Assignment 2 Report

Amirhossein Sohrabbeig - STD: 1744420

2022-03-10

1 Problem 1

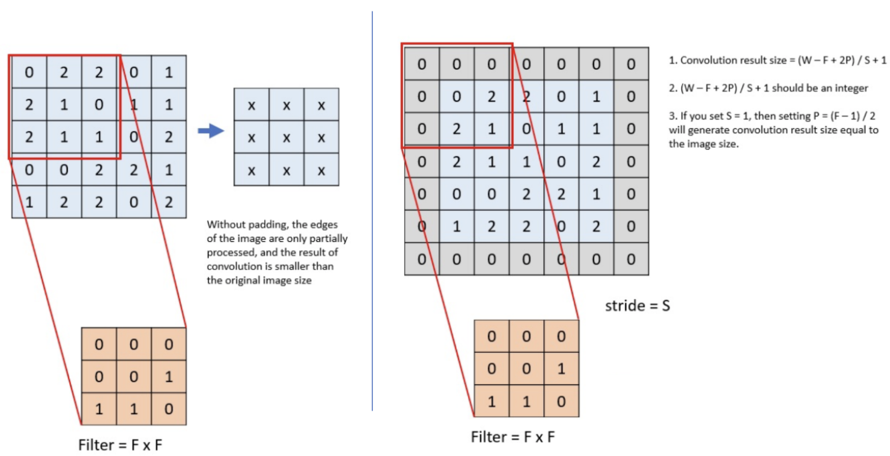

Figure 1: Convolution of images

1.1 padding = 0 and stride = 1

3	3	2
1	2	6
5	6	3

1.2 padding = 1 and stride = 2

4	1	2
1	2	3
2	0	0

2 Problem 2

In this task, we should modify the code so that the whole model (base mode + classifier) is fine-tuned. The provided code includes training the last linear layer on top of the pre-trained model. But we can do even better by **fine-tuning**, which consists of unfreezing the entire model we obtained from the last step and re-training it on the new data with a very low learning rate. Doing this can potentially achieve meaningful improvements by incrementally adapting the pre-trained features to the new data.

First, we should change the trainable attribute of any inner layer into True And then recompile the model so that your changes are taken into account:

```
base_model.trainable = True
model.summary()

model.compile(
    optimizer=keras.optimizers.Adam(1e-5),  # Low learning rate
    loss=keras.losses.BinaryCrossentropy(from_logits=True),
    metrics=[keras.metrics.BinaryAccuracy()],
)

epochs = 10
model.fit(train_ds, epochs=epochs, validation_data=validation_ds)
```

Listing 1: Fine-tuning the whole model

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 150, 150, 3)]     0

 sequential (Sequential)     (None, 150, 150, 3)       0

 tf.__operators__.getitem (S  (None, 150, 150, 3)      0
 licingOpLambda)

 tf.nn.bias_add (TFOpLambda)  (None, 150, 150, 3)      0

 resnet50 (Functional)       (None, 5, 5, 2048)        23587712

 global_average_pooling2d (G  (None, 2048)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 1)                 2049

=================================================================
Total params: 23,589,761
Trainable params: 23,536,641
Non-trainable params: 53,120
_________________________________________________________________
Epoch 1/10
```

```
291/291 [==============================] - 140s 450ms/step - loss: 0.0711 - binary_accuracy: 0.9711 - val_loss: 0.0524 - val_binary_accuracy: 0.9819
Epoch 2/10
291/291 [==============================] - 129s 443ms/step - loss: 0.0461 - binary_accuracy: 0.9827 - val_loss: 0.0469 - val_binary_accuracy: 0.9824
Epoch 3/10
291/291 [==============================] - 129s 443ms/step - loss: 0.0363 - binary_accuracy: 0.9855 - val_loss: 0.0586 - val_binary_accuracy: 0.9802
Epoch 4/10
291/291 [==============================] - 129s 443ms/step - loss: 0.0271 - binary_accuracy: 0.9893 - val_loss: 0.0418 - val_binary_accuracy: 0.9824
Epoch 5/10
291/291 [==============================] - 129s 442ms/step - loss: 0.0207 - binary_accuracy: 0.9923 - val_loss: 0.0442 - val_binary_accuracy: 0.9850
Epoch 6/10
291/291 [==============================] - 129s 442ms/step - loss: 0.0220 - binary_accuracy: 0.9922 - val_loss: 0.0498 - val_binary_accuracy: 0.9832
Epoch 7/10
291/291 [==============================] - 129s 443ms/step - loss: 0.0211 - binary_accuracy: 0.9924 - val_loss: 0.0515 - val_binary_accuracy: 0.9832
Epoch 8/10
291/291 [==============================] - 129s 443ms/step - loss: 0.0141 - binary_accuracy: 0.9954 - val_loss: 0.0463 - val_binary_accuracy: 0.9845
Epoch 9/10
291/291 [==============================] - 129s 442ms/step - loss: 0.0168 - binary_accuracy: 0.9944 - val_loss: 0.0605 - val_binary_accuracy: 0.9828
Epoch 10/10
291/291 [==============================] - 129s 442ms/step - loss: 0.0116 - binary_accuracy: 0.9960 - val_loss: 0.0811 - val_binary_accuracy: 0.9794
<keras.callbacks.History at 0x7f7a4012a050>
```

Listing 2: Outputs of running the fine-tuning code

We can evaluate the the performance of the our model after fine-tuning using the following block of code:

```
# Evaluate the model on the test data using ‘evaluate‘
print("Evaluate on test data")
results = model.evaluate(test_ds, batch_size=128)
print("test loss, test acc:", results)
```

Listing 3: Evaluation of the model

```
Evaluate on test data
73/73 [==============================] - 9s 119ms/step - loss: 0.0863 - binary_accuracy: 0.9794
test loss, test acc: [0.08627454191446304, 0.979363739490509]
```

After ten epochs, our training, validation, and testing accuracies are 0.9960, 0.9794, and 0.9793, respectively, which is a bit of improvement. However, I believe that the model was starting the overfit the data as the training accuracy was constantly increasing, but the validation accuracy was decreasing. This matches our prior expectations since this large model can easily overfit our small dataset after a few epochs.