學號：B05901111 系級：電機三 姓名：陳建成

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%)

直接分別附上Keras程式碼方便說明，

```
model = Sequential([
    BatchNormalization(input_shape=(48, 48, 1)),
    Conv2D(24, 5), Activation('relu'),
    Conv2D(36, 5), Activation('relu'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),

    BatchNormalization(),
    Conv2D(52, 1), Activation('relu'),
    DepthwiseConv2D(5, padding='same'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(rate=0.4),

    BatchNormalization(),
    Conv2D(64, 1), Activation('relu'),
    DepthwiseConv2D(5, padding='same'),
    DepthwiseConv2D(5, padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),

    BatchNormalization(),
    Conv2D(84, 1), Activation('relu'),
    DepthwiseConv2D(5, padding='same'),
    DepthwiseConv2D(5, padding='same'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2)),
    Dropout(rate=0.4),

    Flatten(),

    Dense(90, activation='relu'), Dropout(rate=0.5),
    Dense(7, activation='softmax')
])
```

```
model = Sequential()
model.add(BatchNormalization(input_shape=input_shape))
model.add(Conv2D(64, (5, 5), padding='same'))
model.add(LeakyReLU(0.2))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.4))

model.add(BatchNormalization(input_shape=input_shape))
model.add(Conv2D(128, (5, 5), padding='same'))
model.add(LeakyReLU(0.2))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))

model.add(BatchNormalization(input_shape=input_shape))
model.add(Conv2D(256, (5, 5), padding='same'))
model.add(LeakyReLU(0.2))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.4))

model.add(BatchNormalization(input_shape=input_shape))
model.add(Conv2D(256, (5, 5), padding='same'))
model.add(LeakyReLU(0.2))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Dropout(0.4))

model.add(Flatten())

model.add(Dense(256, use_bias=True))
model.add(LeakyReLU(0.2))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
```

左邊是HW8而右邊是HW3的架構。

原則上主要就是將原本HW3的架構中convolution layers減低filter的數量，並且改成Depthwise與Pointwise的組合這樣，當然最後面的Dense layer也有減少neuron畢竟這裡會明顯影響總參數量。不過在訓練過程中發現，前面一兩層維持原本形式的convolution layer可以大幅有效提升收斂時的準確率（不過filter數量就要更低就是了），因此前面留了兩層一般的convolution layers。

左邊的架構用了73,663個參數，而右邊用了3,258,379個，大約差44倍。

以Kaggle上的結果比較，

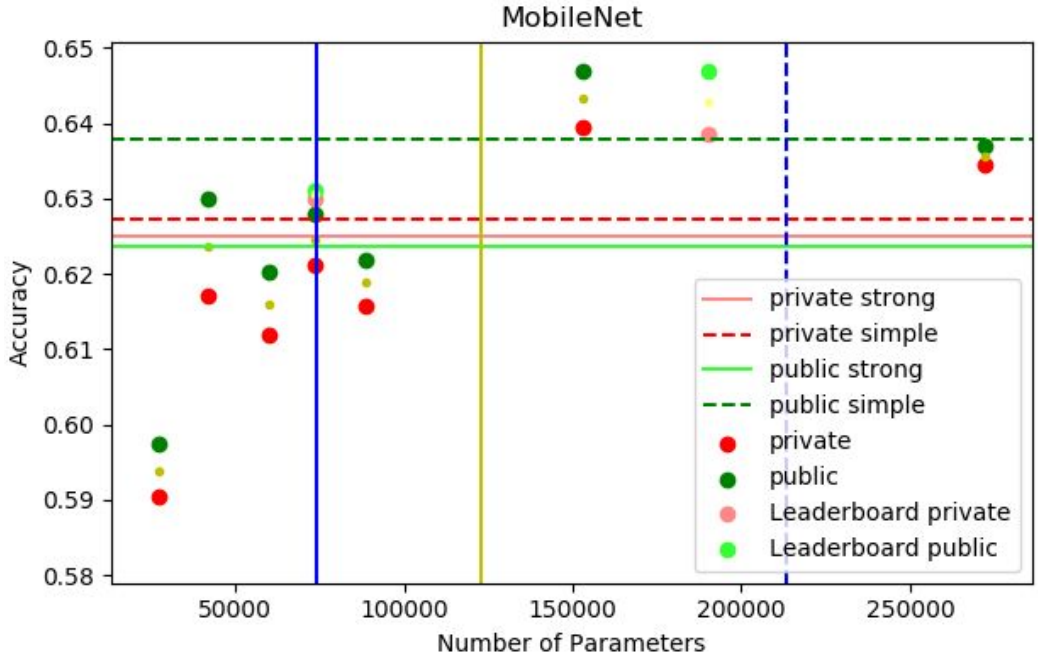
左邊的準確率在private set上是0.62998、public set上是0.63109；

右邊的準確率在private set上是0.67372、public set上是0.69100。

畢竟參數差異相當大，準確率當然也會有差異。

之所以用大約73,000個參數是因為容量大小的限制，經由減低精確度（float32變成float16）可以減到223KB左右。然而後來在嘗試時發現把參數的list分別儲存成各自array的shape，並把所有的參數flatten成一個73,663的array，最後經由npz_compressed可以再減低40%左右的大小，最後上去GitHub的便是這個版本。（不過這也是其實可以把模型擴張到12萬個參數也可以的意思，但後來因為很難訓練而沒有這麼做。其他如knowledge distillation或是用neuron pruning的方法因為訓練效果都不如直接重新訓練MobileNet來得好因此最後就沒有實作了。）

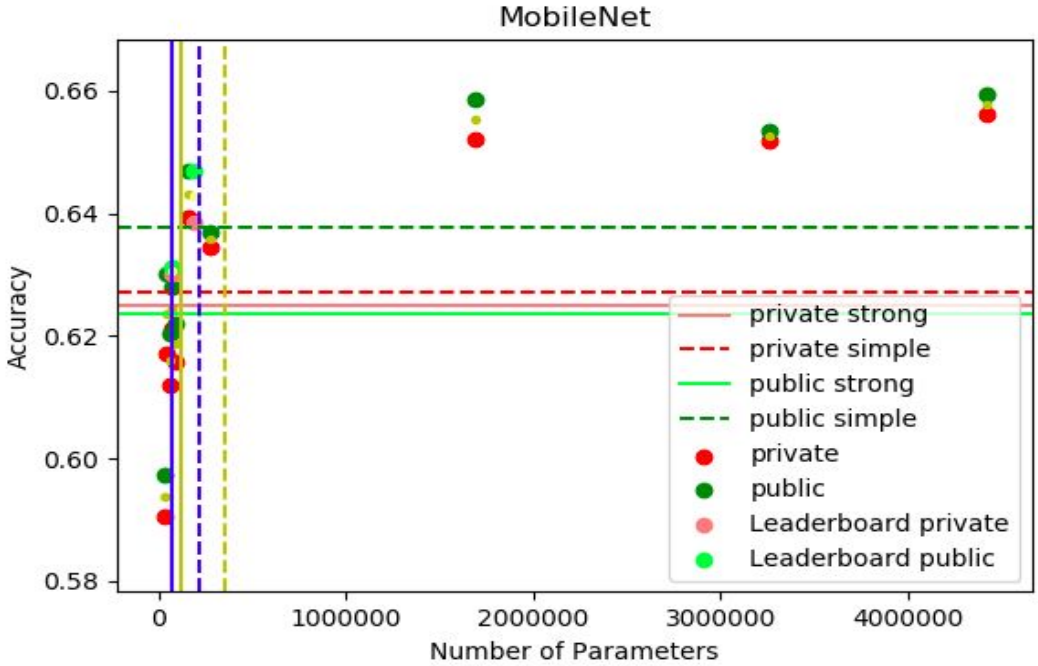
2. 請使用MobileNet的架構，畫出參數量-acc的散布圖（橫軸為參數量，縱軸為accuracy，且至少3個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用train到最好沒關係。）(1%)



這張圖表的acc是根據Kaggle上所得到的數據，然後順便加上了size跟acc的baseline。其中虛線是simple、實線是strong，數據中的小黃點是兩個acc的平均（假設simple和strong的testcases各佔一半）。

關於垂直線的部分，因為我原本只是採用npy.save的方式儲存（藍色垂直線），但後來採用reshape和npz_compressed大幅壓縮了所需空間（大約只需原本的60%）因此參數量的baseline因而變得比較寬鬆（土黃色垂直線）。

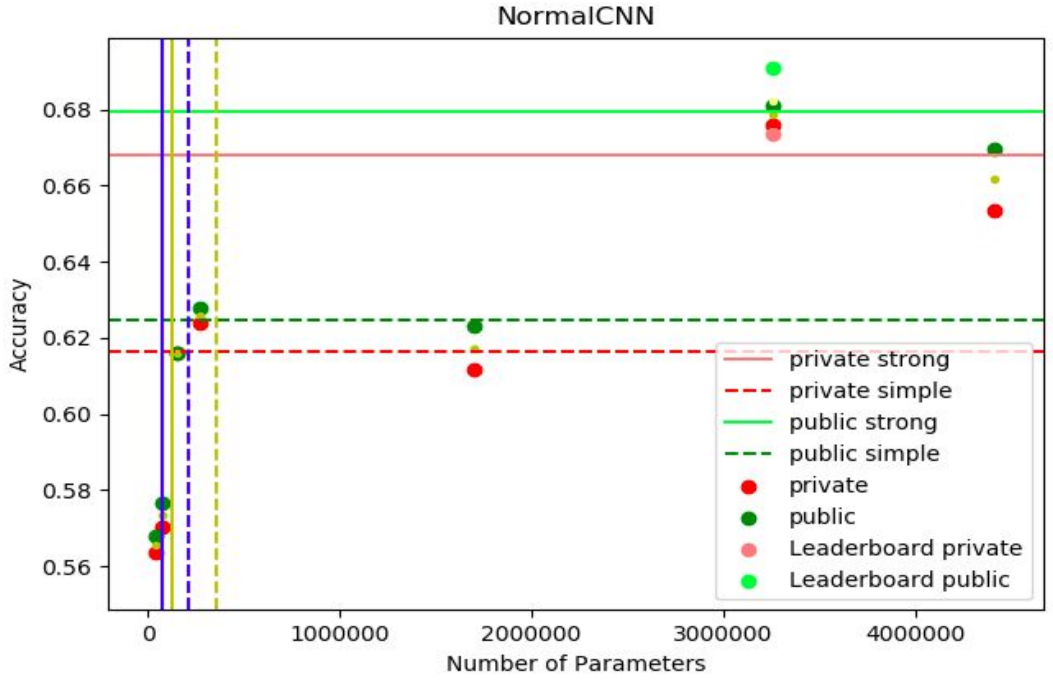
最後圖中相對其他點顏色較為偏淺的6個點是上傳到leaderboard上的資料，原則上是reproduce左上角那個。其實為了下一題比較方便，我還有做參數量更大時的實驗，但是因為會壓縮到許多資料因此獨立到另一張圖表示。



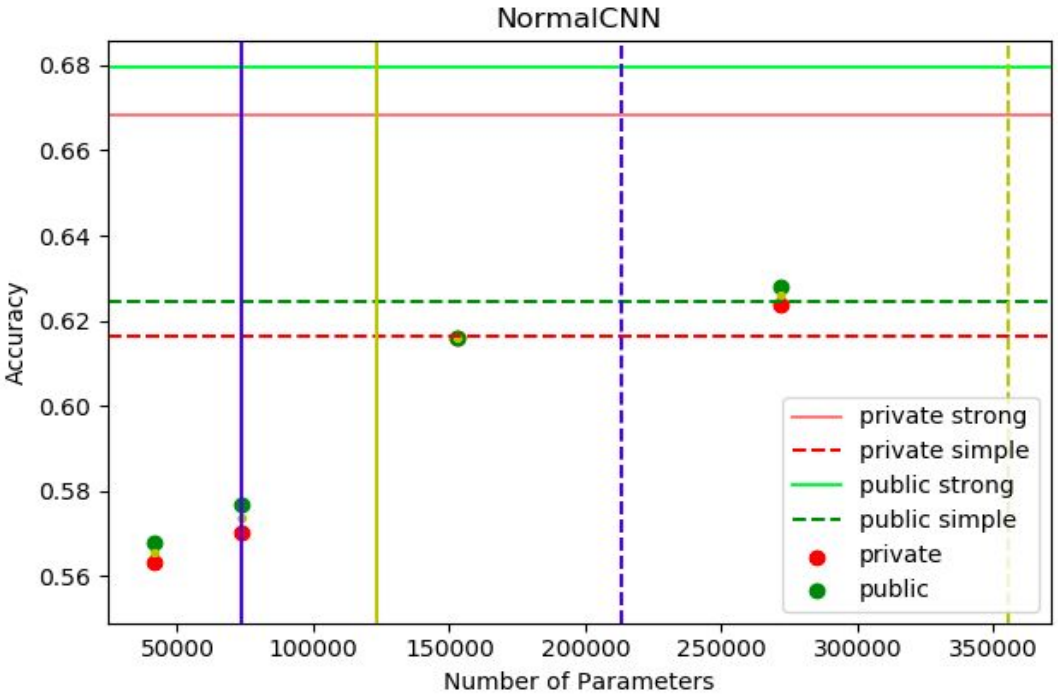
3. 請使用一般CNN的架構，畫出參數量-acc的散布圖（橫軸為參數量，縱軸為accuracy，且至少3個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用train到最好沒關係。）(1%)

採用與上一題相似的方法製作圖表，不過因為一般的CNN train起來還蠻花時間的（不與別人合作是因為想試著比較自己相同的架構下的結果，畢竟每個人架構都有差異），所以其實有些model疑似還可以再train更高一些。

然後下面的baseline是HW3的（畢竟是以HW3類型的模型訓練），而參數量因為沒有限制所以就參考這次作業的。不過為了方便下一題比較，故意一般CNN與MobileNet都選擇了參數量相似的情況去進行了訓練。



同樣的，因為參數少的部分在圖表上不好觀察因此另外進行了放大。



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%)

因為題3的模型並沒有完全train到收斂，因此結果上感覺似乎略差一些，但也另一方面顯示因為模型大所以訓練起來速度比較慢。然而，一般的CNN模型可以達到68%的正確率，這個數值目前我的MobileNet模型還做不到，可見大的模型雖然訓練慢，但是確實是比較容易訓練到好的結果的。

另外，比較有意思的是，在兩者參數相當時，參數多的時候MobileNet好像也只能到66%左右就上不去了，反觀在CNN可以有比較多的潛能繼續網上；但在參數相當少的時候，MobileNet明顯是佔有優勢的，CNN反而變得很難訓練，而且照著自己觀察的訓練曲線大概也不會輕易收斂在太好的準確率上。

可見對於參數少的一般模型來說，訓練確實比參數多時困難很多；而適當改變模型架構確實有助於達成模型壓縮的任務。