學號：R07942115 系級：電信碩一 姓名：謝硯澤

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%)

HW8-Model Architecture:

Layer (type)	Output Shape	Param #
batch_normalization_1 (Batch	(None, 48, 48, 1)	4
conv2d_1 (Conv2D)	(None, 48, 48, 12)	120
conv2d_2 (Conv2D)	(None, 48, 48, 12)	1308
conv2d_3 (Conv2D)	(None, 48, 48, 24)	312
batch_normalization_2 (Batch	(None, 48, 48, 24)	96
max_pooling2d_1 (MaxPooling2	(None, 23, 23, 24)	0
conv2d_4 (Conv2D)	(None, 23, 23, 24)	5208
conv2d_5 (Conv2D)	(None, 23, 23, 24)	600
batch_normalization_3 (Batch	(None, 23, 23, 24)	96
max_pooling2d_2 (MaxPooling2	(None, 11, 11, 24)	0
conv2d_6 (Conv2D)	(None, 11, 11, 36)	7812
conv2d_7 (Conv2D)	(None, 11, 11, 36)	1332
batch_normalization_4 (Batch	(None, 11, 11, 36)	144
max_pooling2d_3 (MaxPooling2	(None, 5, 5, 36)	0
conv2d_8 (Conv2D)	(None, 5, 5, 24)	7800
conv2d_9 (Conv2D)	(None, 5, 5, 24)	600
batch_normalization_5 (Batch	(None, 5, 5, 24)	96
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 24)	0
Flatten (Flatten)	(None, 96)	0
dropout_1 (Dropout)	(None, 96)	0
dense_1 (Dense)	(None, 128)	12416
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 7)	903
softmax1 (Activation)	(None, 7)	0

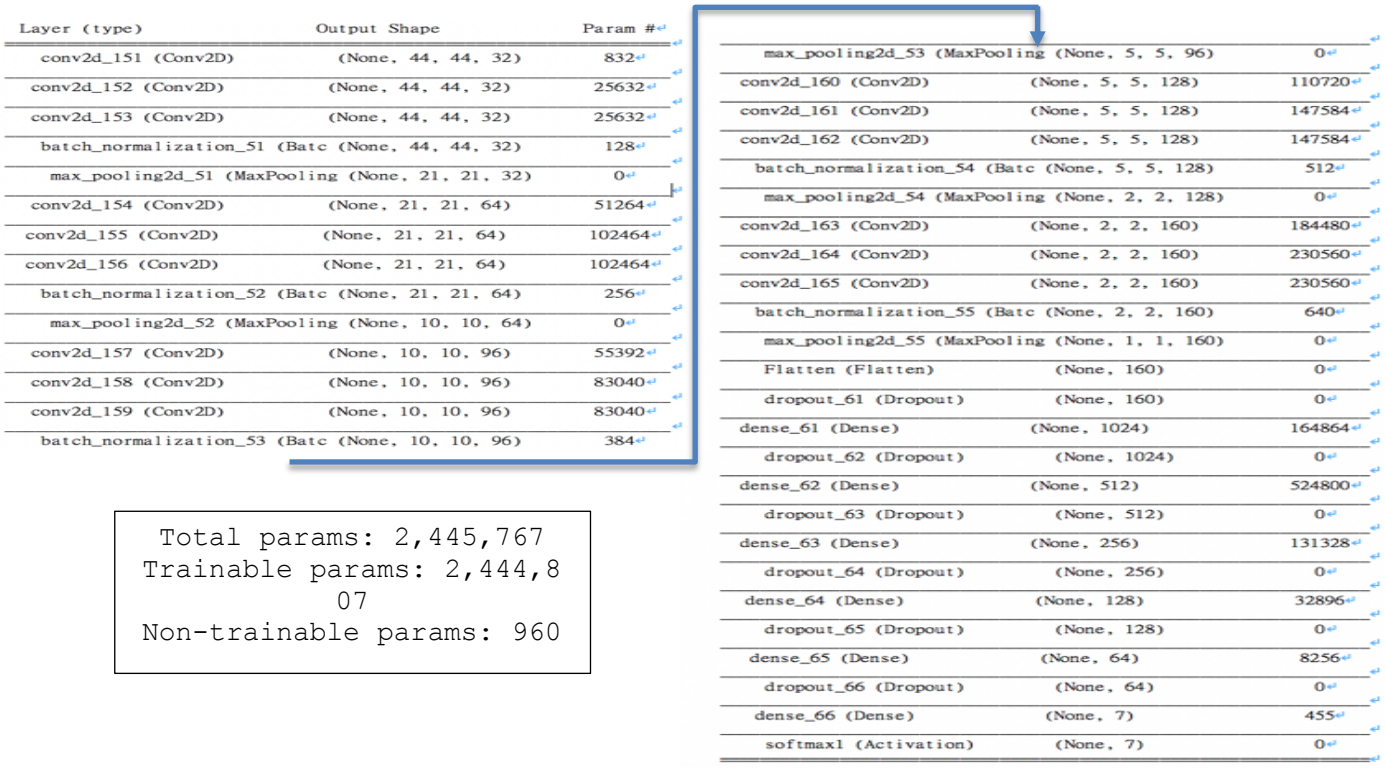
Total params: 38,847
Trainable params: 38,629
Non-trainable params: 218

看上圖的架構可能比較比較不好看出 3*3, 1*1 conv. Layer 加在哪，我以下表來補充說明：

BatchNormalization(input_shape=(48,48,1))
Conv2D(depth*1, (3, 3), activation = 'relu' , padding = 'same')
Conv2D(depth*1, (3, 3), activation = 'relu' , padding = 'same')
Conv2D(depth*2, (1, 1), activation = 'relu' , padding = 'same')
BatchNormalization()
MaxPooling2D(pool_size=(3, 3) , strides = (2,2))
Conv2D(depth*2, (3, 3), activation = 'relu' , padding = 'same')
Conv2D(depth*2, (1, 1), activation = 'relu' , padding = 'same')
BatchNormalization()
MaxPooling2D(pool_size=(2, 2) , strides = (2,2))
Conv2D(depth*3, (3, 3), activation = 'relu' , padding = 'same')
Conv2D(depth*3, (1, 1), activation = 'relu' , padding = 'same')
BatchNormalization()

MaxPooling2D(pool_size=(2, 2) , strides = (2,2))
Conv2D(depth*2, (3, 3), activation = 'relu' , padding = 'same')
Conv2D(depth*2, (1, 1), activation = 'relu' , padding = 'same')
BatchNormalization()
MaxPooling2D(pool_size=(2, 2) , strides = (2,2))
Flatten(name = 'Flatten')
Dropout(0.2)
Dense(128 , activation = 'relu')
Dropout(0.2)
Dense(7, kernel_initializer='normal')
Activation('softmax', name='softmax1')

HW3-Model Architecture



Layer (type)	Output Shape	Param #
conv2d_151 (Conv2D)	(None, 44, 44, 32)	832
conv2d_152 (Conv2D)	(None, 44, 44, 32)	25632
conv2d_153 (Conv2D)	(None, 44, 44, 32)	25632
batch_normalization_51 (Batc	(None, 44, 44, 32)	128
max_pooling2d_51 (MaxPooling	(None, 21, 21, 32)	0
conv2d_154 (Conv2D)	(None, 21, 21, 64)	51264
conv2d_155 (Conv2D)	(None, 21, 21, 64)	102464
conv2d_156 (Conv2D)	(None, 21, 21, 64)	102464
batch_normalization_52 (Batc	(None, 21, 21, 64)	256
max_pooling2d_52 (MaxPooling	(None, 10, 10, 64)	0
conv2d_157 (Conv2D)	(None, 10, 10, 96)	55392
conv2d_158 (Conv2D)	(None, 10, 10, 96)	83040
conv2d_159 (Conv2D)	(None, 10, 10, 96)	83040
batch_normalization_53 (Batc	(None, 10, 10, 96)	384
max_pooling2d_53 (MaxPooling	(None, 5, 5, 96)	0
conv2d_160 (Conv2D)	(None, 5, 5, 128)	110720
conv2d_161 (Conv2D)	(None, 5, 5, 128)	147584
conv2d_162 (Conv2D)	(None, 5, 5, 128)	147584
batch_normalization_54 (Batc	(None, 5, 5, 128)	512
max_pooling2d_54 (MaxPooling	(None, 2, 2, 128)	0
conv2d_163 (Conv2D)	(None, 2, 2, 160)	184480
conv2d_164 (Conv2D)	(None, 2, 2, 160)	230560
conv2d_165 (Conv2D)	(None, 2, 2, 160)	230560
batch_normalization_55 (Batc	(None, 2, 2, 160)	640
max_pooling2d_55 (MaxPooling	(None, 1, 1, 160)	0
Flatten (Flatten)	(None, 160)	0
dropout_61 (Dropout)	(None, 160)	0
dense_61 (Dense)	(None, 1024)	164864
dropout_62 (Dropout)	(None, 1024)	0
dense_62 (Dense)	(None, 512)	524800
dropout_63 (Dropout)	(None, 512)	0
dense_63 (Dense)	(None, 256)	131328
dropout_64 (Dropout)	(None, 256)	0
dense_64 (Dense)	(None, 128)	32896
dropout_65 (Dropout)	(None, 128)	0
dense_65 (Dense)	(None, 64)	8256
dropout_66 (Dropout)	(None, 64)	0
dense_66 (Dense)	(None, 7)	455
softmax1 (Activation)	(None, 7)	0

```
Total params: 2,445,767
Trainable params: 2,444,807
Non-trainable params: 960
```

	HW8	HW3
Total parameters	38,847	2,445,767
Kaggle public score	0.62998	0.66118
Kaggle private score	0.62747	0.64474

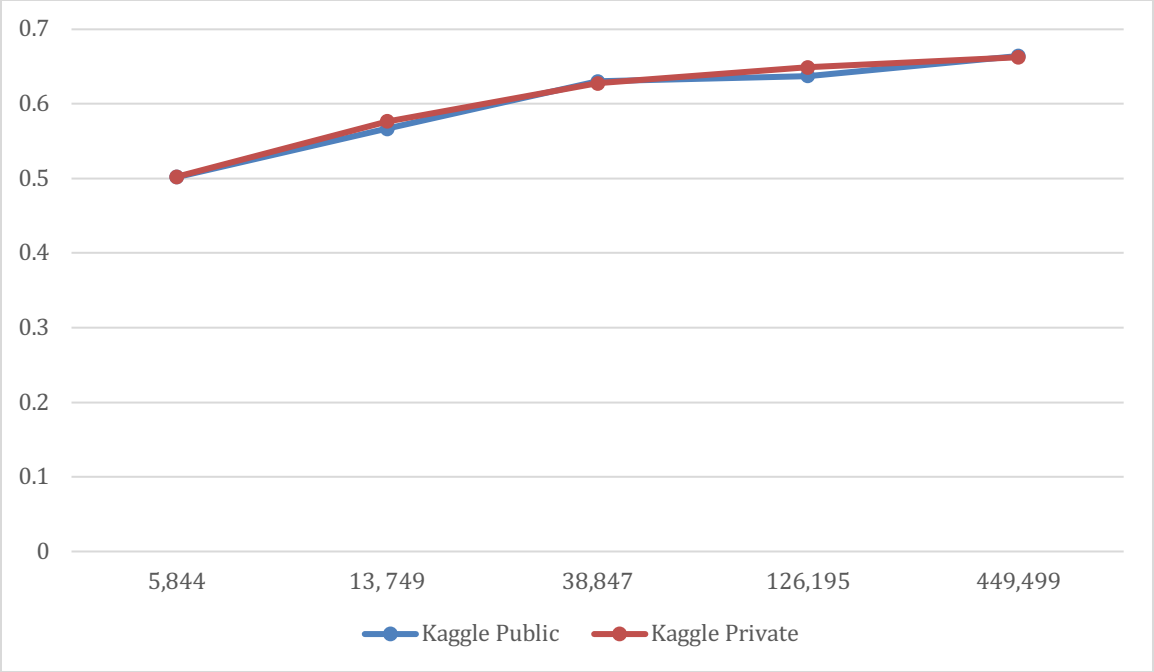
2. 請使用 **MobileNet** 的架構，畫出參數量**-acc** 的散布圖（橫軸為參數量，縱軸為 **accuracy**，且至少 **3** 個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用 **train** 到最好沒關係。）**(1%)**

這題我對 mobilenet 的架構做了一些調整，希望獲得更小的 model size 和更好的 accuracy，架構則與第一題中的 HW8-Model Architecture 一樣。以下分別對五種不同的參數量做訓練，並比較他們的 accuracy。

Layer (type)	Output Shape	Param #
batch_normalization_1 (Batch	(None, 48, 48, 1)	4
conv2d_1 (Conv2D)	(None, 48, 48, 12)	120
conv2d_2 (Conv2D)	(None, 48, 48, 12)	1308
conv2d_3 (Conv2D)	(None, 48, 48, 24)	312
batch_normalization_2 (Batch	(None, 48, 48, 24)	96
max_pooling2d_1 (MaxPooling2	(None, 23, 23, 24)	0
conv2d_4 (Conv2D)	(None, 23, 23, 24)	5208
conv2d_5 (Conv2D)	(None, 23, 23, 24)	600
batch_normalization_3 (Batch	(None, 23, 23, 24)	96
max_pooling2d_2 (MaxPooling2	(None, 11, 11, 24)	0
conv2d_6 (Conv2D)	(None, 11, 11, 36)	7812
conv2d_7 (Conv2D)	(None, 11, 11, 36)	1332
batch_normalization_4 (Batch	(None, 11, 11, 36)	144
max_pooling2d_3 (MaxPooling2	(None, 5, 5, 36)	0
conv2d_8 (Conv2D)	(None, 5, 5, 24)	7800
conv2d_9 (Conv2D)	(None, 5, 5, 24)	600
batch_normalization_5 (Batch	(None, 5, 5, 24)	96
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 24)	0
Flatten (Flatten)	(None, 96)	0
dropout_1 (Dropout)	(None, 96)	0
dense_1 (Dense)	(None, 128)	12416
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 7)	903
softmax1 (Activation)	(None, 7)	0

Total params: 38,847
Trainable params: 38,629
Non-trainable params: 218

實驗說明：上圖為本題所使用的架構，其中我固定整體模型的架構，只去調整每一層 kernel 的數量，也就是去調整輸入到下一層 feature map 的深度。實作方法為設定一個超參數 “depth” ，當 depth=12 時，conv2d_1 的 output shape 為(None, 48, 48, 12); 當 depth=24 時，conv2d_1 的 output shape 則為(None, 48, 48, 24)，以此類推其他層的 output shape。我總共做了五個實驗，depth 分別設為：3, 6, 12, 24, 48，實驗結果如下圖所示：



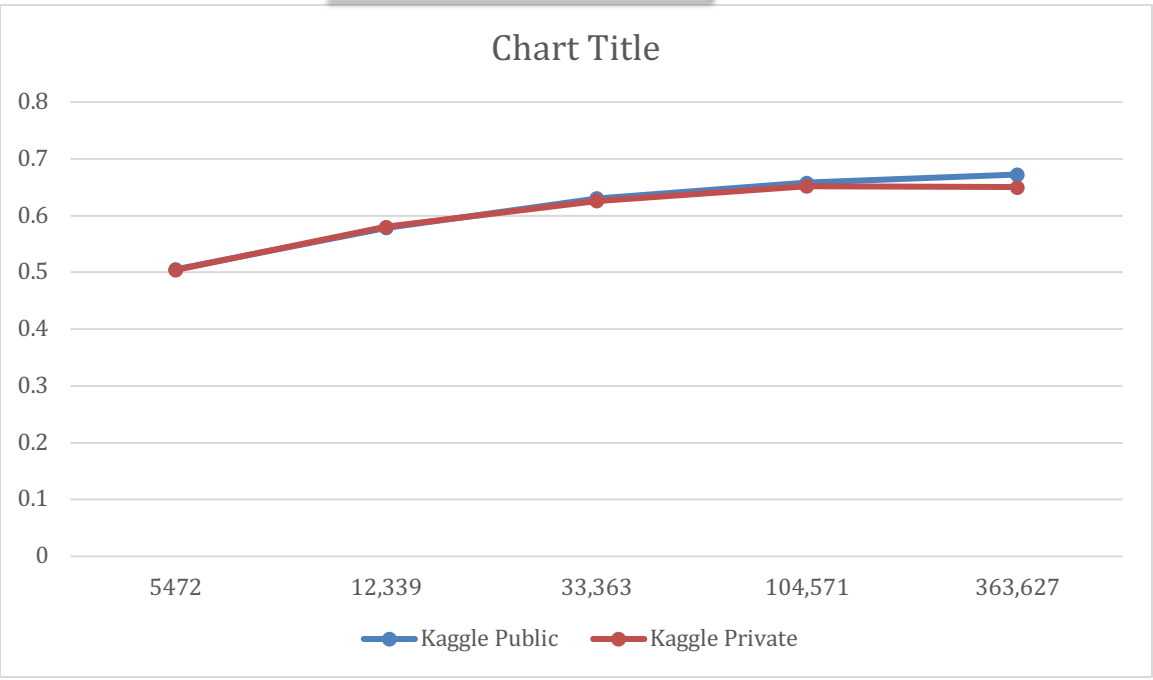
Total params	Kaggle Public	Kaggle Private
5,844	0.50181	0.50236
13, 749	0.56673	0.57648
38,847	0.62998	0.62747
126,195	0.63750	0.64892
449,499	0.66425	0.66258

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%)**

本圖使用的 CNN 架構為基於第二題架構下去做修改，把所有用到 1*1 conv. filter 的 layer 都刪掉，並一樣只調整超參數 “depth” 來控制總參數量。depth 依序設定為：3, 6, 12, 24, 48 如下表所示：

Layer (type)	Output Shape	Param #
batch_normalization_6 (Batch	(None, 48, 48, 1)	4
conv2d_6 (Conv2D)	(None, 48, 48, 12)	120
conv2d_7 (Conv2D)	(None, 48, 48, 12)	1308
batch_normalization_7 (Batch	(None, 48, 48, 12)	48
max_pooling2d_5 (MaxPooling2	(None, 23, 23, 12)	0
conv2d_8 (Conv2D)	(None, 23, 23, 24)	2616
batch_normalization_8 (Batch	(None, 23, 23, 24)	96
max_pooling2d_6 (MaxPooling2	(None, 11, 11, 24)	0
conv2d_9 (Conv2D)	(None, 11, 11, 36)	7812
batch_normalization_9 (Batch	(None, 11, 11, 36)	144
max_pooling2d_7 (MaxPooling2	(None, 5, 5, 36)	0
conv2d_10 (Conv2D)	(None, 5, 5, 24)	7800
batch_normalization_10 (Batc	(None, 5, 5, 24)	96
max_pooling2d_8 (MaxPooling2	(None, 2, 2, 24)	0
Flatten (Flatten)	(None, 96)	0
dropout_3 (Dropout)	(None, 96)	0
dense_3 (Dense)	(None, 128)	12416
dropout_4 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 7)	903
softmax1 (Activation)	(None, 7)	0

Total params: 33,363
Trainable params: 33,169
Non-trainable params: 194



Total params	Kaggle Public	Kaggle Private
5472	0.50543	0.50431
12,339	0.57871	0.58010
33,363	0.63053	0.62607
104,571	0.65812	0.65171
363,627	0.67233	0.65032

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%)

比較題 2, 題 3 我發現，不知道是否是因為我有調整了 mobilenet 的架構，還是其他原因，我覺得只用一般的 CNN 架構，accuracy 也不錯，上述兩題的方法，我並沒有使用到 data augmentation。根據我實驗的結果發現，在參數量相當少的時候（如下面的比較表，參數量皆有達到 strong 的標準），其實題三跟題二的 accuracy 是差不多的，且題三的參數量還比較小。我認為會有這個原因，不知道是不是我們要處理的這個 task 本來就不需要那麼多的參數量就可以達到 63%左右的準確度。如果在其他 task 上，可能 mobilenet 架構的優勢可以更凸顯出來。

	題 2(參數量:38,847)	題 3(參數量:33,363)
Kaggle Public	0.62998	0.63053
Kaggle Private	0.62747	0.62607