزایی استفاده کردم و سه بار با پارامتر های مختلف این داده افزایی را انجام دادم و شبکه را آموزش دادم.

داده افزایی اول:

```
loss: 1.2292 - accuracy: 0.4737
```

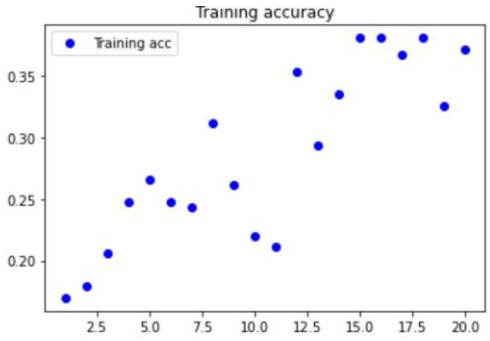


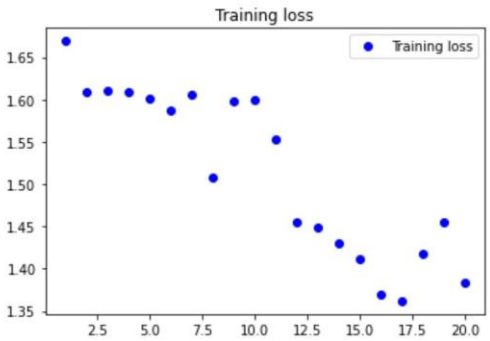
این داده افزایی با پارامتر های زیر انجام شده است:

```
rescale=1./255, rotation_range=40,width_shift_range=0.2,
height_shift_range=0.2,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,fill_mode='nearest'
```

داده افزایی دوم:

```
loss: 1.4348 - accuracy: 0.3860
```

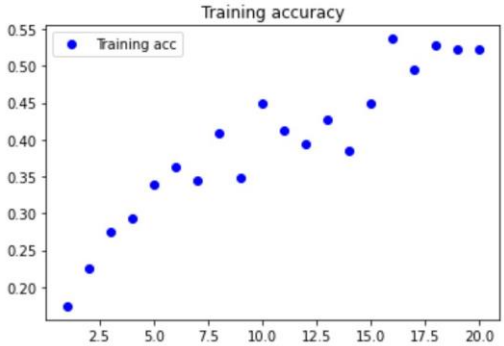


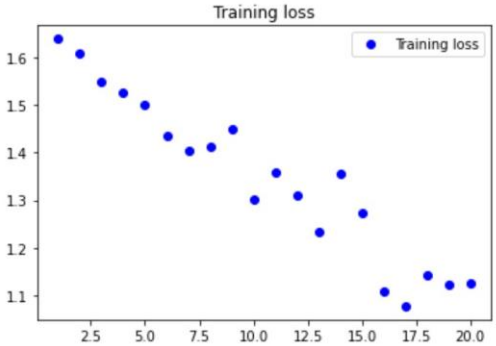


این داده افزایی با پارامتر های همان حالت بالا انجام شد با این تفاوت که به جای چرخش عمودی از چرخش افقی استفاده کردم.

داده افزایی سوم:

```
loss: 1.1881 - accuracy: 0.4211
```



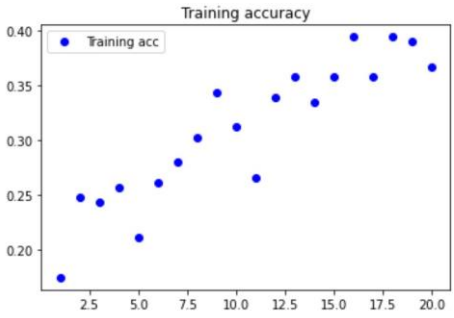


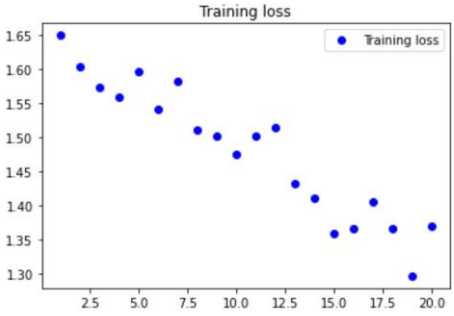
این داده افزایی ترکیب چرخش عمودی و افقی را دارد.

در این حالات مشاهده شد که درصد دقت و خطای شبکه به علت وجود داده های بیشتر بهبود یافت و امکان رخداد اورفیت نیز کمتر شد.

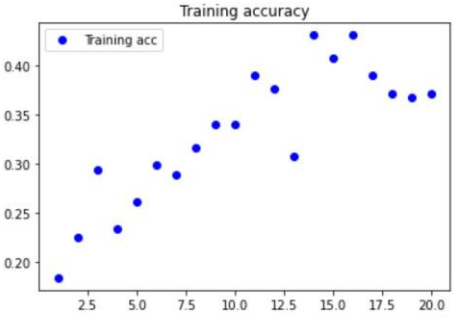
در مورد سوم داده افزایی را همراه با استفاده از لایه dropout با نرخ های نگهداری مختلف استفاده کردم.
برای نرخ های نگهداری مختلف نتایج زیر بدست آمد. در سه حالت اول فقط قبل از لایه آخر که dense است از dropout استفاده شد:

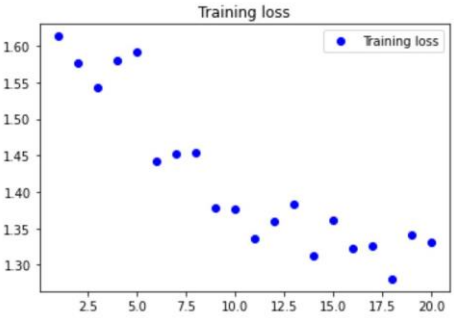
نرخ نگهداری = 0.2



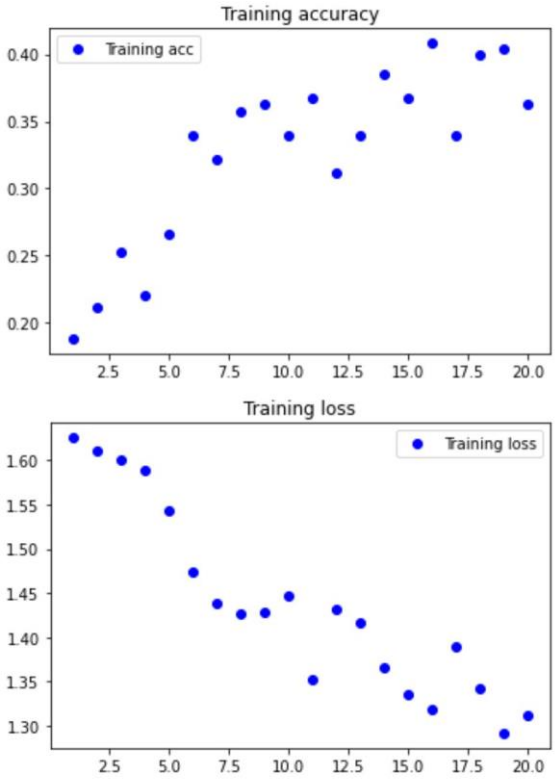


نرخ نگهداری = 0.5

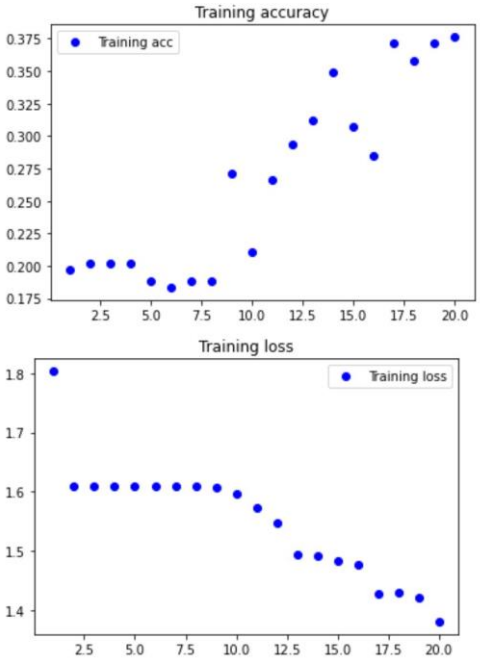




نرخ نگهداری = 0.8



در حالت آخر پس از هر لایه موجود در شبکه از dropout استفاده شد و نرخ نگهداری برای همه لایه ها 0.5 ست شد. نتیجه به صورت زیر شد:



بدترین نتیجه در این حالت رخ داد که آموزش اصلا خوب رخ نداده و اطلاعات مهمی از شبکه از بین رفت. نرخ خطا و آموزش پس از evaluate به ترتیب 1.59 و 0.29 شد که اصلا مطلوب نیست و به شدت کم است که به این معنا است که آموزش خوب اتفاق نیفتاده است.

بین سه حالت بالا نیز برای داده افزایی نوع اول نرخ نگهداری 0.8 بهترین نتیجه را داد که به این معنا است که در قسمت های مختلف اطلاعات متفاوتی موجود است و حذف حدودا نصف اطلاعات بهینه نخواهد بود و باید تا حد امکان قسمت های مختلف حفظ شود.