HW4 成果分析

本次作業實作兩層的神經網路。

資料集狀況

CIFAR10 dataset 的資料是數筆 32*32 pixels 的彩色影像，並且有數個類別。

```
Data type of images: uint8 (value range: [0, 255])
Shape of images: (50000, 32, 32, 3)
Shape of labels: (50000,)
Number of classes: 10
Classes: ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse'
```

把他們分成訓練資料、驗證資料還有測試資料：

```
Number of training data: 30000
Number of validation data: 10000
Number of test data: 10000
```

全連接層

建立兩層全連接層結構:
Input $\rightarrow$ Fully-Connected Layer1 $\rightarrow$ ReLU $\rightarrow$ Fully-Connected Layer2 $\rightarrow$ Softmax
正向傳播、反向傳播和作業三大同小異，但由於我們有連接數層的 Activation function和神經元，所以要將上一層的 `backprop` 結果傳進來進行 Chain rule 。

```
#Forward
out = torch.mm(x, w) + b

#Backward
dw = torch.mm(x.T, backprop) / N
db = torch.mean(backprop, dim=0)
```

ReLU

ReLU 定義為：

$$ReLU(t) = max(0, t)$$

$$ReLU'(t) = \begin{cases} 0, & \text{if } x > 0 \\ 1, & \text{otherwise.} \end{cases}$$

寫成程式碼：

```
#Forward
out = torch.max(t, torch.zeros_like(t))

#Backward
dt = (t > 0).float() * backprop
```

其中， `backprop` 為上一層反向傳播的結果。

Softmax

Softmax 定義為：

$$softmax(\mathbf{S}_i) = \mathbf{p}_i = \frac{exp(\mathbf{S}_i)}{\sum_{j=1}^{\mathbf{C}} exp(\mathbf{S}_j)}$$

$$softmax'(\mathbf{S}_i) = (\hat{y} - y_{\mathbf{one\ hot}})$$

寫成程式碼：

```
#Forward
    prob = torch.exp(net_out) / torch.sum(torch.exp(net_out), dim=1,
keepdim=True)
    pred_y = prob.argmax(dim=1)
    y_one_hot = torch.zeros_like(net_out)
    y_one_hot[np.arange(N), y] = 1

#Backward
    y_one_hot = torch.zeros_like(prob)
    y_one_hot[np.arange(N), y] = 1
    grad = (prob - y_one_hot) / N
```

其中， `backprop` 為上一層反向傳播的結果。

Cross-Entropy Loss Function

最後，再加上 loss function 就可以了。因為要做分類，所以選用 cross-entropy Loss function。

$$CE = -\sum \mathbf{y}_i * log(\mathbf{p}_i)$$

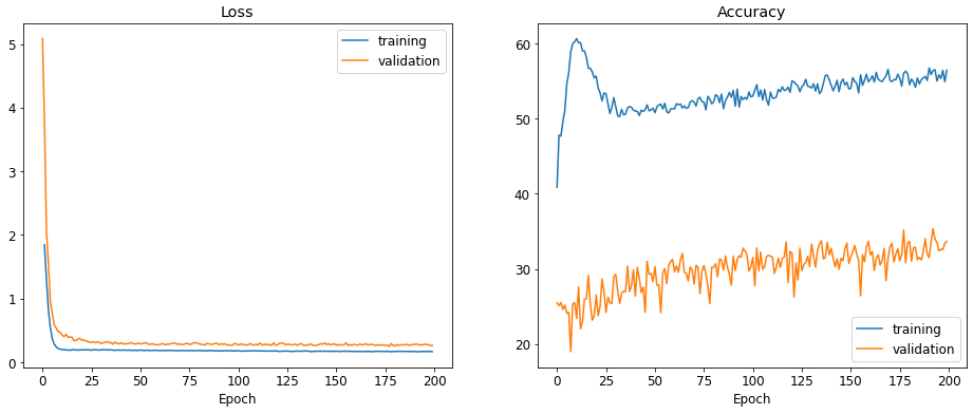
```
loss = -torch.sum(y_one_hot * torch.log(prob)) / N
```

執行結果與分析

將剛剛建立的所有層連在一起，進行訓練。
找到最優的超參數：

```
BATCH_SIZE = 8
LR = 1e-2
LR_DECAY = 1
HIDEEN_DIM = 256
```

訓練結果：



我們發現，驗證資料的準確率遠低於訓練資料。且只有將近40%的準確率。有可能是網路模型的 Capacity 不足，導致判斷失準

改良

嘗試新增了一層隱藏層：

```
# Layers in Deeper_Classifier
layer1 = FullyConnectedLayer()
activation1 = ReLU()
layer2 = FullyConnectedLayer()
activation2 = ReLU()
layer3 = FullyConnectedLayer()
self.net = [layer1, activation1, layer2, activation2, layer3]
self.loss_func = Softmax_CrossEntropy_overflow()
```

測試過程中，發現在進行 Softmax 運算時，因為指數過大造成 overflow 。所以對 Softmax 函數做了一些改善，對 `net_out` 減去他們的最大值：

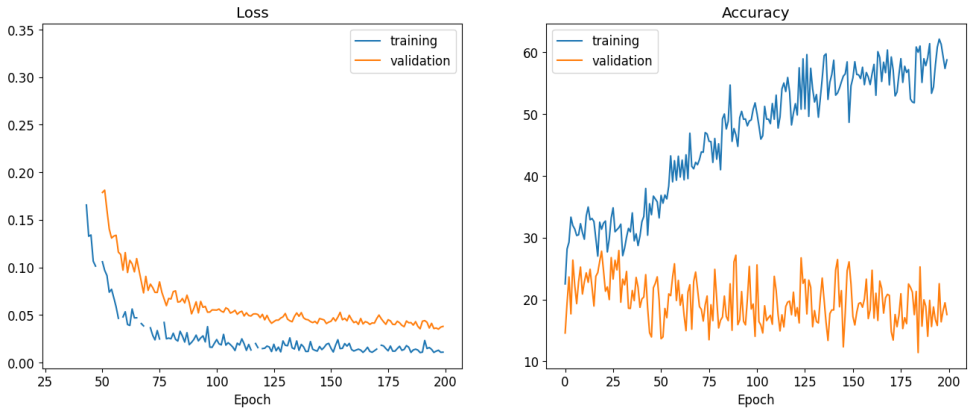
```
# Softmax_CrossEntropy (overflow prevented)

def forward(self, y, net_out):
    N, F = net_out.shape
    net_out_max = torch.max(net_out, dim=1, keepdim=True)[0]
    # prevent overflow
    net_out_exp = torch.exp(net_out - net_out_max)
    prob = net_out_exp / torch.sum(net_out_exp, dim=1, keepdim=True)
    self.cache = (y, prob)
    pred_y = prob.argmax(dim=1)
    y_one_hot = torch.zeros_like(net_out)
    y_one_hot[np.arange(N), y] = 1
    loss = torch.zeros(1)
    loss = -torch.sum(y_one_hot * torch.log(prob)) / N
    return pred_y, loss
```

超參數設置如下：

```
BATCH_SIZE = 128
LR = 1e-2
LR_DECAY = 1
HIDEEN_DIM1 = 1024
HIDEEN_DIM2 = 128
```

結果發現雖然訓練準確率有提升，且看起來仍有持續上升的可能，但是模型 overfitting 了，測試結果 (test_data) 也只有15%的準確率。或許日後的嘗試要再加入 Dropout 看看是否能改善結果。



此外，兩層隱藏層在進行正向傳播時有時 `prop` 的數值及小 (<E-70) 所以在進行 CE Loss 的計算時讓 log 趨近於負無限大 (underflow) 。所以在 loss 曲線圖中間出現了一些斷點。目前也不知道如何解決，希望能於日後的學習中逐步改善。