Project 3: Classification Using Neural Networks and Deep Learning Report

CSE 575: Statistical Machine Learning (Summer 2024) (Samira Ghayekhloo)

Group 2

Project Team: Arjun Dadhwal, Ching-Chun Yuan, Jeffrey Li

Table of Contents

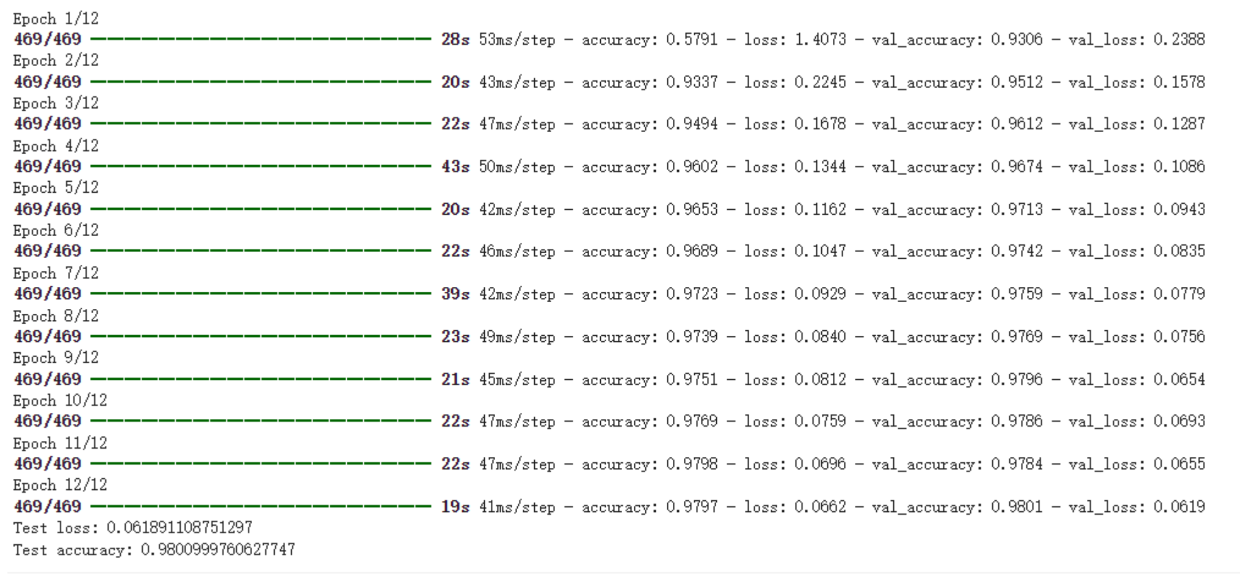
Table of Contents **2**
Introduction 3
Task 1 (Baseline code) **3**
Task 2 (Kernel size 5 * 5) **3**
Task 3 (Change number of feature maps) **4**
Conclusion **5**

Introduction

In this project, we were provided with baseline code, and our tasks involve experimenting with a convolutional neural network. Specifically, we need to modify the parameters of kernels and feature maps, create plots to represent learning errors over epochs, and ultimately present various sets of error and accuracy metrics on the test set.

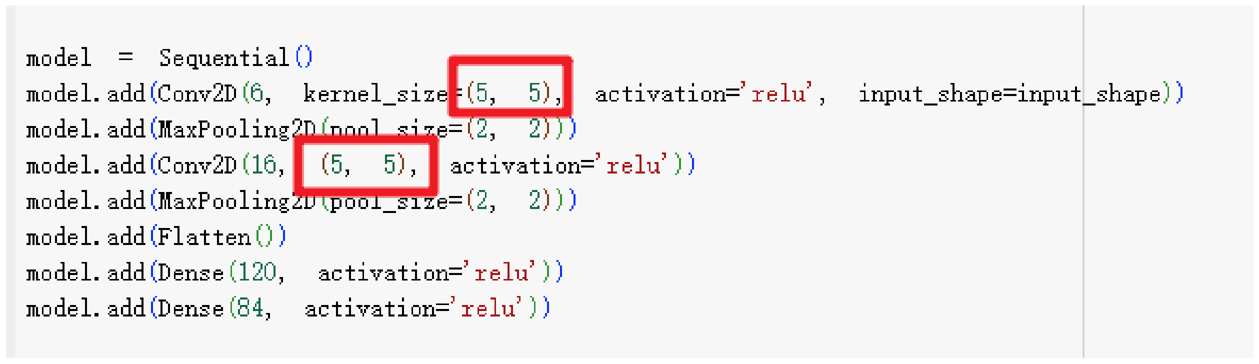
Task 1 (Baseline code)

For running the baseline code, we got a test loss of approximately 0.0619 and an accuracy of approximately 0.98.

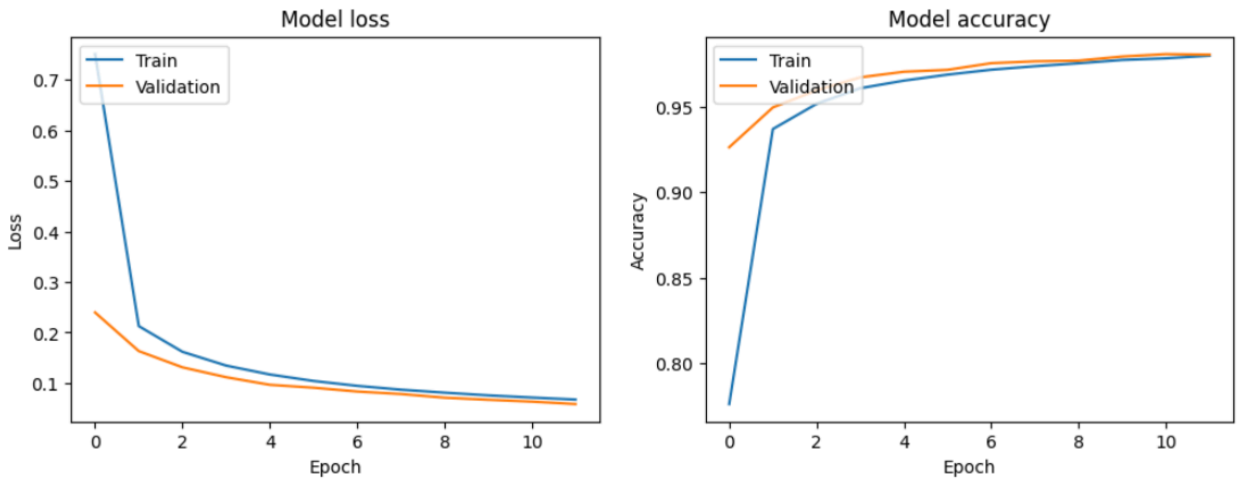


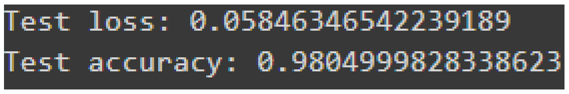
```
Epoch 1/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 28s 53ms/step - accuracy: 0.5791 - loss: 1.4073 - val_accuracy: 0.9306 - val_loss: 0.2388
Epoch 2/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 43ms/step - accuracy: 0.9337 - loss: 0.2245 - val_accuracy: 0.9512 - val_loss: 0.1578
Epoch 3/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22s 47ms/step - accuracy: 0.9494 - loss: 0.1678 - val_accuracy: 0.9612 - val_loss: 0.1287
Epoch 4/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43s 50ms/step - accuracy: 0.9602 - loss: 0.1344 - val_accuracy: 0.9674 - val_loss: 0.1086
Epoch 5/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 42ms/step - accuracy: 0.9653 - loss: 0.1162 - val_accuracy: 0.9713 - val_loss: 0.0943
Epoch 6/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22s 46ms/step - accuracy: 0.9689 - loss: 0.1047 - val_accuracy: 0.9742 - val_loss: 0.0835
Epoch 7/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 39s 42ms/step - accuracy: 0.9723 - loss: 0.0929 - val_accuracy: 0.9759 - val_loss: 0.0779
Epoch 8/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23s 49ms/step - accuracy: 0.9739 - loss: 0.0840 - val_accuracy: 0.9769 - val_loss: 0.0756
Epoch 9/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21s 45ms/step - accuracy: 0.9751 - loss: 0.0812 - val_accuracy: 0.9796 - val_loss: 0.0654
Epoch 10/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22s 47ms/step - accuracy: 0.9769 - loss: 0.0759 - val_accuracy: 0.9786 - val_loss: 0.0693
Epoch 11/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22s 47ms/step - accuracy: 0.9798 - loss: 0.0696 - val_accuracy: 0.9784 - val_loss: 0.0655
Epoch 12/12
469/469 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19s 41ms/step - accuracy: 0.9797 - loss: 0.0662 - val_accuracy: 0.9801 - val_loss: 0.0619
Test loss: 0.061891108751297
Test accuracy: 0.9800999760627747
```

Task 2 (Kernel size 5 * 5)



```
model = Sequential()
model.add(Conv2D(6, kernel_size=(5, 5), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(16, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(120, activation='relu'))
model.add(Dense(84, activation='relu'))
```

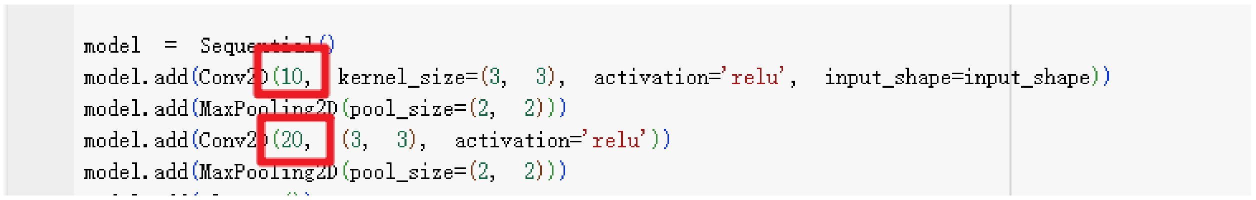




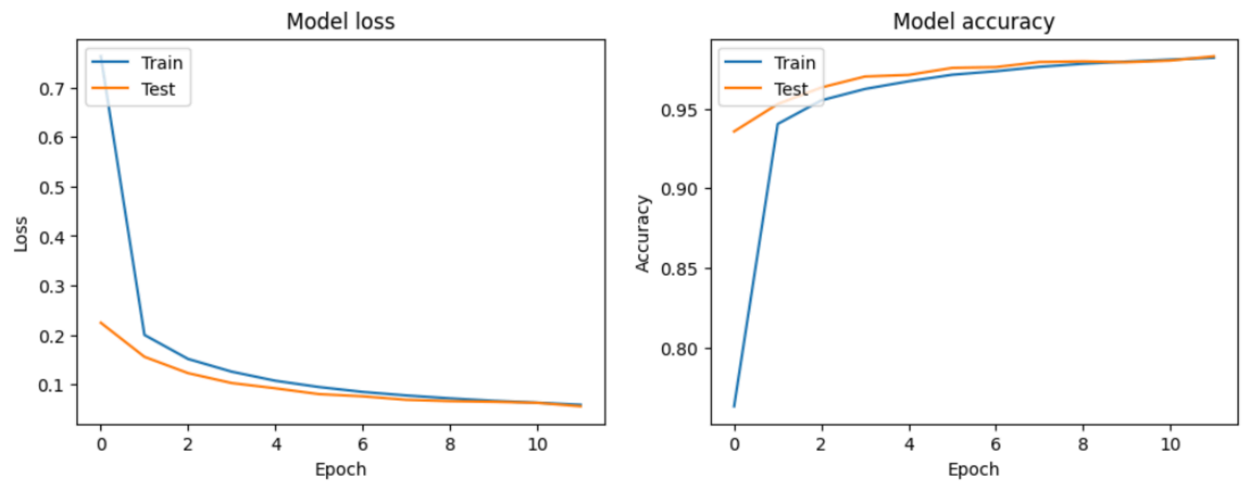
```
Test loss: 0.05846346542239189
Test accuracy: 0.9804999828338623
```

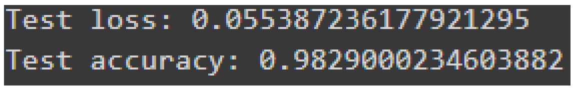
By changing the Kernel size from 3x3 to 5x5 we are increasing the amount of pixels that are included. Meaning that each operation will consider more pixels. From this specific dataset the differences are very small, likely caused by the image having little to no details and a very generic overarching shape. As such increasing the Kernel size would not make a huge difference in the loss and accuracy of the final tests.

Task 3 (Change number of feature maps)



```
model = Sequential()
model.add(Conv2D(10, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(20, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```





```
Test loss: 0.055387236177921295
Test accuracy: 0.9829000234603882
```

Changing the number of feature maps allows the model to capture more patterns from the data provided. From the results this improved the results with the smaller feature maps increasing from 6 to 10 and 16 to 20. The issue with trying to expand it further is potentially causing overfitting. The downside of using the increased feature maps is an increased processing time if there are too many feature maps. As a whole, having a small amount of feature maps is not good, however there needs to be a careful balance to optimize the model without hurting performance.

In our code specifically, we used the default 3x3 rather than the changed 5x5, there are 2 layers that can be changed, the first layer would allow more detailed features to be captured while the second layer would increase the pattern detection after the first layer.

Conclusion

We successfully adapted the default data of a 3x3 Kernel size and 10 feature maps into a 5x5 Kernel size and 20 feature maps. The changes for the kernels were not significantly different, most likely due to the images being used having few details and a simple large shape. The feature maps had an improvement, likely from being able to capture edges of the image for a more accurate result.