به نام خدا

نام: فاطمه زهرا بخشنده **استاد: دکتر محمدرضا محمدی**

شماره دانشجویی: 98522157 **درس مبانی بینایی کامپیوتر**

گزارش تمرین 11:

سوال دوم:

الف) مدل fully connected را به صورت زیر تعریف می کنیم:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 3072)              0

 dense (Dense)               (None, 128)               393344

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 256)               33024

 dense_3 (Dense)             (None, 10)                2570

=================================================================
Total params: 445,450
Trainable params: 445,450
Non-trainable params: 0
```

مدل کانولوشنی را به صورت زیر تعریف می کنیم:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten_1 (Flatten)         (None, 512)               0

 dense_4 (Dense)             (None, 256)               131328

 dense_5 (Dense)             (None, 10)                2570

=================================================================
Total params: 227,146
Trainable params: 227,146
Non-trainable params: 0
```

تعداد پارامتر های مدل اول تقریبا دو برابر پارامتر های مدل دوم است.

برای مدل fully connected دقت و خطا روی داده های تست به صورت زیر است:

```
Loss and Accuracy on Test set :
313/313 [==============================] - 1s 2ms/step - loss: 1.6584 - accuracy: 0.4002
```

برای مدل کانولوشنی دقت و خطا روی داده های تست به صورت زیر است:

```
Loss and Accuracy on Test set :
313/313 [==============================] - 1s 3ms/step - loss: 1.4323 - accuracy: 0.6853
```

می بینیم که در مدل کانولوشنی، دقت 28% افزایش یافته است. در اینجا loss به اندازه 0.2 کم شده است. بیشتر اوقات مشاهده می کنیم که دقت با کاهش خطا افزایش می یابد. اما همیشه اینطور نیست. دقت و ضرر تعاریف متفاوتی دارند و چیزهای مختلفی را می سنجند. اغلب به نظر می رسد که نسبت معکوس دارند، اما هیچ رابطه ریاضی بین این دو معیار وجود ندارد.

1. در واقع تابع loss را تعریف می کنیم تا وزن ها هر دفعه طوری تنظیم شوند که loss، minimize شود. اما هیچ رابطه ای بین خطا و دقت وجود ندارد. و نمی توانیم انتظار داشته باشیم این دو در یک زمان دقیقا با همدیگر تغییر کنند.
   accuracy وقتی بیشتر می شود که مدل برای نمونه های بیشتری پاسخ درست بدهد. برای مثال ممکن است در یک iteration خروجی مدل به ازای تعدادی نمونه، به خروجی اصلی (labels) آن ها از دفعه پیش کمی نزدیکتر شود، اما همچنان مدل برای آن ها پاسخ درستی ندهد پس در این case خطا کاهش می یابد اما دقت افزایش نمی‌یابد. (مثلا اگر تابع فعالسازی sigmoid داریم، و label نمونه ما 1 است، احتمال 0.2 و 0.3 هردو خروجی 0 می دهند اما loss در حالت دوم کمتر می شود، ولی accuracy بیشتر نمی شود). پس همیشه کم شدن loss به معنی بیشتر شدن دقت نیست.

2. همچنین ممکن است به دقت خیلی خوبی برسیم اما همچنان loss داشته باشیم چون با اینکه مدل در نهایت به ازای نمونه های زیادی پاسخ درست می دهد، هنوز خروجی مدل به ازای نمونه ها، می‌تواند به خروجی اصلی (labels) آن ها نزدیکتر شود. (مثلا اگر تابع فعالسازی sigmoid داریم و label نمونه ما 1 است، احتمال 0.67 خروجی 1 می دهد اما loss انتظار دارد این احتمال به 1.0 برسد). پس این حالت نیز ممکن است.

3. همچنین باید توجه داشته باشیم زمانی می توانیم دو مدل را با اندازه loss آن ها مقایسه کنیم که دو مدل از تابع loss یکسان استفاده کنند. چون توابع خطای مختلف، خروجی و order متفاوتی از هم دارند. پس بهتر است دو مدل را از نظر accuracy مقایسه کنیم.

ب) در مدل fully connected مدت زمان اجرای هر ایپاک 5s و در مدل کانولوشنی اکثرا 6s و برخی هم 7s است.

پ) در حالت کلی خیر. مدت زمان اجرا بستگی به تعداد عملیات ریاضی در هر epoch دارد. البته وقتی تعداد پارامتر های مدل کمتر باشد، تعداد محاسبات کمتری برای مشتق گیری و آپدیت وزن ها داریم، اما زمان اجرا به عملیات ریاضی دیگری نیز جز اینها بستگی دارد.

ویژگی مثبت مدل CNN اشتراک گذاری وزن ها است. پس تعداد پارامتر کمتری دارد. اما از همین پارامتر ها برای محاسبات ریاضی در کل عکس و مکان های مختلف آن استفاده می کند. یعنی تعداد محاسبات زیادی دارد. در این سوال نیز دیدیم هر epoch مدل کانولوشنی بیشتر طول کشید.

یا مثلا شبکه های RNN تعداد پارامتر های خیلی کمی دارند. اما زمان آموزش طولانی دارند، چون همان وزن‌های کم را باید در طول زمان share کنند و محاسبات زیادی انجام می دهد.

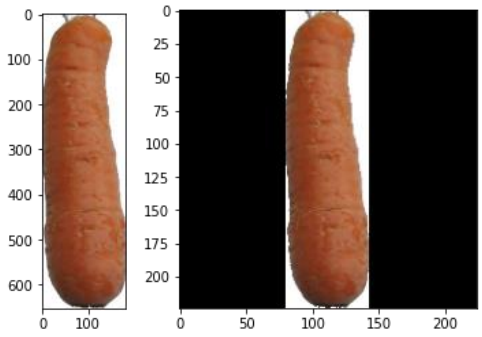
نکته: عملیات ها هم دو دسته اند. عملیاتی که قابل vectorize شدن دارند، و عملیاتی که حتما باید پشت سر هم انجام شوند.

همچنین زمان اجرا به ابعاد ورودی نیز بستگی دارد چون ابعاد ورودی روی تعداد پارامتر های مدل fully connected و روی تعداد عملیات ریاضی مدل کانولوشنی تاثیر دارد.

منابع: لینک و لینک

سوال سوم:

الف) ابتدا از تابع resize در OpenCV استفاده می کنیم. و بعد بزرگتر تصویر را به 224 رسانده و بعد کوچکتر آن را نیز به نسبت تغییر بعد بزرگتر، عوض می کنیم. سپس به اندازه نیاز، به اطراف بعد کوچکتر تصویر، border اضافه می کنیم تا ابعاد تصویر 224 در 224 شود.



ب) ابتدا لایه های کانولوشنی مدل Resnet را با وزن های رندوم به مدل sequential خود اضافه می کنیم. top layer آن مربوط به دسته بندی 1000 تصویر است، پس مورد نیاز ما نیست. ابتدا یک لایه flatten می زنیم. دو لایه dense مربوط به تسک خودمان به آن اضافه می کنیم به طوری که 24 نورون خروجی داشته باشیم.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 24)                12312

=================================================================
Total params: 24,649,112
Trainable params: 24,595,992
Non-trainable params: 53,120
_________________________________________________________________
```

مدل را train می کنیم.

پ) این دفعه لایه های کانولوشنی مدل Resnet را با وزن های imagenet به مدل sequential خود اضافه می کنیم. top layer آن مربوط به دسته بندی 1000 تصویر است، پس مورد نیاز ما نیست. ابتدا یک لایه flatten می زنیم. از یک لایه dropout استفاده کرده، سپس یک لایه dense با 512 نورون و یک لایه خروجی با 24 نورون به آن اضافه می کنیم. لایه های کانولوشنی Resnet را فریز می کنیم. در این حالت تقریبا 1 میلیون پارامتر trainable داریم.

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 3s 0us/step
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 flatten_1 (Flatten)         (None, 2048)              0

 dropout (Dropout)           (None, 2048)              0

 dense_2 (Dense)             (None, 512)               1049088

 dense_3 (Dense)             (None, 24)                12312

=================================================================
Total params: 24,649,112
Trainable params: 1,061,400
Non-trainable params: 23,587,712
_________________________________________________________________
```

مدل را train می کنیم.

ت) دقت و خطای دو مدل روی داده تست:

```
resnet.evaluate(test_generator)
```

```
33/33 [==============================] - 17s 511ms/step - loss: 3.0231 - acc: 0.5826
[3.023050546646118, 0.5826366543769836]
```

```
fine_tune_resnet.evaluate(test_generator)
```

```
33/33 [==============================] - 19s 543ms/step - loss: 0.4047 - acc: 0.8624
[0.4047434329986572, 0.8623794317245483]
```

با آموزش دو مدل روی داده train نتایج زیر را مشاهده می کنیم:

1. مدل اول خیلی زود داده آموزشی را حفظ می کند و دقت آن به 98% می رسد. دقت مدل دوم روی داده آموزشی تقریبا به 80% می رسد.
2. دقت مدل اول روی داده تست 58% می شود که به این معنی است که به دلیل داشتن مدل پیپیده و تعداد داده های بسیار کم، و شروع از وزن های کاملا رندوم، روی داده آموزشی overfit شده است. اما مدل دوم generalization بسیار خوبی داشته است و روی داده تست به دقت 86% رسیده است!
3. آموزش مدل اول خیلی طول کشید چون مدل Resnet مدلی با لایه های زیاد و تقریبا 25 میلیون پارامتر است. اما زمان آموزش مدل دوم خیلی کمتر بود چون لایه های کانولوشنی Resnet را فریز کرده و پارامتر های trainable مدل به 1 میلیون کاهش یافتند.
4. نتیجه می گیریم وقتی تعداد دیتای خیلی زیادی نداریم برای بهترین نتیجه می توانیم از یک مدل pretrained با وزن های آموخته شده توسط یک task بزرگتر با دیتای بیشتر، استفاده کنیم، و آن را fine tune کنیم.

منابع: لینک و لینک