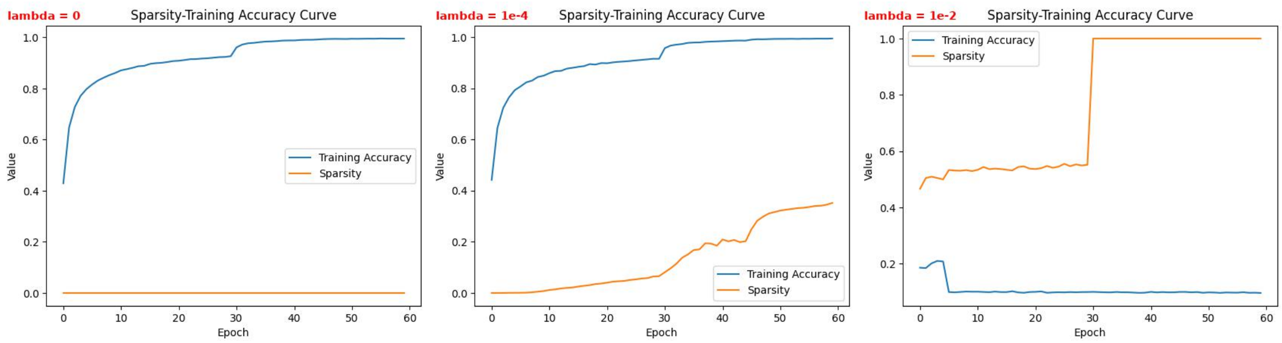
EAI lab 4

Model Pruning

數據所 RE6121011 徐仁瓏

1. Sparsity-Epoch & Training Accuracy-Epoch Plot

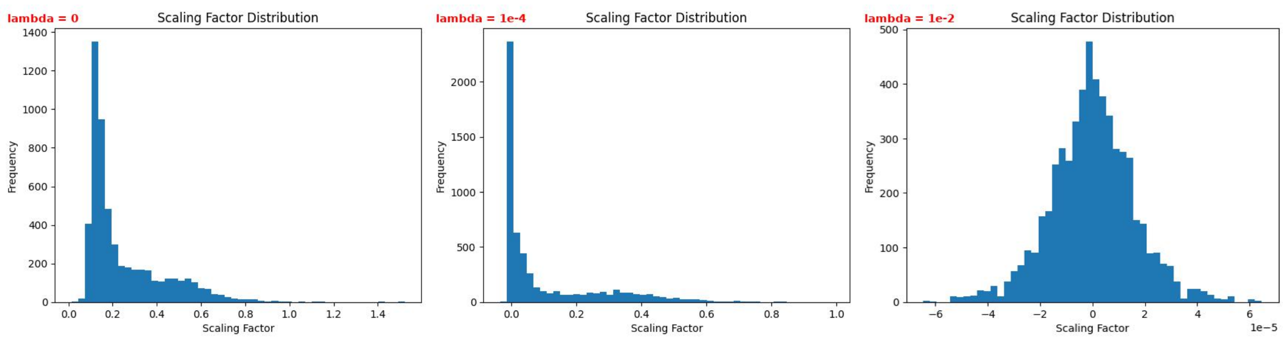


從左圖至右圖分別為當lambda值為0、0.0001、0.01時的設定，藍線為Training Accuracy隨epoch數增加的變化，橘線為Sparsity程度隨epoch數增加的變化。

- 當lambda值為0時，表示不進行L1-regularization，可以發現Sparsity（橘線）幾乎為0，因為我們沒有限制模型要進行正則化，所以不會將模型權重壓縮至靠近0的位置。
- 當lambda值為0.0001時，對模型進行些微的正則化，模型有越來越Sparsity的趨勢。
- 當lambda值為0.01時，可能這個lambda值有點太大了，導致模型的Sparsity程度（橘線）達到100%，也造成Training Accuracy（藍線）非常低，模型訓練不起來的情況。

所以從上述可知，當lambda值越低，模型越不Sparsity；當lambda值越高，模型越Sparsity。

2. Scaling Factor Distribution



從左圖至右圖分別為當lambda值為0、0.0001、0.01時Scaling Factor的分佈。

- 當lambda值為0時，這是模型沒有進行任何正則化時，Scaling Factor的原始分佈。
- 當lambda值為0.0001時，Scaling Factor的分佈有越來越集中於0的趨勢。

- 當lambda值為0.01時，Scaling Factor的分佈幾乎完全在0附近（注意：此時Scaling Factor的單位為1e-5）。

所以從上述可知，當lambda值越高時，模型越往0集中，即表示模型越Sparsity，與第一題的結論相同。

3. Pruning 50% channels

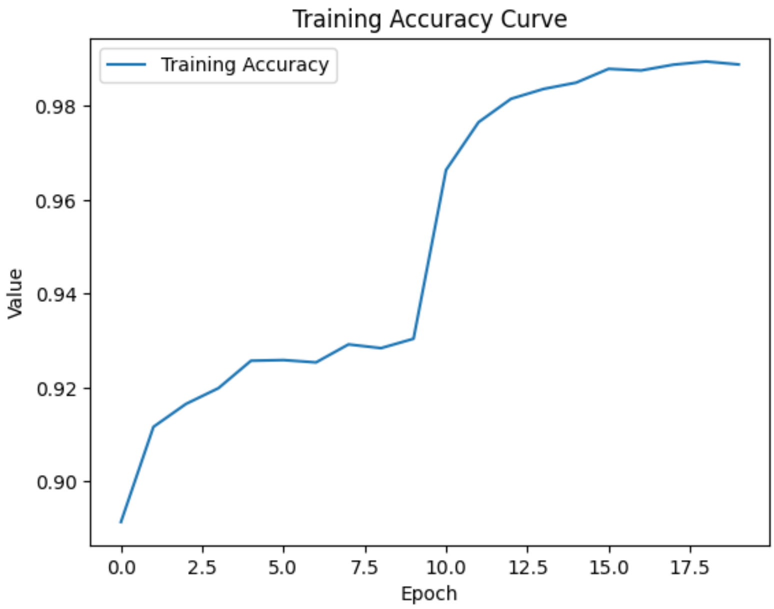
- Prune完fine-tune前：
    - Test Accuracy為10%。

```
vgg(
  (feature): Sequential(
    (0): Conv2d(3, 51, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(51, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(51, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (8): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (11): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): ReLU(inplace=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (14): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (15): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): Conv2d(256, 255, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (18): BatchNorm2d(255, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): ReLU(inplace=True)
    (20): Conv2d(255, 255, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (21): BatchNorm2d(255, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): ReLU(inplace=True)
...
Files already downloaded and verified

Test set: Accuracy: 1000/10000 (10.0%)

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

tensor(0.1000)
```

- Prune完fine-tune後
    - Test Accuracy為92.6%。

```
Train Epoch: 19 [0/50000 (0.0%)]        Loss: 0.066163
Train Epoch: 19 [10000/50000 (20.0%)]   Loss: 0.007639
Train Epoch: 19 [20000/50000 (40.0%)]   Loss: 0.024447
Train Epoch: 19 [30000/50000 (60.0%)]   Loss: 0.003823
Train Epoch: 19 [40000/50000 (80.0%)]   Loss: 0.046713
Train Accuracy for Epoch 19: 98.92%

Test set: Average loss: 0.2777, Accuracy: 9261/10000 (92.6%)
```

- Training Accuracy Plot



4. Pruning 90% channels

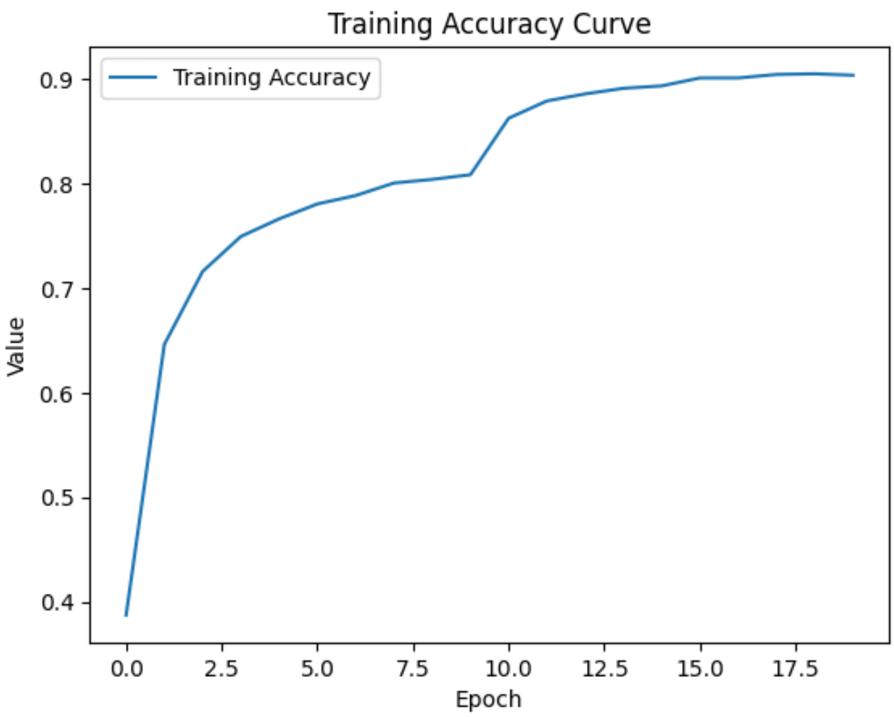
- Prune完fine-tune前：
  - Test Accuracy為10%。

```
vgg(
  (feature): Sequential(
    (0): Conv2d(3, 27, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(27, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(27, 63, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (4): BatchNorm2d(63, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (7): Conv2d(63, 89, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (8): BatchNorm2d(89, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (9): ReLU(inplace=True)
    (10): Conv2d(89, 124, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (11): BatchNorm2d(124, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (12): ReLU(inplace=True)
    (13): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (14): Conv2d(124, 123, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (15): BatchNorm2d(123, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (16): ReLU(inplace=True)
    (17): Conv2d(123, 60, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (18): BatchNorm2d(60, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): ReLU(inplace=True)
    (20): Conv2d(60, 11, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (21): BatchNorm2d(11, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (22): ReLU(inplace=True)
...
    (51): ReLU(inplace=True)
  )
  (classifier): Linear(in_features=53, out_features=10, bias=True)
)
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
Files already downloaded and verified

Test set: Accuracy: 1000/10000 (10.0%)
```

- Prune完fine-tune後：
  - Test Accuracy為87.1%。

```
Train Epoch: 19 [0/50000 (0.0%)]        Loss: 0.226688
Train Epoch: 19 [10000/50000 (20.0%)]   Loss: 0.418237
Train Epoch: 19 [20000/50000 (40.0%)]   Loss: 0.256122
Train Epoch: 19 [30000/50000 (60.0%)]   Loss: 0.262042
Train Epoch: 19 [40000/50000 (80.0%)]   Loss: 0.317818
Train Accuracy for Epoch 19: 90.40%

Test set: Average loss: 0.4319, Accuracy: 8709/10000 (87.1%)


 TRAIN PRUNED MODEL DONE!
```

  - Training Accuracy Plot



5. Problem

一開始不太清楚三個程式碼代表什麼意思，不太了解開怎麼著手進行，後來再深入觀察後慢慢了解到，本次作業是想讓我們先使用sparsity_train.ipynb檔來將VGG模型變得較Sparsity，之後再透過vggprune.ipynb檔將較不重要的channel刪除，得到pruning過後的model，最後再使用train_prune_model.ipynb將model重新train過一次，讓他得到與原始沒剪枝模型一樣高準確度的模型。

此外，還有一個困難點是做Sparsity Regularization的部分，需要計算梯度，一開始看不太懂該怎麼計算這部分的梯度，後來才了解到計算絕對值的梯度即是直接取他的正負號，了解到這點後才能完成這部分的梯度計算。

最後應該是對於cfg和cfg_mask變數的理解，再和chatGPT協作後，並仔細印出這些程式碼的變數後，才慢慢知道它的意義。而理解後還需要思考他的0和1的意思，是保留通道還是刪除通道，這些皆需要理解過後才可以寫出程式碼，我覺得這部分是我認為最難的部分。