EAI Lab 1/NM6131027/林偉琦

Task 1 :

1. 在 Task 1 部分主要是需要先自行建立一些參數，包含 ReLU, Softmax 等，以及 NN 網路在 Forward 以及 Backward 的部分，這邊根據助教給的資訊都能快速建立完成。接著就是切分 train, val, test 這邊我們選擇使用 6:2:2 比例去切分，在 optimizer 的部分助教有提供 SGD 作為參考，但這邊後續使用 AdamW ，主要是考量到 AdamW 的優勢：自適應學習率 **(Adaptive Learning Rate)** 與 更有效的權重衰減 **(Decoupled Weight Decay)**，使其能有效率且穩定地在複雜的損失函數平面上找到最佳解。

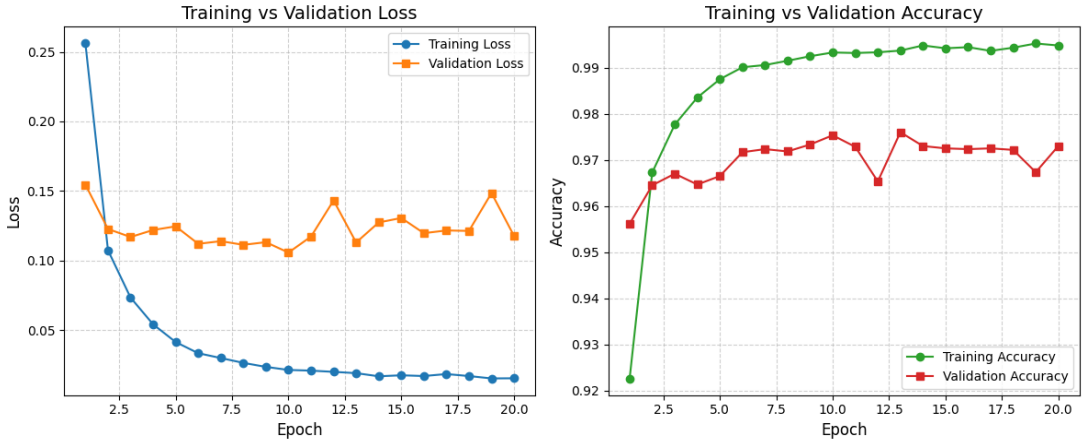
2. Hyper Parameters:
   - Learning Rate : 1e-4
   - Epoch : 20

3. 模型架構

```python
class MLP(Module):
    def __init__(self) -> None:
        # Assume the iuput feature of MNIST is 784d
        self.fc1 = Linear(784, 128)
        self.relu1 = ReLU()
        self.fc2 = Linear(128, 10)
        self.softmax = Softmax()

    def forward(self, x):
        # 順序: Linear -> ReLU -> Linear -> Softmax
        x = x.reshape(x.shape[0], -1)
        out = self.fc1(x)
        out = self.relu1.forward(out)
        out = self.fc2.forward(out)
        out = self.softmax.forward(out)
        return out

    def backward(self, dy):
        # 反向順序：Softmax → fc2 → ReLU → fc1
        dy = self.softmax.backward(dy)
        dy = self.fc2.backward(dy)
        dy = self.relu1.backward(dy)
        dx = self.fc1.backward(dy)
        return dx
```

MLP 的架構依序由一個輸入層、一個隱藏層和一個輸出層所組成，模型先把影像攤平成 784 維的向量當作輸入，接著經過第一層 128 個神經元的隱藏層，並且使用 ReLU 作為 activation function，接著模型通過一個 fully connect layer 的輸出層，把學到的特徵轉成 10 個類別的預測分數。

4. 結果



訓練時一開始將 lr 設為 1e-4 表現就算穩定，最終在 test_acc = 0.9736

Task 2 :

1. 這份作業是 CIFAR10，最主要是先需要先手刻出 Resnet 18 的架構，然後再來是 Data Augmentation，我們這邊使用了 RandomRotation 角度選擇 10 度，讓圖像不會和原先差太多，另外還有 ResizedCrop，主要是因爲 CIFAR10 原先照片的大小為 32*32，但是原始的 ResNet 是採用 InageNet 訓練集去做

training，並且是先放大到 256，再隨機裁成 224 的大小。而 CIFAR-10 的原圖只有 32 x 32，所以將 CIFAR-10 透過 Resize 放大到 256 再隨機裁成 224，透過做此步驟能讓準確率上升不少。後續則和 Task1 沒有太大差別，Learning rate 的部分有試過 1e-3, 1e-4，後者的收斂速度較快，另外為了讓效果更好，也有加入 Learning Rate Scheduler，也有做比較。

2. Data augmentation
   - RandomCrop

此實驗須比較使用不同 data augmentation 方法的差異。完全不使用任何 data augmentation 的狀況下，test_acc = 76.58，加上 RandomCrop 和 RandomHorizontalFlip 之後 test_acc 提升到 86 左右。在其他實驗中 data augmentation 也都採用 RandomCrop 和 RandomHorizontalFlip。

```
transform_train = transforms.Compose([
    transforms.Randomcrop(32, padding=4),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(), # Transform to tensor
    transforms.Normalize(mean=train_mean, std=train_std), # Normalization
])
```

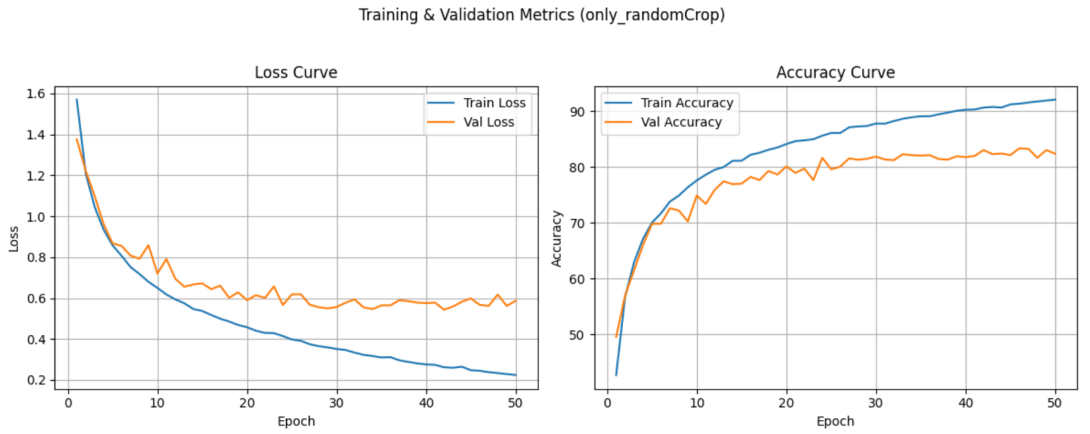
   - Resize

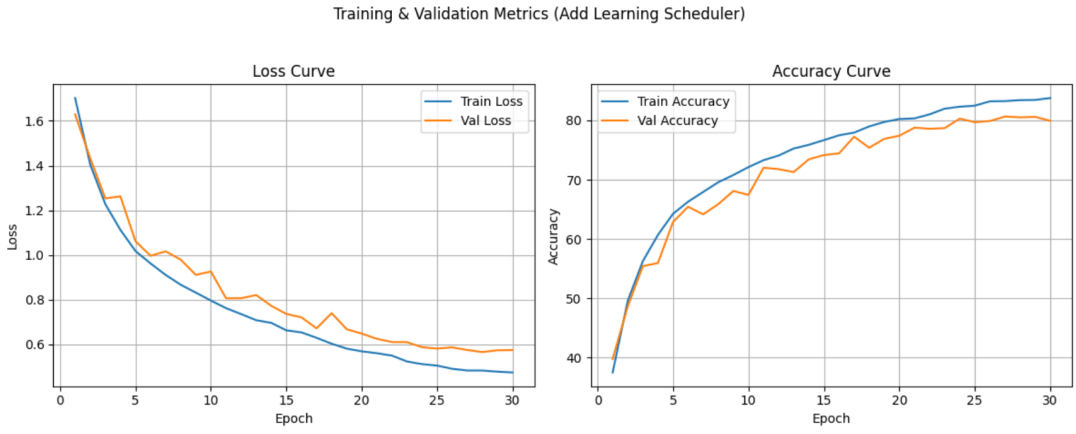
```python
##### data augmentation & normalization #####
transform_train = transforms.Compose([
    transforms.RandomResizedCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(10),
    transforms.ToTensor(), # Transform to tensor
    transforms.Normalize(mean=train_mean, std=train_std), # Normalization
])

transform_test = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=test_mean, std=test_std),
])
```

3. Test 結果：

```
Best Model Test Accuracy: 90.03%
```

4. 不同參數比較：





5. 遇到的問題及解決方法：

在 Task1 遇到的問題是要如何設計 MLP Layer，一開始的層數設定過多導致表現不好，後續則是慢慢調整，加入 MNIST 並不會有太複雜的資料，所以用 2-3 層 Layer 就能有不錯的表現了。

在 Task2 則是一開始沒有做 Data Augmentation 時，只有單純條整 lr, epoch 表現沒有到特別好，在網路搜尋資料後才知道可能是沒有做 ResizedCrop 的部分，因此稍作調整後結果跟準確率就有上升了。