

人工智慧模型設計與應用 Lab3

NM6121030 余振揚

1. Outline:

在此次 lab 總共建了 6 個 model，針對 Batch Size、Adjust Learning Rate 以及 Data Augmentation 做實驗與對照。並在最好的 model 中達到 Test Accuracy>90%的條件。

以下是各個模型 Test Accuracy 的統計表：

	Non-adjusting Learning Rate	Adjusting Learning Rate	Adjust Learning Rate & Batch Size = 256
Base	0.866	0.860	0.856
Data Augmentation	0.908	0.915	0.908

Base Model 為不添加任何 Data Augmentation 方法的模型，僅包含 Normalize。

Data Augmentation Model 添加了許多 transform 函數，將在後續提及。

Non-adjusting Learning Rate 及 adjusting Learning Rate 的測試中，batch size 皆為 128。

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，做正規化(Normalization):

```
# 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)
])

# for test, we only need to normalize data
transform_test = transforms.Compose([
    transforms.ToTensor(),
    # data normalization # standardization: (image - train_mean) / train_std
    transforms.Normalize(train_mean, train_std)
])
```

對於 test dataset 也必須以 train dataset 的分佈來做正規化，因為在現實生活中不可能知道 test dataset 的資料分佈，我們只能預先假設未來的資料會跟現有的訓練資料有相同的資料分佈。

3. Batch Size:

由於 GPU 的平行運算，使得可以一次 train 更大的數量，但越大的 batch size 意味著模型將一次看到多個 data，多少會影響 gradient descent 的尋找最佳解的方向，本次比較 128 及 256 對模型的影響。

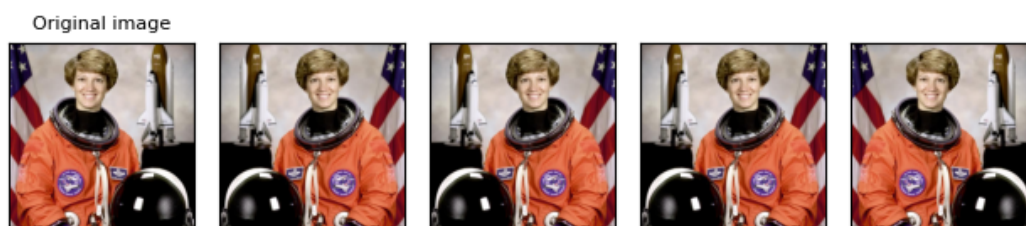
4. Data Augmentation:

三個 Data Augmentation Model 均使用到同樣參數的 transform 函數，如下：

```
# 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
    # data normalization # standardization: (image - train_mean) / train_std
    transforms.Normalize(train_mean, train_std)
])
```

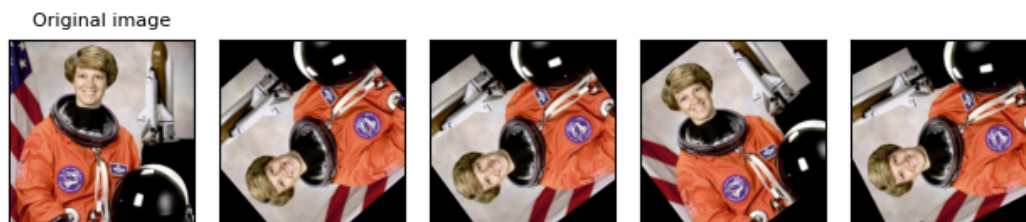
- **RandomHorizontalFlip:**

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



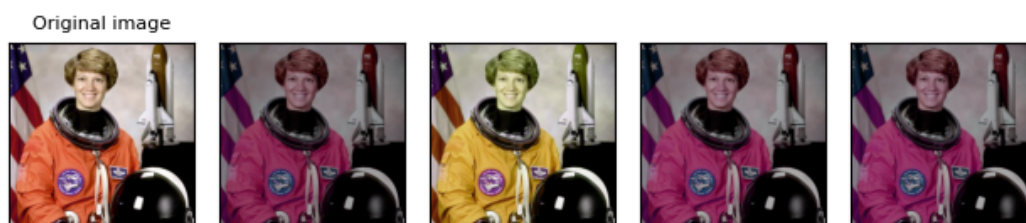
- **RandomRotation:**

`transforms.RandomRotation(degrees)` 隨機旋轉圖像一定的角度度數。



- **ColorJitter:**

`transforms.ColorJitter()` 隨機調整圖像的亮度、對比度、飽和度和色調。



- **RandomErasing:**

`transforms.RandomErasing()` 隨機遮蔽圖像的一部分來強制模型學習更多訊息。



5. 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。

```
# learning rate shedule
def adjust_learning_rate(optimizer, 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 optimizer.param_groups:
        param_group['lr'] = lr
```

但 source code 裡其實有使用現有的 SGD optimizer，所以不寫這個函式，learning rate 還是會自適應的下降。所以也將使用 adjust_learning_rate 的有無納入評估模型的考量。

6. ResNet 18 Architecture:

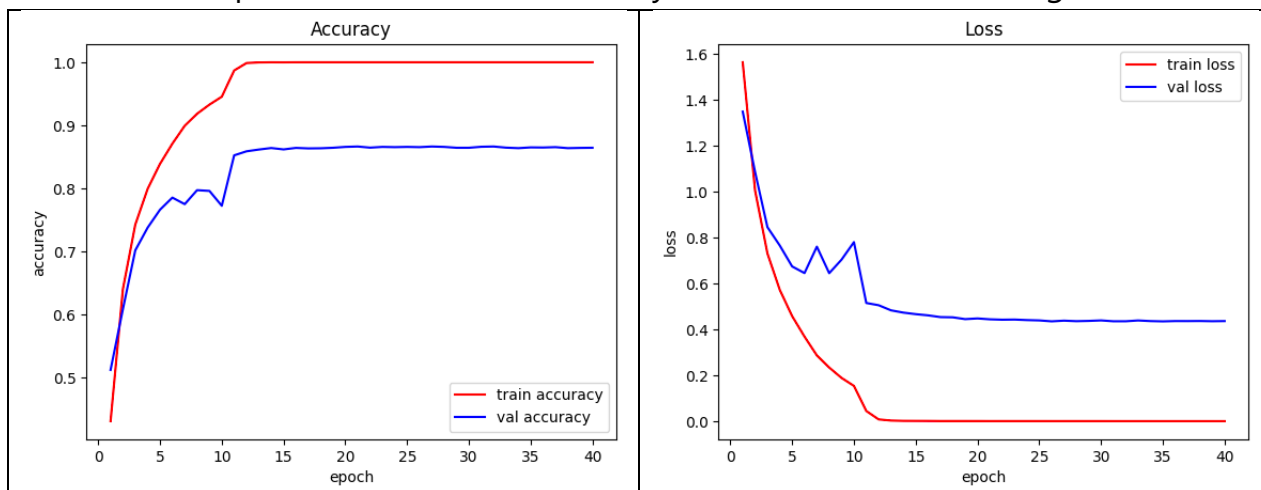
若是按照原論文的方式去設計模型，第一層 convolution layer 的 filter size 應該為 7 X 7 大小，但助教給的 source code 中 size 為 3 X 3，詢問過後才發現是為了針對 CIFAR-10 資料集所做的小更動。原始的 ResNet paper 中，作者是使用 ImageNet 資料集做 training，而 model 的輸入影像有 224 X 224 大小，但 CIFAR-10 的圖片僅有 32 X 32 大小，因此將 filter 大小從 7 X 7 更改為 3 X 3 是為了適應這種小尺寸的圖片。過大的 filter 可能會導致模型在小圖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

```
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:

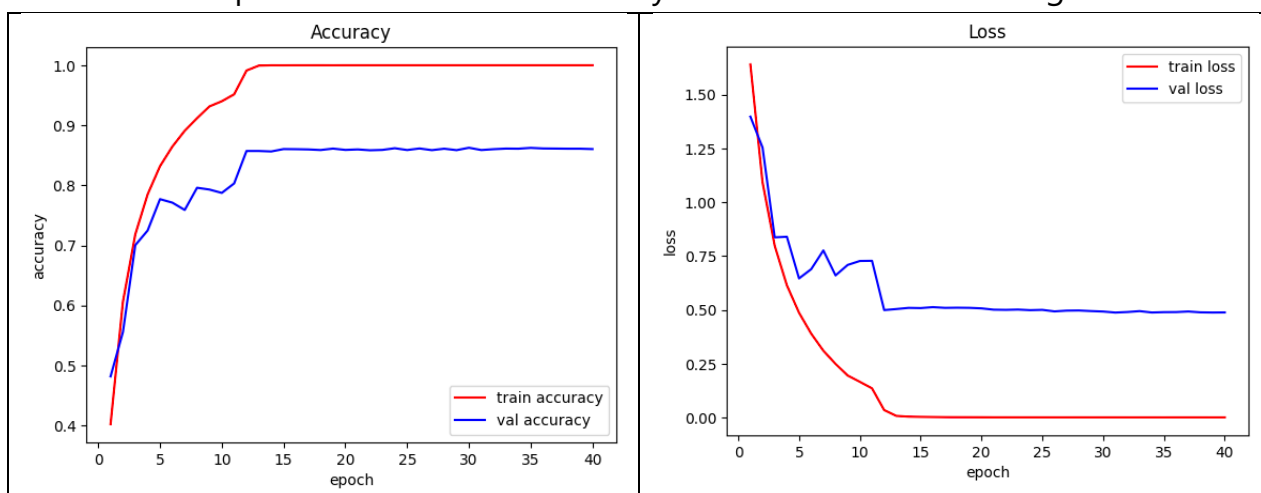
- **Base Model (Non-adjusting Learning Rate):**

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



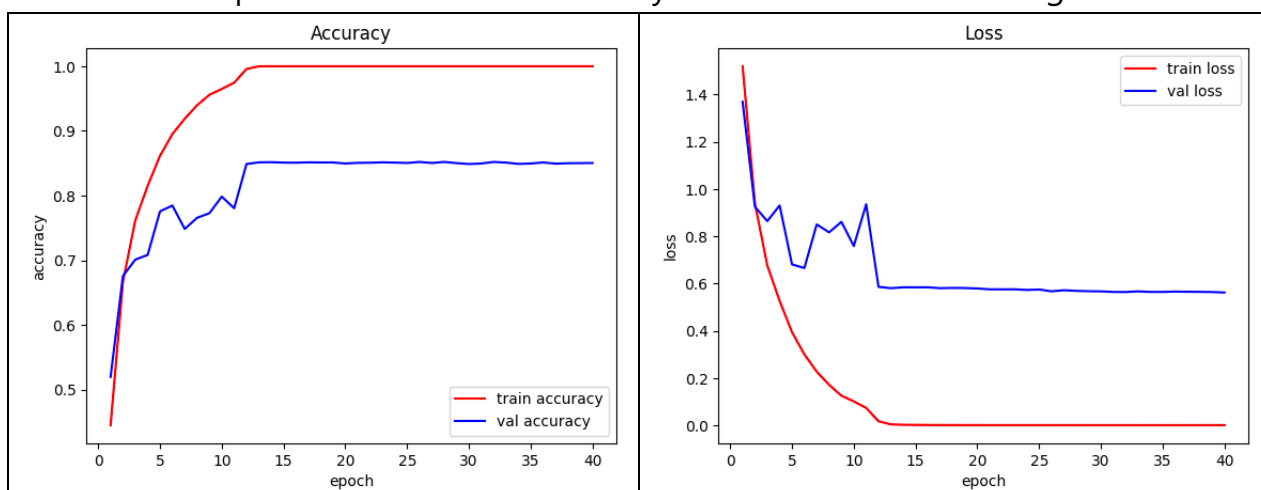
- **Base Model (Adjusting Learning Rate):**

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



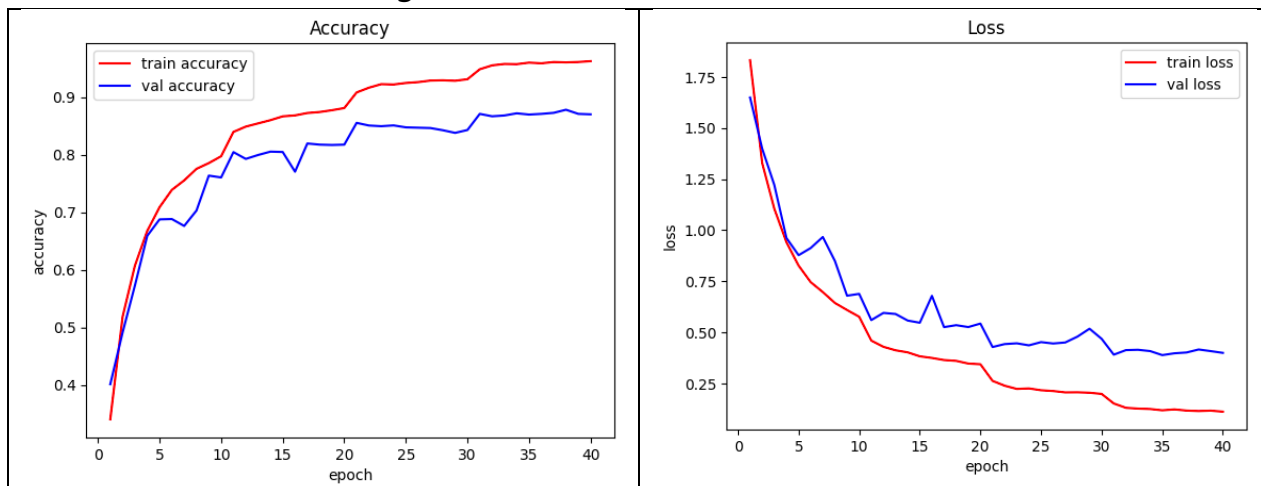
- **Base Model (Adjusting Learning Rate & Batch Size = 256):**

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



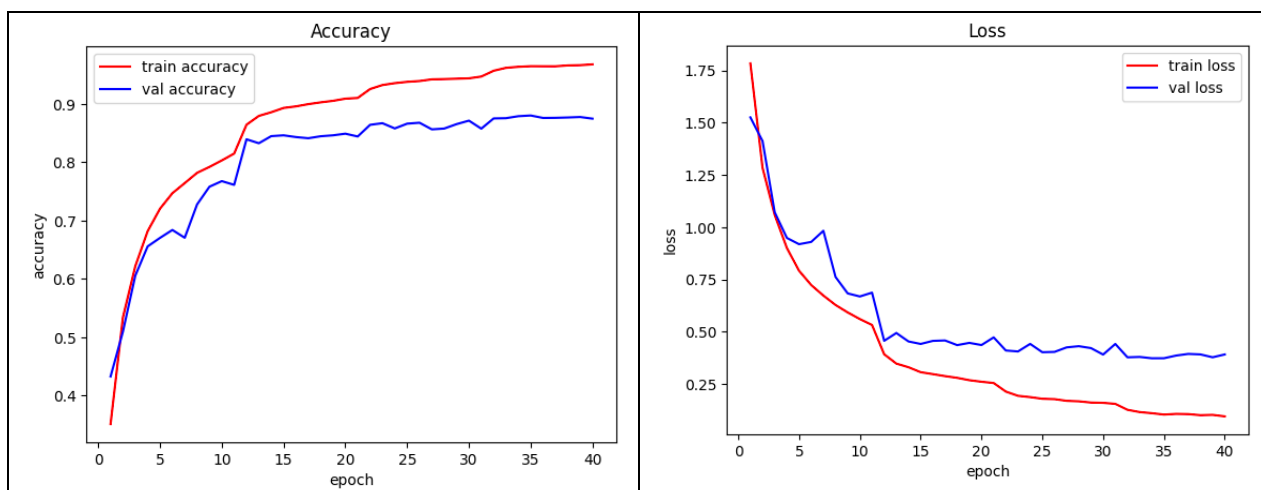
- **Data Augmentation Model (Non-adjusting Learning Rate):**

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 可以防止類似 Base Model 提早 overfitting 的發生。



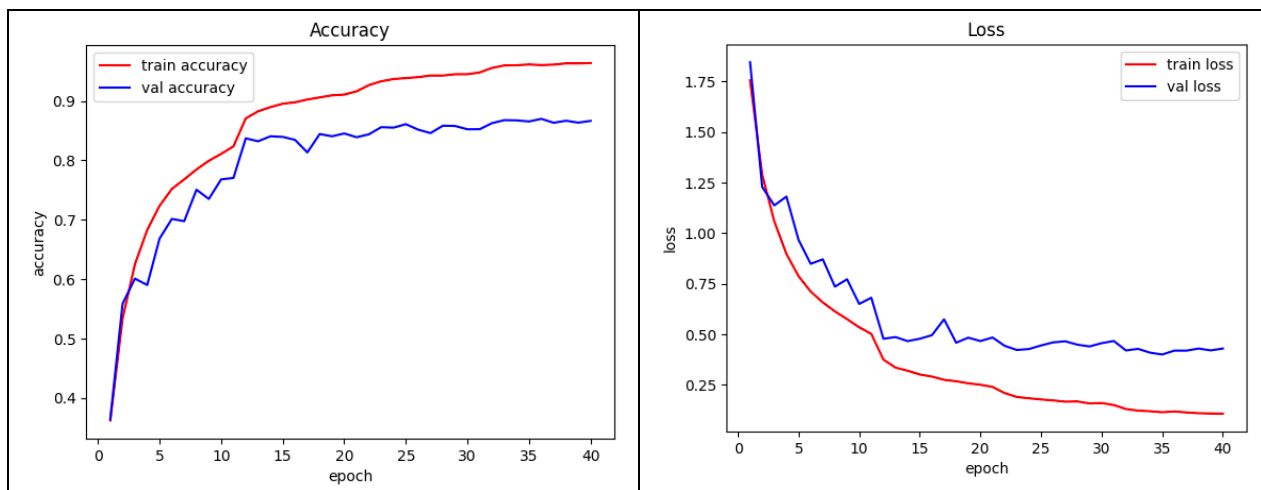
- **Data Augmentation Model (Adjusting Learning Rate):**

同上，其實模型表現的差不多。



- **Data Augmentation Model (Adjusting Learning Rate & Batch Size = 256):**

同上，其實模型表現的差不多。



8. Test Accuracy and Loss:

- Base Model (Adjusting Learning Rate): (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
```

- Data Augmentation Model (Adjusting Learning Rate): (Loss: 0.278, Accuracy: 0.915)

```
Epoch: 40  
learning rate: 0.001  
Train loss: 0.098 | Train acc: 0.968  
100%|██████████| 40/40 [07:48<00:00, 11.72s/it]  
Val loss: 0.372 | Val acc: 0.881  
  
Test loss: 0.278 | Test acc: 0.915
```

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。對於即使使用 Data Augmentation 的方法，但最後還是無法繼續讓 validation accuracy 上升，我認為可以在 ResNet 的 Fully Connected Layer 中加入 Dropout 的做法，但由於對 ResNet 還不夠熟悉，所以無法實現該技術。

10. Reference:

- [Illustration Of Transforms](#)
- [Random Erasing Data Augmentation](#)