EAI Lab2 Report

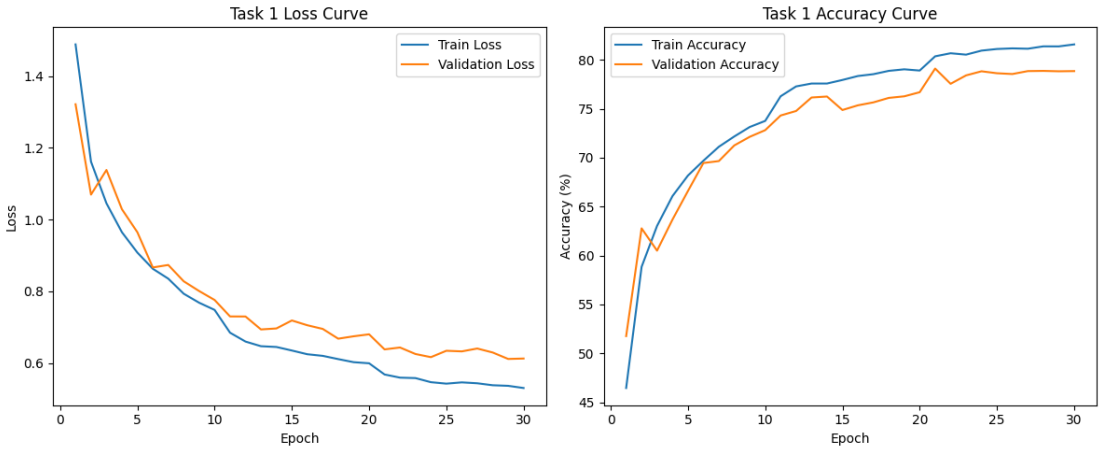
Task 1 – create your own CNN model

(1) Print model summary(including parameters) and plot model(5%)

```
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
           Linear-13                 [-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): 1.42
Params size (MB): 0.44
Estimated Total Size (MB): 1.87
----------------------------------------------------------------
```

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

```
Finished Task 1 Training
Task 1 Test Accuracy: 80.76%
```



(3) Describe how do you choose your best model(5%)

在這次實驗中，我設計了一個三層卷積的 SimpleCNN 模型，每層的設定皆以 kernel size = 3*3,padding = 1 去訓練，模型的每層卷積後面都接了一個 2*2 池化層，用來減少特徵圖的尺寸，這樣可以讓模型更專注於重要的特徵，同時降低計算量。模型最後通過全連接層將學到的特徵用來進行分類預測。

在整個訓練過程中，選擇使用 CrossEntropy 函數來衡量模型預測結果和實際標籤之間的差距。

為了加速模型的訓練和優化，使用了 Adam 優化器。並將學習率初始設置為 0.001，並設定了一個學習率調整策略，每訓練 10 個週期，學習率減半，這樣可以讓模型在訓練後期逐步穩定，減少不必要的波動。

此訓練進行 30 個 epoch，在訓練時每個週期結束後，我們都會用驗證集來檢驗模型的表現，並且 print 出 Train Loss, Train Acc, Val Loss, Val Acc 的值來確認每個 epoch 的訓練狀況，如果當前週期的模型在驗證集上的準確率比之前任何一次都好，我們會把這個模型保存下來。

當模型完成所有訓練週期後，我們會加載這個保存的最佳模型，並在測試集上進行最終的評估。

Task 2 – ResNet 18 implementation

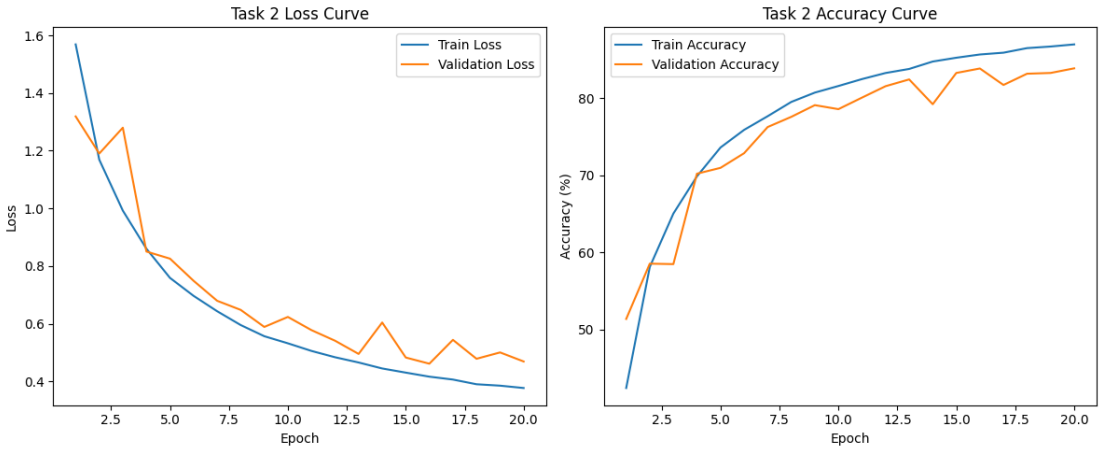
(4) Print model summary(including parameters) and plot model(5%)

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 64, 32, 32]           1,728
       BatchNorm2d-2           [-1, 64, 32, 32]             128
              ReLU-3           [-1, 64, 32, 32]               0
            Conv2d-4           [-1, 64, 32, 32]          36,864
       BatchNorm2d-5           [-1, 64, 32, 32]             128
              ReLU-6           [-1, 64, 32, 32]               0
            Conv2d-7           [-1, 64, 32, 32]          36,864
       BatchNorm2d-8           [-1, 64, 32, 32]             128
         ResBlock-9           [-1, 64, 32, 32]               0
           Conv2d-10           [-1, 64, 32, 32]          36,864
      BatchNorm2d-11           [-1, 64, 32, 32]             128
             ReLU-12           [-1, 64, 32, 32]               0
           Conv2d-13           [-1, 64, 32, 32]          36,864
      BatchNorm2d-14           [-1, 64, 32, 32]             128
        ResBlock-15           [-1, 64, 32, 32]               0
           Conv2d-16          [-1, 128, 16, 16]          73,728
      BatchNorm2d-17          [-1, 128, 16, 16]             256
             ReLU-18          [-1, 128, 16, 16]               0
           Conv2d-19          [-1, 128, 16, 16]         147,456
      BatchNorm2d-20          [-1, 128, 16, 16]             256
           Conv2d-21          [-1, 128, 16, 16]           8,192
      BatchNorm2d-22          [-1, 128, 16, 16]             256
        ResBlock-23          [-1, 128, 16, 16]               0
           Conv2d-24          [-1, 128, 16, 16]         147,456
      BatchNorm2d-25          [-1, 128, 16, 16]             256
             ReLU-26          [-1, 128, 16, 16]               0
           Conv2d-27          [-1, 128, 16, 16]         147,456
      BatchNorm2d-28          [-1, 128, 16, 16]             256
        ResBlock-29          [-1, 128, 16, 16]               0
           Conv2d-30            [-1, 256, 8, 8]         294,912
      BatchNorm2d-31            [-1, 256, 8, 8]             512
             ReLU-32            [-1, 256, 8, 8]               0
           Conv2d-33            [-1, 256, 8, 8]         589,824
      BatchNorm2d-34            [-1, 256, 8, 8]             512
           Conv2d-35            [-1, 256, 8, 8]          32,768
      BatchNorm2d-36            [-1, 256, 8, 8]             512
        ResBlock-37            [-1, 256, 8, 8]               0
           Conv2d-38            [-1, 256, 8, 8]         589,824
      BatchNorm2d-39            [-1, 256, 8, 8]             512
             ReLU-40            [-1, 256, 8, 8]               0
```

```
            Conv2d-41          [-1, 256, 8, 8]         589,824
       BatchNorm2d-42          [-1, 256, 8, 8]             512
          ResBlock-43          [-1, 256, 8, 8]               0
            Conv2d-44          [-1, 512, 4, 4]       1,179,648
       BatchNorm2d-45          [-1, 512, 4, 4]           1,024
              ReLU-46          [-1, 512, 4, 4]               0
            Conv2d-47          [-1, 512, 4, 4]       2,359,296
       BatchNorm2d-48          [-1, 512, 4, 4]           1,024
            Conv2d-49          [-1, 512, 4, 4]         131,072
       BatchNorm2d-50          [-1, 512, 4, 4]           1,024
          ResBlock-51          [-1, 512, 4, 4]               0
            Conv2d-52          [-1, 512, 4, 4]       2,359,296
       BatchNorm2d-53          [-1, 512, 4, 4]           1,024
              ReLU-54          [-1, 512, 4, 4]               0
            Conv2d-55          [-1, 512, 4, 4]       2,359,296
       BatchNorm2d-56          [-1, 512, 4, 4]           1,024
          ResBlock-57          [-1, 512, 4, 4]               0
AdaptiveAvgPool2d-58          [-1, 512, 1, 1]               0
            Linear-59                [-1, 10]           5,130
================================================================
Total params: 11,173,962
Trainable params: 11,173,962
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 13.63
Params size (MB): 42.63
Estimated Total Size (MB): 56.27
----------------------------------------------------------------
```

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

```
Task 2 Test Accuracy: 86.69%
```



(6) Describe how do you choose your best model(5%)

這次實驗中，這個模型的設計理念基於 ResNet 的結構，核心目標是通過動態調整每一層的殘差塊數量來控制網絡的深度，以 ResBlock 作為網絡的基本構建單位，並在每個殘差塊中使用 4（跳躍連接），使網絡在學習複雜特徵時能夠保持梯度的有效傳遞。

在這個設計中，make_layer 函數扮演了關鍵角色。它根據每層所需的殘差塊數量（由 num_blocks 參數控制），動態生成不同數量的 ResBlock。這樣可以靈活地調整每一層的深度，同時確保特徵圖的尺寸和通道數隨著網絡深度的加深而逐步增加。

具體舉例來說，像是網絡的第一層卷積操作將輸入的 RGB 圖像轉換為 64 通道的特徵圖，然後通過四個不同的層來逐步加深網絡。每一層的特徵圖深度和空間尺寸由 make_layer 函數動態決定。這裡的 stride 參數決定了特徵圖的尺寸縮放，例如在 layer2 之後，特徵圖的空間尺寸會縮小一半，而通道數從 64 增加到 128。這樣的設計保證了網絡能夠學習更深層次的特徵，同時減少計算成本。

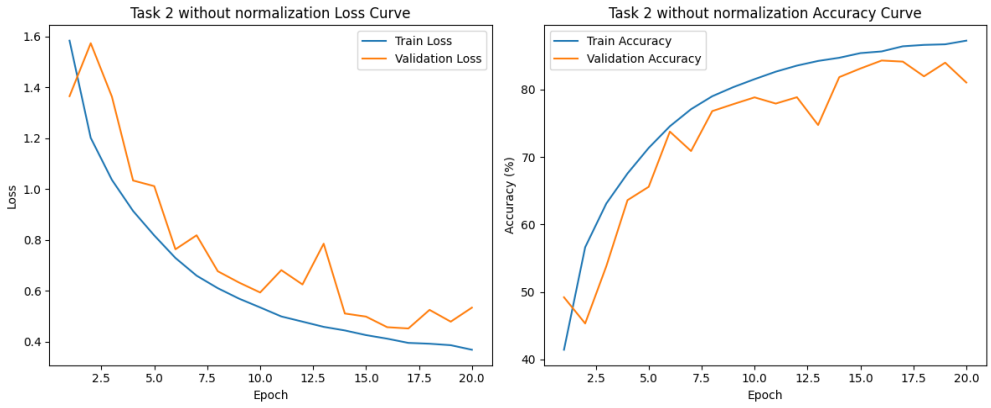
在優化模型的過程中，使用了 Adam 優化器。學習率初始設置為 0.001，並通過 weight_decay 來防止過擬合。

此訓練進行 30 個 epoch，在訓練時每個週期結束後，我們都會用驗證集來檢驗模型的表現，並且 print 出 Train Loss, Train Acc, Val Loss, Val Acc 的值來確認每個 epoch 的訓練狀況，如果當前週期的模型在驗證集上的準確率比之前任何一次都好，我們會把這個模型保存下來。

當模型完成所有訓練週期後，我們會加載這個保存的最佳模型，並在測試集上進行最終的評估。

(7) Experiment on the following and compare the result with baseline

a. Input image normalization (15%)



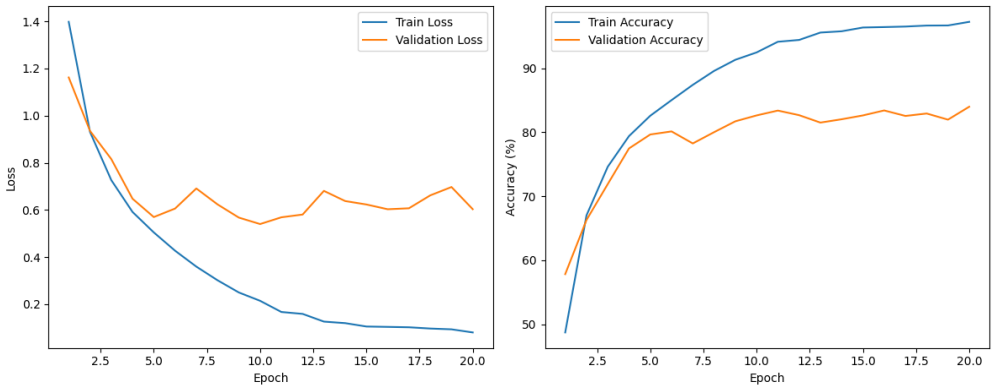
（圖）無使用 normalization 的情況下 plot 出來的 accurancy 和 loss 值

以訓練的成果而言，兩者準確率最終的大小並無太明顯的差異，但從圖表我們可以發現若無進行 normalization，模型在初期訓練時較不穩定，

準確率跟 loss 值都是從較不理想的值開始訓練的，此外收斂速度也明顯比有進行 normalization 的模型訓練來得慢，猜測是因為無 normalization 導致輸入數值範圍較廣，訓練初期產生較大的梯度，導致優化器需要更長的時間來進行調整。
整體而言，我認爲儘管結果上差異不大，但透過 normalization 來去改善模型早期訓練的狀況仍是很重要的一環。

b. Data augmentation (15%)

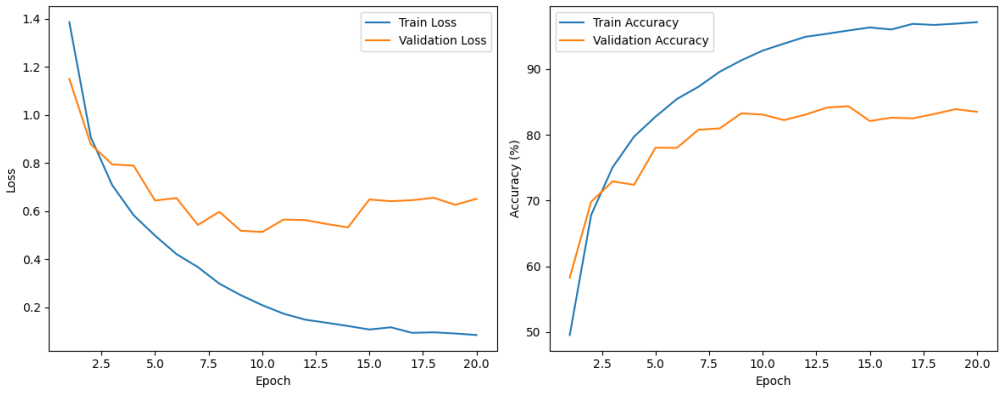
```
Task 2 without data_augmentation Test Accuracy: 84.11%
```



（圖）無使用 Data augmentation 的情況下 plot 出來的 accurancy 和 loss 值

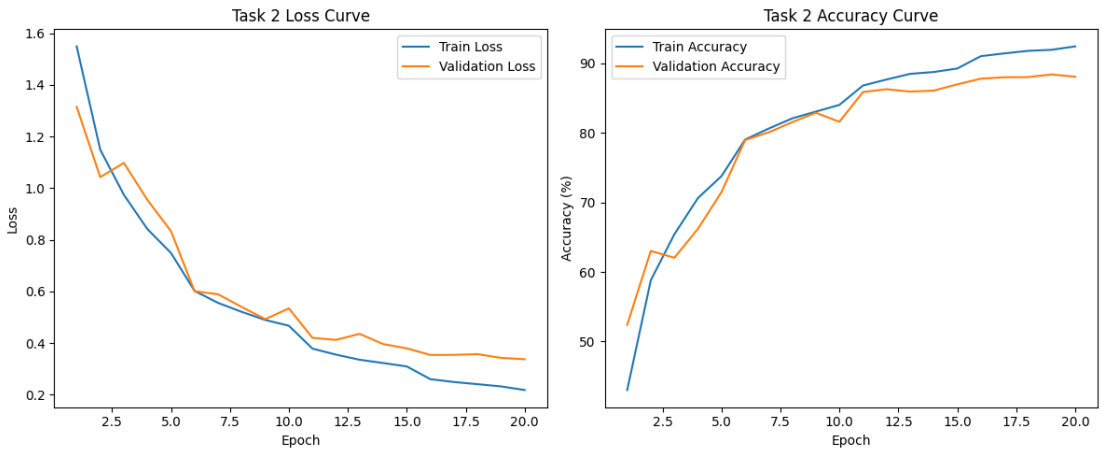
從圖表中我們可以發現，若無使用 data augmentation 的情況下， Train Accuracy 隨著訓練的進行穩步提升，最終接近 100%，同時 Train Loss 也在不斷降低，這表明模型在訓練過程中學得非常好。然而，當觀察到驗證集的表現時，情況則有些不同。
Validation Loss 在訓練的早期隨著 epoch 的增加有所下降，但在訓練的中後期，驗證損失開始波動甚至略微上升，而 Validation Accuracy 則在達到一定水準後趨於平緩，並且與訓練準確率之間出現了明顯的差距。這種現象顯示出模型正在逐漸「過擬合」，意味著模型在訓練集上表現得很好，但在驗證集上並沒有同樣的提升，甚至在最終的 test accurancy 比有使用 data augmentation 來得更低。
整體而言沒有進行 data augmentation 的情況下雖然在訓練上表現良好但驗證集上性能明顯受限，因此透過 data augmentation 來去使模型能夠充分學習也是很重要的。

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



（圖）將學習率從原本的 0.001 改為 0.01 的情況下 plot 出來的 accurancy 和 loss 值

從左邊的損失圖中可以發現 Train Loss 持續下降，且隨著訓練逐漸趨近於 0，這表示模型在訓練集上學得很好。然而 Validation Loss 下降趨勢在初期較為顯著，但在後期（大約第 10 個 epoch 之後）開始波動，並沒有隨著 epoch 增加繼續下降，反而呈現略微上升的情況。這說明模型在驗證集上的性能未能進一步提升，甚至出現了退步的現象。
在右邊的準確率圖中，Train Accuracy 不斷提高，最終接近 100%，顯示出模型在訓練集上幾乎完美地擬合了數據。而 Validation Accuracy 則在早期快速提升，但隨著訓練的進行，達到一定水平後也開始出現波動，並且在 80% 左右徘徊，無法繼續顯著提升。
這樣的現象表明，將學習率提高到 0.01 使模型能夠更快地學習，但同時也加速了模型過擬合的進程。高學習率的好處是模型能夠在前幾個 epoch 中快速學到訓練數據的特徵，因此你會發現模型的 Train Loss 和 Train Accuracy 在早期下降和上升得很快。然而，隨著訓練的進行，高學習率可能導致模型在參數空間中的大幅度跳躍，使得模型難以精細調整權重，從而在驗證集上的表現變得不穩定。



（圖）採用 stepLR 學習率調整策略的情況下 plot 出來的 accurancy 和 loss 值

```
Task 2 Test Accuracy: 90.67%
```

在本次模型訓練中，我們採用了 StepLR 作為學習率調整策略，並設定每 5 個 epoch 將學習率減半。這種設定有助於模型在初期保持較快的學習速度，並在後期逐漸減少學習率，從而穩定模型的收斂。對比使用與未使用學習率調整的情況，可以明顯觀察到模型的表現得到了顯著提升。圖中展示了損失和準確率的變化曲線，清楚顯示出學習率調整對模型訓練效果的影響。

從損失曲線的變化來看，訓練損失和驗證損失在最初幾個 epoch 內都快速下降，表明模型在初期迅速學習到數據中的模式並顯著減少了錯誤。當學習率在第 5 個 epoch 和第 10 個 epoch 等階段減半時，模型的收斂速度有所放緩，但損失值依然持續下降。最終，訓練損失收斂至接近 0.2，而驗證損失在最後的幾個 epoch 中穩定於一個較低的水準，說明模型成功避免了過度擬合，並具有良好的泛化能力。

準確率曲線同樣顯示出，模型的準確率在初始階段快速提升，特別是前 5 個 epoch，這是因為當時的學習率較高，使模型能夠迅速學習數據中的規律。在學習率減半後，準確率增長趨於平緩，但仍在穩定提升。最終，模型在訓練集上的準確率達到了約 91%，而在驗證集上的準確率則穩定在 88% 左右，這表明模型不僅在訓練過程中表現良好，且在驗證過程中也展現出優秀的泛化能力。

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

在這次實驗中，在設計 task1 model 的過程時，我多次對模型的結構進行調整，嘗試了多種參數組合，並測試了不同的優化器和學習率調整策略。甚至也嘗試了像是加入 Dropout 層 等方法，但準確率並未顯著提升。

經過多次嘗試後，我最終選擇了增加一層新的卷積層，這才帶來了準確率的明顯提高。這次的經驗讓我了解到，在模型的設計過程中，細節的調整可能不會立即產生預期的效果，但透過持續的測試和調整，找到適合的架構是非常關鍵的。

其次，與之前的實驗相比，這次的訓練過程顯著耗時更久。由於每次訓練所需的時間較長，我在使用 Colab 進行模型訓練時，經常遇到 訓練中斷 或 限流 的問題，導致訓練過程被強制停止。

為了解決這個問題，我嘗試使用多個 Google 帳號，並將它們設為 Colab 的共同編輯者，這樣能夠延續訓練的進程，避免因為限流導致的訓練中斷。雖然這不是最理想的解決方案，但至少能夠在現有資源下完成實驗。