作業4:Use a CNN-based neural network to classify the scene is ​indoors or outdoors

班級: 碩專班

學生:5108056016張仲威

教授: 吳俊霖 教授

目錄

[一. 主要演算法 2](#_Toc59987423)

[二. 程式片斷 2](#_Toc59987424)

[三. 測試資料 6](#_Toc59987425)

[四. 結果呈現 6](#_Toc59987426)

[五. 討論 7](#_Toc59987427)

[六. 結論 8](#_Toc59987428)

# 主要演算法

主要概念為訓練一個CNN二分類模型,我用**keras** 設計一個**CNN** Model,

Keras會在**ImageDataGenerator**迭代過程中，自動的為data/train，data/test，data/validation內部生產訓練標籤，標籤依據就是在data/train，data/test，data/validation下面的分類目錄，本作業是/outdoor和/intdoor目錄文件夾作為兩分類

# 程式片斷

1.訓練:

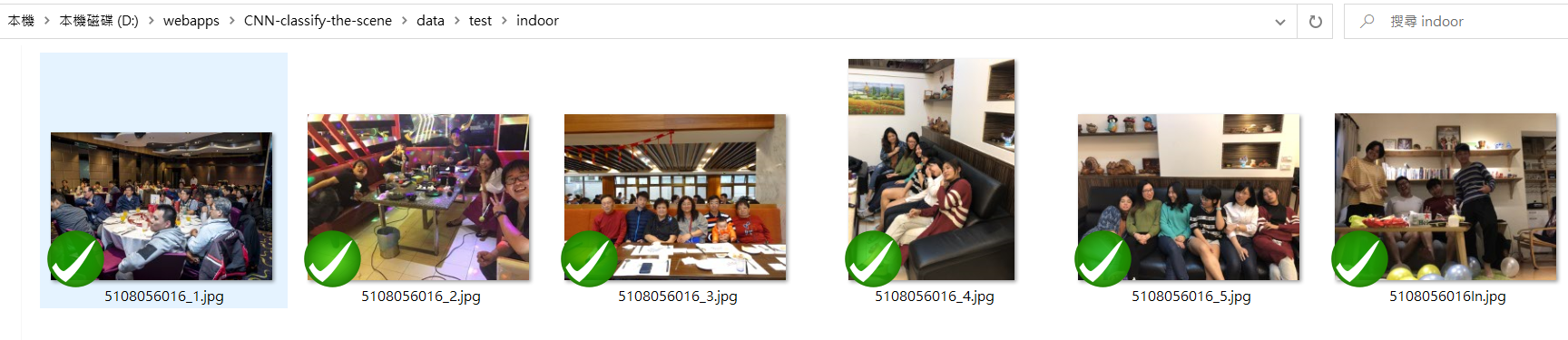
1. **def** model():
2. model = models.Sequential() # 初始序貫模型(有序)
4. KERNEL\_SIZE = (3, 3)
6. model.add(layers.Conv2D(filters=32, kernel\_size=KERNEL\_SIZE, activation='relu', input\_shape=(150, 150, 3)))
7. model.add(layers.MaxPooling2D((2, 2)))
9. model.add(layers.Conv2D(filters=64, kernel\_size=KERNEL\_SIZE, activation='relu'))
10. model.add(layers.MaxPooling2D((2, 2)))
12. model.add(layers.Conv2D(filters=128, kernel\_size=KERNEL\_SIZE, activation='relu'))
13. model.add(layers.MaxPooling2D((2, 2)))
15. model.add(layers.Conv2D(filters=128, kernel\_size=KERNEL\_SIZE, activation='relu'))
16. model.add(layers.MaxPooling2D((2, 2)))
18. model.add(layers.Flatten())
19. model.add(layers.Dense(512, activation='relu'))
21. model.add(Dropout(0.5))
23. model.add(layers.Dense(1, activation='sigmoid'))
25. model.compile(loss='binary\_crossentropy', # 定義損失函數(交叉熵用於二分類)
26. optimizer=optimizers.RMSprop(lr=1e-3), # 定義優化器(方均根傳播)與學習率
27. metrics=['accuracy'])
29. **return** mode
30. **def** loss(history):
31. history\_dict = history.history
32. loss\_values = history\_dict['loss']
33. val\_loss\_values = history\_dict['val\_loss']
34. epochs = range(1, len(loss\_values) + 1)
35. plt.plot(epochs, loss\_values, 'b', label='Training loss')
36. plt.plot(epochs, val\_loss\_values, 'r', label='Validation loss')
37. plt.title('Training and validation loss')
38. plt.xlabel('Epochs')
39. plt.ylabel('Loss')
40. plt.legend()
41. plt.grid()
42. plt.show()
43. **def** accuracy(history):
44. history\_dict = history.history
45. acc = history\_dict['accuracy']
46. val\_acc = history\_dict['val\_accuracy']
47. epochs = range(1, len(acc) + 1)
48. plt.plot(epochs, acc, 'g', label='Training acc')
49. plt.plot(epochs, val\_acc, 'r', label='Validation acc')
50. plt.title('Training and validation accuracy')
51. plt.xlabel('Epochs')
52. plt.ylabel('Accuracy')
53. plt.legend()
54. plt.grid()
55. plt.show()
56. **def** train(model):
57. # 圖像預處理
58. train\_datagen = ImageDataGenerator(rescale=1. / 255)
59. validation\_datagen = ImageDataGenerator(rescale=1. / 255)
61. train\_generator = train\_datagen.flow\_from\_directory(
62. train\_dir,
63. target\_size=(150, 150),
64. batch\_size=256,
65. class\_mode='binary')
67. validation\_generator = validation\_datagen.flow\_from\_directory(
68. validation\_dir,
69. target\_size=(150, 150),
70. batch\_size=64,
71. class\_mode='binary')
73. history = model.fit\_generator(
74. train\_generator,
75. epochs=30, # 訓練30次
76. validation\_data=validation\_generator
77. )
79. model.save(model\_file\_name)
81. loss(history)
82. accuracy(history)

2.預測:

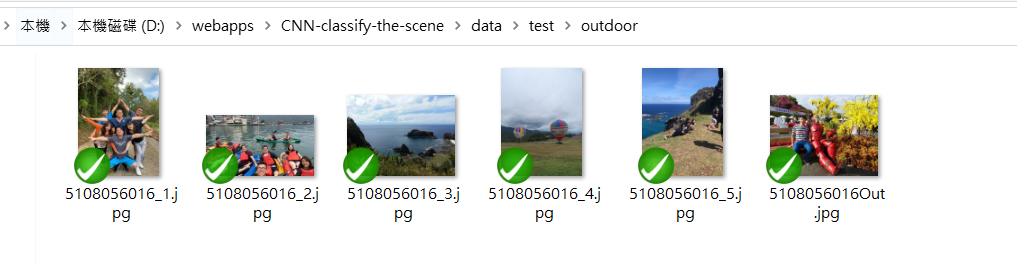
1. # 顯示出預測結果
2. **def** result(model, count):
3. # 圖像預處理
4. test\_datagen = ImageDataGenerator(rescale=1. / 255)
5. test\_generator = test\_datagen.flow\_from\_directory(
6. test\_dir,
7. target\_size=(150, 150),
8. batch\_size=256,
9. class\_mode='binary')
11. text\_labels = []
12. plt.figure(figsize=(30, 20))
13. # 取第一個結果
14. **for** batch, label **in** test\_generator:
15. pred = model.predict(batch)
16. **for** i **in** range(count):
17. true\_reuslt = label[i]
18. **print**('true\_reuslt',true\_reuslt)
19. **print**('pred:',pred[i])
20. **print**('==='\*40)
21. **if** pred[i] > 0.5:
22. text\_labels.append('outdoor')
23. **else**:
24. text\_labels.append('indoor')
26. # 4列，畫圖
27. plt.subplot(count / 4 + 1, 4, i + 1)
28. plt.title('predict: ' + text\_labels[i])
29. imgplot = plt.imshow(batch[i])
31. plt.show()
33. **break**

# 測試資料

室內:



室外:

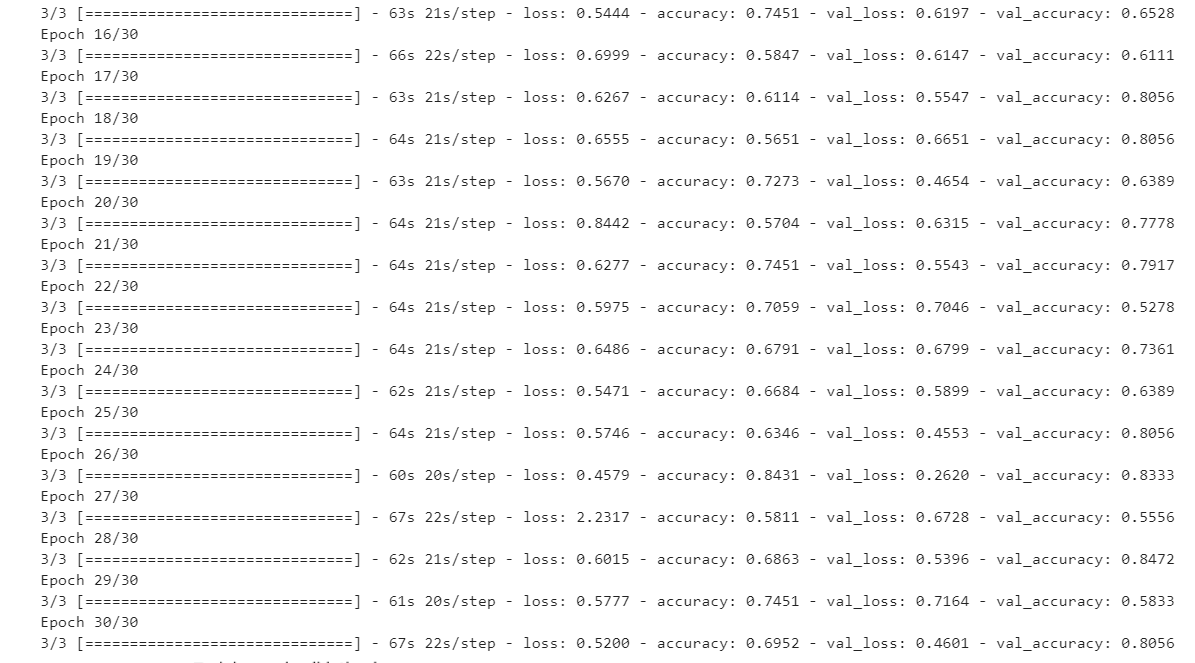


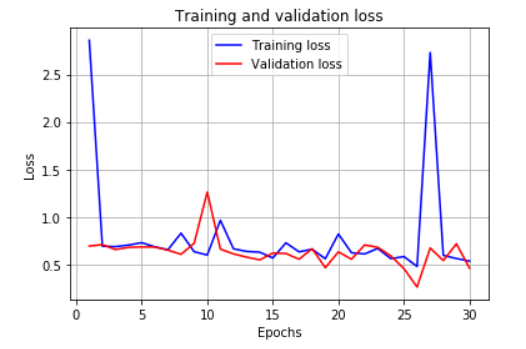
# 結果呈現



# 討論

下圖為訓練的過程, 我訓練30次,看的出來loss有慢慢降低的趨勢,





Loss的曲線圖有動盪的趨勢,感覺有點不穩,像在第26次loss突然升高,但最終還是慢慢降低



準確度的部分從一開始的0.5上升到0.8

# 結論

以最後預測的10張圖結果,有9張預測正確,有1張預測錯誤,預測的準確度還蠻高的