# 實驗一

## 前處理

### 一、圖片轉換

這邊針對圖片做一些差異化處理，以下是其處理細節:

1. `transforms.RandomResizedCrop(224)`: 這個轉換隨機對圖像進行裁剪和縮放，使圖像的大小變為 224x224 像素，幫助模型更好地適應不同尺寸的輸入圖像。

2. `transforms.RandomHorizo​​ntalFlip()`:隨機水平翻轉的轉換，以增加數據的多樣性。

3. `transforms.RandomRotation(30)`: 這個轉換隨機旋轉圖像，最多旋轉 30 度。

4. `transforms.ToTensor()`: 這個轉換將圖像轉換為 PyTorch 張量格式。

5. `transforms.Normalize(mean=train\_mean, std=train\_std)`: 這個轉換對圖像進行正則化，通過減去平均值（mean）並除以標準差（std）。這是我計算自訓練集。

### 二、設置分割

Batch 設置為 64

## 模型設計

### 一、初始特徵提取

這裡首先定義了一個初始的卷積層，這個卷積層對輸入影像進行處理。具體來說，它有以下特點：

1. `nn.Conv2d(3, 32, kernel\_size=7, stride=2, padding=3)`：這是一個卷積層，輸入通道數為3（RGB 彩色影像），輸出通道數為32，卷積核大小為7x7，步幅（stride）為2，填充（padding）為3。這個卷積層負責對輸入影像進行初步特徵提取，並降低影像的尺寸。
2. `nn.BatchNorm2d(32)`：接著是一個批量標準化層，用來穩定神經網路的訓練，防止梯度消失或爆炸的問題。
3. `nn.RELU()`：採用 RELU 激活函數，用來引入非線性性質到網絡中。

### 二、IBConv Blocks

接下來，定義了一系列的Inverted Bottleneck Convolution Blocks。我是閱讀自資料參考一。每個區塊有不同的輸入和輸出通道數，以及卷積核的設定。這些區塊可以用於提取更高級的特徵。

### 三、Fully connected layers

在卷積部分之後，定義了幾個全連接層，這些層通常用於將卷積層的輸出映射到最終的輸出分類。具體來說，這裡有以下層次：

1. `nn.Linear(80, 512)`：一個全連接層，將卷積部分的輸出映射到512維的特徵向量。
2. `nn.RELU()`：再次使用RELU激活函數。
3. `nn.BatchNorm1d(512)`：批量標準化層，用於穩定訓練過程。
4. `nn.Dropout()`：Dropout層，用於隨機丟棄部分神經元，以防止過度擬合，機率設置為 0.5。

### 四、Output layer

最後，使用一個全連接層 `nn.Linear(512, 5)` 作為輸出層，輸出5個類別的分類結果。

### 五、訓練方法

1. 增加 `early\_stopping\_patience` 是用來設定早停機制的參數，即在連續20個輪次中如果驗證損失不再改善就提前停止訓練。
2. 使用 Adam 優化法，設置權重衰減0.0001。
3. 學習率設置為0.001

### 六、標記未標註資料加強準確度

1. 將偽標籤設置需要0.85 臨界點來標記
2. 使用 Adam 優化法，設置權重衰減0.001。
3. 學習率設置為0.0001

## 總結：

1. 採用Inverted Bottleneck Convolution Blocks，來提升訓練效率，對比低深度高Channel能達到更好效率 (9s/Epoch with 3060 GPU)。
2. 採用Adam 達到更快訓練。
3. 由正規化技術 Batch-Normalize、搭配Dropout、權重衰減，以及訓練時Early-Stopping，以避免過度擬合。
4. 在使用未標記資料訓練時提高權重衰減，並降低學習率

## 實驗結果:

測定獲取: 0.75694測試準確度，對比訓練0.82準確度，代表泛度仍須提升

一張含有 螢幕擷取畫面, 繪圖, 繪圖軟體, 多媒體軟體 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 繪圖, 行, 繪圖軟體 的圖片

自動產生的描述

# 實驗二

## 前處理

### 設置分割

變更Batch 設置為 32

## 模型設計

### 激活函數變更

將全部激活函數變更為 `nn.SELU()`，觀測結果與出對比用

## 總結：

1. 我將對比 Selu 與 Relu 的差異，我認為對我的架構上有些許提升，但是賞需多次實驗度比
2. 較小的 Batch 似乎能提升泛度
3. 總一成效提升4.3%

## 實驗結果:

測定獲取: 0.80092測試準確度，對比訓練0.83準確度，有顯著提升，說明在該設計使用Selu具有較好成效

一張含有 螢幕擷取畫面, 繪圖 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 繪圖, 行 的圖片

自動產生的描述

# 實驗三

## 前處理

### 圖片轉換

1. 更新訓練集的平均值與標準差
2. 去除干擾性訓練資料

## 總結：

移除干擾資料後，我的模型成效下降0.7%

## 實驗結果:

測定獲取: 0.793262驗證準確度，對比訓練 0.83 準確度，測試 0.7723，代表有過度擬合跡象

一張含有 螢幕擷取畫面, 繪圖, 繪圖軟體, 多媒體軟體 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 繪圖, 繪圖軟體 的圖片

自動產生的描述

備註: 早停會回朔參數到最後儲存最佳準確驗證模型的參數

# 實驗四

## 前處理

### 圖片轉換

防範過度擬合，新增額外轉換

1. `transforms.RandomVerticalFlip()`隨機垂直翻轉的轉換，以增加數據的多樣性。
2. `transforms.ColorJitter(...)`: 隨機調整圖像的亮度、對比度、飽和度和色調。
3. `transforms.RandomGrayscale(p=0.1)`: 以 0.1 的概率將圖像轉換為灰度。
4. `transforms.RandomAffine(...)`: 進行隨機仿射變換，這裡主要是平移。

### 訓練方法

使用 PyTorch 的 torch.optim .lr\_scheduler.ReduceLROnPlateau 模塊。

1. 'min': 監控的指標，這裡是最小化某個指標。
2. patience=5: 如果在 5 個 epoch 內指標沒有改善，則減少學習率。
3. factor=0.5: 學習率減少的倍數。

## 總結：

1. 訓練資料變得更多樣化能提升模型泛度，然而卻會損失訓練時的效率
2. 自動減低學習率對後期微調有不錯成效
3. 成效降低0.6%

## 實驗結果:

測定獲取: 0. 786957驗證準確度，對比訓練 0.67 準確度，測試 0.79107，過度擬合在此有所緩解，但是訓練的梯度變得相當難優化

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 文字, 字型 的圖片

自動產生的描述

備註: 這是基於實驗二模型進行調整

# 實驗五

圖片轉換

防範過度擬合，新增額外轉換

1. 取消 `transforms.RandomGrayscale(p=0.1)`
2. 取消 `transforms.RandomAffine(...)`

## 模型設計

### 一、初始特徵提取

這一層用於對原始圖像進行初步的特徵提取。使用了3個通道（RGB）輸入和32個輸出通道。

### 二、IBConv Blocks With Squeeze-and-Excitation and Residual Connect

每個區塊都可能包含深度可分離卷積（Depthwise separable convolution）、Squeeze-and-Excitation（SE）模塊等。這些區塊有時會被重複使用，特別是當輸入和輸出通道數相同的時候。在IBConv區塊中，如果輸入和輸出通道數相同，添加一個殘差連接（Residual Connection）和Dropout層以防止過擬合。

### 三、Final Convolution

這一層將80個通道的特徵圖轉換為1280個通道。

### 四、Fully connected layers

取消 Dropout

## 訓練方法

1. 學習率改成0.003
2. weight\_decay 改成1e-5
3. 在每個提升valid accuracy後降低threshold 0.025

## 總結：

1. SE使用可以提升準確度，但是使我的每個epochs 多25% 時間 (約10s 採用 3060 計算)
2. Residual 可以加強深層網路學習，不過對比8個Block與15個Block更深層網路準確度提升有限，推測可能是資料過少
3. 自動降低 threshold可以在 Semi提升些許準確度，且有持續提升空間
4. 在Residual卷積層使用 Dropout比在全連接使用具有較好成效
5. 成效顯著提升3.2%

## 實驗結果:

測定獲取: 0.827667驗證準確度，對比訓練0.870700準確度，測試 0.81924，過度擬合出現在訓練階段，但是總成效提升

一張含有 螢幕擷取畫面, 繪圖, 繪圖軟體, 行 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 文字 的圖片

自動產生的描述

上方是Block 8

一張含有 螢幕擷取畫面, 繪圖, 多媒體軟體, 繪圖軟體 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 多媒體軟體, 繪圖軟體, 繪圖 的圖片

自動產生的描述

上方是Block 11，在後期出現比較嚴重過擬合，但是Valid走輕微上升

一張含有 螢幕擷取畫面, 繪圖軟體, 多媒體軟體, 繪圖 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 繪圖軟體, 繪圖, 多媒體軟體 的圖片

自動產生的描述

上方是Block 15

# 實驗六

## 訓練方法

1. 新增 EMA 在 Semi-Supervise (ema\_decay = 0.999)，不直接使用反向回饋，而是由學生反饋更新，具有較好穩定性
2. 將Pseudo-Label改成每個epoch都會更新，即便上回合已經被標定
3. 每次epoch結束時選擇較佳模型來預測下回合的Persudo-Lable

## 總結：

我採用 EMA 與原始模型對比驗證準確度，發現這樣能使標籤在每epoch刷新時具有較好的品質穩定性，最終得到較好成效，不過很意外的是，我的EMA Valid準確度很長時間是低於原始模型，即便loss下降，推測可能是過度自信的Pseudo-Label會對模型有負面影響。

## 實驗結果:

測定獲取: 0.8262驗證準確度，對比訓練0.87380準確度，測試 0.82629，對於末端訓練提升泛度有比較好幫助

# 實驗七

## 訓練方法

1. 新增 FixMatch半監督演算法，模型以教師-學生的方式訓練，教師會用較簡單Argumentation，而學生會作較複雜的Argumentation，優化器對真實標籤與偽標籤使用不同loss來計算，如下圖

* 真實標籤loss function

一張含有 字型, 白色, 文字, 設計 的圖片

自動產生的描述

* 偽標籤 loss function

一張含有 字型, 文字, 白色, 書法 的圖片

自動產生的描述

## 總結：

驗證準確率有輕微上升，但是測試準確度保持一致。或許這方法在此模型成效有限

## 實驗結果:

測定獲取: 0.83317驗證準確度，測試 0.82629，沒有顯著提升 (訓練時我沒有紀錄混砸資料的準確度)

# 實驗八

## 訓練方法

1. 新增 EMA 在 Semi-Supervise (ema\_decay = 0.999)，不直接使用反向回饋，而是由學生反饋更新，具有較好穩定性
2. 將Pseudo-Label改成每個epoch都會更新，即便上回合已經被標定
3. 每次epoch結束時選擇較佳模型來預測下回合的Persudo-Lable

## 總結：

我採用 EMA 與原始模型對比驗證準確度，發現這樣能使標籤在每epoch刷新時具有較好的品質穩定性，最終得到較好成效，不過很意外的是，我的EMA Valid準確度很長時間是低於原始模型，即便loss下降，推測可能是過度自信的Pseudo-Label會對模型有負面影響。

## 實驗結果:

測定獲取: 0.8262驗證準確度，對比訓練0.87380準確度，測試 0.82629，對於末端訓練提升泛度有比較好幫助

# 實驗九

## 訓練方法

1. 使用Temperature=2在監督學習階段

## 總結：

沒有觀測刀顯著差異

## 實驗結果:

測定獲取: 0.83317驗證準確度，測試 0.82629，沒有變化

一張含有 螢幕擷取畫面, 繪圖, 繪圖軟體 的圖片

自動產生的描述

一張含有 螢幕擷取畫面, 文字, 繪圖, 繪圖軟體 的圖片

自動產生的描述

# 架構圖:

<https://github.com/whats2000/Deep-Learning/blob/main/PyTorch/Class/pytorch103/model_architecture.png>

# 程式原始碼:

<https://github.com/whats2000/Deep-Learning/blob/main/PyTorch/Class/pytorch103/A3_semi_supervised_flower_classification.ipynb>

# 資料參考:

不同類型卷積方塊設計方法: <https://github.com/FrancescoSaverioZuppichini/BottleNeck-InvertedResidual-FusedMBConv-in-PyTorch>

Residual Inverted Bottleneck Convolution:

<https://paperswithcode.com/method/inverted-residual-block>

自動轉換選擇:  
<https://arxiv.org/pdf/1805.09501.pdf>

關於 SE 使用:

<https://hackmd.io/@machine-learning/HkH1jUArI>

半監督策略 FixMatch:

<https://arxiv.org/ftp/arxiv/papers/2001/2001.07685.pdf>