Deep learning hw4

109610025 陳品妍

Task 1:

```
Saving..
Training: 100%|████████████████| 65/65 [00:03<00:00, 18.12it/s]
Testing: 100%|████████████████| 10/10 [00:00<00:00, 17.12it/s]
05 Nov 2024 06:59:20
Epoch:  17 |train loss: 0.2112 |train accuracy: 94.74 |validation loss: 0.5217 |validation accuracy: 83.12 |learning rate: 1.0e-03 |train time: 3.59 |te
Training: 100%|████████████████| 65/65 [00:03<00:00, 18.10it/s]
Testing: 100%|████████████████| 10/10 [00:00<00:00, 17.60it/s]
05 Nov 2024 06:59:24
Epoch:  18 |train loss: 0.1997 |train accuracy: 93.77 |validation loss: 0.6855 |validation accuracy: 81.25 |learning rate: 1.0e-03 |train time: 3.60 |te
Training: 100%|████████████████| 65/65 [00:04<00:00, 15.15it/s]
Testing: 100%|████████████████| 10/10 [00:00<00:00, 12.99it/s]05 Nov 2024 06:59:29
Epoch:  19 |train loss: 0.2001 |train accuracy: 95.13 |validation loss: 0.5022 |validation accuracy: 81.88 |learning rate: 1.0e-03 |train time: 4.30 |te
Best accuracy: 85.62
```

只有 task1 是用 colab，後來轉到 server 上

Best accuracy : 85.62

調整包括 batch_size 以及 lr。發現 batch_size 下降可導致 accuracy 上升，猜測應該是梯度更新時，可以跳出一些 local minimum(小 batchsize 可以產生更大的梯度 difference)。且 batch size 下降，整體更新的頻率上升，可以有更多次梯度更新，使模型可以更加適應資料。

Task2:

w/ prune_rate = 0.15

```
After fine-tune
Validation loss: 0.4588 Validation accuracy: 84.38
```

w/ prune_rate = 0.1

```
After fine-tune
Validation loss: 0.5983 Validation accuracy: 85.00
```

w/ prune_rate = 0.05

```
After fine-tune
Validation loss: 0.4882 Validation accuracy: 85.62
```

```
(features): ModuleList(
  (0): _Layer(
    (conv0): Conv1d(20, 20, kernel_size=(25,), stride=(2,), groups=20)
    (conv1): Conv1d(20, 133, kernel_size=(1,), stride=(1,))
    (relu): ReLU()
    (bn): BatchNorm1d(133, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (pool): AvgPool1d(kernel_size=(2,), stride=(2,), padding=(0,))
  )
```

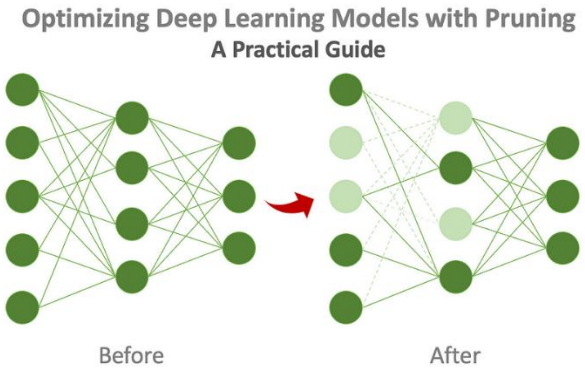
Task 3:

```
05 Nov 2024 21:08:30
Epoch: 246 |train loss: 0.3719 |train accuracy: 90.26 |validation loss: 0.8947 |validation accuracy: 70.00 |learning rate: 3.9e-05 |tra
in time: 1.40 |test time: 0.18
05 Nov 2024 21:08:32
Epoch: 247 |train loss: 0.2909 |train accuracy: 91.33 |validation loss: 0.8995 |validation accuracy: 71.25 |learning rate: 2.0e-05 |tra
in time: 1.36 |test time: 0.20
05 Nov 2024 21:08:33
Epoch: 248 |train loss: 0.4266 |train accuracy: 90.17 |validation loss: 0.9498 |validation accuracy: 70.00 |learning rate: 2.0e-05 |tra
in time: 1.35 |test time: 0.19
05 Nov 2024 21:08:35
Epoch: 249 |train loss: 0.3917 |train accuracy: 90.46 |validation loss: 0.9821 |validation accuracy: 72.50 |learning rate: 2.0e-05 |tra
in time: 1.51 |test time: 0.19
Best accuracy: 73.75
```

Best accuracy:73.75

Briefly explain pruning:

What&why : Pruning 是一種用在深度學習的技巧，用途在於優化 nn model，使它更小也更有效率。透過減少 values of weight tensors，可以使在訓練模型以及計算中的時間下降，也可以節省運算資源(尤其式多數 nn 的運算涵蓋到很多浮點數的乘法運算，浮點數 32bit 需要用 4byte，而整數只需要 1byte)。例如，CNN 的演算法架構往往很複雜且參數量巨大。



圖片來源: https://medium.com/@jan_marcel_kezmann/optimizing-deep-learning-models-with-pruning-a-practical-guide-163e990c02af

有趣的是，Pruning 的發想來自於人類的神經發展，在人類青少年時期時，腦神經會開始進行大量修剪，將較少用到的神經切掉，以將能量集中以及強化在常用到的神經路徑。

How:

1. Iterative pruning(pruning scheduling): pruning 類似於 regularization(但 regularization 意在避免模型 overfitting)，其結果會使 accuracy 下降，所以在實驗的過程當中，也要避免將 pruning_rate 設為太大，導致 network 被破壞到無可恢復的狀態。應該要先從小的 pruning_rate 開始 iterate 再進行 fine-tuning。在論文<Structured Pruning for Deep Neural Networks with Adaptive Pruning Rate Derivation Based on Con>當中有提到早期 one-shot pruning 的方法對 model 進行一次高 pruning rate 的 pruning，但研究證實 iterative pruning 比較不會使 accuracy 大量下降，並且可以得到每一層的 pruning rate 基於該層的梯度變化來推導 pruning rate。
2. Weight pruning:

將不重要的權重丟棄，例如:$f(x) = x + 5x^2$，當我們改變係數 1 的時候，y 只會發生較小變化，相較於 5 來說。因此我們可以選擇將 1 丟棄。而在 nn 中，定義權重的重要性則跟 loss function 有關。例如，在更新梯度變化的過程當中，可能某些參數相較於其他參數有更大幅度的量去進行更新，該 weight 則可以被視為相對重要的參數。權重 pruning 的一般流程為：

1. 根據權重大小進行排序。
2. 推導出合適的剪枝率（此處可參考相關文獻的具體方法）。
3. 設定閾值，將小於閾值的權重設為零。

通常來說，閥值可以設定為整個網路抑或者是該層 nn 的最小梯度差。

3. Sparsity Matrix：在 pruning 過程中，許多權重被設置為零，逐漸形成稀疏矩陣。稀疏矩陣主要包含零值，這些零值表示對模型輸出影響較小的權重。剪枝後的稀疏矩陣既保持模型結構完整，又大幅降低計算需求。其優點包括內存節省。稀疏矩陣僅需存儲非零元素及其索引，可以顯著減少模型的內存佔用，特別是對於大型網絡。另外，還可以使模型計算加速，支援稀疏運算的硬體（如 TPU 和一些 GPU）可以利用稀疏矩陣的特性加速推理，節省能耗。可應用在嵌入式設備和資源有限的環境。然而，並非所有硬體都支援稀疏矩陣運算，可能在特定環境中無法顯著加速。精度維護：高稀疏度可能造成精度下降，因此需要合理設計 pruning 策略來控制稀疏度。

Briefly explain quantization:

What & why: Quantization 是將模型中的權重和激活值從浮點數（如 FP32）降低到較低位元的整數（如 INT8）的技術。浮點數運算需要較大的儲存空間和運算資源，而量化可以顯著減少記憶體佔用、降低功耗並提升 forward 速度。這對於資源有限的設備（如嵌入式裝置）特別重要，因為這類裝置的運算能力和電力都十分有限。儘管浮點數能提供更大的範圍和更高的精度，但在許多應用中，模型轉換為整數格式後依然能夠維持可接受的性能。這樣的轉換會導致一定的精度損失，但透過有效的技術可以將這些精度損失降到最小。

How:

Quantization 包括 Post-training Quantization 和 quantization aware training，目標在於確保 quantizaiton 後依然保持 accuracy(誤差可能會隨層數增加而增加)。

Post-training quantization: 將訓練好的 weights 轉成整數，這種方法較快速且不用改變訓練過程，然而卻會導致某些 model 的 accuracy 下降。在論文中提到結果大部分 model 進行 post-training quantization 的 accuracy 有稍微下降，但每個模型適合 calibration 方法(量化範圍)不盡相同。

Partial quantization: 在論文中有提到一種有別於 ptq 的量化方法，旨在平衡準確率以及效率，避免全部層量化導致 accuracy 下降太大。Partial quantization 一次量化一層，然後算出正確率，在排列 layers 對 model 影響程度，並忽略影響大

的 layers 或者是保留浮點數的表示，專注量化於其他 Layers，可有效減少損失。

Quantization aware training: 先用 floating point 訓練模型後再用 integer weight 用整數來 fine-tune，以量化效果進行模擬，使模型在訓練中可以自我調整。值得注意的是，quantize operation(例如 rounding)在某些數值上無法為分，所以會舞法使用 back propagation。為了解決，qat 引入 straight through error，使範圍間的微分倒數為 1，其餘為 0。這個方的運算較為複雜且訓練成本高但是可以達到比較好的 accuracy。

Compare the difference from above methods of compression:

	quantization	pruning
目的	減少 floating point 來減少占用空間	移除較為不重要的參數
種類	PTQ, QAT 等	非結構化:不考慮網路中的位置，刪除不重要的單個 weight 結構化: 刪除整個通道，層級單元
誤差比較(在相同壓縮率下)	較小	較大
性能	量化後的 model 優於 pruning 模型	在高壓縮率下，pruning 可能有更好性能，但需要 fine-tuning
應用	小型裝置、低功率、對精確度要求低的應用	大參數模型，例如 CNN LSTM
優點	高效能、節省功耗、快速	減少模型大小、減少硬體資源
缺點	準確度下降需要仰賴校正或訓練來維持性能	• Pruning_rate 需要精心計算，避免破壞模型結構 • 需要空間來儲存 sparsity matrix 的 index message，導致額外開銷(硬體需求高)
可組合性	可跟 pruning 結合使用	可跟 quantization 結合

Reference:

论文解读《Structured Pruning for Deep Neural Networks with Adaptive Pruning

Rate Derivation Based on Con》 journal of advances in information technology-CSDN 博客

https://arxiv.org/pdf/1710.09282.pdf

[2004.09602] Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation (arxiv.org)

https://arxiv.org/abs/2307.02973