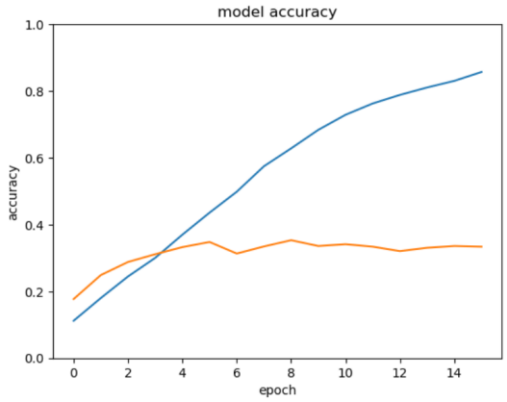
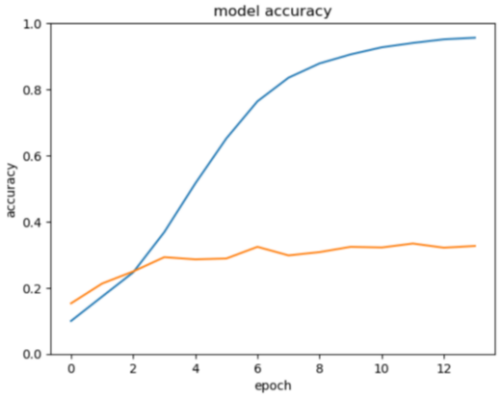
目錄

調整圖片大小

128x128



224x224



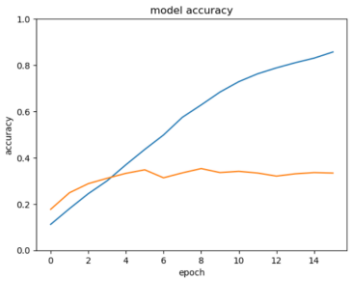
樣本權重

在訓練過程中，如果設定 Class Weight 參數，會計算出每個樣本的權重，然後在訓練過程中將這些權重納入損失函數中進行更新，以保證對稀有類別的樣本更加關注，從而提高模型的預測性能。

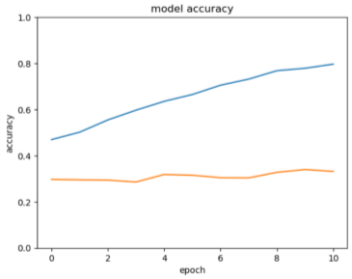
公式一(大於等於 1): 最大樣本數 / 該標籤樣本數

公式二(介於 0~1): 總和樣本數 / (標籤數 x 該標籤樣本數)

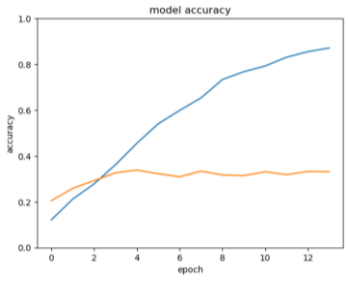
不設權重



公式一



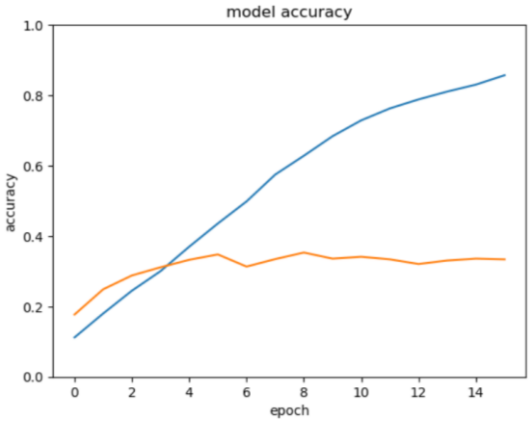
公式二

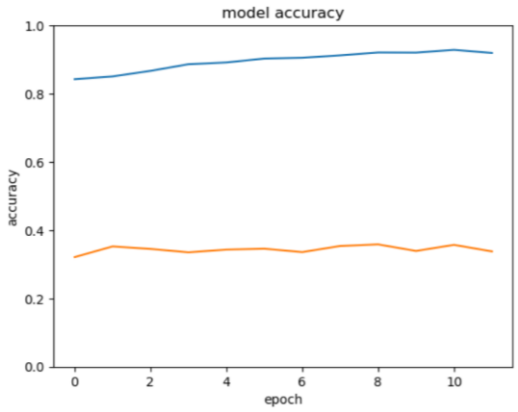


調整批次大小

在每次訓練（一個 batch）後，權重都會被更新一次，通常情況下，較大的 batch size 可以加快模型訓練的速度，但是可能會影響模型的收斂效果；而較小的 batch size 可以提高模型的收斂效果，但是訓練速度較慢。

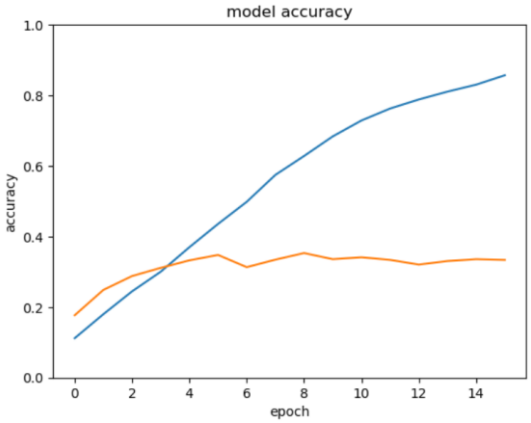
64 32



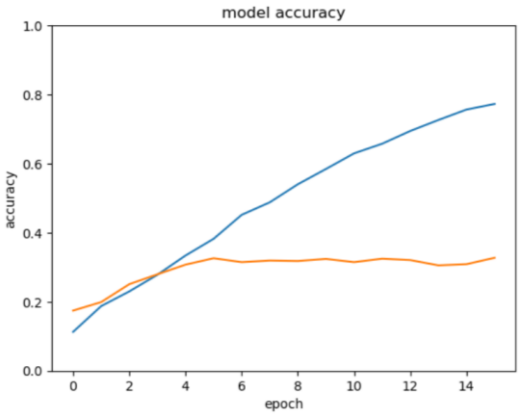


調整打亂緩衝區大小

7000

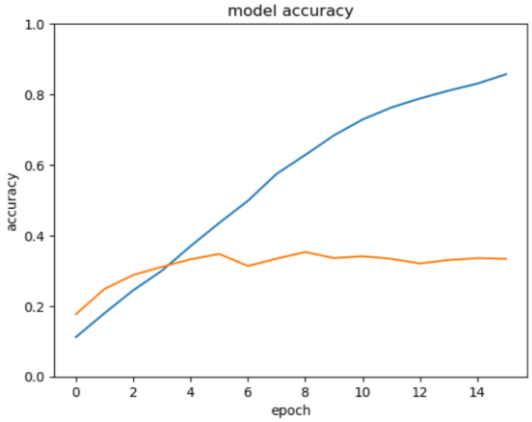


70000

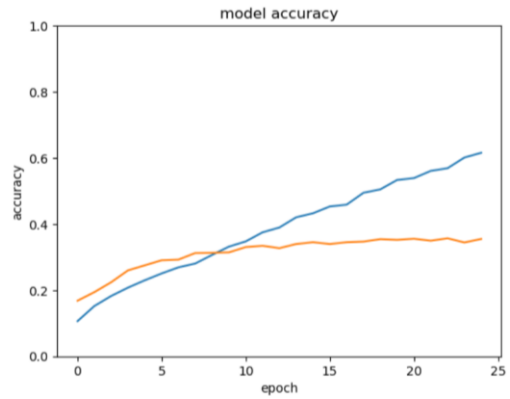


設定學習率

預設

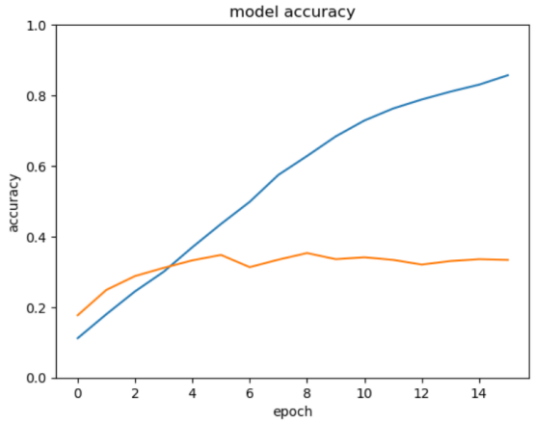


0.0001

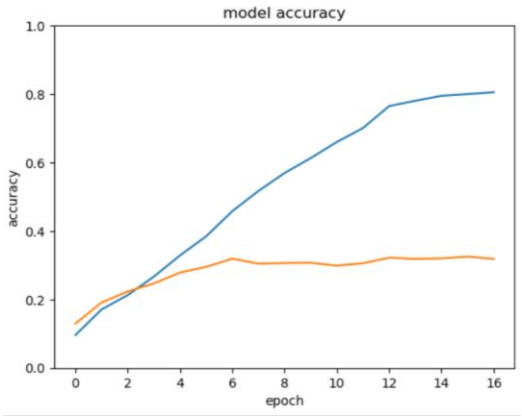


設定學習率衰退

預設



ReduceLROnPlateau



調整模型

使用多層小卷積核堆疊可以達到和使用大卷積核相同的感受野，但是其參數和計算量更少，更經濟。此外，多層小卷積核堆疊相較於大卷積核可以引入更多的非線性，因此能夠獲得更好的效果。

Layer (type)	Output Shape	Param #
conv2d_61 (Conv2D)	(None, 128, 128, 32)	896
max_pooling2d_46 (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_62 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_47 (MaxPooling2D)	(None, 32, 32, 64)	0
flatten_23 (Flatten)	(None, 65536)	0
dense_47 (Dense)	(None, 256)	16777472
dropout_23 (Dropout)	(None, 256)	0
dense_48 (Dense)	(None, 50)	12850

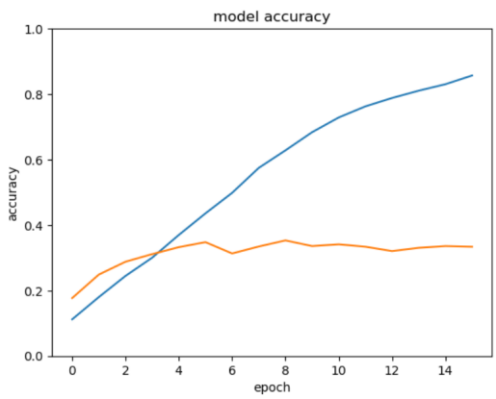
Total params: 16,809,714
Trainable params: 16,809,714
Non-trainable params: 0

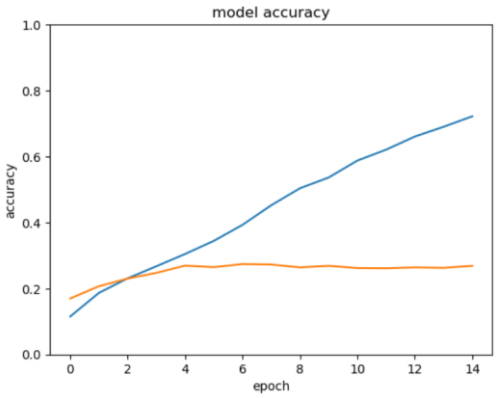
原始模型

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 126, 126, 8)	224
conv2d_14 (Conv2D)	(None, 124, 124, 16)	1168
conv2d_15 (Conv2D)	(None, 122, 122, 16)	2320
max_pooling2d_6 (MaxPooling2D)	(None, 61, 61, 16)	0
conv2d_16 (Conv2D)	(None, 59, 59, 32)	4640
conv2d_17 (Conv2D)	(None, 57, 57, 16)	4624
conv2d_18 (Conv2D)	(None, 55, 55, 16)	2320
max_pooling2d_7 (MaxPooling2D)	(None, 27, 27, 16)	0
flatten_3 (Flatten)	(None, 11664)	0
dense_7 (Dense)	(None, 256)	2986240
dropout_3 (Dropout)	(None, 256)	0
dense_8 (Dense)	(None, 50)	12850

Total params: 3,014,386
Trainable params: 3,014,386
Non-trainable params: 0

多層(小卷積)架構





[【CNN 基础】为什么要用较小的卷积核 卷积核大小的影响 SinHao22 的博客-CSDN 博客]

[How to Improve the Accuracy of Your Image Recognition Models (freecodecamp.org)]

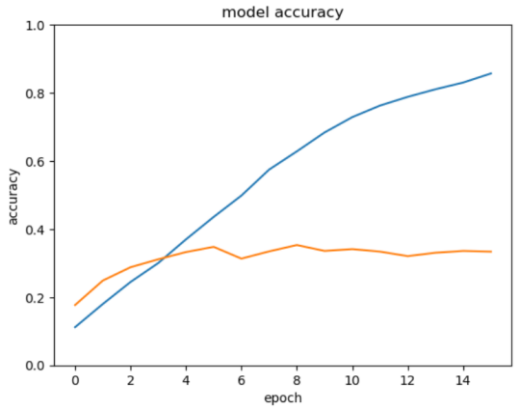
Batchnormalization

BatchNormalization 的作用是在每個批次（batch）上對前一層的激活值進行重新規範化，即使其輸出數據的均值接近於 0，其標準差接近於 1，通常用於加速深度學習模型的收斂，提高模型的精度和穩定性。

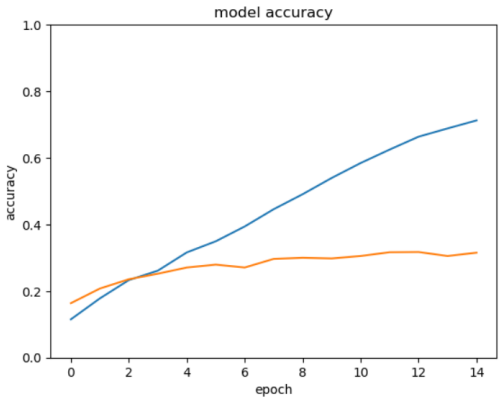
Layer (type)	Output Shape	Param #
conv2d_71 (Conv2D)	(None, 126, 126, 32)	896
batch_normalization (BatchNormalization)	(None, 126, 126, 32)	128
max_pooling2d_50 (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_72 (Conv2D)	(None, 61, 61, 64)	18496
batch_normalization_1 (BatchNormalization)	(None, 61, 61, 64)	256
max_pooling2d_51 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten_26 (Flatten)	(None, 57600)	0
dense_53 (Dense)	(None, 256)	14745856
batch_normalization_2 (BatchNormalization)	(None, 256)	1024
dropout_26 (Dropout)	(None, 256)	0
dense_54 (Dense)	(None, 50)	12850

Total params: 14,779,506
Trainable params: 14,778,802
Non-trainable params: 704

沒有 BatchNormalization



有 BatchNormalization



參考:

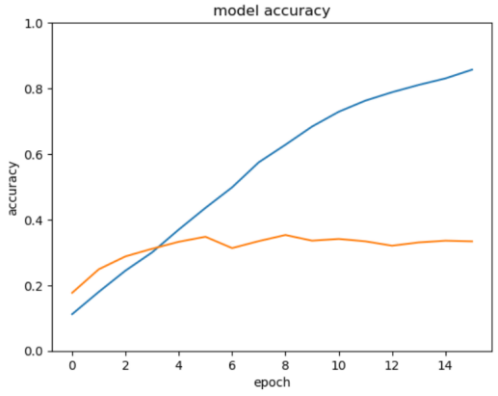
https://stackoverflow.com/questions/47143521/where-to-apply-batch-normalization-on-standard-cnns

Batch Normalization 介紹. 隨著神經網路越來越深，為了使模型更加穩定，Batch… | by 李謦伊 | 謦伊的閱讀筆記 | Medium

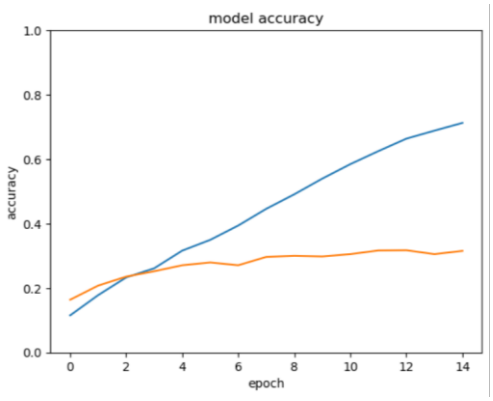
AveragePooling

AveragePooling 更適合提取圖像中的整體特徵信息，因為它計算的是輸入數據中所有元素的平均值，所以它更能反映整體特徵。

MaxPooling



AveragePooling

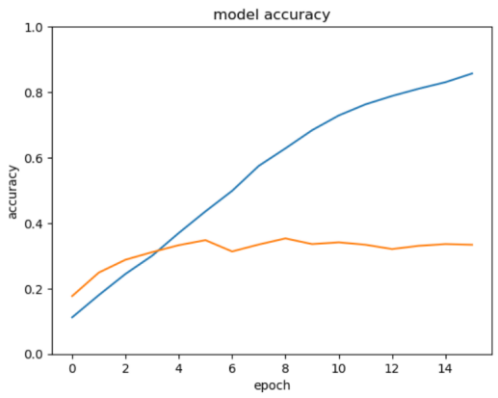


Padding

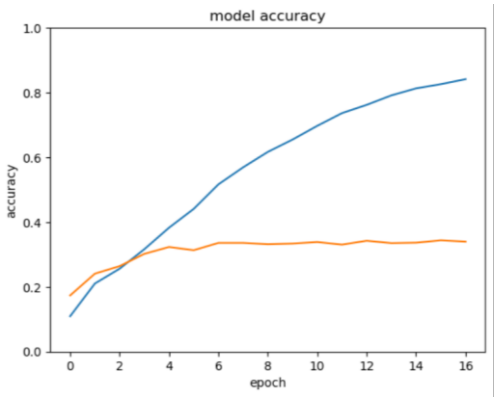
使用 same padding 的主要目的是為了保護原圖信息，保持輸出特徵圖的大小與輸入圖像大小相同，使得更深層的層的輸入依然保持足夠的信息量，並且避免在填充像素中引入噪聲，『Same Padding』在使用 filter size=3, Padding =1, Stride=1 時，會讓輸出 Feature map 與輸入圖像維持一樣的尺寸

Output = (Input+2p-f)/s+1

沒有 Padding



Same Padding



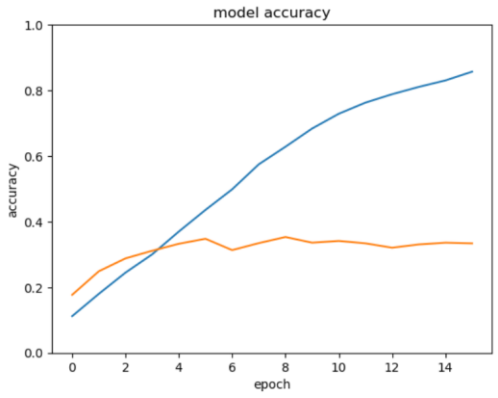
參考:

https://cinnamonaitaiwan.medium.com/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-cnn%E5%8E%9F%E7%90%86-keras%E5%AF%A6%E7%8F%BE-432fd9ea4935

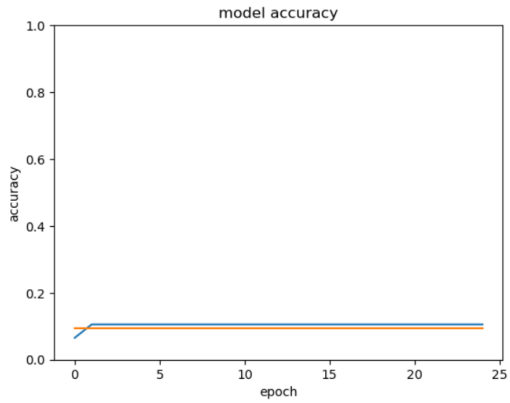
https://blog.csdn.net/AnneQiQi/article/details/103573917

更改激活函數

Relu



Sigmoid



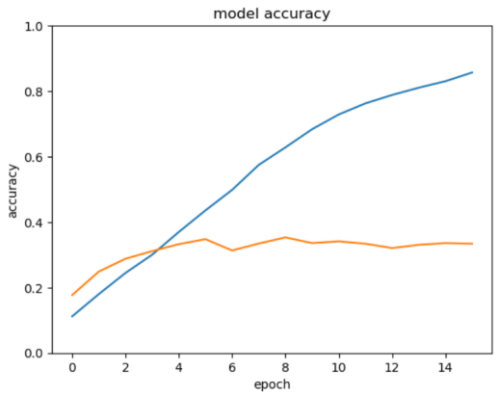
數據增強

```
data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal_and_vertical"),
    layers.RandomRotation(0.2),
])
```

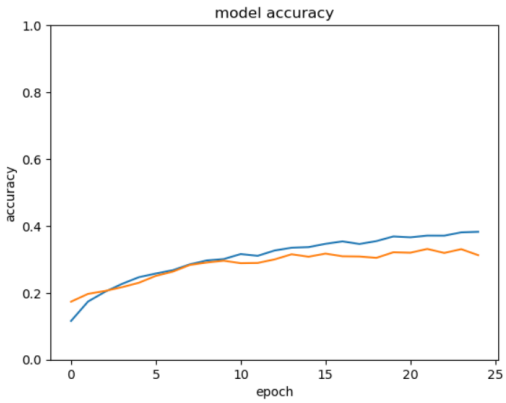
```
model = keras.Sequential(
    [
        layers.Input(shape=input_shape),
        data_augmentation,
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(256, activation="relu"),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

```
Layer (type)                 Output Shape              Param #
=================================================================
sequential_13 (Sequential)   (64, 128, 128, 3)         0

conv2d_33 (Conv2D)           (64, 126, 126, 32)        896

max_pooling2d_18 (MaxPoolin  (64, 63, 63, 32)          0
g2D)

conv2d_34 (Conv2D)           (64, 61, 61, 64)          18496

max_pooling2d_19 (MaxPoolin  (64, 30, 30, 64)          0
g2D)

flatten_9 (Flatten)          (64, 57600)               0

dense_19 (Dense)             (64, 256)                 14745856

dropout_9 (Dropout)          (64, 256)                 0

dense_20 (Dense)             (64, 50)                  12850

=================================================================
Total params: 14,778,098
Trainable params: 14,778,098
Non-trainable params: 0
```

原始資料



垂直、水平翻轉與隨機旋轉



其他架構

預設參數 ：

Image Size: 224 x 224

Batch Size : 32

Class Weight: Yes

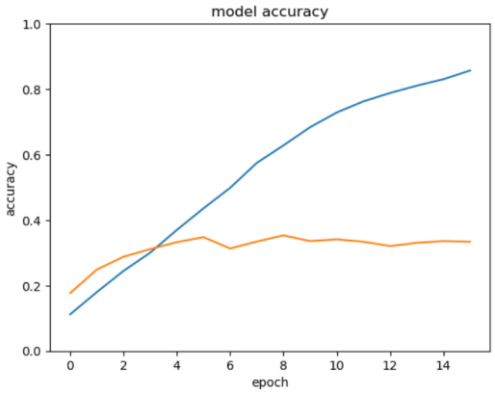
Learning Rate: 0.01

ReduceLROnPlateau: Yes

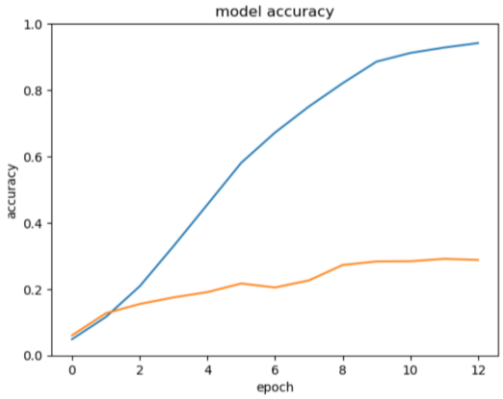
原始架構

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 224, 224, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_7 (Conv2D)	(None, 112, 112, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_8 (Conv2D)	(None, 56, 56, 96)	55392
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 96)	0
conv2d_9 (Conv2D)	(None, 28, 28, 96)	83040
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 96)	0
flatten_1 (Flatten)	(None, 18816)	0
dense_2 (Dense)	(None, 512)	9634304
dense_3 (Dense)	(None, 50)	25650

Total params: 9,819,314
Trainable params: 9,819,314
Non-trainable params: 0



花朵辨識架構



參考:

https://www.kaggle.com/code/rajmehra03/flower-recognition-cnn-keras

預設參數 ：

Image Size: 224 x 224

Batch Size : 32

Class Weight: Yes

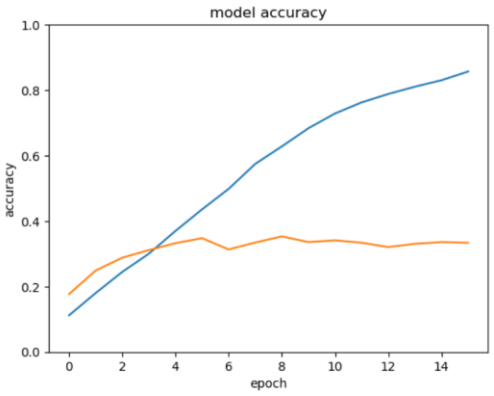
Learning Rate: 0.01

ReduceLROnPlateau: Yes

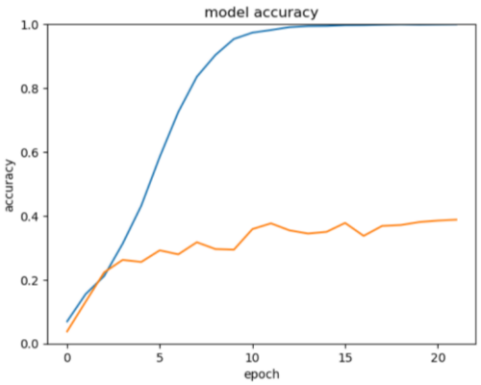
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
batch_normalization (BatchNormalization)	(None, 224, 224, 32)	128
conv2d_1 (Conv2D)	(None, 224, 224, 32)	9248
batch_normalization_1 (BatchNormalization)	(None, 224, 224, 32)	128
conv2d_2 (Conv2D)	(None, 112, 112, 32)	25632
batch_normalization_2 (BatchNormalization)	(None, 112, 112, 32)	128
dropout (Dropout)	(None, 112, 112, 32)	0
conv2d_3 (Conv2D)	(None, 112, 112, 64)	18496
batch_normalization_3 (BatchNormalization)	(None, 112, 112, 64)	256
conv2d_4 (Conv2D)	(None, 112, 112, 64)	36928
batch_normalization_4 (BatchNormalization)	(None, 112, 112, 64)	256
conv2d_5 (Conv2D)	(None, 56, 56, 64)	102464
batch_normalization_5 (BatchNormalization)	(None, 56, 56, 64)	256
dropout_1 (Dropout)	(None, 56, 56, 64)	0
flatten (Flatten)	(None, 200704)	0
dense (Dense)	(None, 128)	25690240
batch_normalization_6 (BatchNormalization)	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 50)	6450

Total params: 25,892,018
Trainable params: 25,891,186
Non-trainable params: 832

原始架構



高級數字辨識架構



參考:

https://www.kaggle.com/code/cdeotte/how-to-choose-cnn-architecture-mnist

預設參數 ：

Image Size: 224 x 224

Batch Size : 32

Class Weight: Yes

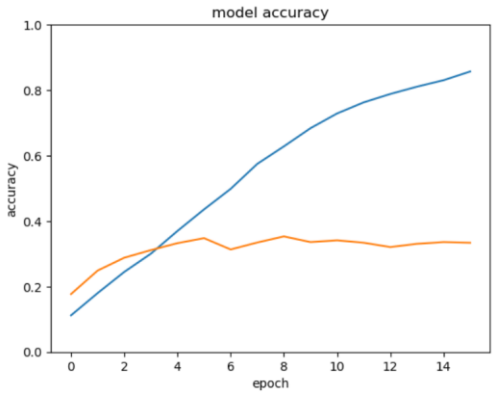
Learning Rate: 0.01

ReduceLROnPlateau: Yes

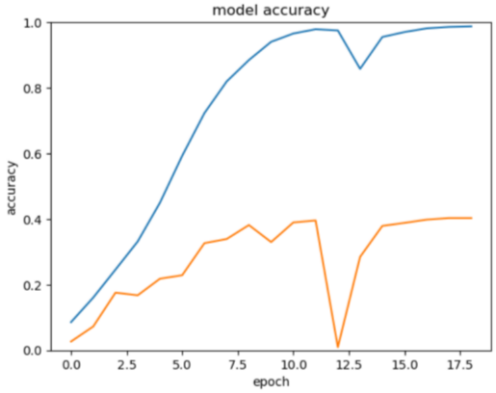
Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 224, 224, 32)	2432
batch_normalization_7 (BatchNormalization)	(None, 224, 224, 32)	128
max_pooling2d_4 (MaxPooling2D)	(None, 112, 112, 32)	0
dropout_3 (Dropout)	(None, 112, 112, 32)	0
conv2d_11 (Conv2D)	(None, 112, 112, 64)	51264
batch_normalization_8 (BatchNormalization)	(None, 112, 112, 64)	256
max_pooling2d_5 (MaxPooling2D)	(None, 56, 56, 64)	0
dropout_4 (Dropout)	(None, 56, 56, 64)	0
conv2d_12 (Conv2D)	(None, 56, 56, 128)	204928
batch_normalization_9 (BatchNormalization)	(None, 56, 56, 128)	512
max_pooling2d_6 (MaxPooling2D)	(None, 28, 28, 128)	0
dropout_5 (Dropout)	(None, 28, 28, 128)	0
flatten_2 (Flatten)	(None, 100352)	0
dense_4 (Dense)	(None, 512)	51380736
batch_normalization_10 (BatchNormalization)	(None, 512)	2048
dropout_6 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 50)	25650

Total params: 51,667,954
Trainable params: 51,666,482
Non-trainable params: 1,472

原始架構



貓狗辨識架構



27/27 [==============================] - 3s 39ms/step - loss: 2.5745 - accuracy: 0.40
Test loss: 2.574482025299072
Test accuracy: 0.40838322043418884

參考:

https://www.kaggle.com/code/uysimty/keras-cnn-dog-or-cat-classification

心得

方法	校驗資料精準度變化
調整圖片大小	整體微升
樣本權重(公式 1)	不變
樣本權重(公式 2)	不變
調整批次大小	整體上升
調整打亂緩衝區大小	不變
設定學習率	整體上升
設定學習率衰退	不變
調整模型	不變
Batch Normalization	不變
Average Pooling	不變
Same Padding	不變
更改激活函數	整體下降
數據增強	整體上升
其他架構	不變

我在自己的 Jupyter Notebook 上使用 GTX 3060 顯卡進行訓練，同時使用了 Early Stop、Workers 和 Multiprocessing 來提高訓練效率。經過多次測試後，我仍無法達到 50%的準確度，而且大部分問題都出現在過度擬合上。一開始我以為過度擬合只是指訓練精度的尾端急劇上升的情況，但上網查詢後才發現，訓練和驗證資料的準確度之間的差距過大也是過度擬合的一種表現。最後由於時間限制，我無法改善這個問題，但我認為可以使用正則化或增加隨機丟棄神經元的方法來改善這個問題，此外令我感到困惑的是，在準確率穩定上升時，偶爾會突然出現一 epoch 急遽下降的情況。。