## 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、MBConv Blocks

接下來，定義了一系列的Mobile Inverted Bottleneck Convolution Blocks。我是閱讀自資料參考一，這些區塊通常用於深度學習中的移動應用，以減少模型的計算量，同時保持性能。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. 我首次採用Mobile Inverted Bottleneck Convolution Blocks，來提升訓練效率。
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
2. 測試較小的 batch 是否能提升訓練效率

### 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：

1. 我將對比有無移除干擾資料，進而對比成效

### 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。

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

### 總結：

我發現到將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，在此實驗我首次嘗試用自動調整學習率進行微調

### 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、MBConv 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MBConv區塊中，如果輸入和輸出通道數相同，添加一個捷徑連接（Shortcut 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

1. 取消 Dropout

訓練方法

* 學習率改成0.003
* weight\_decay 改成1e-5

### 總結：

SE使用可以提升準確度，Residual 可以加強深層網路學習

### 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，但是總提成效提升

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

### 資料參考:

不同類型，卷積方塊設計: <https://blog.csdn.net/deephub/article/details/124684557>

Mobile 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>