ML-HW7 Network Compression Report

學號：b06901066　系級：電機三　姓名：孟妍

1. 請從 Network Pruning/Quantization/Knowledge Distillation/Low Rank Approximation/Design Architecture 選擇兩者實做並詳述你的方法，將同一個大 model 壓縮至接近相同的參數量，並紀錄其 accuracy。 (2%)

・Origin model:

原始 model 就是一個有 8 層 convolution layer 的 CNN model，最後接上一層 FC layer，參數量為 2168203，架構如下：

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 256, 256]             448
       BatchNorm2d-2         [-1, 16, 256, 256]              32
              ReLU-3         [-1, 16, 256, 256]               0
         MaxPool2d-4         [-1, 16, 128, 128]               0
            Conv2d-5         [-1, 32, 128, 128]           4,640
       BatchNorm2d-6         [-1, 32, 128, 128]              64
              ReLU-7         [-1, 32, 128, 128]               0
         MaxPool2d-8           [-1, 32, 64, 64]               0
            Conv2d-9           [-1, 64, 64, 64]          18,496
      BatchNorm2d-10           [-1, 64, 64, 64]             128
             ReLU-11           [-1, 64, 64, 64]               0
        MaxPool2d-12           [-1, 64, 32, 32]               0
           Conv2d-13          [-1, 128, 32, 32]          73,856
      BatchNorm2d-14          [-1, 128, 32, 32]             256
             ReLU-15          [-1, 128, 32, 32]               0
        MaxPool2d-16          [-1, 128, 16, 16]               0
           Conv2d-17          [-1, 256, 16, 16]         295,168
      BatchNorm2d-18          [-1, 256, 16, 16]             512
             ReLU-19          [-1, 256, 16, 16]               0
        MaxPool2d-20            [-1, 256, 8, 8]               0
           Conv2d-21            [-1, 256, 8, 8]         590,080
      BatchNorm2d-22            [-1, 256, 8, 8]             512
             ReLU-23            [-1, 256, 8, 8]               0
        MaxPool2d-24            [-1, 256, 4, 4]               0
           Conv2d-25            [-1, 256, 4, 4]         590,080
      BatchNorm2d-26            [-1, 256, 4, 4]             512
             ReLU-27            [-1, 256, 4, 4]               0
        MaxPool2d-28            [-1, 256, 2, 2]               0
           Conv2d-29            [-1, 256, 2, 2]         590,080
      BatchNorm2d-30            [-1, 256, 2, 2]             512
             ReLU-31            [-1, 256, 2, 2]               0
        MaxPool2d-32            [-1, 256, 1, 1]               0
AdaptiveAvgPool2d-33            [-1, 256, 1, 1]               0
           Linear-34                  [-1, 11]           2,827
================================================================
Total params: 2,168,203
Trainable params: 2,168,203
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 50.91
Params size (MB): 8.27
Estimated Total Size (MB): 59.93
----------------------------------------------------------------
```

(1) Architecture Design + Weight Quantization:

將原本的 model 的 convolution Layer 換成 Dw & Pw 後，參數量就小很多，只剩下 256779，架構如下：

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 256, 256]             448
       BatchNorm2d-2         [-1, 16, 256, 256]              32
             ReLU6-3         [-1, 16, 256, 256]               0
          MaxPool2d-4         [-1, 16, 128, 128]               0
            Conv2d-5         [-1, 16, 128, 128]             160
       BatchNorm2d-6         [-1, 16, 128, 128]              32
             ReLU6-7         [-1, 16, 128, 128]               0
            Conv2d-8         [-1, 32, 128, 128]             544
          MaxPool2d-9           [-1, 32, 64, 64]               0
           Conv2d-10           [-1, 32, 64, 64]             320
      BatchNorm2d-11           [-1, 32, 64, 64]              64
            ReLU6-12           [-1, 32, 64, 64]               0
           Conv2d-13           [-1, 64, 64, 64]           2,112
         MaxPool2d-14           [-1, 64, 32, 32]               0
           Conv2d-15           [-1, 64, 32, 32]             640
      BatchNorm2d-16           [-1, 64, 32, 32]             128
            ReLU6-17           [-1, 64, 32, 32]               0
           Conv2d-18          [-1, 128, 32, 32]           8,320
         MaxPool2d-19          [-1, 128, 16, 16]               0
           Conv2d-20          [-1, 128, 16, 16]           1,280
      BatchNorm2d-21          [-1, 128, 16, 16]             256
            ReLU6-22          [-1, 128, 16, 16]               0
           Conv2d-23          [-1, 256, 16, 16]          33,024
           Conv2d-24          [-1, 256, 16, 16]           2,560
      BatchNorm2d-25          [-1, 256, 16, 16]             512
            ReLU6-26          [-1, 256, 16, 16]               0
           Conv2d-27          [-1, 256, 16, 16]          65,792
           Conv2d-28          [-1, 256, 16, 16]           2,560
      BatchNorm2d-29          [-1, 256, 16, 16]             512
            ReLU6-30          [-1, 256, 16, 16]               0
           Conv2d-31          [-1, 256, 16, 16]          65,792
           Conv2d-32          [-1, 256, 16, 16]           2,560
      BatchNorm2d-33          [-1, 256, 16, 16]             512
            ReLU6-34          [-1, 256, 16, 16]               0
           Conv2d-35          [-1, 256, 16, 16]          65,792
AdaptiveAvgPool2d-36            [-1, 256, 1, 1]               0
           Linear-37                  [-1, 11]           2,827
================================================================
Total params: 256,779
Trainable params: 256,779
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 52.50
Params size (MB): 0.98
Estimated Total Size (MB): 54.23
----------------------------------------------------------------
```

用這個小 model train 200 個 epoch，觀察其 training 的 validation accuracy，大約只會收斂到 0.75 左右，經過 WQ 把 model 壓到 268471 bytes，上傳 kaggle 得到 accuracy 為 0.81589

(2) Knowledge Distillation + Weight Quantization

把小 model 拿去對 teacher_resnet18.bin(acc= 88.41%)的 model 做 knowledge distillation，train 200 個 epoch，觀察其 training 的 validation accuracy，就可以收斂到 0.79 左右，再把 train 好的小 model 再去做 WQ，參數量不變。上傳 kaggle 得到 accuracy 為 0.84399

*比較：在同樣的 model 架構與同樣參數設定下，有做 Knowledge Distillation 的準確率就提升不少，而且上傳 kaggle 的成績竟然比 validation accuracy 高不少，但原本有做 Architecture Design 的 model 表現也沒有非常差，因此若需要減少參數量可以先使用 Architecture Design 換成 Dw & Pw convolution，要得到更好的準確率可以再選擇一個好 model 做 KD。另外，因為此次作業有限制上傳的 model 大小必須在 300000 bytes 以內，因此我都還有再用 WQ 壓過限制，但從結果看起來也許 WQ 並不會對準確率造成太大影響。

2. [Knowledge Distillation] 請嘗試比較以下 validation accuracy (兩個 Teacher Net 由助教提供)以及 student 的總參數量以及架構，並嘗試解釋為甚麼有這樣的結果。你的 Student Net 的參數量必須要小於 Teacher Net 的參數量。(2%)

x. Teacher net architecture and # of parameters: torchvision's ResNet18, with 11,182,155 parameters.

y. Student net architecture and # of parameters:

a. Teacher net (ResNet18) from scratch: 80.09%

b. Teacher net (ResNet18) ImageNet pretrained & fine-tune: 88.41%

c. Your student net from scratch:

d. Your student net KD from (a.):

e. Your student net KD from (b.):

---

y. Student net 使用的是助教提供的 architecture design 的 colab 的 model 架構，參數量為 256779，架構如下：

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 256, 256]             448
       BatchNorm2d-2         [-1, 16, 256, 256]              32
             ReLU6-3         [-1, 16, 256, 256]               0
          MaxPool2d-4         [-1, 16, 128, 128]               0
            Conv2d-5         [-1, 16, 128, 128]             160
       BatchNorm2d-6         [-1, 16, 128, 128]              32
             ReLU6-7         [-1, 16, 128, 128]               0
            Conv2d-8         [-1, 32, 128, 128]             544
          MaxPool2d-9           [-1, 32, 64, 64]               0
           Conv2d-10           [-1, 32, 64, 64]             320
      BatchNorm2d-11           [-1, 32, 64, 64]              64
            ReLU6-12           [-1, 32, 64, 64]               0
           Conv2d-13           [-1, 64, 64, 64]           2,112
         MaxPool2d-14           [-1, 64, 32, 32]               0
           Conv2d-15           [-1, 64, 32, 32]             640
      BatchNorm2d-16           [-1, 64, 32, 32]             128
            ReLU6-17           [-1, 64, 32, 32]               0
           Conv2d-18          [-1, 128, 32, 32]           8,320
         MaxPool2d-19          [-1, 128, 16, 16]               0
           Conv2d-20          [-1, 128, 16, 16]           1,280
      BatchNorm2d-21          [-1, 128, 16, 16]             256
            ReLU6-22          [-1, 128, 16, 16]               0
           Conv2d-23          [-1, 256, 16, 16]          33,024
           Conv2d-24          [-1, 256, 16, 16]           2,560
      BatchNorm2d-25          [-1, 256, 16, 16]             512
            ReLU6-26          [-1, 256, 16, 16]               0
           Conv2d-27          [-1, 256, 16, 16]          65,792
           Conv2d-28          [-1, 256, 16, 16]           2,560
      BatchNorm2d-29          [-1, 256, 16, 16]             512
            ReLU6-30          [-1, 256, 16, 16]               0
           Conv2d-31          [-1, 256, 16, 16]          65,792
           Conv2d-32          [-1, 256, 16, 16]           2,560
      BatchNorm2d-33          [-1, 256, 16, 16]             512
            ReLU6-34          [-1, 256, 16, 16]               0
           Conv2d-35          [-1, 256, 16, 16]          65,792
AdaptiveAvgPool2d-36            [-1, 256, 1, 1]               0
           Linear-37                  [-1, 11]           2,827
================================================================
Total params: 256,779
Trainable params: 256,779
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.75
Forward/backward pass size (MB): 52.50
Params size (MB): 0.98
Estimated Total Size (MB): 54.23
----------------------------------------------------------------
```

c. 把 student net 從頭 train 200 epoch，validation accuracy 大概只會到 75%左右

```
[179/200] 38.70 sec(s) Train Acc: 0.876748 Loss: 0.011884 | Val Acc: 0.741399 loss: 0.031150
[180/200] 38.69 sec(s) Train Acc: 0.870363 Loss: 0.011804 | Val Acc: 0.720991 loss: 0.033658
[181/200] 38.71 sec(s) Train Acc: 0.874620 Loss: 0.011857 | Val Acc: 0.751020 loss: 0.031401
[182/200] 38.89 sec(s) Train Acc: 0.876444 Loss: 0.011724 | Val Acc: 0.748105 loss: 0.030374
[183/200] 39.06 sec(s) Train Acc: 0.877965 Loss: 0.011312 | Val Acc: 0.740233 loss: 0.032051
[184/200] 38.89 sec(s) Train Acc: 0.883844 Loss: 0.010952 | Val Acc: 0.735277 loss: 0.031747
[185/200] 38.14 sec(s) Train Acc: 0.876140 Loss: 0.011590 | Val Acc: 0.751020 loss: 0.030439
[186/200] 39.73 sec(s) Train Acc: 0.881208 Loss: 0.011340 | Val Acc: 0.744898 loss: 0.032135
[187/200] 39.90 sec(s) Train Acc: 0.878877 Loss: 0.011684 | Val Acc: 0.755685 loss: 0.030067
saving model with accurracy: 0.7556851311953353
[188/200] 39.61 sec(s) Train Acc: 0.875305 Loss: 0.011531 | Val Acc: 0.725656 loss: 0.034517
[189/200] 39.34 sec(s) Train Acc: 0.878978 Loss: 0.011149 | Val Acc: 0.743149 loss: 0.029543
[190/200] 39.62 sec(s) Train Acc: 0.881614 Loss: 0.010970 | Val Acc: 0.727988 loss: 0.032993
[191/200] 38.92 sec(s) Train Acc: 0.876039 Loss: 0.011553 | Val Acc: 0.739359 loss: 0.033777
[192/200] 39.26 sec(s) Train Acc: 0.879384 Loss: 0.011170 | Val Acc: 0.743440 loss: 0.029928
[193/200] 38.78 sec(s) Train Acc: 0.876850 Loss: 0.011215 | Val Acc: 0.729155 loss: 0.033473
[194/200] 38.58 sec(s) Train Acc: 0.877053 Loss: 0.011213 | Val Acc: 0.727697 loss: 0.034264
[195/200] 38.54 sec(s) Train Acc: 0.882323 Loss: 0.010868 | Val Acc: 0.748105 loss: 0.031669
[196/200] 38.69 sec(s) Train Acc: 0.883539 Loss: 0.010769 | Val Acc: 0.737901 loss: 0.031668
[197/200] 38.59 sec(s) Train Acc: 0.878877 Loss: 0.011072 | Val Acc: 0.750146 loss: 0.030596
[198/200] 40.38 sec(s) Train Acc: 0.880701 Loss: 0.011026 | Val Acc: 0.741108 loss: 0.032962
[199/200] 39.93 sec(s) Train Acc: 0.884148 Loss: 0.010978 | Val Acc: 0.736152 loss: 0.034552
[200/200] 39.99 sec(s) Train Acc: 0.887391 Loss: 0.010537 | Val Acc: 0.733528 loss: 0.033757
```

d. 使用 student net 對 a.的 model (teacher_resnet18_from_scratch.bin acc=80.09%)做 KD，同樣 train 200 個 epoch，validation accuracy 就明顯比 c.好很多，大約可到 79%

```
epoch 151: train loss: 1.7458, acc 0.8727 valid loss: 2.2667, acc 0.7749
epoch 152: train loss: 1.7364, acc 0.8679 valid loss: 2.1561, acc 0.7767
saving model with accuracy: 0.7959183673469388
epoch 153: train loss: 1.7222, acc 0.8672 valid loss: 2.3077, acc 0.7959
epoch 154: train loss: 1.7531, acc 0.8688 valid loss: 2.4053, acc 0.7688
epoch 155: train loss: 1.7742, acc 0.8685 valid loss: 2.2834, acc 0.7848
epoch 156: train loss: 1.7543, acc 0.8705 valid loss: 2.8532, acc 0.7373
epoch 157: train loss: 1.7383, acc 0.8685 valid loss: 2.3311, acc 0.7752
epoch 158: train loss: 1.7256, acc 0.8719 valid loss: 2.2745, acc 0.7895
epoch 159: train loss: 1.7122, acc 0.8738 valid loss: 2.3299, acc 0.7790
epoch 160: train loss: 1.7166, acc 0.8669 valid loss: 2.4186, acc 0.7694
epoch 161: train loss: 1.7265, acc 0.8695 valid loss: 2.3226, acc 0.7679
epoch 162: train loss: 1.7213, acc 0.8713 valid loss: 2.3262, acc 0.7778
epoch 163: train loss: 1.7028, acc 0.8760 valid loss: 2.5222, acc 0.7691
epoch 164: train loss: 1.6880, acc 0.8777 valid loss: 2.3705, acc 0.7601
epoch 165: train loss: 1.7142, acc 0.8718 valid loss: 2.3014, acc 0.7703
epoch 166: train loss: 1.6838, acc 0.8764 valid loss: 2.3781, acc 0.7487
epoch 167: train loss: 1.6908, acc 0.8708 valid loss: 2.1238, acc 0.7915
epoch 168: train loss: 1.7137, acc 0.8700 valid loss: 2.3518, acc 0.7513
epoch 169: train loss: 1.6840, acc 0.8715 valid loss: 2.4226, acc 0.7767
epoch 170: train loss: 1.6652, acc 0.8767 valid loss: 2.2235, acc 0.7837
```

e. 使用 student net 對 b.的 model (teacher_resnet18.bin acc= 88.41%) 做 KD，同樣 train 200 個 epoch，validation accuracy 又比 d.更好一些，大約可以到 82%

```
saving model with accuracy: 0.8174927113702624
epoch 162: train loss: 3.2900, acc 0.8731 valid loss: 3.9577, acc 0.8175
epoch 163: train loss: 3.2960, acc 0.8726 valid loss: 4.2516, acc 0.7921
epoch 164: train loss: 3.2907, acc 0.8784 valid loss: 4.2193, acc 0.8058
epoch 165: train loss: 3.2142, acc 0.8796 valid loss: 4.1475, acc 0.8090
epoch 166: train loss: 3.3300, acc 0.8742 valid loss: 4.8476, acc 0.7708
epoch 167: train loss: 3.3160, acc 0.8736 valid loss: 4.1765, acc 0.8009
epoch 168: train loss: 3.2872, acc 0.8731 valid loss: 3.9536, acc 0.8099
saving model with accuracy: 0.8221574344023324
epoch 169: train loss: 3.2197, acc 0.8803 valid loss: 3.8889, acc 0.8222
epoch 170: train loss: 3.1863, acc 0.8798 valid loss: 4.6665, acc 0.7974
epoch 171: train loss: 3.2403, acc 0.8811 valid loss: 4.4845, acc 0.8117
epoch 172: train loss: 3.2578, acc 0.8755 valid loss: 4.4176, acc 0.8093
epoch 173: train loss: 3.2392, acc 0.8762 valid loss: 4.1890, acc 0.8102
epoch 174: train loss: 3.1847, acc 0.8810 valid loss: 4.1756, acc 0.8073
epoch 175: train loss: 3.1424, acc 0.8801 valid loss: 4.2264, acc 0.8017
epoch 176: train loss: 3.2358, acc 0.8810 valid loss: 3.9958, acc 0.8204
epoch 177: train loss: 3.1742, acc 0.8781 valid loss: 3.8218, acc 0.8169
epoch 178: train loss: 3.2631, acc 0.8775 valid loss: 4.4224, acc 0.7915
epoch 179: train loss: 3.2500, acc 0.8792 valid loss: 4.2512, acc 0.8108
epoch 180: train loss: 3.2090, acc 0.8801 valid loss: 4.9056, acc 0.7895
```

*比較：使用 accuracy 最高的 teacher model 做 knowledge distillation，validation accuracy 最高，實際在 kaggle 的準確率也最高，因為 student model 本身架構較小，所以從頭 train 的結果並沒有很好，但做了 knowledge distillation，準確率就能提高不少。

3. [Network Pruning] 請使用兩種以上的 pruning rate 畫出 X 軸為參數量，Y 軸為 validation accuracy 的折線圖。你的圖上應該會有兩條以上的折線。(2%)

這題我用了 0.95,0.9,0.85 三種 pruning rate，每次 pruning 完有再 finetune 兩個 epoch。下圖是對助教提供的 student_custom_small.bin 做 pruning 所得到的結果：