人工智慧模型設計與應用 Lab3

NM6121030 余振揚

1. **Outline:**

   在此次 lab 建了兩個 model，兩者在 Hyper Parameters 固定的情況下，僅增加 Data Augmentation。在最好的 model 中達到 Test Accuracy>90%的條件。

2. **Input Image Normalization:**

   在 CIFAR10 的 training 資料中，可計算出所有影像 RGB 通道的 mean 及 std。

   - mean: [0.490703, 0.48110706, 0.44542626]
   - std: [0.24697259, 0.2435216, 0.26159507]

   在建立 dataset 時，做正規劃(Normalization):

```python
# data augmentation & normalization
transform_train = transforms.Compose([
    # data augmentation
    transforms.ToTensor(),
    # data normalization    # standardization: (image - train_mean) / train_std
    transforms.Normalize(train_mean, train_std)
])
```

3. **Data Augmentation:**

   僅最好的 model 有做 data augmentation，使用到的 data augmentation 技術如下：

   - **RandomHorizontalFlip:**

     transforms.RandomHorizontalFlip(p=0.5) 以概率 p 對圖像進行隨機水平翻轉。

   - **RandomRotation:**

     transforms.RandomRotation(degrees) 隨機旋轉圖像一定的角度度數。degrees 是旋轉的最大角度範圍。

   - **ColorJitter**:

     transforms.ColorJitter(brightness=0, contrast=0, saturation=0, hue=0) 隨機調整圖像的亮度、對比度、飽和度和色調。

   - **RandomErasing**:

     transforms.RandomErasing(p=0.5, scale=(0.02, 0.33), ratio=(0.3, 3.3), value=0, inplace=False) 以概率 p 對圖像進行隨機刪除，通過隨機遮蔽圖像的一部分來強制模型學習更多訊息。

```python
# data augmentation & normalization
transform_train = transforms.Compose([
    # data augmentation
    transforms.RandomHorizontalFlip(),  # 50% chance to flip the image horizontally
    transforms.RandomRotation(10),  # rotate the image by 10 degrees
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1),
    transforms.ToTensor(),
    transforms.RandomErasing(),  # randomly selects a rectangle region in an image a
    # data normalization    # standardization: (image - train_mean) / train_std
    transforms.Normalize(train_mean, train_std)
])
```

4. **Learning Rate and Update Strategy:**

   每 10 個 Epoch 就降低 Learning Rate 以防止過大的 Lr 造成震盪。

```python
# learning rate shedule
def adjust_learning_rate(optim, epoch):
    # define your lr scheduler
    if epoch < 10:    # 0~9
        lr = 0.05
    elif epoch < 20:    # 10~19
        lr = 0.01
    elif epoch < 30:    # 20~29
        lr = 0.005
    else:               # 30~40
        lr = 0.001

    for param_group in optim.param_groups:
        param_group['lr'] = lr
```

5. **Batch Size:**

   為了做出實驗組與對照組，及比較 DataAugmentation 對模型的影響，這邊的 Batch Size 為原始助教提供的 128，並未做更動。

6. **ResNet 18 Architecture:**

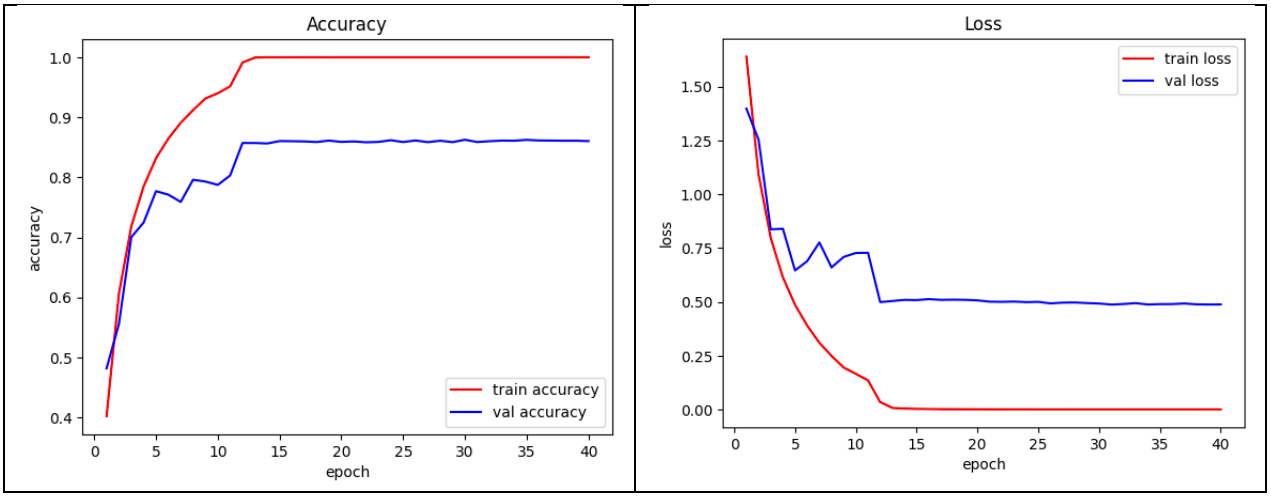
   若是按照原論文的方式去設計模型，第一層 convolution layer 的 filter size 應該為 7 X 7 大小，但助教給的 source code 中 size 為 3 X 3，詢問過後才發現是為了針對 CIFAR-10 資料集所做的小更動。原始的 ResNet paper 中，作者是使用 ImageNet 資料集做 training，而 model 的輸入影像有 224 X 224 大小，但 CIFAR-10 的圖片僅有 32 X 32 大小，因此將 filter 大小從 7 X 7 更改為 3 X 3 是為了適應這種小尺寸的圖片。過大的 filter 可能會導致模型在小圖片上過度擬合。

```python
self.conv1 = nn.Sequential(
    nn.Conv2d(3, 64, kernel_size=3, stride=1, padding=1, bias=False),  # for CIFAR10, use 3x3 ke
    # nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=1, bias=False),   # original paper uses
    nn.BatchNorm2d(64),
    nn.ReLU(),
)
```

7. **Train and Validation Plot:**

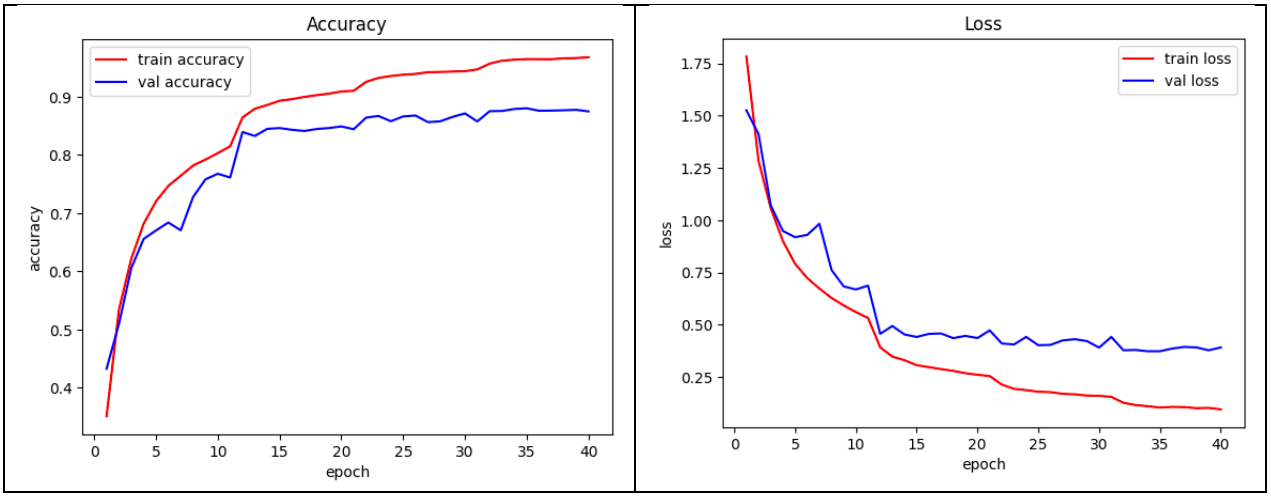
- **Base Model:**

  可發現在第 15epoch 時，validation accuracy 不再提升，應為 overfitting。



- **Fine-Tuned Model (with data augmentation):**

  經過多個 data augmentation 的轉換，雖然 validation 在後續的表現上沒有持續進步，但可發現 train accuracy 並沒有在第 15 epoch 時直接衝到接近 1，至少還是有些震盪，此表示 data augmentation 可以防止 overfitting 的發生。



8. Test Accuracy and Loss:
   - Base Model: (Loss: 0.494, Accuracy: 0.860)

```
Epoch: 40
learning rate:  0.001
Train loss: 0.002 | Train acc: 1.000
100%|          | 40/40 [07:29<00:00, 11.24s/it]
Val loss: 0.489 | Val acc: 0.860

Test loss: 0.494 | Test acc: 0.860
```

   - Fine-tuned Model: (Loss: 0.288, Accuracy: 0.912)

```
Epoch: 40
learning rate:  0.001
Train loss: 0.095 | Train acc: 0.968
100%|          | 40/40 [07:44<00:00, 11.62s/it]
Val loss: 0.391 | Val acc: 0.875

Test loss: 0.288 | Test acc: 0.912
```

9. Feedback:

   這次的 Lab 讓我對 ResNet 有更多的了解，之前都是直接 import ResNet，並沒有特別了解裡面的參數，而閱讀完原始 paper 後，對建立 layer 有更多的認知，且也學到了針對不同大小的 image 要做相對應的變動。另外也學到了多種 Data Augmentation 的方法，以降低 overfitting。