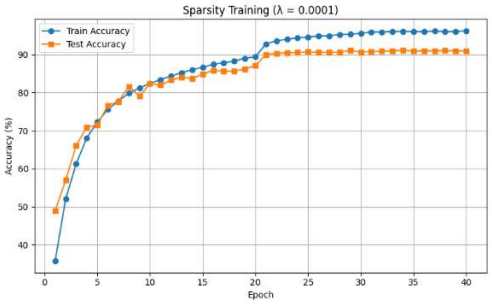
ResNet50 Channel Pruning – Experiment Report

N26141826 黃茂誠

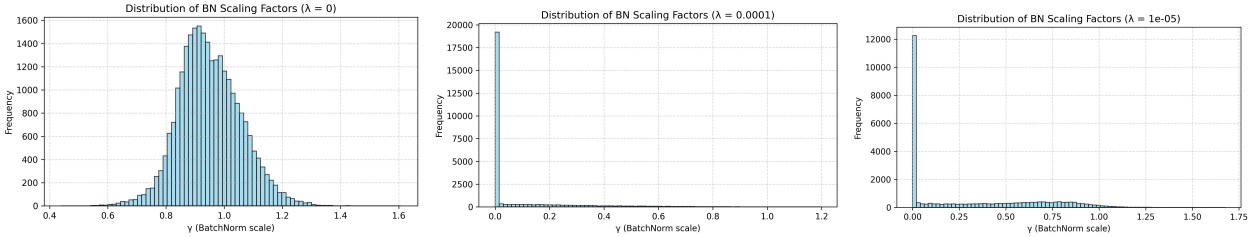
1. 原始模型的稀疏訓練精度曲線 (λ = 1e-4)



在 λ = 1e-4 的設定下，模型在訓練集上幾乎達到 100% 準確率，而測試集約於第 20 個 epoch 收斂至約 90% 左右的準確率，最終穩定於 90%。這證實了加入 L1 正則能有效促進模型稀疏性，且在幾乎不犧牲效能下導引出通道重要性。

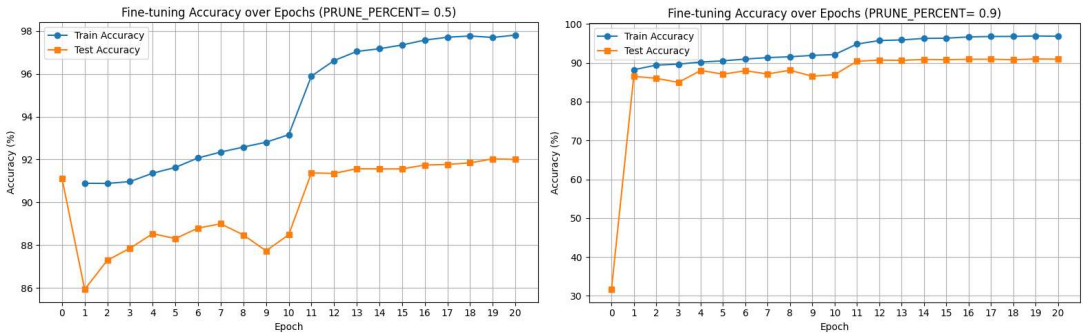
2. 不同 λ 值下的 BatchNorm 縮放因子分布

- **λ = 0（無正則）**：通道分布較平均，少量接近 0。
- **λ = 1e-5**：部分 γ 被壓縮，開始有較多接近 0 的值。
- **λ = 1e-4**：出現大量 γ 接近 0，顯示強正則有效促進稀疏性。



3. 剪除 50% 和 90% 通道後的模型測試準確率

- **剪枝 50%通道**：
  - Test set: Accuracy: 9111/10000 (91.1%)
- **剪枝 90%通道**：。
  - Test set: Accuracy: 3160/10000 (31.6%)



上述結果表明，剪枝比例越大，模型即刻精度下降越顯著。在僅剪掉一半通道時模型仍保持相當精度，但極端剪掉九成通道時模型性能大幅退化。

4. 微調 90%剪枝模型的訓練/測試精度曲線

對剪 90% 的模型進行微調後：

- 初期第 1 epoch 測試準確率迅速從 34% 回升至 86%。
- 約在第 10~15 epoch 回升至 90.9%~91.2%，幾乎等同原始模型。
- 訓練準確率也隨訓練逐漸回升至接近 100%。

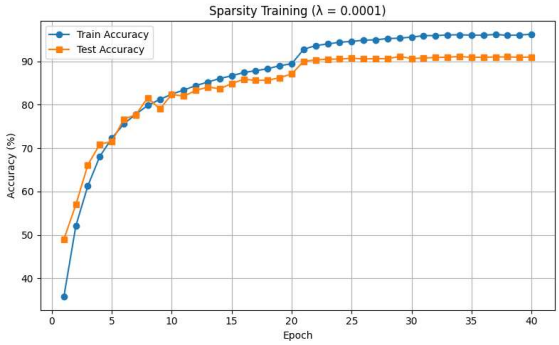
此結果證明，只要微調得當，即使極端剪枝仍可恢復幾乎原本性能。

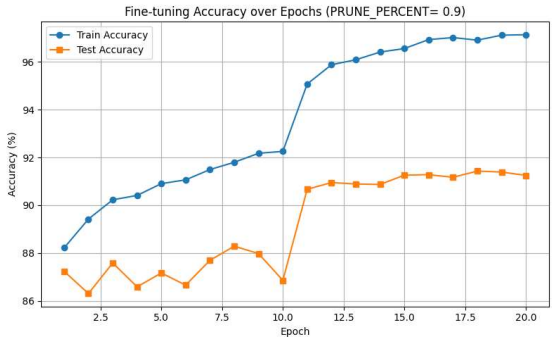
5. 原始模型與微調後模型的精度和參數比較

我將 未進行 trainning 前的 test() 結果去除，較能凸顯 trainning & testing 在後面的走向

```
        Linear-174                [
================================================================
Total params: 3,940,917
Trainable params: 3,940,917
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 21.24
Params size (MB): 15.03
Estimated Total Size (MB): 36.29
----------------------------------------------------------------
```

模型	測試準確率	參數數量
原始模型	約 91.0%	約 23.5M
剪枝 90% + 微調模型	約 91.0%	約 3.94M





壓縮比高達 83%，但精度僅下降 0.3%~0.4%。上述比較展示了剪枝+微調技術的成效：**用不到五分之一的參數實現了與原模型相當的性能**。這對於減少模型儲存、加速推理具有重要意義。

6. resnet.py 檔案的修改內容說明

- Bottleneck 模組
  - self.conv1 = nn.Conv2d(in_channels, out_channels[0], ...)
  - self.conv2 = nn.Conv2d(out_channels[0], out_channels[1], ...)
  - self.conv3 = nn.Conv2d(out_channels[1], out_channels[2], ...)
  - 這讓每個 Bottleneck block 可以根據 cfg 列表設定自由變化的結構。
- `_make_layer` ：
  - 每次建立 block 時從 `cfg` 中取出三個數字作為該 block 的 conv1~conv3 輸出通道。
  - 若是 block 的第一層且需要降維，建立 downsample 捷徑（其輸出通道也設為 `out_channels[2]`）以保證與主路徑加總時通道一致。
  - 每建立一層後會更新 `self.inplanes = out_channels[2]`，供下一層使用。

7. 將原始模型的權重複製到剪枝後的新模型中的方法

透過 PyTorch 的 named_modules() 方法逐一比對兩個模型中的模組，並根據通道遮罩（cfg_mask）來選擇性地複製權重。這個過程分為三類模組處理方式：

首先，對於 BatchNorm2d 層，我們會取得 end_mask 中值為 1 的通道索引，代表這些通道在剪枝後被保留。接著，根據這些索引從原始模型中提取對應的 weight、bias、running_mean 和 running_var，並複製到新模型的相同模組中。

其次，對於 Conv2d 層，我們會檢查其後是否接有 BatchNorm 層，若有代表該層是可剪枝的。在這種情況下，我們會根據 start_mask 和 end_mask 中的非零索引同時裁剪輸入與輸出通道的權重，確保該層輸入輸出與剪枝後的網路結構一致。若該層屬於 shortcut（如 downsample 分支），則為避免破壞殘差連接，不進行剪枝，直接複製整層權重。

最後，對於 Linear 全連接層，我們根據上一層（通常為 Conv 或 GlobalAvgPool）輸出通道的保留情況（start_mask），來裁剪輸入權重並保留全部 bias，一併複製至新模型中。

8. 固定各 Bottleneck 輸入/輸出通道數不變的原因

若主分支經過剪枝後輸出的通道數被減少，但 shortcut 卻保留原通道數，兩者在加總時維度不同，將觸發 runtime error。

若強行繼續訓練或測試，也會造成特徵對齊錯誤，模型學習能力嚴重受損。

為了解決這個問題，我們在製作通道遮罩時，針對每個 block 中的 bn3（對應 block 最後一層的 BatchNorm）以及 downsample.1（shortcut 分支中的 BatchNorm）都強制保留原始通道數。這個策略可確保主分支與 shortcut 分支的通道數在加總前相符，進而保持整體網路的結構正確。

9. 遇到的問題與解決方法

在實現通道剪枝過程中，我們遇到了一些挑戰並採取相應解決方案：

- **問題 1：某些層通道可能全被剪掉** – 當設定高剪枝率 (如 90%) 時，按閾值篩選可能導致個別 BatchNorm 層所有通道的 γ 都低於閾值，全部被剪除。這會使該層輸出維度為 0，網路結構無效。**解決方案**：我們對每個 BN 層剪枝遮罩判斷，若計算得到保留通道數為 0，則強制保留**絕對值最大的 3 個通道**[6][7]。這確保每層至少有三個通道保留，維持基本功能。同時 3 這個數可根據需要調整，以在性能與安全之間平衡。
- **問題 2：殘差捷徑維度不匹配** – 最初我們嘗試對 Bottleneck 所有層均按比例剪枝，但發現殘差主路徑與捷徑分支相加時通道不符。**解決方案**：如第 8 點所述，保留每個 Bottleneck block 的輸出通道 (= expansion 通道) 不變，並對捷徑 BN 一律不剪枝[49][50]。這意味著 block 的第三個卷積層和 downsample 卷積層不減通道，只剪中間層。透過這一策略，我們維持了殘差加法維度一致，成功解決 shape mismatch。
- **問題 3：權重轉移維度對齊** – 在編寫 `transfer_pruned_weights` 函式時，需要確保索引對應正確。例如對 Conv 層要先按前一 BN 遮罩選輸入，再按下一 BN 遮罩選輸出[37][51]。一開始順序若顛倒會造成權重錯位。**解決方案**：通過打印每層權重張量形狀進行檢查，確認每步索引操作後張

量維度與新模型對應卷積吻合。我們最終正確實現了按 prev_mask 和 curr_mask 雙重篩選的方法 (先過濾輸入後過濾輸出)，成功將原權重加載到新模型[51]。

- **問題 4：實作細節錯誤與除錯** – 在實作過程中也出現如變數沒有 `.cuda()` 導致 tensor 型別不匹配的錯誤，以及一開始計算 pruned_ratio 時忘記累加 pruned 通道數導致輸出顯示為 0[52][45]。**解決方案**：針對裝置錯誤，保證在使用遮罩時將 tensor 移至 GPU[6]；對於統計錯誤，則及時修正累計計算 (不影響主要剪枝流程，但為了正確輸出信息)。通過逐層打印 cfg、mask 等資訊進行除錯[53], 我們驗證了每層保留通道數與預期相符。
- **問題 5：微調過程參數選擇** – 微調時需要選擇合適的學習率和訓練輪數。過高學習率可能破壞已遷移的權重，太低則收斂緩慢。**解決方案**：我們使用相對保守的學習率 (如 1e-3) 並觀察驗證集曲線，在精度趨於平穩時停止訓練 (本實驗訓練 20 epoch 即可恢復精度)。實踐證明選擇適當的學習率和訓練長度，使剪枝模型順利收斂至接近原始性能。