學號：B06902066 系級：資工二 姓名：蔡秉辰

1. 請比較你本次作業的架構，參數量、結果和原 HW3 作業架構、參數量、結果做比較。(1%)

(1) HW3 之架構、參數量：

HW3 有使用 ensemble，而這次作業是改自 HW3 中的其中一個 model，該 model 架構如下：

Conv2D(32, (3, 3)) + BatchNormalization( )

Conv2D(32, (3, 3)) + BatchNormalization( )

Conv2D(64, (3, 3)) + PReLu( ) + MaxPooling2D(2, 2) + BatchNormalization( )

Conv2D(64, (3, 3)) + PReLu( ) + BatchNormalization( ) + Dropout(0.1)

Conv2D(128, (3, 3)) + PReLu( ) + BatchNormalization( ) + Dropout(0.2)

Conv2D(128, (3, 3)) + PReLu( ) + MaxPooling2D(2, 2) + BatchNormalization( ) + Dropout(0.2)

Flatten( )

Dense(512) + PReLu( ) + BatchNormalization( ) + Dropout(0.5)

Dense(256) + PReLu( ) + BatchNormalization( ) + Dropout(0.5)

Dense(128) + PReLu( ) + BatchNormalization( ) + Dropout(0.5)

總參數：10,079,847。有用 ModelCheckpoint 和 Data Augmentation：rotation_range = 25, horizontal_flip = True, width_shift_range = 0.1, height_shift_range = 0.1, zoom_range = 0.2

(2) 本次使用之架構、參數量：

Conv2D(14, (3, 3)) + BatchNormalization( )

Conv2D(14, (3, 3)) + MaxPooling2D(2, 2) + BatchNormalization( )

Conv2D(28, (3, 3)) + BatchNormalization( )

Conv2D(28, (3, 3)) + LeakyReLu( )

MaxPooling2D(2, 2) + BatchNormalization( ) + Dropout(0.1)

Conv2D(56, (3, 3)) + LeakyReLu( ) + BatchNormalization( )

Conv2D(56, (3, 3)) + LeakyReLu( )

MaxPooling2D(2, 2) + BatchNormalization( ) + Dropout(0.2)

Flatten( ) + Dense(7, activation = 'softmax')

總參數：69909。有使用 ModelCheckpoint 和做 Data Augmentation(增強參數和 HW3 相同)。

除此之外，本次並非存整個 model，而是只有存 weight。且在存之前會先將其轉成'float16'之後再去進行儲存。

(3) 結果之比較：

	Private Accuracy	Public Accuracy
HW3 的其中一個 model(model 2)	0.66703	0.66007
HW8	0.63304	0.63861

可以看出，本次作業所使用之 model 較 HW3 的 model 小上非常多(已挑

選 hw3 最小之 model 做比較)，但就預測準確率而言(忽略 ensemble)，準確率僅約往下降 2~3%左右而已。

2. 請使用 MobileNet 的架構，畫出參數量-acc 的散布圖（橫軸為參數量，縱軸為 accuracy，且至少 3 個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用 train 到最好沒關係。）(1%)

(1) 架構：

三個 model 架構相同，只是使用之參數量不同。

(i) 第一個 model 如下：

Conv2D(8, (3, 3)) + BatchNormalization( )

DepthwiseConv2D((3, 3)) + Conv2D(8, (1, 1)) + BatchNormalization( )

Maxpooling2D( ) + BatchNormalization( )

DepthwiseConv2D((3, 3)) + Conv2D(16, (1, 1)) + BatchNormalization( )

DepthwiseConv2D((3, 3)) + Conv2D(16, (1, 1)) + BatchNormalization( ) + LeakyReLU( )

Maxpooling2D( ) + BatchNormalization( ) + Dropout(0.1)

DepthwiseConv2D((3, 3)) + Conv2D(32, (1, 1)) + BatchNormalization( ) + LeakyReLU( )

DepthwiseConv2D((3, 3)) + Conv2D(32, (1, 1)) + BatchNormalization( ) + LeakyReLU( )

Maxpooling2D( ) + BatchNormalization( ) + Dropout(0.1)

Flatten( ) + Dense(7, activation = 'softmax')

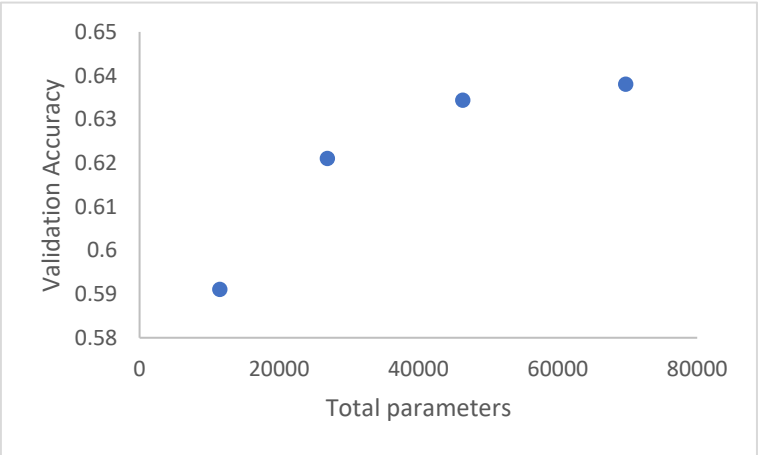
(ii) 第二個 model 為將以上 Conv2D 的 filter 數從[8, 8, 16, 16, 32, 32]增加成[16, 16, 32, 32, 64, 64]

(iii) 第三個 model 則是將 Conv2D 的 filter 數增為[32, 32, 64, 64, 128, 128]

(iv) 第四個 model 為將 Conv2D 的 filter 數設為[24, 24, 48, 48, 96, 96]

(2) 參數量 - validation acc 散布圖 & 表格數據：

註：均 train 100 個 epoch、均有用和第一題相同之 Data Augmentation、且均有使用 ModelCheckpoint。



參數量	11487	26935	46351	69735
Validation Acc	0.59100	0.62100	0.63433	0.63800

3. 請使用一般 CNN 的架構，畫出參數量-acc 的散布圖（橫軸為參數量，縱軸為 accuracy，且至少 3 個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用 train 到最好沒關係。）(1%)

(1) 架構：

和上題一樣，三個 model 架構相同，只是使用之參數量不同。

(i) 第一個 model 如下：

Conv2D(10, (3, 3)) + BatchNormalization( )

Conv2D(10, (3, 3)) + BatchNormalization( )

Maxpooling2D( ) + BatchNormalization( )

Conv2D(20, (3, 3)) + BatchNormalization( )

Conv2D(20, (3, 3)) + BatchNormalization( ) + LeakyReLU( )

Maxpooling2D( ) + BatchNormalization( ) + Dropout(0.1)

Conv2D(40, (3, 3)) + BatchNormalization( ) + LeakyReLU( )

Conv2D(40, (3, 3)) + BatchNormalization( ) + LeakyReLU( )

Maxpooling2D( ) + BatchNormalization( ) + Dropout(0.1)

Flatten( ) + Dense(7, activation = 'softmax')

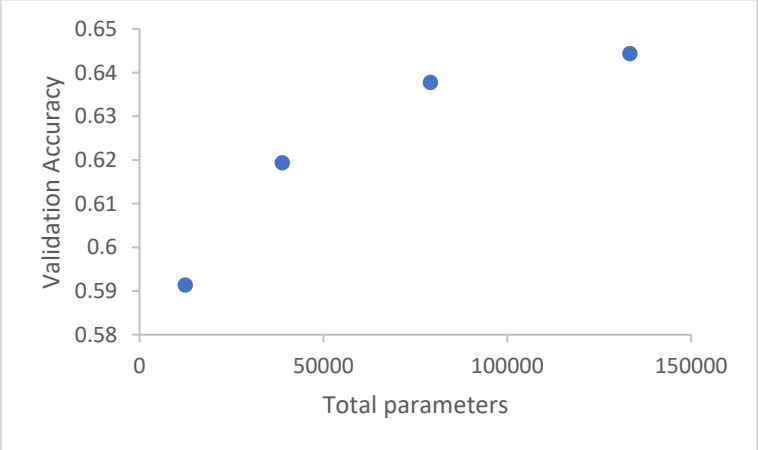
(ii) 第二個 model 為將以上 Conv2D 的 filter 數從[10, 10, 20, 20, 40, 40]增加成[15, 15, 30, 30, 60, 60]

(iii) 第三個 model 則是將 Conv2D 的 filter 數增為[20, 20, 40, 40, 80, 80]

(iv) 第四個 model 則是將 Conv2D 的 filter 數降為[5, 5, 10, 10, 20, 20]

(2) 參數量 – validation acc 散布圖 & 表格數據：

註：均 train 100 個 epoch。均有用和第一題相同之 Data Augmentation、且有用 ModelCheckpoint。



參數量	12417	38777	79087	133347
Validation Acc	0.59133	0.61933	0.63767	0.64433

4. 請你比較題 2 和題 3 的結果，並請針對當參數量相當少的時候，如果兩者參數量相當，兩者的差異，以及你認為為什麼會造成這個原因。(2%)

第二題和第三題中，參數相當(相差 10000 以下)時有以下三組：

	Prob_2	Prob_3
參數量	11487	12417
Validation Acc	0.59100	0.59133

	Prob_2	Prob_3
參數量	46351	38777
Validation Acc	0.63433	0.61933

	Prob_2	Prob_3
參數量	69735	79087
Validation Acc	0.63800	0.63767

可以看出，在參數量相當小的時候(11487, 12417)，在 validation set 的準確率非常相近；但在參數量略大時((46351, 38777)和(69735, 79085))卻有發生 MobileNet 表現較好、兩者表現差不多這兩種情形。我認為可能造成之原因有以下：

(1) 我認為，100 個 epoch 基本上只是大致收斂而已。在本次用來繳交 kaggle 的 model 中(無論是 CNN 或是 MobileNet)，我使用 epoch 數為 400，在 200~400 個 epoch 時，validation accuracy 仍會有略為的上升(通常為 0.5~2%)。

(2) 在此次的 case 中，MobileNet 的 Accuracy(validation/testing set 均是如此)變動度相當大(initial state 並沒有固定)，在同樣參數下，可能跑出 Validation Acc 從 60.x%到 63.x%都有發生過；甚至 kaggle 上有發生同一個 model 中，public 和 private score 相差到快 3%的情形(其他 model 頂多差不到 1%)。而 CNN 完全沒有這種情形發生。

故我認為，若將 MobileNet train 到完全收斂時，其”平均而言”的表現會較 CNN model 好一點點，但變動幅度比 CNN model 大上許多。