學號：b05901070　系級：電機四　　姓名：蔡昌廷

1. 請從 Network Pruning/Quantization/Knowledge Distillation/Low Rank Approximation 選擇兩個方法(並詳述)，將同一個大 model 壓縮至同等數量級，並討論其 accuracy 的變化。 (2%)

   此題我分別實作 Knowledge Distillation 跟 Low Rank Approximation 的方式來檢測不同方法壓縮後的差別，以下兩者的大 model 皆是 hw3 自己的 model，Accuracy 為 87.029%，接著為方法說明。

   Knowledge Distillation：
   此處的大 model 我是使用 hw3 自己的 model，小的 model 是自己建一個參數較小的 model，參數總數為 540683，透過 KD 的方法，loss 計算方式依據比例 0.2 參考與大 model output 的 loss 做 KL Divergence、0.8 根據 true label 計算出的 cross entropy，接著用 100 epoch 去實測，learning rate 為 0.001。

   Low Rank Approximation：
   此處我將 hw3 的 model CNN 的部分改成使用 depthwise 與 pointwise 的方式來減少參數量，此外為了盡量維持跟 KD 方法一樣的參數量，我些微調整 model 後半部 fully connected 部分的 neuron 個數，最後參數量為 540683，最後也是直接用 100 epoch 去實測，learning rate 為 0.001。

   以下為實測的結果

	Training Accuracy	Validation Accuracy
Knowledge Distillation	84.35 %	72.71 %
Low Rank Approximation	79.65 %	74.31 %

   從以上結果發現 KD 在 training 上會比 Low Rank Approximation 更快達到好的效果，我認為這部分是因為學習到大 model 的知識，原本大 model 在 training data 就表現的不錯，因此透過 KD 的方式可以使 training 在同樣 epoch 更快達到好的效果。至於在 validation 上則是 Low Rank Approximation 的效果較好，我想是因為整體來說 Low Rank Approximation 是根據大 model 的架構去調整，因此大 model performance 不差的情況下，其 validation 的表現也會趨近於原 model 的結果。

以下三題只需要選擇兩者即可，分數取最高的兩個。

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. 以下為 student net 的架構與參數量，參數量為 256779

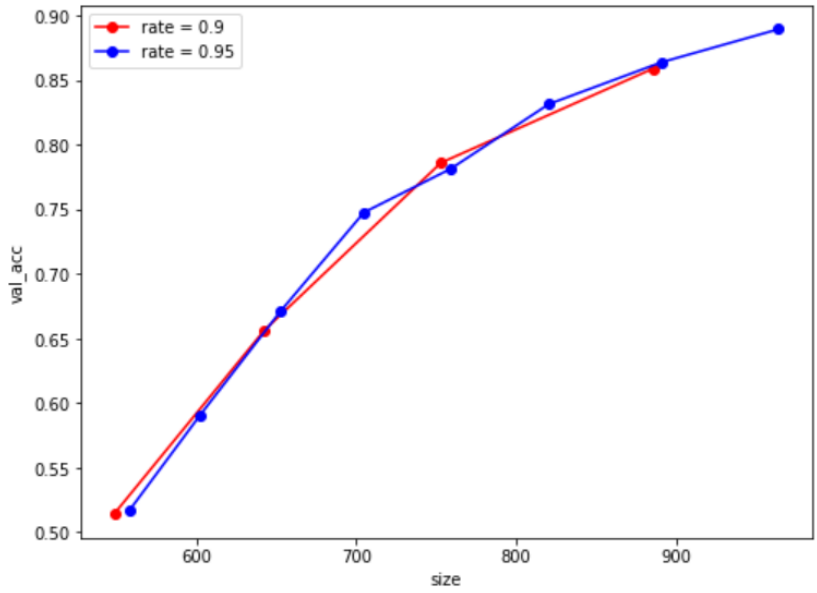
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 256, 256]             448
       BatchNorm2d-2         [-1, 16, 256, 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-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
          Dropout-37                  [-1, 256]               0
           Linear-38                   [-1, 11]           2,827
================================================================
Total params: 256,779
Trainable params: 256,779
Non-trainable params: 0
----------------------------------------------------------------
```

以下 validation accuracy 皆以第 100 epoch 為準

c. My student net from scratch： 71.78 %
d. My student net KD from (a.)：76.41 %
e. My student net KD from (b.)：79.59 %

從以上結果可以看出 student net 在沒有經過 KD 而是直接 train 的情況下，表現都比有使用 KD 的結果還要差，即使是多跑一些 epoch，accuracy 大約還是在 71、72 左右，由此可以看出使用知識蒸餾讓小 model 去學習大 model 的知識是有效提升小 model performance 的方法，透過知識蒸餾可以讓小 model 學到不同 label 之間相似的程度，更能避免分類錯誤的發生。此外使用 performance 較好的大 model 當作 teacher 的話，其 student net 的 performance 會表現的較優秀，因此在做知識蒸餾時，選取好的 teacher 也是重要的一部分。

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%)



從上圖可以發現即使用不同的 pruning rate 去 prune model，其最後對 validation 的結果似乎影響不太大，自己也實驗很多次也嘗試過不同的 rate，但結果大致不會差太多，我認為在 pruning 完之後再對小 model 重新訓練一次 weight 的過程極為重要，能夠使得小 model 的表現不會一下就表現太差，可能也是這個原因導致這次實作的結果，至於越 prune 越多後，其 accuracy 會越來越低也是合理的現象，畢竟從原本 train 好的 model 拿掉部分的 neuron 本來就不太可能維持原本的 performance。

4. [Low Rank Approx / Model Architecture] 請嘗試比較以下 validation accuracy，並且模型大小須接近 1 MB。 (2%)

a. 原始 CNN model (用一般的 Convolution Layer) 的 accuracy

b. 將 CNN model 的 Convolution Layer 換成參數量接近的 Depthwise & Pointwise 後的 accuracy
c. 將 CNN model 的 Convolution Layer 換成參數量接近的 Group Convolution Layer (Group 數量自訂，但不要設為 1 或 in_filters)