學號：R07942091 系級：電信碩一 姓名：許博閔

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%)

   本次 mobilenet 架構 : 下圖是 print(model)的結果，先經過一層普通的 CNN layer，再過 4 層的 Depthwise Separable Convolution layer，將 feature average pooling 後通過 2 層的 fully connected layer 後得到預測的結果。總參數量為 109761

   準確率: kaggle public:0.64279　kaggle private:0.62970

   model size:224224bytes

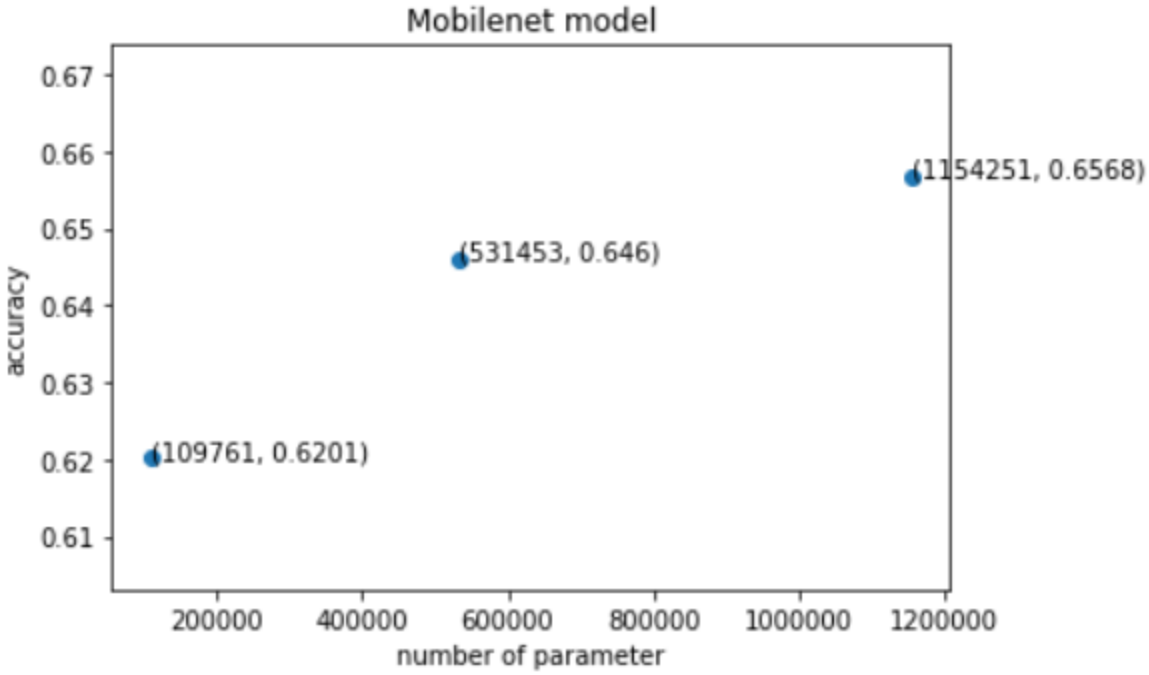
```
Net(
  (model): Sequential(
    (0): Sequential(
      (0): Conv2d(1, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.05)
    )
    (1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=64, bias=False)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.05)
      (3): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): LeakyReLU(negative_slope=0.05)
    )
    (2): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=128, bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.05)
      (3): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): LeakyReLU(negative_slope=0.05)
    )
    (3): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=128, bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.05)
      (3): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
      (5): LeakyReLU(negative_slope=0.05)
    )
    (4): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=128, bias=False)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.05)
      (3): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (5): LeakyReLU(negative_slope=0.05)
    )
    (5): AvgPool2d(kernel_size=3, stride=3, padding=0)
  )
  (fc1): Sequential(
    (0): Linear(in_features=256, out_features=100, bias=True)
    (1): ReLU()
    (2): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): Dropout(p=0.5)
  )
  (fc2): Linear(in_features=100, out_features=7, bias=True)
)
```

HW3 的 model 架構為 5 層的普通 CNN layear 和 3 層的 fully connected layer ，總參數量為 5758791，準確率: kaggle public:0.71914　kaggle private:0.70660

兩者相比，從架構上來看，其實都有 5 層 CNN layers，但因為 mobilenet 是 depthwise Separable CNN layers，因此 mobilenet 總參數量大約是 HW3 的五分之一，但準確率只有少了約 8%。

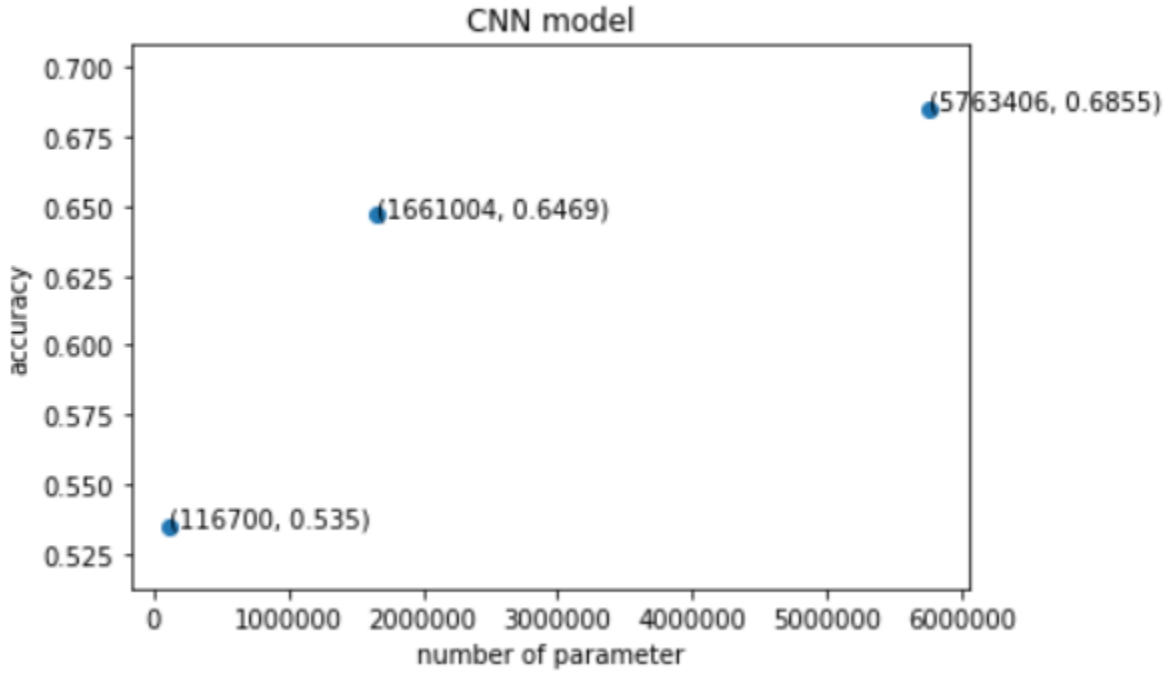
2. 請使用 MobileNet 的架構，畫出參數量-acc 的散布圖（橫軸為參數量，縱軸為 accuracy，且至少 3 個點，參數量選擇時儘量不要離的太近，結果選擇只要大致收斂，不用 train 到最好沒關係。）(1%)

   X 軸為參數量，Y 軸為準確率




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%)

   X 軸為參數量，Y 軸為準確率




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%)

   2、3 題皆用 adam optimizer，learning rate = 0.001，用 torchvision transform 做 data augmentation(水平翻轉、旋轉、平移、縮放)，並 train 100 個 epoch，取 validation dataset 的準確率當結果。

   根據結果，當參數量多(超過 $10^6$)且參數量相當時，mobilenet 和普通 CNN layer 的準確率其實差不多，但當參數量少(大約 $10^5$)且參數量相當時，mobilenet 的準確率比普通 CNN layer 好了 10%左右。

   我認為 mobilenet 在參數量少時會表現比普通 CNN 好的原因是壓縮 model size 的方法不同，普通 CNN 要縮小 model size 最簡單的方法就是減少 filter 的數量，但這樣就會使 CNN 提取出來的 feature 變少，而 mobilenet 使用 Depthwise Separable CNN layer，藉由 channel wise 的 filter 和之後的 1*1 的 filter，用比較少的參數，但仍維持和尚未縮小 CNN model 所提取的 feature 相同的量，因此在相同參數量的狀況下，mobilenet 提出的 feature 數量是比較多的，因此可以有比較好的結果。