

**計算智慧 期末報告**

U0433050 范瑜萍

一、做的改變(未更改辨識圖片)

* 重點程式碼

# 如同範例使用150\*150\*3的輸入圖片

img\_input = layers.Input(shape=(150, 150, 3))

# CONV->RELU->POOL

# 第一層 convolution 使用32個3\*3的filters並搭配RELU激活

# 批標準化以統一數據規格

# 以3\*3的MaxPooling縮小圖片

# 使用dropout在相同資料下避免overfitting

x = layers.Conv2D(32, 3, activation='relu')(img\_input)

x = layers.BatchNormalization()(x)

x = layers.MaxPooling2D(3)(x)

x = layers.Dropout(0.25)(x)

# (CONV=>RELU)\*2 ->POOL 堆疊多個CONV與RELU獲得更多特徵

x = layers.Conv2D(64, 3, activation='relu')(x)

x = layers.BatchNormalization()(x)

x = layers.Conv2D(64, 3, activation='relu')(x)

x = layers.BatchNormalization()(x)

x = layers.MaxPooling2D(2)(x)

x = layers.Dropout(0.25)(x)

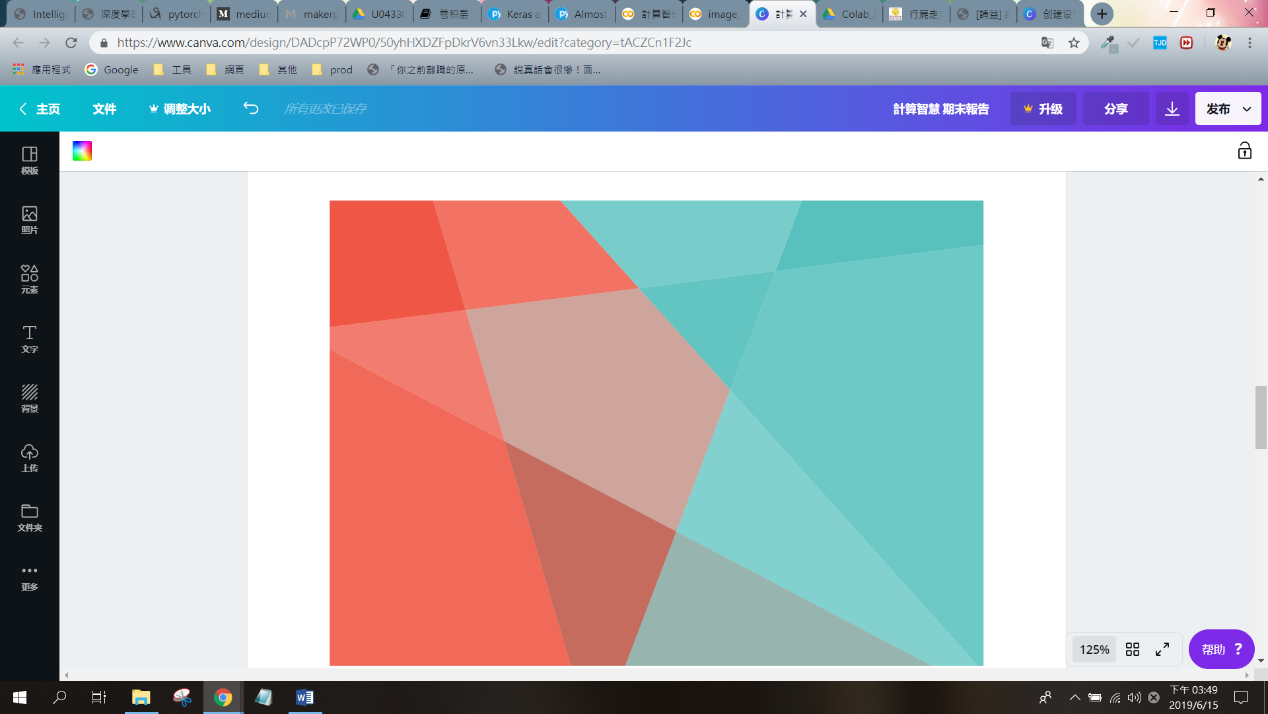
# (CONV=>RELU)\*2 ->POOL

x = layers.Conv2D(128, 3, activation='relu')(x)

x = layers.BatchNormalization()(x)

x = layers.Conv2D(128, 3, activation='relu')(x)

x = layers.BatchNormalization()(x)

x = layers.MaxPooling2D(2)(x)

x = layers.Dropout(0.25)(x)

x = layers.Flatten()(x)

x = layers.Dense(1024, activation='relu')(x)

x = layers.BatchNormalization()(x)

x = layers.Dropout(0.5)(x)

# Create output layer with a single node and sigmoid activation

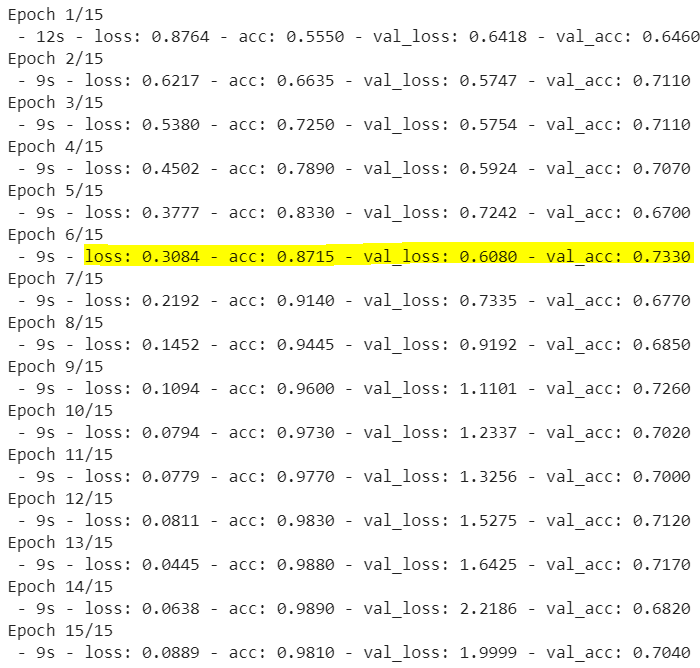
output = layers.Dense(1, activation='sigmoid')(x)

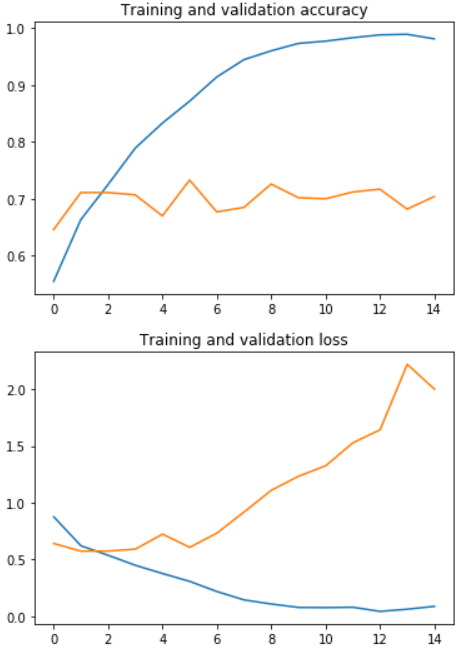
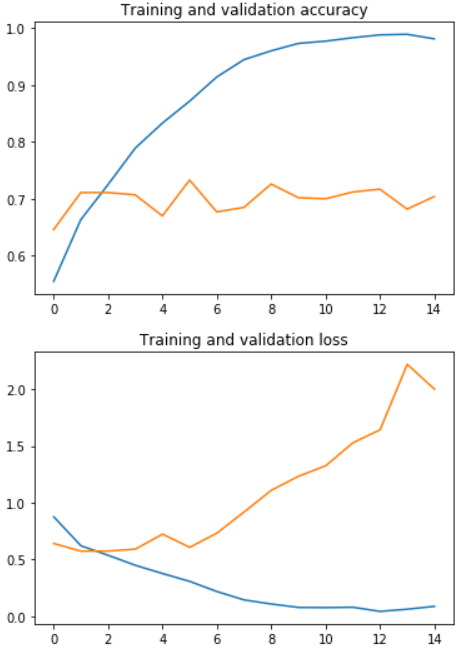
* 完整模型架構

|  |  |  |
| --- | --- | --- |
| Layer (type) | Output Shape | Param # |
| input\_1 (InputLayer) | (None, 150, 150, 3) | 0 |
| conv2d\_1 (Conv2D) | (None, 148, 148, 32) | 896 |
| batch\_normalization\_21 (Batch | (None, 148, 148, 32) | 128 |
| max\_pooling2d\_1 | (MaxPooling2 (None, 49, 49, 32) | 0 |
| dropout\_1 (Dropout) | (None, 49, 49, 32) | 0 |
| conv2d\_2 (Conv2D) | (None, 47, 47, 64) | 18496 |
| batch\_normalization\_1 (Batch | (None, 47, 47, 64) | 256 |
| conv2d\_3 (Conv2D) | (None, 45, 45, 64) | 36928 |
| batch\_normalization\_2 (Batch | (None, 45, 45, 64) | 256 |
| max\_pooling2d\_2 (MaxPooling2 | (None, 22, 22, 64) | 0 |
| dropout\_3 (Dropout) | (None, 22, 22, 64) | 0 |
| conv2d\_4 (Conv2D) | (None, 20, 20, 128) | 73856 |
| batch\_normalization\_3 (Batch | (None, 20, 20, 128) | 512 |
| conv2d\_5 (Conv2D) | (None, 18, 18, 128) | 147584 |
| batch\_normalization\_4 (Batch | (None, 18, 18, 128) | 512 |
| max\_pooling2d\_3 (MaxPooling2 | (None, 9, 9, 128) | 0 |
| dropout\_4 (Dropout) | (None, 9, 9, 128) | 0 |
| flatten\_1 (Flatten) | (None, 10368) | 0 |
| dense\_1 (Dense) | (None, 1024) | 10617856 |
| batch\_normalization\_5 (Batch | (None, 1024) | 4096 |
| dropout\_5 (Dropout) | (None, 1024) | 0 |
| dense\_2 (Dense) | (None, 1) | 1025 |
| Total params: 10,902,401  Trainable params: 10,899,521  Non-trainable params: 2,880 | | |

二、正確率與損失率比較(未更改辨識圖片)

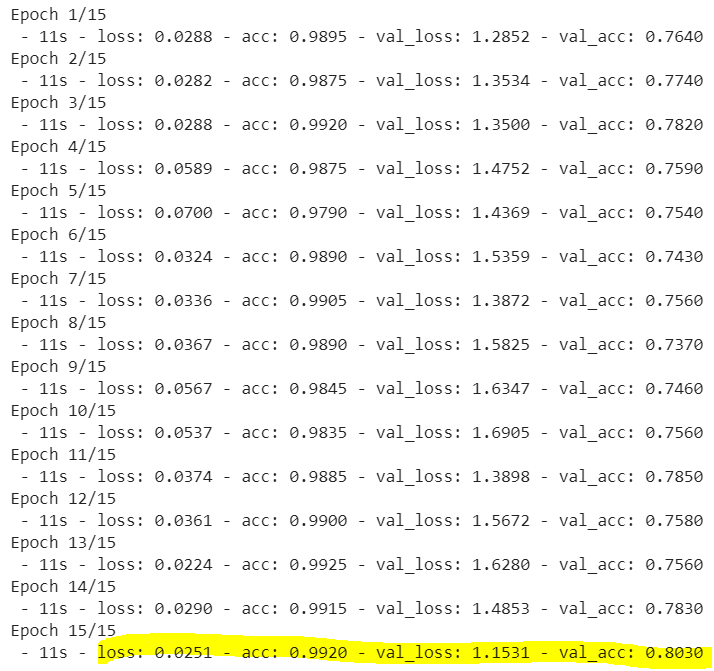
* **舊**訓練方式與參數



藍線為訓練準確率或損失率，橘線為驗證準確率或損失率。

* **新**訓練方式與參數



藍線為訓練準確率或損失率，橘線為驗證準確率或損失率。

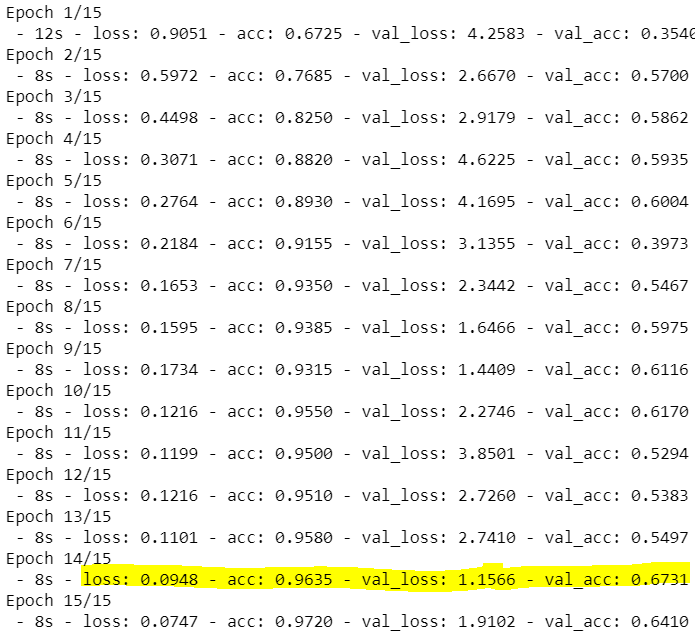
* **結果**：新訓練方式之驗證準確率比舊訓練方式好。

三、做的改變(更改辨識圖片)

將貓狗辨識更改為「湯匙、叉子」之辨識，並沿用先前之新訓練方式與參數，叉子之訓練圖共有994張，驗證圖共有151張；湯匙之訓練圖共有1966張，驗證圖共有242張。



四、正確率與損失率比較(更改辨識圖片)



藍線為訓練準確率或損失率，橘線為驗證準確率或損失率。

* **結果**：在15次迭代中，最好的表現為驗證損失率1.1566，及驗證準確率0.6731，而訓練準確率已達0.9635，代表如果需要再增加準確度，訓練圖片集需要更多樣的圖片。

五、程式載點

<https://github.com/a850228/-/tree/master/U0433050%20%E8%A8%88%E7%AE%97%E6%99%BA%E6%85%A7%20%E6%9C%9F%E6%9C%AB%E5%A0%B1%E5%91%8A>

縮網址：<https://reurl.cc/Rlnbx>

六、附註資料

【Maker玩AI】使用Google Colaboratory免費資源學AI，正是時候！

<https://makerpro.cc/2018/06/learn-ai-by-google-colaboratory/>

Keras and Convolutional Neural Networks (CNNs)

<https://www.pyimagesearch.com/2018/04/16/keras-and-convolutional-neural-networks-cnns/>

The Knifey-Spoony image data-set

<https://github.com/Hvass-Labs/knifey-spoony>