學號：B06902042　系級：資工三　姓名：劉愷為

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%)

   大 model : 我用助教提供的 teacher_resnet18.bin

   參數總量 : 11182155，在本機測試的 val_acc 為 : 88.12%

   a. Knowledge Distillation：重構一個小 model，並讓他去學習大 model。我使用 colab 中 KD 和 Architecture Design 去 train

   Parameters: 257863，本機的 val_acc: 85.63%

   b. Low Rank Approximation：將大 model 中的 convolution layer 換成 DW&PW。

   將 nn.Conv2d 寫成

   ```
   # Depthwise Convolution
   nn.Conv2d(in_chs, out_chs, kernel_size, stride, padding, groups=in_chs)
   # Pointwise Convolution
   nn.Conv2d(in_chs, out_chs, 1)
   ```

   其中為配合參數量級，在其中兩個 convolution layer 只做一層 DW&PW。

   Parameters: 289694，本機的 val_acc: 75.73%

   c. ：從 val_acc 的變化可發現，Knowledge Distillation 的表現明顯優於 Low Rank Approximation，推測我所實作的 Knowledge Distillation 有包含 DW+PW 的優點還有 Knowledge Distillation 的優點，因此 validation accuracy 有較好的判斷結果

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: 使用 colab 上的 architecture

```
StudentNet(
  (cnn): Sequential(
    (0): Sequential(
      (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=64)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (2): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=64)
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): Conv2d(64, 128, kernel_size=(1, 1), stride=(1, 1))
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (3): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=128)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): Conv2d(128, 128, kernel_size=(1, 1), stride=(1, 1))
      (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (4): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=128)
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): Conv2d(128, 256, kernel_size=(1, 1), stride=(1, 1))
    )
    (5): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=256)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    )
    (6): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=256)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    )
    (7): Sequential(
      (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=256)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU6()
      (3): Conv2d(256, 256, kernel_size=(1, 1), stride=(1, 1))
    )
    (8): AdaptiveAvgPool2d(output_size=(1, 1))
  )
  (fc): Sequential(
    (0): Linear(in_features=256, out_features=11, bias=True)
  )
)
start training, parameter total:277963
```

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

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

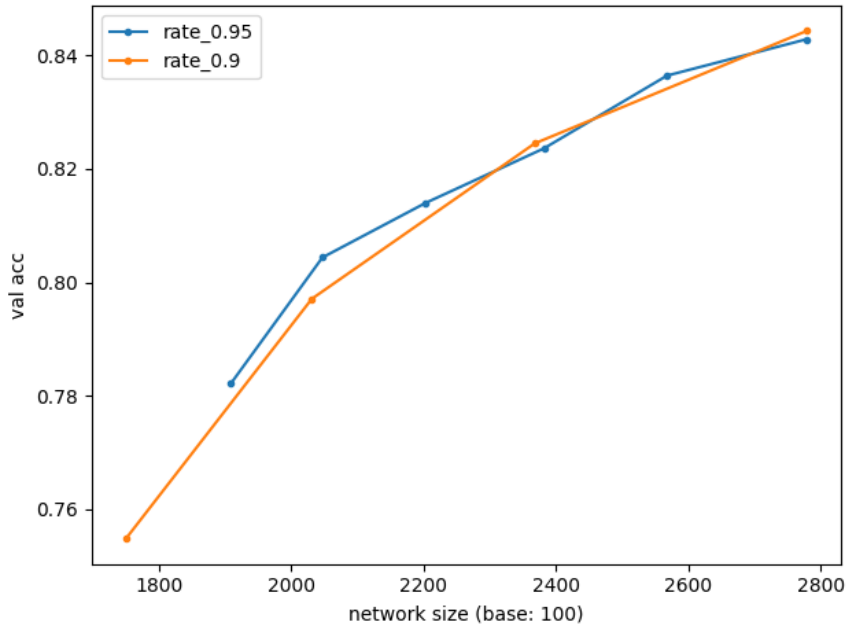
c. Your student net from scratch: 77.45%

d. Your student net KD from (a.): 80.56%

e. Your student net KD from (b.): 83.16%

從上面各個模型在 validation set 上的準確率，可以看出沒有經過 KD 的 Student net 準確率最低。我猜測可能的原因是 Teacher net 不只告訴學生正確答案，還會把哪些類別有比較接近的關係告訴學生，所以學生學習成果會比較好；從後兩個 event 我們可以看到 KD from (b)的學生準確率較高，猜測原因是經過 ImageNet pretrained 以及微調參數的 Teacher net 給出的機率分布較正確， 可想而知學生會學的比較好。

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%)



這個圖為 pruning rate = 0.95 and 0.9 分別 iteration 6, 4, 2 次的結果。

經由上圖可以發現，到最後兩條曲線的變化幅度幾乎一樣，而在 pruning rate 較接近 1 時，size 的變化對於 accuracy 的影響並不大。隨著 model 的 size 越來越小，三條 accuracy 的變化幅度及差異越來越大，下降的速度也變快了。