學號：B04505028 系級：工科四 姓名：林秀銓

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%)

hw3:

架構

```
def build_model():
    filt_size = (3, 3)
    model = Sequential()
    model.add(Convolution2D(32, filt_size, input_shape=(48,48,1), activation='relu', padding='same'))
    model.add(Convolution2D(32, filt_size, activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2,2)))
    model.add(Dropout(0.5))

    model.add(Convolution2D(64, filt_size, activation='relu', padding='same'))
    model.add(Convolution2D(64, filt_size, activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2,2)))
    model.add(Dropout(0.5))

    model.add(Convolution2D(128, filt_size, activation='relu', padding='same'))
    model.add(Convolution2D(128, filt_size, activation='relu', padding='same'))
    model.add(Convolution2D(128, filt_size, activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2,2)))
    model.add(Dropout(0.5))

    model.add(Convolution2D(256, filt_size, activation='relu', padding='same'))
    model.add(Convolution2D(256, filt_size, activation='relu', padding='same'))
    #model.add(Convolution2D(256, filt_size, activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(MaxPooling2D((2,2)))
    model.add(Dropout(0.5))


    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(1024, activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(7))
    model.add(Activation('softmax'))

    print(model.summary())
    return model
```

參數量

```
================================
Total params: 4,746,471
Trainable params: 4,741,415
Non-trainable params: 5,056
```

結果

Private Score	Public Score
0.65310	0.66815

hw8:

架構

```
x = _conv_block(img_input, 8, alpha)
x = _depthwise_conv_block(x, 8, alpha, depth_multiplier, block_id=1)

x = _conv_block(x, 16, alpha)
x = _depthwise_conv_block(x, 16, alpha, depth_multiplier, block_id=3)

x = MaxPool2D(pool_size=(2, 2), padding="same")(x)

x = _conv_block(x, 32, alpha)
x = _depthwise_conv_block(x, 32, alpha, depth_multiplier, block_id=5)

x = _depthwise_conv_block(x, 32, alpha, depth_multiplier, strides=(2, 2), block_id=6)

x = MaxPool2D(pool_size=(2, 2), padding="same")(x)
x = _conv_block(x, 64, alpha)
x = _depthwise_conv_block(x, 64, alpha, depth_multiplier, block_id=8)
x = _depthwise_conv_block(x, 64, alpha, depth_multiplier, block_id=9)
x = MaxPool2D(pool_size=(2, 2), padding="same")(x)
# x = _depthwise_conv_block(x, 64, alpha, depth_multiplier, block_id=10)
# x = _depthwise_conv_block(x, 64, alpha, depth_multiplier, block_id=11)

# x = _depthwise_conv_block(x, 1024, alpha, depth_multiplier,
#                           strides=(2, 2), block_id=12)
# x = _depthwise_conv_block(x, 1024, alpha, depth_multiplier, block_id=13)

if pooling == 'avg':
    x = AveragePooling2D(padding="same")(x)
elif pooling == 'max':
    x = MaxPool2D(padding="same")(x)

x = Flatten()(x)
x = Dense(30)(x)
x = BatchNormalization()(x)
x = LeakyReLU(0.2)(x)
x = Dropout(0.2)(x)
x = Dense(7)(x)
x = Activation('softmax')(x)
```

```python
def _conv_block(inputs, filters, alpha, kernel=(3, 3), strides=(1, 1)):
    channel_axis = 1 if K.image_data_format() == 'channels_first' else -1
    filters = int(filters * alpha)
    x = Conv2D(filters, kernel,
               padding='same',
               use_bias=False,
               strides=strides)(inputs)
    x = BatchNormalization(axis=channel_axis)(x)
    return Activation(relu6)(x)


def _depthwise_conv_block(inputs, pointwise_conv_filters, alpha,
                          depth_multiplier=1, strides=(1, 1), block_id=1):
    channel_axis = 1 if K.image_data_format() == 'channels_first' else -1
    pointwise_conv_filters = int(pointwise_conv_filters * alpha)

    x = DepthwiseConv2D((3, 3),
                        padding='same',
                        depth_multiplier=depth_multiplier,
                        strides=strides,
                        use_bias=False,
                        name='conv_dw_%d' % block_id)(inputs)
    x = BatchNormalization(axis=channel_axis, name='conv_dw_%d_bn' % block_id)(x)
    x = Activation(relu6, name='conv_dw_%d_relu' % block_id)(x)

    x = Conv2D(pointwise_conv_filters, (1, 1),
               padding='same',
               use_bias=False,
               strides=(1, 1),
               name='conv_pw_%d' % block_id)(x)
    x = BatchNormalization(axis=channel_axis, name='conv_pw_%d_bn' % block_id)(x)
    return Activation(relu6, name='conv_pw_%d_relu' % block_id)(x)
```

參數量

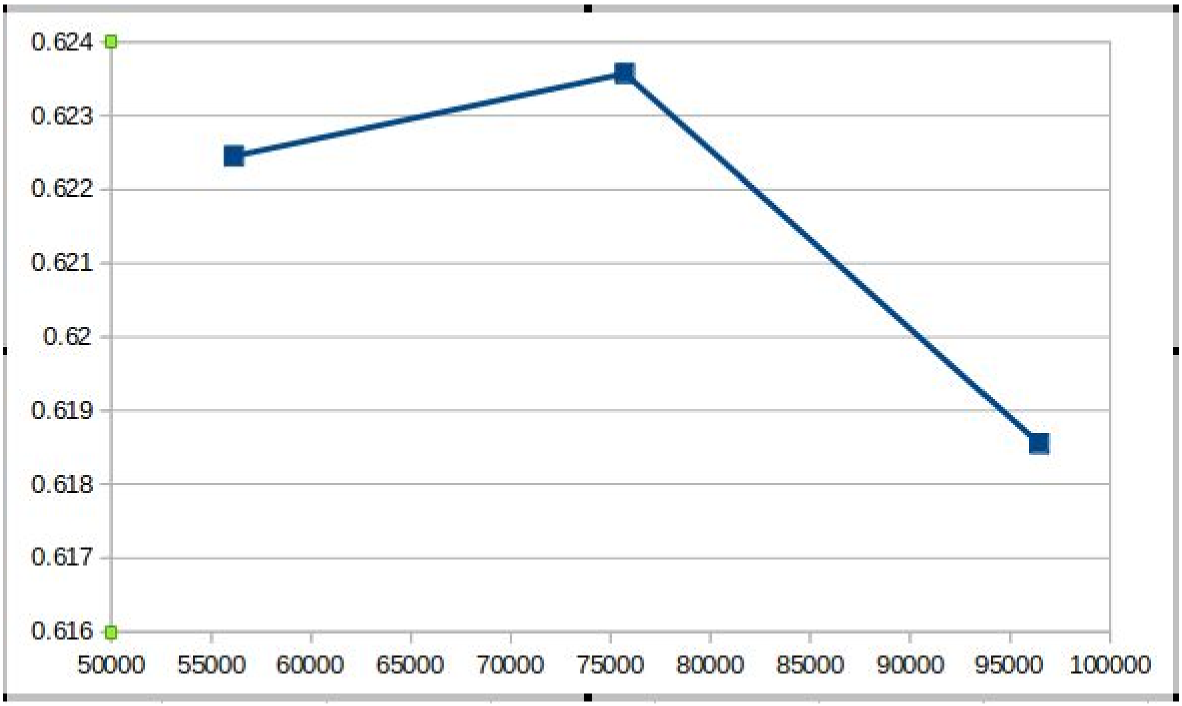
```
==============================
Total params: 56,135
Trainable params: 54,903
Non-trainable params: 1,232
```

結果

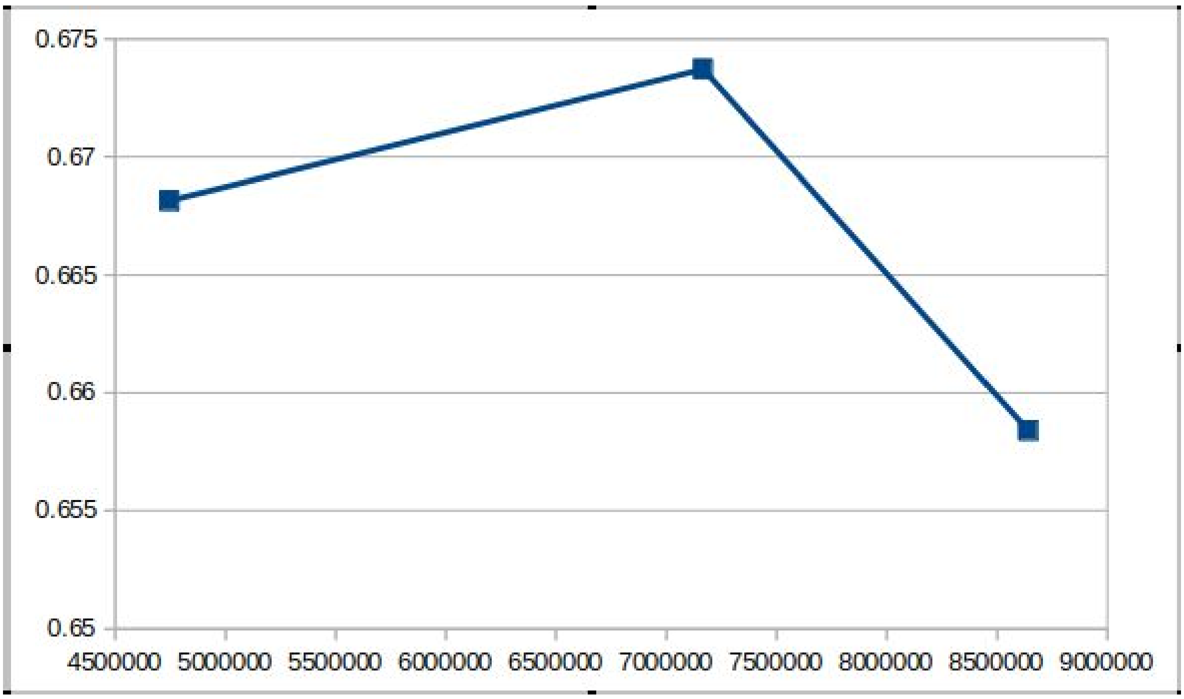
Private Score	Public Score
0.61855	0.62245

兩種架構的參數量相差了約85倍，辦事兩者結果只相差了約4%，因此MobileNet確實可以壓縮參數量，且對結果不會影響太大。

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%)



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%)



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%)

由2和3題的結果可以發現，增加參數量都可以提昇正確率，但若增加太多則會overfitting。

而當參數量相當時，MobileNet參數量：56135，一般CNN參數量：42167，結果分別為下左圖和下右圖，可以發現兩者差異非常大，可能是因為CNN的layer跟kernel數量較少，且mobilenet的參數量和計算量都遠低於同樣效果的CNN，所以同樣參數量的mobilenet會有必同樣參數量的cnn更好的performance，造成結果相差這麼多。

Private Score	Public Score
0.61855	0.62245

Private Score	Public Score
0.38450	0.39648