HW11

بابک بهکام کیا

سوال 2)

در ابتدا دو مدل یکی با لایه های fully connected و دیگری با لایه های Convolutional می سازیم.

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

 dense (Dense)               (None, 64)                196672

 dense_1 (Dense)             (None, 32)                2080

 dense_2 (Dense)             (None, 16)                528

 dense_3 (Dense)             (None, 10)                170

=================================================================
Total params: 199,450
Trainable params: 199,450
Non-trainable params: 0
_________________________________________________________________
```

```
# Fully connected model
fc_model = keras.Sequential()
fc_model.add(keras.layers.Input(shape=x_train[0].shape))
# Write your code here
# Add Flatten layer and few Dense layers
fc_model.add(Flatten())
# fc_model.add(Dense(128, activation='sigmoid'))
fc_model.add(Dense(64, activation='sigmoid'))
fc_model.add(Dense(32, activation='sigmoid'))
fc_model.add(Dense(16, activation='sigmoid'))
fc_model.add(Dense(10, activation='sigmoid'))
fc_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 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, 10)                5130

=================================================================
Total params: 98,378
Trainable params: 98,378
Non-trainable params: 0
_________________________________________________________________
```

```
Conv_model = keras.Sequential()
Conv_model.add(keras.layers.Input(shape=x_train[0].shape))
# write your code here
# add few Conv layers and Flatten layer
Conv_model.add(Conv2D(32, 3, activation='relu'))
Conv_model.add(MaxPool2D())
Conv_model.add(Conv2D(64, 3, activation='relu'))
Conv_model.add(MaxPool2D())
Conv_model.add(Conv2D(128, 3, activation='relu'))
Conv_model.add(MaxPool2D())
# you can use pool layers after Conv layers
Conv_model.add(Flatten())
Conv_model.add(Dense(10, activation='softmax'))
Conv_model.summary()
```

سپس این دو مدل را با دیتاست داده شده آموزش می دهیم و تست می کنیم.

```
Fully Connected Model
Epoch 1/5
1563/1563 [==============================] - 8s 3ms/step - loss: 2.1602 - accuracy: 0.1621
Epoch 2/5
1563/1563 [==============================] - 7s 4ms/step - loss: 2.0803 - accuracy: 0.1827
Epoch 3/5
1563/1563 [==============================] - 5s 3ms/step - loss: 2.0670 - accuracy: 0.1883
Epoch 4/5
1563/1563 [==============================] - 5s 3ms/step - loss: 2.0523 - accuracy: 0.2000
Epoch 5/5
1563/1563 [==============================] - 5s 3ms/step - loss: 2.0323 - accuracy: 0.2188


Loss and Accuracy on Test set :
313/313 [==============================] - 1s 3ms/step - loss: 2.0170 - accuracy: 0.2368


Convolutional Model
Epoch 1/5
1563/1563 [==============================] - 11s 4ms/step - loss: 1.4952 - accuracy: 0.4565
Epoch 2/5
1563/1563 [==============================] - 6s 4ms/step - loss: 1.1699 - accuracy: 0.5900
Epoch 3/5
1563/1563 [==============================] - 6s 4ms/step - loss: 1.0381 - accuracy: 0.6379
Epoch 4/5
1563/1563 [==============================] - 6s 4ms/step - loss: 0.9538 - accuracy: 0.6679
Epoch 5/5
1563/1563 [==============================] - 6s 4ms/step - loss: 0.8921 - accuracy: 0.6909


Loss and Accuracy on Test set :
313/313 [==============================] - 1s 3ms/step - loss: 0.9860 - accuracy: 0.6624
```

الف) خطای کمتر همیشه به معنای دقت بیشتر نیست ولی در اکثر مواقع اینگونه است.

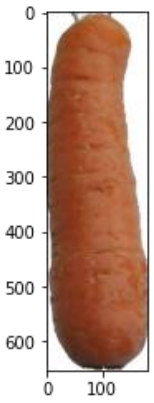
ب) در مدل اول 3ms و در مدل دوم 4ms

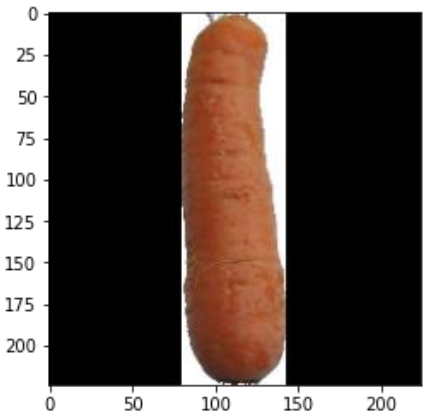
ج) بله، هر چقدر تعداد پارامتر کمتر باشد، مدت زمان مورد نیاز برای آموزش نیز کمتر است. زیرا تعداد پارامتر کمتری برای یادگرفتن وجود دارد.

سوال 3)

در ابتدا تابع resize_image را تکمیل می کنیم تا تمام عکس ها در سایز (224,224) باشند.

```python
def resize_img(img, desired_size = 224):
    # write your code here
    new_img = my_img = tf.image.resize_with_pad(img,
                                                224,
                                                224,
                                                method=tf.image.ResizeMethod.BILINEAR,
                                                antialias=False
                                               )
    new_img /= 255
    new_img = np.array(new_img,np.float64)
    return new_img
```





در مرحله بعد طبق داک سوالات مدل resnet50 را با شرایط خواسته شده load می کنیم.

```python
pre_trained_random_resnet = tf.keras.applications.resnet50.ResNet50(include_top=False,
                                                                     weights=None,
                                                                     input_tensor=None,
                                                                     input_shape=(224,224,3),
                                                                     pooling=None,
                                                                     classes=24,
                                                                     )
```

```python
resnet = tf.keras.models.Sequential()
# Write your code here
resnet.add(pre_trained_random_resnet)
resnet.add(Conv2D(1024,1))
resnet.add(Flatten())
resnet.add(Dense(1024, activation='relu'))
resnet.add(Dense(512, activation='relu'))
resnet.add(Dense(256, activation='relu'))
resnet.add(Dense(128, activation='relu'))
resnet.add(Dense(64, activation='relu'))
resnet.add(Dense(24, activation='softmax'))
resnet.summary()
```

که در نهایت نتیجه آموزش این مدل به شکل زیر شد:

```
resnet.fit(train_generator, epochs=1)
```

```
65/65 [==============================] - 118s 2s/step - loss: 5.3496 - acc: 0.0799
<keras.callbacks.History at 0x7f58e51668e0>
```

در مرحله بعد دباره از مدل resnet50 استفاده می کنیم اما با تنظیمات متفاوت:

```
pre_trained_imagenet_resnet = tf.keras.applications.resnet50.ResNet50(include_top=False,
                                                                       weights='imagenet',
                                                                       input_tensor=None,
                                                                       input_shape=(224,224,3),
                                                                       pooling=None,
                                                                       classes=24,
                                                                       )


for layer in pre_trained_imagenet_resnet.layers:
  layer.trainable = False
```

```
fine_tune_resnet = tf.keras.models.Sequential()
# write your code here

fine_tune_resnet.add(pre_trained_imagenet_resnet)
fine_tune_resnet.add(Conv2D(1024,1))
fine_tune_resnet.add(Flatten())
fine_tune_resnet.add(Dense(1024, activation='relu'))
fine_tune_resnet.add(Dense(512, activation='relu'))
fine_tune_resnet.add(Dense(256, activation='relu'))
fine_tune_resnet.add(Dense(128, activation='relu'))
fine_tune_resnet.add(Dense(64, activation='relu'))
fine_tune_resnet.add(Dense(24, activation='softmax'))

fine_tune_resnet.summary()

```

و نتیجه زیر به دست آمد:

```
fine_tune_resnet.fit(train_generator, epochs=1)
```

```
65/65 [==============================] - 73s 1s/step - loss: 5.8428 - acc: 0.0473
<keras.callbacks.History at 0x7f568f3de9a0>
```

و در آخر این دو مدل را با داده تست امتحان می کنیم

ت)

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

```
33/33 [==============================] - 34s 1s/step - loss: 3.1529 - acc: 0.0752
[3.152940273284912, 0.07524115592241287]
```

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

```
33/33 [==============================] - 35s 1s/step - loss: 3.1753 - acc: 0.0521
[3.175304651260376, 0.05209003388881683]
```

با توجه به نتایج مدلی که تمام وزن هایش را آموزش دادیم نتیجه بهتری در ابتدا دارد، زیرا مدل دوم دارای وزن های یک تسک دیگر را دارد و باید مدل را کمی بیشتر آموزش دهیم تا بتواند از آن اطلاعات استفاده بکند. از آنجایی که فقط 1 اپیاک train کردیم، نتایج آن درکل بد است.

لینک های استفاده شده:

https://stackoverflow.com/questions/49643907/clipping-input-data-to-the-valid-range-for-imshow-with-rgb-data-0-1-for-floa

https://www.tensorflow.org/api_docs/python/tf/keras/applications/resnet50/ResNet50

https://stackoverflow.com/questions/46610732/how-to-freeze-some-layers-when-fine-tune-resnet50

https://www.tensorflow.org/api_docs/python/tf/image/resize_with_pad

https://www.tensorflow.org/api_docs/python/tf/image/resize