Machine Learning HW8 Report

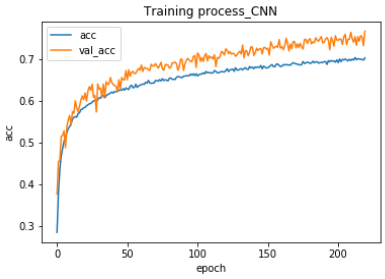
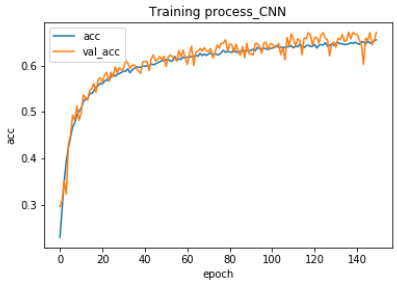
學號：B05901040　系級：電機三　姓名：蔡松達

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%)

   模型架構與參數量如下表所示，HW3 跑了 220 epochs，HW8 跑了 150 epochs，兩者 batch size 皆設為 100，皆有實作 data augmentation。

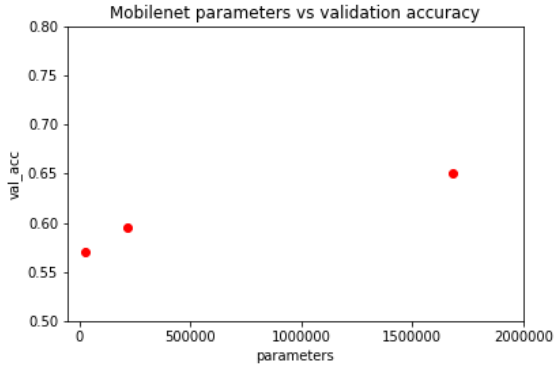
Model	CNN model (HW3)	Mobilenet model (HW8)
Model 架構	總參數量：1332807 在此架構中每三層相同的 Convolution 後面會接一層 Batch normalization 層（參數採預設值），後面再接一層 LeakyRelu(alpha=0.3)作為 activation，接著使用一層 Maxpooling，最後再接一層 Dropout(rate=0.3)，總共搭建三組如此的架構，以下僅寫出三組中 Convolution 及 Maxpooling 層各自的參數： 第一組：Conv2D(32, kernel_size = (3, 3), strides=(1, 1), padding='same' , input_shape = (48, 48, 1)) MaxPooling2D(pool_size=(2, 2), strides = (2,2), padding='same') 第二組： Conv2D(64, kernel_size = (3, 3), strides=(1, 1), padding='same') MaxPooling2D(pool_size=(2, 2), strides = (2, 2), padding='same') 第三組：Conv2D(128, kernel_size = (2, 2), strides=(1, 1), padding='valid') MaxPooling2D(pool_size=(2, 2), strides = (2, 2), padding='valid') 最後接上下列幾層： Flatten() Dense(units = 512)	總參數量：73095 在此架構中先搭一層原本的 Convolution，接著連接 8 組的 depthwise+pointwise convolution，最後再接一層 Maxpooling 及 Dropout(0.2)後進行 Flatten，並連接 Dense(64)→Dense(7) output，每層 Convolution、Depthwise Convolution、Pointwise Convolution 後面皆有連接 Batchnormalization 層以及 Leaky Relu(alpha=0.3)作為 activation function，參數如下列 (Pointwise convolution 的 kernel_size = (1, 1), strides=(1, 1), padding='same')： 第一層 Convolution：Conv2D(8, kernel_size = (3, 3), strides=(1, 1), padding='same', input_shape = (48, 48, 1)) 八組的 Depthwise+Pointwise Convolution: (1) DepthwiseConv2D(kernel_size = (3, 3), strides=(1, 1), padding='same') PointwiseConv2D(filter: 16) (2) DepthwiseConv2D(kernel_size = (3, 3), strides=(1, 1), padding='same') PointwiseConv2D(filter: 16) (3) DepthwiseConv2D(kernel_size = (3,

	BatchNormalization() Dropout(0.3) LeakyReLU(alpha=0.3) Dense(units = 7, activation = 'softmax')	3), strides=(2, 2), padding='valid') PointwiseConv2D(filter: 32) (4) DepthwiseConv2D(kernel_size = (3, 3), strides=(1, 1), padding='same') PointwiseConv2D(filter: 32) (5) DepthwiseConv2D(kernel_size = (3, 3), strides=(2, 2), padding='valid') PointwiseConv2D(filter: 64) (6) DepthwiseConv2D(kernel_size = (3, 3), strides=(1, 1), padding='same') PointwiseConv2D(filter: 64) (7) DepthwiseConv2D(kernel_size = (3, 3), strides=(2, 2), padding='valid') PointwiseConv2D(filter: 128) (8) DepthwiseConv2D(kernel_size = (2, 2), strides=(1, 1), padding='valid') PointwiseConv2D(filter: 128) Maxpooling 層： MaxPooling2D(pool_size=(2, 2), strides = (2, 2), padding='valid') 最後接上： Dropout(0.2)→Flatten()→ Dense(64)→BatchNormalization()→ Dropout(0.2)→LeakyRelu(0.3)→ Dense(7, activation: softmax)
Public score	0.69100	0.63666
Private score	0.67316	0.62552
Testing score	0.68208	0.63109

Training process		

兩次 training 的過程其實有不少相異之處，包含參數量差距甚大，epoch 次數不同，但從 training 過程的 acc 變化圖可以大致比較兩者在 train 150 epochs 之後，CNN 架構的 validation acc 已經達到 0.7 左右，而 Mobilenet 則大約為 0.64 左右。至於從架構來看，HW3 的架構使用三組 Convolution（每組都放三層一樣的 Conv2D），因此總共有九層 Convolution；而 HW8 的架構使用一層 Conv2D 以及 8 組的 Depthwise+pointwise Convolution，等同於大致上使用了 9 層的 convolution，只差在 filter 數量有所不同（較 HW3 少）。雖然由此結果 CNN 的效果較佳，但其使用十幾倍以上的參數量，反觀 Mobilenet 也疊了大致上相同層數的 Convolution，參數量卻少了許多，也能達到一定的效果，因此可以在參數量與 accuracy 之間取捨，以此依據來選擇是要使用 CNN 還是 Mobilenet 架構。

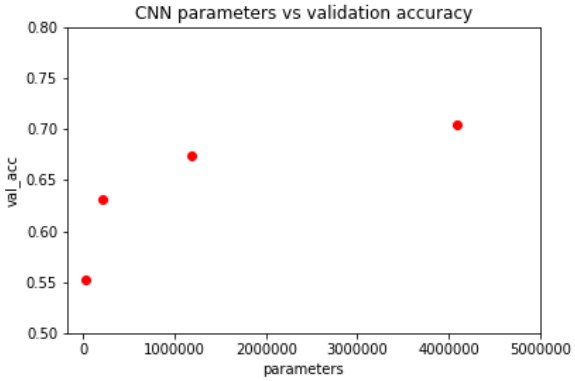
2. 請使用 MobileNet 的架構，畫出參數量-acc 的散布圖（橫軸為參數量，縱軸為 accuracy，且至少 3 個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用 train 到最好沒關係。）(1%)



參數量	validation accuracy （做 augmentation 後使用原本的 training data 做 validation set）
28775	0.5707
213127	0.5947
1684535	0.6503

此處呈現出三個點（皆 train 50 epochs），可以簡單觀察出 Mobilenet 在參數量相當少的時候就能達到 0.57 的效果，但其實隨著參數量增加，accuracy 增加的速度相當明顯的趨緩，可見 MobileNet 較適合用在總參數量被限制在較少的時候比較有優勢。

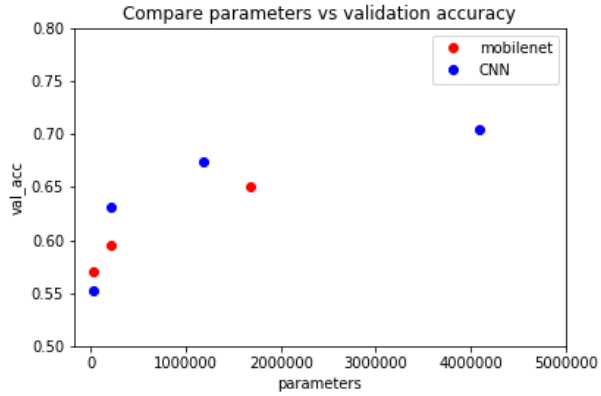
3. 請使用一般 CNN 的架構，畫出參數量-acc 的散布圖（橫軸為參數量，縱軸為 accuracy，且至少 3 個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用 train 到最好沒關係。）(1%)



參數量	validation accuracy （做 augmentation 後使用原本的 training data 做 validation set）
28911	0.5518
218807	0.6315
1192679	0.6737
4098087	0.7043

此處呈現出四個點（皆 train 50 epochs），可以發覺在參數量很少的時候 accuracy 比起 MobileNet 還低，但是達到二十萬左右的參數量後就能夠勝出 MobileNet，成長的幅度雖然也漸漸趨緩，但整體而言在接近的參數量下，accuracy 皆在 MobileNet 之上（除了參數量非常少的狀況之外）。

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%)



此圖比較兩者參數量 v.s 正確率進行作圖，可以觀察到上述提到的情形，也就是 MobileNet 在參數相當少的情況下較具優勢（大約五萬到十萬個參數以內），但超過此參數量後使用 CNN 的架構便能取得明顯優勢，也就是普遍而言 CNN 較能夠表達出整張圖的特徵，而 Mobilenet 與 CNN 參數相當時往往是疊的較深的情況（因為 MobileNet 精簡了每層的參數量，在 filter 數量相

差不遠的情況下，Mobilenet 想與 CNN 架構之參數相當必然是疊的較深），而太深的架構有時候較難 train 的起來；但是在參數量非常少的時候，也就是大約幾萬個參數的情況下，CNN 的架構可能會變得相當淺或是 filter 數量極少，而 MobileNet 此時的深度跟 filter 數量恰好落在較為合適的情況，達到的效果是比較好的，此外從圖中能觀察出 Mobilenet 整體的 accuracy 落在較小的 range 中，CNN 的 accuracy range 則較大，可能是使用 CNN 架構仍舊比較能夠表達出特徵之間的關聯性，畢竟 CNN 每層使用的參數比起 Mobilenet 來的多，這些比 Mobilenet 多出來的參數能夠將特徵描述的更仔細，但在參數量極少的狀況下，CNN 疊的層數不夠多，又花了不少參數在描述較為細膩的特徵，因此整題而言較難 train 起來；而 MobileNet 因為充分利用了僅有的參數在描述各種大方向的特徵，捨去那些比較細的關聯性，雖然沒辦法描述的很細膩，卻能夠大致 cover 到較多明顯的 feature，這也是為什麼只有在參數量極少的時候 MobileNet 才具優勢。