EAI Lab2 Report

Task 1 – create your own CNN model

(1) Print model summary (including parameters) and plot model (5%)
第一個任務中，我使用的模型包含三個 convolutional layers，每個 layer 包含一個 Conv2d、BatchNorm2d、ReLU 和 MaxPool2d，每經過一個 convolutional layer，activation 的 spatial size (both height and width) 會減半、channel 數則加倍，最後則是 Linear layer 輸出 10 個類別的結果，並加上 Dropout 做 regularization 以避免 overfitting。

```
FLOPs: 10571776.0
Params: 114186.0
----------------------------------------------------------------
       Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 32, 32]             896
       BatchNorm2d-2          [-1, 32, 32, 32]              64
              ReLU-3          [-1, 32, 32, 32]               0
         MaxPool2d-4          [-1, 32, 16, 16]               0
            Conv2d-5          [-1, 64, 16, 16]          18,496
       BatchNorm2d-6          [-1, 64, 16, 16]             128
              ReLU-7          [-1, 64, 16, 16]               0
         MaxPool2d-8            [-1, 64, 8, 8]               0
            Conv2d-9           [-1, 128, 8, 8]          73,856
      BatchNorm2d-10           [-1, 128, 8, 8]             256
             ReLU-11           [-1, 128, 8, 8]               0
        MaxPool2d-12           [-1, 128, 4, 4]               0
          Dropout-13           [-1, 128, 4, 4]               0
           Linear-14                 [-1, 10]          20,490
================================================================
Total params: 114,186
Trainable params: 114,186
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 1.44
Params size (MB): 0.44
Estimated Total Size (MB): 1.88
----------------------------------------------------------------
```

(2) Print test accuracy, plot epoch-train accuracy, epoch-val accuracy, epoch-train loss, epoch-val loss (10%)

Training configuration:

- Data normalization with mean and standard deviation calculated from the first image in the training set
- Data augmentation: RandomCorp, RandomHorizontalFlip, and RandomRotation
- Batch size: 128
- Optimizer: Adam
- Base learning rate: 0.001
- Learning rate update strategy: halve per 10 epochs (StepLR)
- Number of epochs: 80

依照上面的配置訓練模型後做 testing，準確度達到 81.7%，觀察訓練過程中的 loss 和 accuracy 顯示，整體訓練過程大致穩定沒有劇烈震盪，且沒有出現 training accuracy 持續下降但 validation accuracy 反而上升的狀況，因此沒有 overfitting，最後 loss 和 accuracy 曲線也趨於平穩收斂。

```
Test loss: 0.533, Test accuracy: 0.817
```

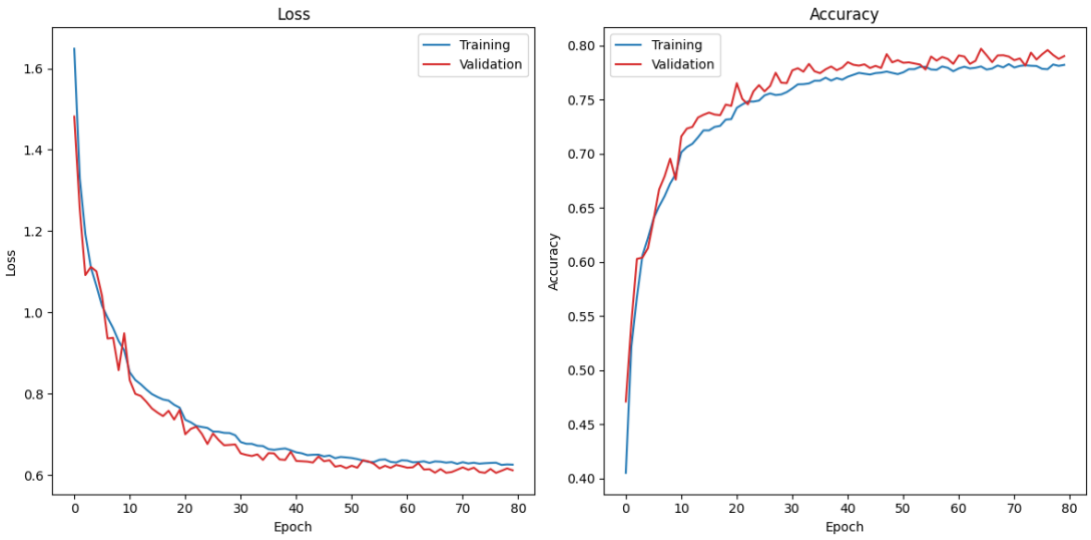
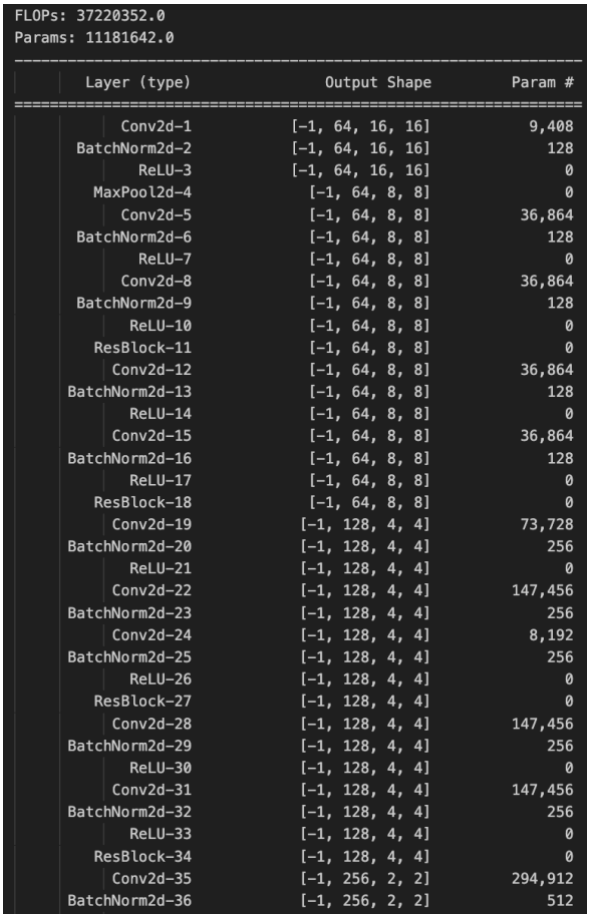


Figure 1 SimpleCNN

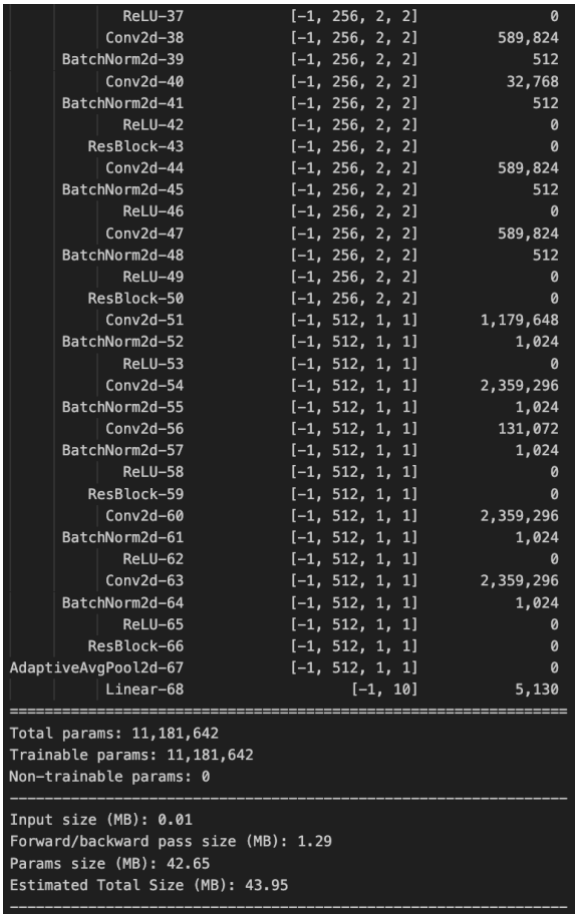
(3) Describe how do you choose your best model (5%)
在訓練過程中，選擇當前 validation accuracy 最高的 checkpoint 存下來。

Task 2 – ResNet 18 implementation

(4) Print model summary (including parameters) and plot model (5%)
ResNet18 模型實作如下：



```
FLOPs: 37220352.0
Params: 11181642.0
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
          Conv2d-1          [-1, 64, 16, 16]           9,408
     BatchNorm2d-2          [-1, 64, 16, 16]             128
            ReLU-3          [-1, 64, 16, 16]               0
       MaxPool2d-4            [-1, 64, 8, 8]               0
          Conv2d-5            [-1, 64, 8, 8]          36,864
     BatchNorm2d-6            [-1, 64, 8, 8]             128
            ReLU-7            [-1, 64, 8, 8]               0
          Conv2d-8            [-1, 64, 8, 8]          36,864
     BatchNorm2d-9            [-1, 64, 8, 8]             128
           ReLU-10            [-1, 64, 8, 8]               0
      ResBlock-11            [-1, 64, 8, 8]               0
          Conv2d-12            [-1, 64, 8, 8]          36,864
     BatchNorm2d-13            [-1, 64, 8, 8]             128
           ReLU-14            [-1, 64, 8, 8]               0
          Conv2d-15            [-1, 64, 8, 8]          36,864
     BatchNorm2d-16            [-1, 64, 8, 8]             128
           ReLU-17            [-1, 64, 8, 8]               0
      ResBlock-18            [-1, 64, 8, 8]               0
          Conv2d-19           [-1, 128, 4, 4]          73,728
     BatchNorm2d-20           [-1, 128, 4, 4]             256
           ReLU-21           [-1, 128, 4, 4]               0
          Conv2d-22           [-1, 128, 4, 4]         147,456
     BatchNorm2d-23           [-1, 128, 4, 4]             256
          Conv2d-24           [-1, 128, 4, 4]           8,192
     BatchNorm2d-25           [-1, 128, 4, 4]             256
           ReLU-26           [-1, 128, 4, 4]               0
      ResBlock-27           [-1, 128, 4, 4]               0
          Conv2d-28           [-1, 128, 4, 4]         147,456
     BatchNorm2d-29           [-1, 128, 4, 4]             256
           ReLU-30           [-1, 128, 4, 4]               0
          Conv2d-31           [-1, 128, 4, 4]         147,456
     BatchNorm2d-32           [-1, 128, 4, 4]             256
           ReLU-33           [-1, 128, 4, 4]               0
      ResBlock-34           [-1, 128, 4, 4]               0
          Conv2d-35           [-1, 256, 2, 2]         294,912
     BatchNorm2d-36           [-1, 256, 2, 2]             512
```



```
           ReLU-37           [-1, 256, 2, 2]               0
         Conv2d-38           [-1, 256, 2, 2]         589,824
    BatchNorm2d-39           [-1, 256, 2, 2]             512
         Conv2d-40           [-1, 256, 2, 2]          32,768
    BatchNorm2d-41           [-1, 256, 2, 2]             512
           ReLU-42           [-1, 256, 2, 2]               0
      ResBlock-43           [-1, 256, 2, 2]               0
         Conv2d-44           [-1, 256, 2, 2]         589,824
    BatchNorm2d-45           [-1, 256, 2, 2]             512
           ReLU-46           [-1, 256, 2, 2]               0
         Conv2d-47           [-1, 256, 2, 2]         589,824
    BatchNorm2d-48           [-1, 256, 2, 2]             512
           ReLU-49           [-1, 256, 2, 2]               0
      ResBlock-50           [-1, 256, 2, 2]               0
         Conv2d-51           [-1, 512, 1, 1]       1,179,648
    BatchNorm2d-52           [-1, 512, 1, 1]           1,024
           ReLU-53           [-1, 512, 1, 1]               0
         Conv2d-54           [-1, 512, 1, 1]       2,359,296
    BatchNorm2d-55           [-1, 512, 1, 1]           1,024
         Conv2d-56           [-1, 512, 1, 1]         131,072
    BatchNorm2d-57           [-1, 512, 1, 1]           1,024
           ReLU-58           [-1, 512, 1, 1]               0
      ResBlock-59           [-1, 512, 1, 1]               0
         Conv2d-60           [-1, 512, 1, 1]       2,359,296
    BatchNorm2d-61           [-1, 512, 1, 1]           1,024
           ReLU-62           [-1, 512, 1, 1]               0
         Conv2d-63           [-1, 512, 1, 1]       2,359,296
    BatchNorm2d-64           [-1, 512, 1, 1]           1,024
           ReLU-65           [-1, 512, 1, 1]               0
      ResBlock-66           [-1, 512, 1, 1]               0
AdaptiveAvgPool2d-67           [-1, 512, 1, 1]               0
         Linear-68                 [-1, 10]           5,130
================================================================
Total params: 11,181,642
Trainable params: 11,181,642
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 1.29
Params size (MB): 42.65
Estimated Total Size (MB): 43.95
----------------------------------------------------------------
```

(5) Print test accuracy, plot epoch-train accuracy, epoch-val accuracy, epoch-train loss, epoch-val loss (10%)

Baseline configuration:

- Data normalization with mean and standard deviation calculated from the first image in the training set
- Data augmentation: RandomCorp, RandomHorizontalFlip, and RandomRotation
- Batch size: 64
- Optimizer: Adam
- Base learning rate: 0.001
- Learning rate update strategy: halve per 10 epochs (StepLR)

依照上述配置訓練模型，準確度達到 85.3%，觀察訓練過程中的 loss 和 accuracy 顯示，整體訓練過程大致穩定沒有劇烈震盪，雖然沒有出現 training accuracy 持續下降但 validation accuracy 反而上升的狀況，不過在訓練後期隨著 training loss 持續下降，validation 卻沒有跟著明顯改善，因此可能有 overfitting 的狀況。不過因為 validation set 是從 training set 隨機

選出來的，有做 data augmentation，而 test set 則沒有，因此 test accuracy 比 validation accuracy 高屬合理狀況。

```
Model loaded from ./task2/resnet18_64.pt (44.80703 MB)
Test loss: 0.464, Test accuracy: 0.853
```

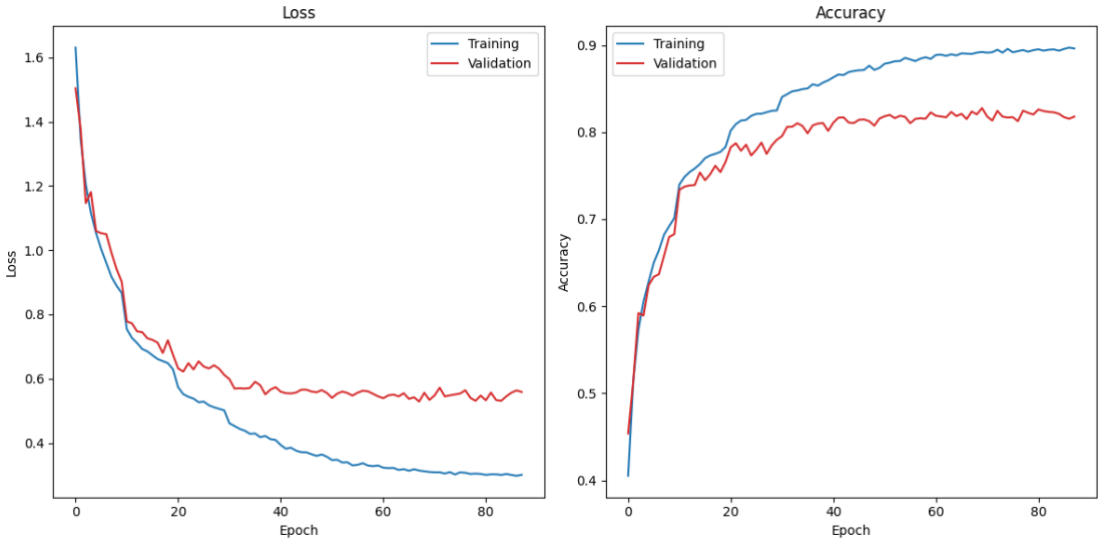


Figure 2 ResNet-18 baseline

(6) Describe how do you choose your best model (5%)
在訓練過程中，選擇當前 validation accuracy 最高的 checkpoint 存下來。

(7) Experiment on the following and compare the result with baseline
以下將會以第(5)題訓練出來的模型作為 baseline，依序探討 data normalization、data augmentation、learning rate 對模型效能的影響。

a. Input image normalization (15%)

	with data normalization (baseline)	without data normalization
Test accuracy	0.853 (figure 2)	0.845 (figure 3)

從 loss-accuracy curve 中可以發現沒有做 data normalization 時，在訓練初期 validation loss 和 validation accuracy 的震盪較明顯，因此可以推論 data normalization 可以讓最佳化的過程更加穩定。

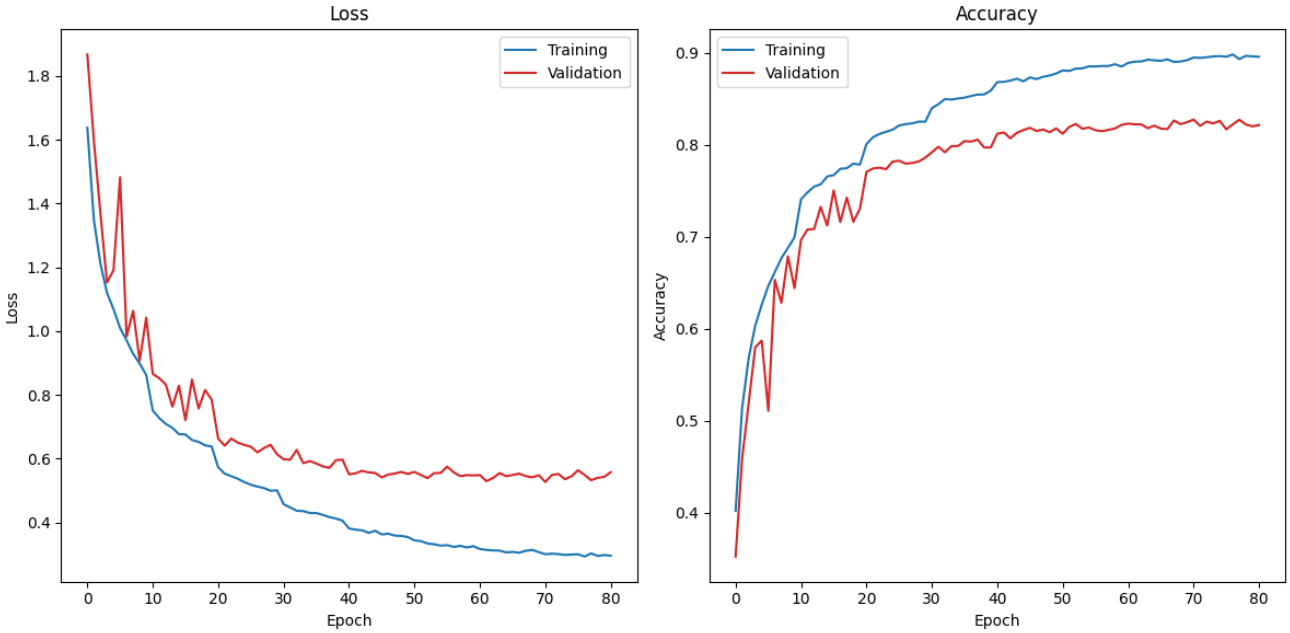


Figure 3 Without data normalization

b. Data augmentation (15%)

	with data augmentation (baseline)	without data augmentation
Test accuracy	0.853 (figure 2)	0.783 (figure 4)

沒有 data augmentation 時，validation loss 從第 8 個 epoch 開始就不再隨著 training loss 下降，發生嚴重的 overfitting，即便因為 training set 變得比較簡單使得 training accuracy 很快達到 90%以上，但 validation 卻始終達不到 80%，因此可以推論 data augmentation 對於預防模型 overfitting 以及提升模型的泛化能力有非常顯著的效果。

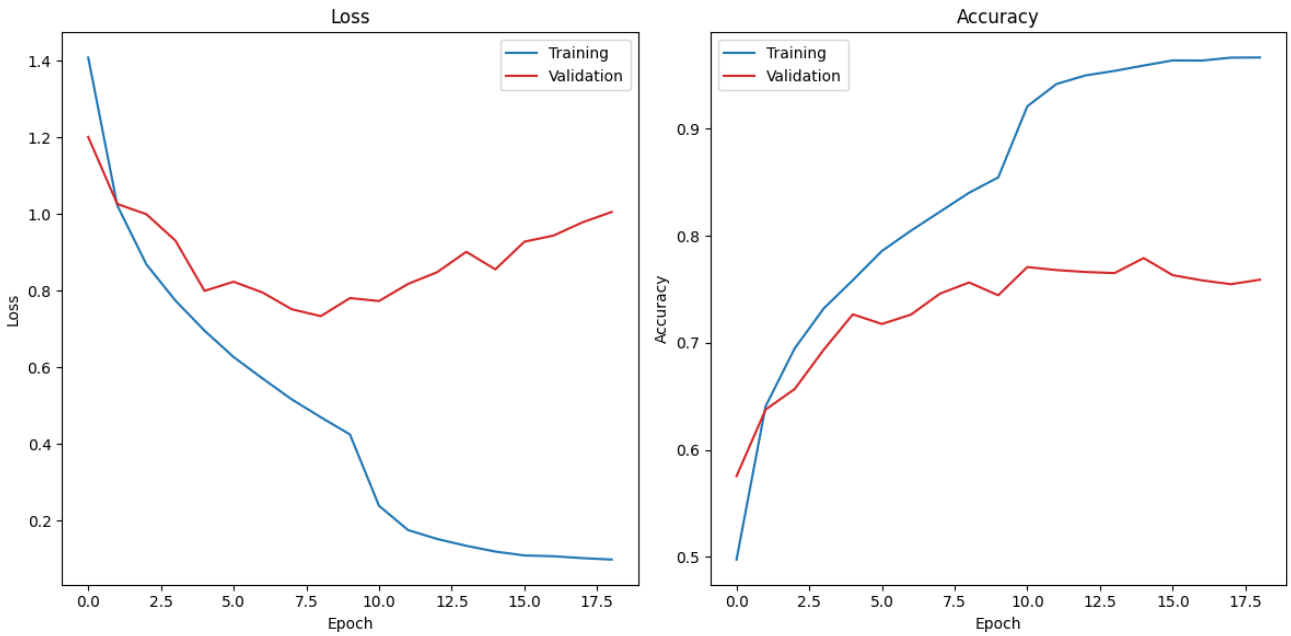


Figure 4 Without data augmentation

c. Different base learning rate and update strategy (15%)

	Step LR (halve every 10 epochs)	Constant LR
Base LR = 0.01	0.715 (figure 5)	0.521 (figure 7)
Base LR = 0.001	0.853 (figure 2, baseline)	0.797 (figure 8)
Base LR = 0.0001	0.796 (figure 6)	0.820 (figure 9)

下圖 5、圖 6 為使用一樣的 learning rate update strategy 但改用不同的初始值，可以發現同樣是每 10 個 epoch 學習率減半，有較高初始學習率的訓練過程有比較多震盪，且權重更新的步長較大，不容易收斂至最佳的值，因此 test accuracy 表現得比 baseline 差，而初始學習率太小時，雖然訓練過程更加平緩，但在更新至最佳權重前就會停下來，會更加依賴權重初始化才能達到更好的準確度。

圖 7、圖 8 和圖 9 則是固定 learning rate，當 learning rate 較大時，訓練過程中 loss 和 accuracy 曲線震盪較明顯，當 learning rate 較小時，訓練過程中 loss 和 accuracy 曲線則較為平緩。具體來說，圖 7 的 learning rate 0.01 過大，以致於模型無法收斂，自第 16 個 epoch 開始 validation loss 就不再下降，圖 8 的 learning rate 0.001 還是過大，自第 40 個 epoch 以後 validation loss 就不再下降，而圖 9 的 learning rate 0.0001 則和 baseline 在訓練後期的 learning 差不多，但因為前期的步長較小，因此收斂速度不如 baseline 的配置。

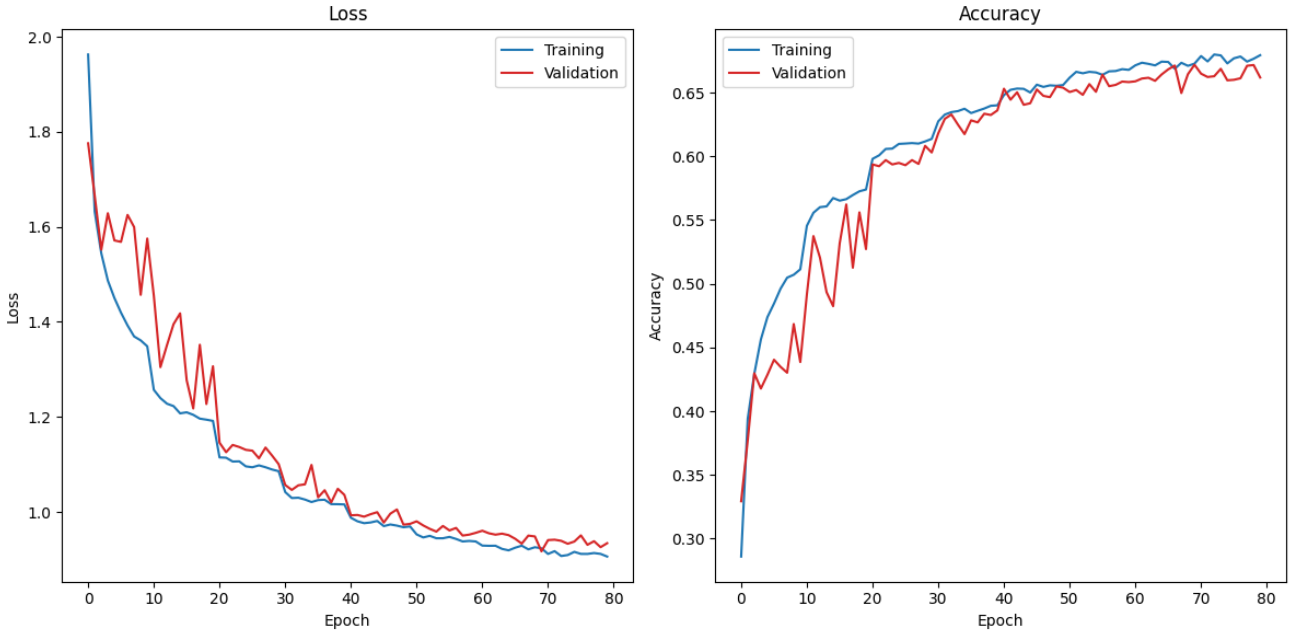


Figure 5 Step LR with inital value 0.01

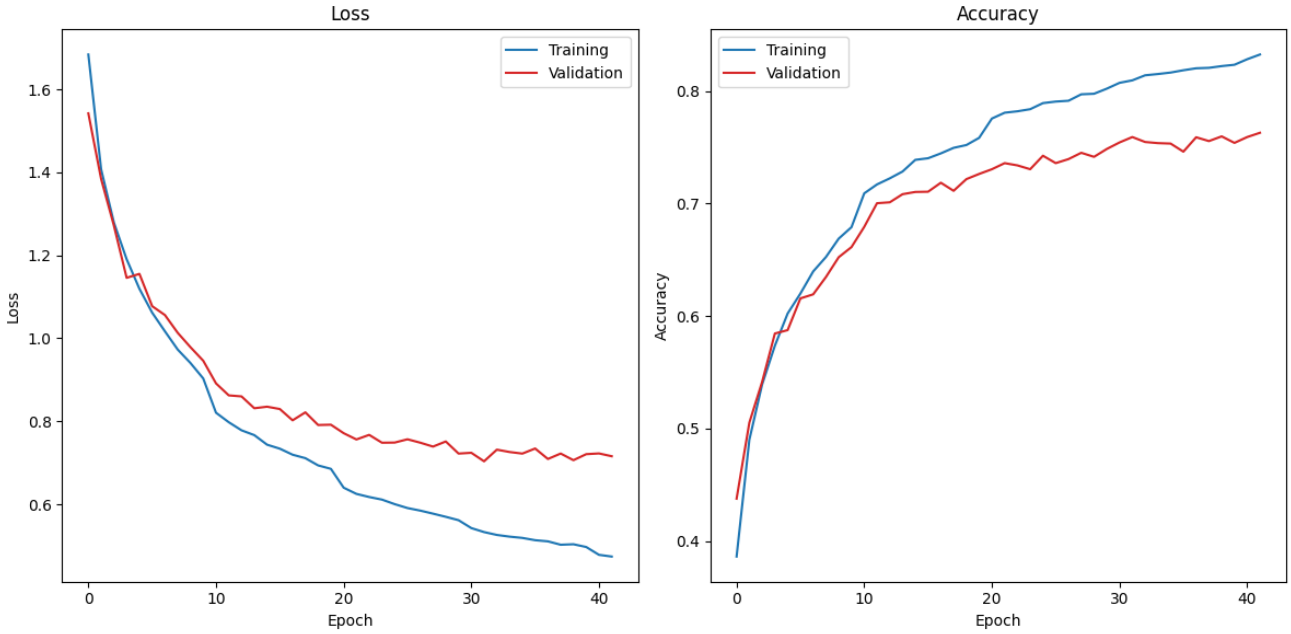


Figure 6 Step LR with inital value 0.0001

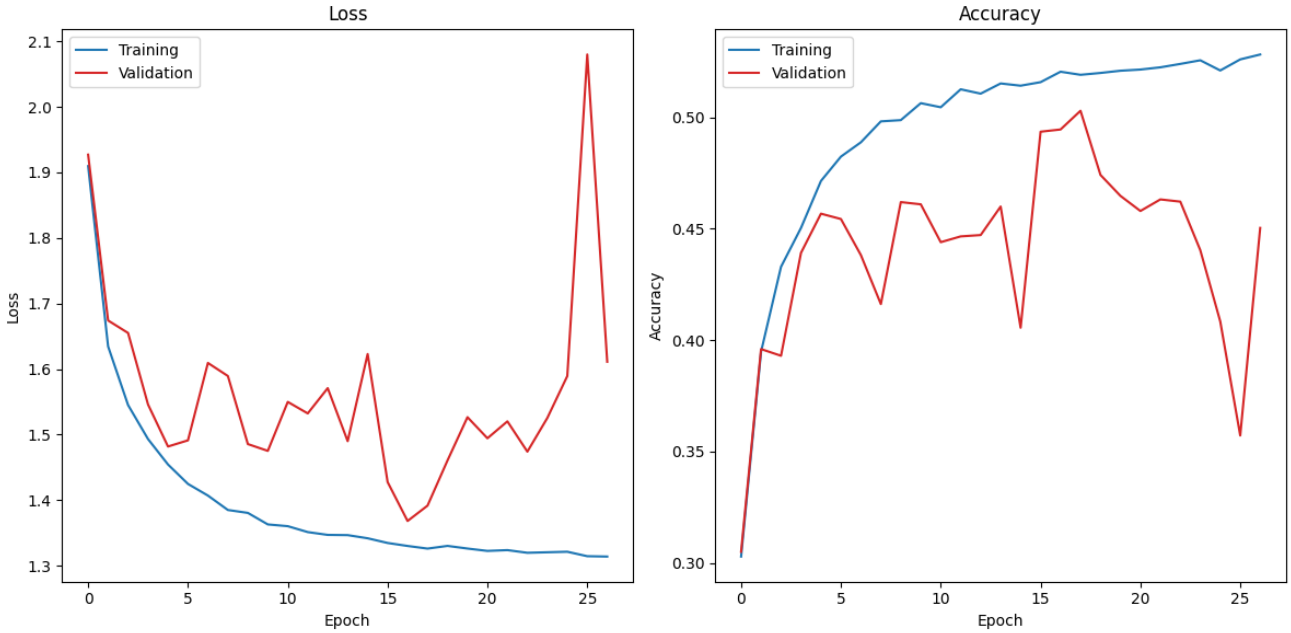


Figure 7 Constant LR 0.01

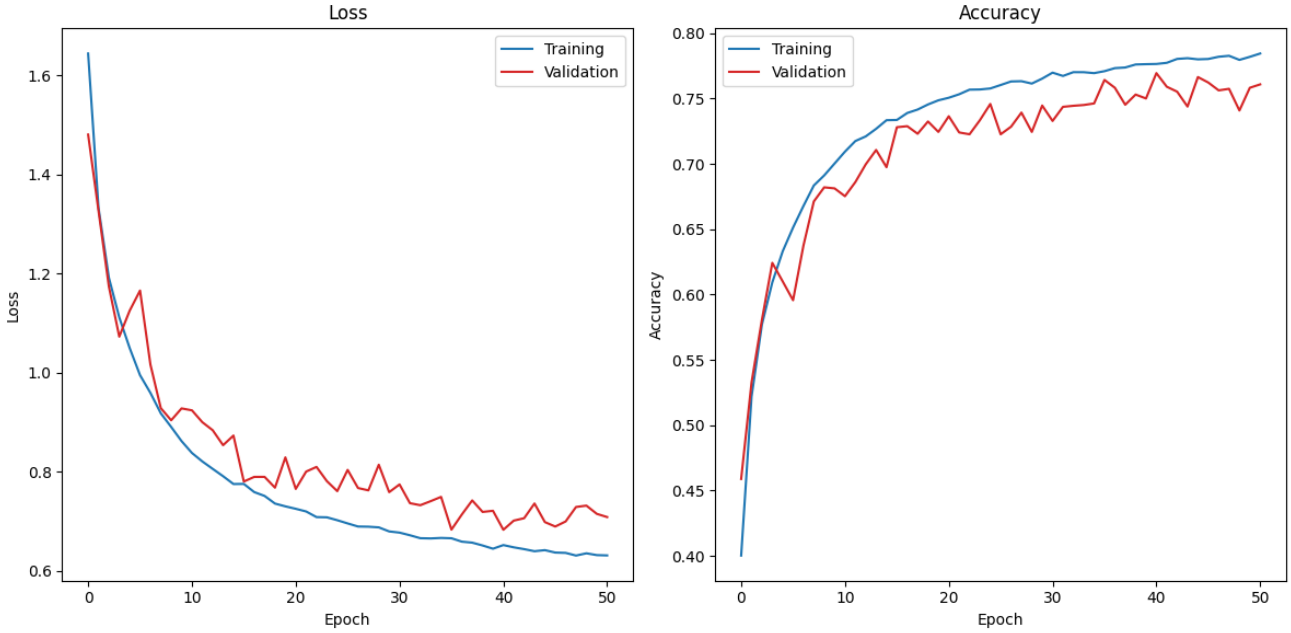


Figure 8 Constant LR 0.001

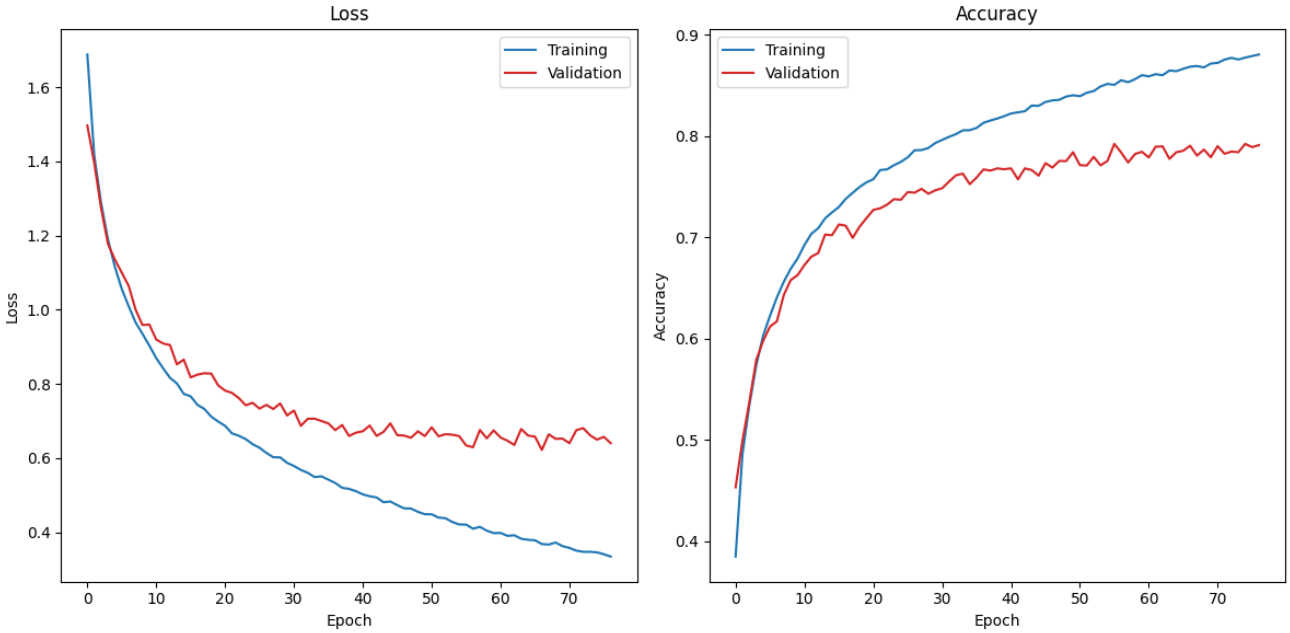


Figure 9 Constant LR 0.0001

(8) What challenges did you encounter, how did you solve them, and what are your thoughts and suggestions regarding this lab? (15%)

Task 1 原本想用兩個 convolutional layer 來訓練，但發現不管怎麼調整 learning rate 或 optimization algorithm，都達不到 80%以上的準確度，後來多加了一層 convolutional layer 後準確度才得到改善。

Task 2 有規定使用 ResNet-18，嘗試了不同的 learning rate、optimizer、learning rate scheduler 和不同的 data augmentation，都難以達到 85%以上的準確度，後來仔細再重新看了助教的錄影後才想到可以調整 batch size，調整為 64 後才終於達到 85%。

助教給程式碼中 train model 的部分有縮排上的錯誤，下次可以更正一下。另外，這個 lab 的目的應該是要教會學生如何調整超參數以訓練出最佳的模型，因此未來的講義或講解影片或許可以說明更多在各種情況下如何判斷應該調整什麼超參數，例如看到 validation loss 震盪太大可以把 learning rate 調小，或是當 validation loss 經過一定 epochs 後都沒有繼續改善可以做 early stop 等等。