دانشکده مهندسی کامپیوتر

تمرین سری ۱۴ بینایی کامپیوتر

نام درس

مبانی بینایی کامپیوتر

نام دانشجو

زهرا انوریان

نام استاد درس

دکتر محمدی

زمستان ۱۳۹۹

سوالات

۱. الف) علت به وجود آمدن مشکل underfitting در شبکه چیست و چه زمانی رخ می‌دهد؟ (۵ نمره)
پاسخ: زمانی این اتفاق رخ می‌دهد که مدل ما به دلیل اینکه داده‌ی زیادی یاد نگرفته است، نمی‌تواند به درستی پیش‌بینی کند و دقت خوبی ندارد و دارای خطاست.

ب) مشکل بیش‌برازش (overfitting) چیست؟ چه راه‌هایی برای رفع آن پیشنهاد می‌کنید؟ (۵ نمره)
پاسخ: زمانی این اتفاق رخ می‌دهد که مدل ما داده‌های آموزش را حفظ کرده باشد یعنی دقت آن بر روی داده‌های آموزش زیاد و بر روی داده‌های تست کم باشد و به خوبی پیش‌بینی نکند و برای حل این مشکل می‌توان مسئله را برایش سخت‌تر کرد. برای این کار می‌توان از dropout (حذف برخی از نورون‌ها از شبکه با یک rateای به صورت تصادفی) استفاده کرد (در واقع به شبکه نویز اضافه می‌کنیم) و همچنین می‌توان با استفاده از الگوریتم‌های داده افزایی از حفظ کردن داده‌های ورودی توسط مدل جلوگیری کنیم.

۲. استفاده از شبکه‌های pre-train یکی از روش‌های موثر برای مقداردهی اولیه پارامترهای یک شبکه است. در مورد نحوه استفاده از این نوع یادگیری و مزایای آن، توضیح دهید. (۱۰نمره)
پاسخ: اگر دیتاست ما کوچک بود می‌توان از شبکه‌های pre-train استفاده کرد به این صورت که کل شبکه به صورت فریز قرار گیرد و فقط لایه‌ی آخر آن که دسته‌بندی داده‌ها را برعهده دارد برحسب دسته‌های مسئله‌ی خود آموزش داد اما اگر دیتاست ما بزرگ بود می‌توان تنها لایه‌های کانولوشن را فریز گذاشت و لایه‌های مربوط به استخراج ویژگی را با دیتاست خود آموزش داد تا ویژگی‌های بدست آمده دقیق‌تر باشد و یا حتی می‌توان به جای مقداردهی اولیه‌ی پارامترهای شبکه از پارامترهای شبکه‌ی آموزش ددیده استفاده کرد و اگر دیتاست متفاوت از شبکه‌ی آموزش دیده بود می‌توان در هر لایه‌ای از ابتدای شبکه دسته‌بند را اعمال کرد و تنها ویژگی‌های عام را توسط شبکه‌ی آموزش دیده بدست آورد. مزیت استفاده از شبکه‌های آموزش دیده، افزایش سرعت در آموزش، بالا بردن دقت شبکه و حتی آسان‌تر شدن کار می‌باشد.

۳. اگر یک تصویر ۳ کاناله با ابعاد ۲۸ در ۲۸ داشته باشیم، تعداد پارامترها را برای یک لایه کانوولوشنی با تعداد ۱۲۸ فیلتر ۵×۵ را در حالتی که از کانوولوشن ساده و کانوولوشن با عمق جداپذیر (Depthwise Separable Convolution) استفاده می‌شود (به ازای depth_multiplier برابر با ۲)، به دست آورده و با هم مقایسه کنید. (۱۰نمره)
پاسخ:

$$Parametes: F.F.K.D_1 + k(bias) = (5 * 5 * 128 * 3) + 128 = 9728$$

$$Parameters: 2 * (5 * 5 + 1) + 128 * (3 * 2 + 1) = 306 + 17152 = 948$$

۴. (۷۰ نمره) در این سوال میخواهیم یک مسئله دسته‌بندی را با استفاده از تکنیک‌های داده‌افزایی و انتقال یادگیری انجام دهیم.

در این سوال از مجموعه‌داده cars استفاده می‌کنیم که شامل ۱۶۱۸۵ تصویر است و ۱۹۶ کلاس دارد. تعداد تصاویر آموزشی در این مجموعه‌داده برابر با ۸۱۴۴ و تعداد تصاویر آزمون برابر با ۸۰۴۱ است. برای دانلود این مجموعه‌داده کافی است مراحلی که در فایل HW14.ipynb ذکر شده است انجام دهید تا مجموعه‌داده در Google Drive شما ذخیره گردد. پس از انجام این مراحل پوشه‌هایی به نام stanford_car_dataset در درایو شما ایجاد شده است که محتویات آن باید مطابق شکل زیر باشد:

داخل هر کدام از پوشه‌های train و test، ۱۹۶ پوشه دیگر وجود دارد که داخل هر کدام از آن‌ها تصاویر مربوط به یکی از ۱۹۶ کلاس است و نام این پوشه‌ها نشان‌دهنده‌ی آن کلاس است. برای مثال در پوشه 2012BMW 1 Series Convertible تصاویر مربوط به کلاس 2012BMW 1 Series Convertible وجود دارد. در ادامه توضیح خواهیم داد که چگونه این ساختار ذخیره‌سازی به ما کمک خواهد کرد تا مجموعه‌داده را به مدل یادگیری عمیق ورودی بدهیم.

معمولاً برای آموزش مدل‌های یادگیری عمیق، از مجموعه‌داده‌های نسبتاً بزرگی استفاده می‌شود که حتی پیشرفته‌ترین سخت‌افزارها، حافظه کافی برای پردازش داده‌ها به صورت یکجا و یکپارچه را ندارند. به همین دلیل است که ما باید راه‌های دیگری برای انجام کارآمد آن پیدا کنیم. در ادامه قصد داریم به شما نشان دهیم که چگونه مجموعه‌داده را در چندین هسته و در زمان اجرا تولید کرده و بلافاصله آن را به مدل یادگیری عمیق خود بدهیم. بدین منظور کلاسی به نام ImageDataGenerator در keras پیاده‌سازی شده است که همزمان با تولید batchای از تصاویر در زمان اجرا، می‌تواند اعمال مختلف داده‌افزایی مانند rotation، flipping و ... را انجام دهد. باتوجه به ساختار ذخیره‌سازی مجموعه‌داده cars

که قبلا ذکر شد؛ در این مسئله می‌توانیم به راحتی از تابع flow_from_directory برای تولید تصاویر استفاده کنیم. در غیر این صورت باید خودمان با توجه به نیاز مسئله کلاس generator را پیاده‌سازی می‌کردیم (اگر علاقه‌مند به نوشتن custom generator هستید؛ از این لینک استفاده نمائید). برای آشنایی بیشتر با ImageDataGenerator و نحوه استفاده از آن برای داده‌افزایی، از مثال موجود در لینک استفاده نمائید.

الف) در این قسمت شما باید برای هرکدام از داده‌های آموزشی و داده‌های تست، به ترتیب دو generator با استفاده از ImageDataGenerator و flow_from_directory با نام‌های train_datagen و test_datagen بسازید. سپس از resent50 (موجود در کراس) با **وزنه‌ای تصادفی** استفاده کنید و یک مدل برای این مسئله طراحی کنید. پس از آموزش مدل، آن را بر روی داده‌های تست ارزیابی کنید.

پاسخ: با استفاده از batch_size=80 و epoch=25 توانستم به دقت ۹۵درصد بر روی داده‌های آموزش و به دقت ۱۹درصد بر روی داده‌های تست برسم (البته با epochهای بیشتر بر روی داده‌های تست می‌توان به درصد بهتری نیز رسید) که این نشان‌دهنده‌ی overfiting مدل می‌باشد که در واقع داده‌های آموزش را حفظ کرده و دقتش در آن بالاست ولی در داده‌ی تست که جدید است و تا به حال آن را ندیده ضعیف عمل می‌کند. حال برای رفع این مسئله می‌توان از dropout و یا داده‌افزایی استفاده کرد که در بخش بعد داده‌افزایی را بررسی می‌کنیم.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 2048)              23587712
_________________________________________________________________
dense (Dense)                (None, 196)               401604
=================================================================
Total params: 23,989,316
Trainable params: 23,936,196
Non-trainable params: 53,120
```

```
Epoch 24/25
102/102 [==============================] - 102s 993ms/step - loss: 0.2119 - accuracy: 0.9438
Epoch 25/25
102/102 [==============================] - 105s 1s/step - loss: 0.1722 - accuracy: 0.9535
<tensorflow.python.keras.callbacks.History at 0x7f2b08f9dda0>
```

```
model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
101/101 [==============================] - 85s 833ms/step - loss: 5.6033 - accuracy: 0.1897
[5.603324890136719, 0.18965302407741547]
```

ب) در این قسمت، از مدلی که در قسمت الف طراحی کرده‌اید؛ استفاده کنید و این بار مدل را با داده‌افزایی آموزش دهید. بنابراین برای داده‌های آموزشی یک generator با استفاده از ImageDataGenerator و flow_from_directory با نام train_datagen_aug بسازید که برای داده‌های آموزشی نیز داده‌افزایی

انجام دهد. پس از آموزش مدل، آن را بر روی داده‌های تست ارزیابی کنید. نتیجه حاصل از این قسمت را با قسمت الف مقایسه نمائید.

پاسخ: در این قسمت با استفاده از تابع ImageDataGenerator و به کمک پارامترهای آن در داده‌های آموزش تغییر ایجاد می‌کنیم و از حفظ کردن داده‌ها توسط مدل جلوگیری می‌کنیم. با تعیین epoch=50 و batch_size=80 به ۸۸درصد بر روی داده‌های آموزش و برروی داده‌های تست به ۴۲درصد رسیدم.

```
Epoch 49/50
102/102 [==============================] - 168s 2s/step - loss: 0.4272 - accuracy: 0.8740
Epoch 50/50
102/102 [==============================] - 171s 2s/step - loss: 0.3903 - accuracy: 0.8850
<tensorflow.python.keras.callbacks.History at 0x7f1250b4d240>
```

```
model.evaluate(test_generator, batch_size=BATCH_SIZE)
```

```
101/101 [==============================] - 1793s 18s/step - loss: 3.3639 - accuracy: 0.4257
[3.3639113903045654, 0.4256933331489563]
```

پ) در این قسمت از شبکه از قبل آموزش‌دیده ResNet50 بر روی مجموعه‌داده Imagenet استفاده کنید (لینک) و پس از آن یک دسته‌بند قرار دهید. موقع آموزش یک بار تمامی وزن‌های شبکه ResNet50 را freeze کنید و مابقی وزن‌ها را آموزش دهید. در بار دیگر بگذارید تا وزن‌های شبکه ResNet50 نیز آموزش ببینند (fine tuning). برای آموزش این مدل در هر دو حالت از generator (train_datagen_aug با داده‌افزایی) استفاده نمائید. پس از آموزش مدل، آن را بر روی داده‌های تست ارزیابی کنید. نتیجه حاصل از این قسمت را با قسمت الف و ب مقایسه نمائید.

پاسخ: ابتدا وزن‌های شبکه‌ی imagnet را برای شبکه‌ی resnet50 درنظر می‌گیریم و کل شبکه را فریز می‌کنیم و یک لایه‌ی dense که دارای ۱۰۲۴ نورون است را به شبکه‌ی resnet50 اضافه می‌کنیم. حال با epoch=50 و batch_size=80 به ۴۵درصد دقت بر روی داده‌های تست می‌رسیم.

```
Epoch 47/50
102/102 [==============================] - 222s 2s/step - loss: 0.4317 - accuracy: 0.8719 - val_loss: 2.8020 - val_accuracy: 0.4512
Epoch 48/50
102/102 [==============================] - 221s 2s/step - loss: 0.4115 - accuracy: 0.8756 - val_loss: 2.9014 - val_accuracy: 0.4333
Epoch 49/50
102/102 [==============================] - 222s 2s/step - loss: 0.4329 - accuracy: 0.8730 - val_loss: 2.8095 - val_accuracy: 0.4476
Epoch 50/50
102/102 [==============================] - 222s 2s/step - loss: 0.4073 - accuracy: 0.8803 - val_loss: 2.8265 - val_accuracy: 0.4419
```

و برای fine tuning نیز ۳۰ لایه‌ی آخر شبکه‌ی resnet50 با وزن‌های imagenet را trainable درنظر می‌گیریم و بقیه را فریز می‌کنیم و از الگوریتم SGD (با momentum) برای بهینه‌سازی استفاده می‌کنیم. با epoch=35 و batch_size=80 به ۷۵درصد دقت بر روی داده‌های تست می‌رسیم.

```
Epoch 25/35
102/102 [==============================] - 232s 2s/step - loss: 0.0532 - accuracy: 0.9851 - val_loss: 1.3232 - val_accuracy: 0.7071
Epoch 26/35
102/102 [==============================] - 233s 2s/step - loss: 0.0413 - accuracy: 0.9891 - val_loss: 1.1602 - val_accuracy: 0.7514
Epoch 27/35
102/102 [==============================] - 233s 2s/step - loss: 0.0419 - accuracy: 0.9867 - val_loss: 1.3729 - val_accuracy: 0.7141
Epoch 28/35
102/102 [==============================] - 232s 2s/step - loss: 0.0447 - accuracy: 0.9867 - val_loss: 1.2395 - val_accuracy: 0.7301
Epoch 29/35
102/102 [==============================] - 233s 2s/step - loss: 0.0332 - accuracy: 0.9913 - val_loss: 1.2434 - val_accuracy: 0.7385
Epoch 30/35
102/102 [==============================] - 233s 2s/step - loss: 0.0372 - accuracy: 0.9887 - val_loss: 1.6215 - val_accuracy: 0.6757
Epoch 31/35
102/102 [==============================] - 233s 2s/step - loss: 0.0410 - accuracy: 0.9877 - val_loss: 1.5911 - val_accuracy: 0.6847
Epoch 32/35
102/102 [==============================] - 232s 2s/step - loss: 0.0382 - accuracy: 0.9896 - val_loss: 1.1970 - val_accuracy: 0.7405
Epoch 33/35
102/102 [==============================] - 232s 2s/step - loss: 0.0374 - accuracy: 0.9888 - val_loss: 1.2060 - val_accuracy: 0.7417
Epoch 34/35
102/102 [==============================] - 234s 2s/step - loss: 0.0347 - accuracy: 0.9905 - val_loss: 1.2535 - val_accuracy: 0.7500
Epoch 35/35
102/102 [==============================] - 233s 2s/step - loss: 0.0328 - accuracy: 0.9907 - val_loss: 1.3180 - val_accuracy: 0.7268
```

چند نکته برای طراحی مدل یادگیری عمیق:

- برای آموزش مدل از تابع fit استفاده نمائید و برخلاف قبل، به جای ورودی دادن داده‌های آموزشی و برچسب آن، کافی است generator داده‌های آموزشی را به تابع fit ورودی بدهید.

نیازی به استفاده از fit_generator در نسخه جدید تنسورفلو نیست.

- برای ارزیابی مدل نیز کافی است از تابع evaluate استفاده نمائید و generator داده‌های تست را به آن ورودی دهید.

- موفق باشید.