AI training HW3

STuser19 賴昱凱

Best accuracy: 90.11%

Optimizer:

原先的 training skills 使用的是

Normalization: None, Activation function: ReLU, Dropout: 0.2

Optimizer: SGD (lr=0.0001)

Scheduler: None

可以看到 accuracy 為 55.57%，十分不準確，因此我的第一個想法就是更改 Optimizer，因為原先使用的 SGD 並沒有考慮先前的梯度，只使用當下梯度進行，因此非常容易被局部的梯度影響導致不容易降到 local minimum，因此我考慮了幾個不同的 Optimizer，包括 SGD with weight、Adam、AdamW、Lamb 等，更改之後的 accuracy 如下:

SGD with weight (momentum=0.9, weight_decay=0.0005): 85.72%

Adam (lr = 0.0001): 88.11%

AdamW (lr = 0.0001): 87.51%

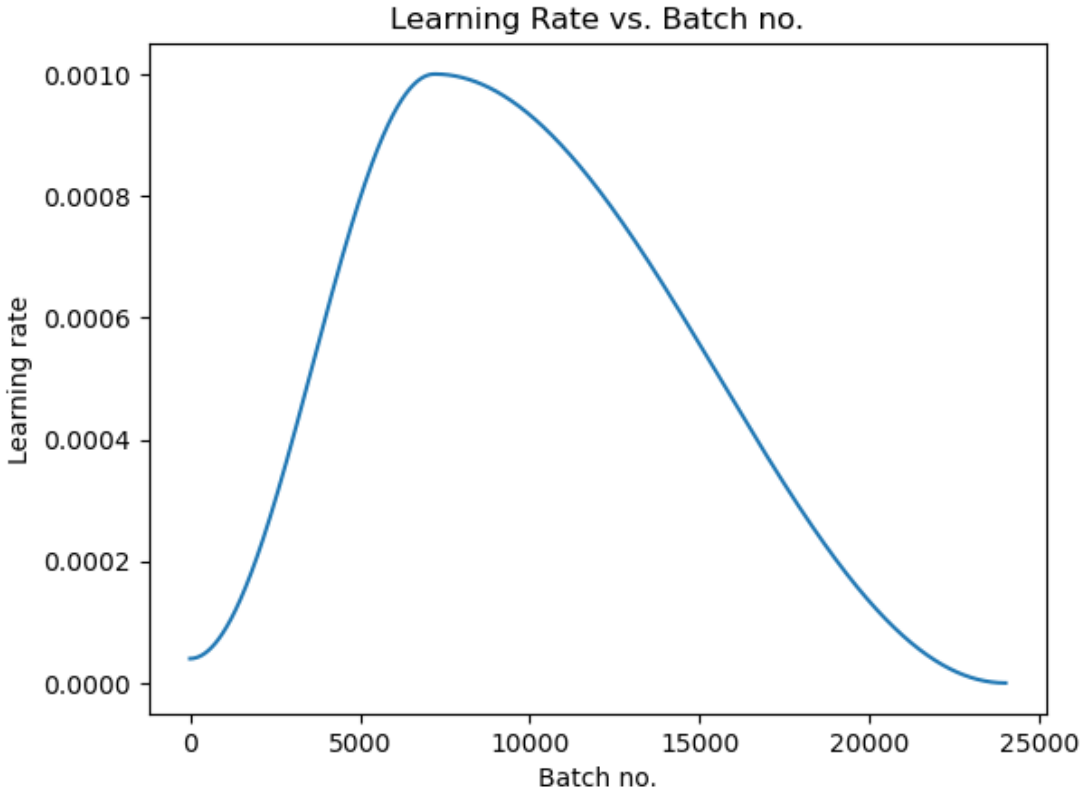
Lamb (lr = 0.0001): 82.96%

其中 Lamb 的準確率最低，因為 Lamb 適合的是 batch size 很大的情況，可是在本模型中 batch size 預設在 20，是一個很小的 batch，因此表現不佳也是可預期的。而 SGD with weight 也因為架構相對簡單，相較 Adam 來說，比較無法靈活改變 weight。因此由比較的結果，後續我皆使用 Adam 來當作 Optimizer。

Scheduler:

而第二步我把重點放在 Learning rate 的變化，由於 Optimizer 的 Learning rate 都是固定不變的，因此我們需要自己設定 Scheduler 來適當改變 training 過程中的 Learning rate。

一開始我的想法是讓 Learning rate 有個 warning up 的趨勢，讓梯度先在初始點附近尋找適當的方向再加大步伐，因此我選擇了 OneCycleLR，並先都使用預設的參數設定，Max_lr 則設定成 0.001，得到結果 accuracy 為 89.03%



Dropout:

Dropout 的比例對於 accuracy 也有很大影響，由於這個模型小，架構也簡單，因此將 dropout 的比例降低是有利於避免在訓練模型時過度依賴某些樣本導致 overfitting，進而使訓練結果不佳，而我多次嘗試也發現的確有此項趨勢。

Dropout(0.5): 88.06%

Dropout(0.2): 89.03%

Dropout(0.1): 89.05%

因此後續使用 Dropout(0.1)進行訓練。

Normalization:

更改相關參數無法再有效增加 accuracy，因此我把改進方向轉至 Normalization 上，我在前兩層計算結束後使用了 Batch Normalization，accuracy 結果為 88.88%，反而變低了，我推測是由於會需要計算 Batch 的 mean 與 variance，需要一定的樣本數量，因此原先的 batch size = 20 太少了，比統計學中對於大樣本數的定義 30 個還少，因此我將 Batch size 增加至 100，使每一次 Batch 的樣本分佈更接近母體，結果提升到了 89.38%，證明我的猜想是正確的。

Activation function:

接下來我想了解不同的 activation function 的效果如何，測試結果如下:

ReLU: 89.52%

PReLU: 88.48%

GELU: 89.57%

Softmax: 88.05%

Sigmoid: 85.44%

ReLU 及 GELU 在此模型上表現差不多，且 ReLU 計算量明顯小於 GELU，因此之後就繼續使用 ReLU。

至於 Softmax 表現差的原因應該是因為當我們選擇使用 CrossEntropyLoss 當作 loss function 時，就已經在該函數中使用過 Softmax 了，因此再多使用一個並不會讓結果更好。而 Sigmoid 則是適合用在二分法，不適合用在此題目中，因此表現特別差。

調整參數:

在此時的情況下，我發現 warning up 並不會增加 accuracy，反而增加初始的 learning rate 可以讓 loss 有更大幅度的下降，因此我將參數調整並尋找最佳的組合，最後使用的 learning rate 如下:

torch.optim.lr_scheduler.OneCycleLR(optimizer, max_lr=0.0035, steps_per_epoch=len(train_loader), epochs=n_epochs, div_factor=0.7, final_div_factor=100, pct_start=0.4)

而最後測試出來最好的 training skills:

Normalization: Batch Normalization

Activation function: ReLU

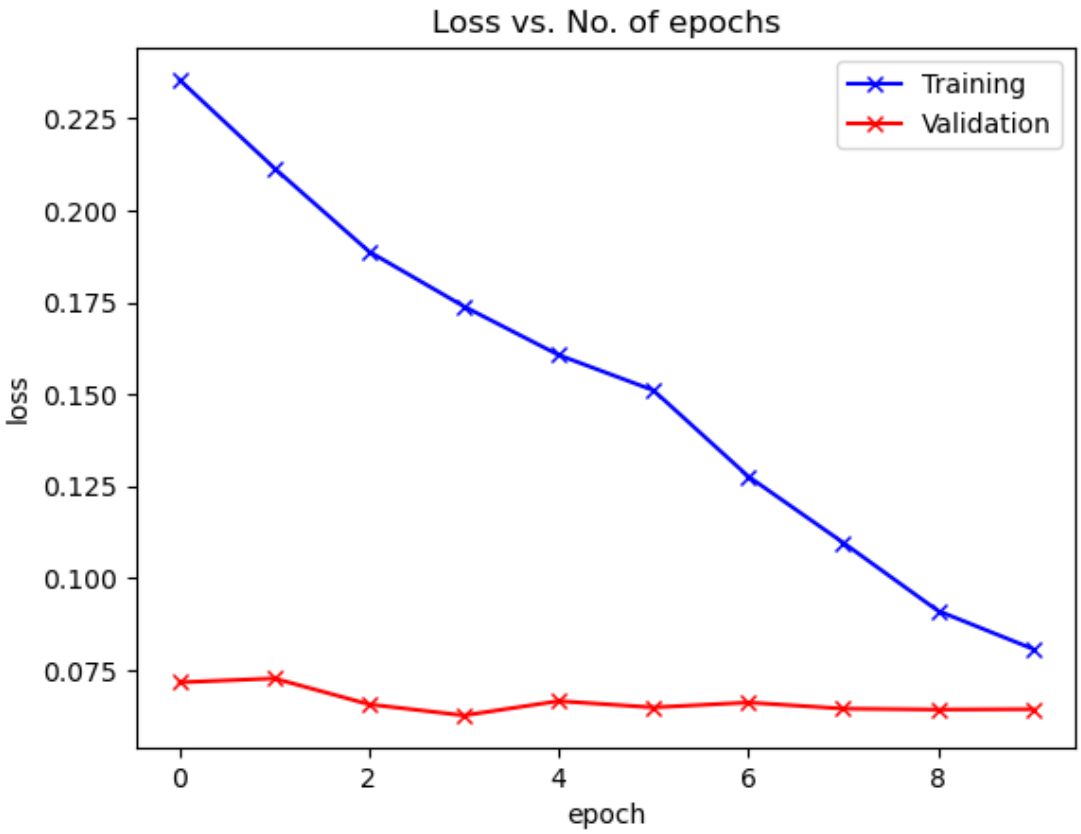
Dropout: 0.1

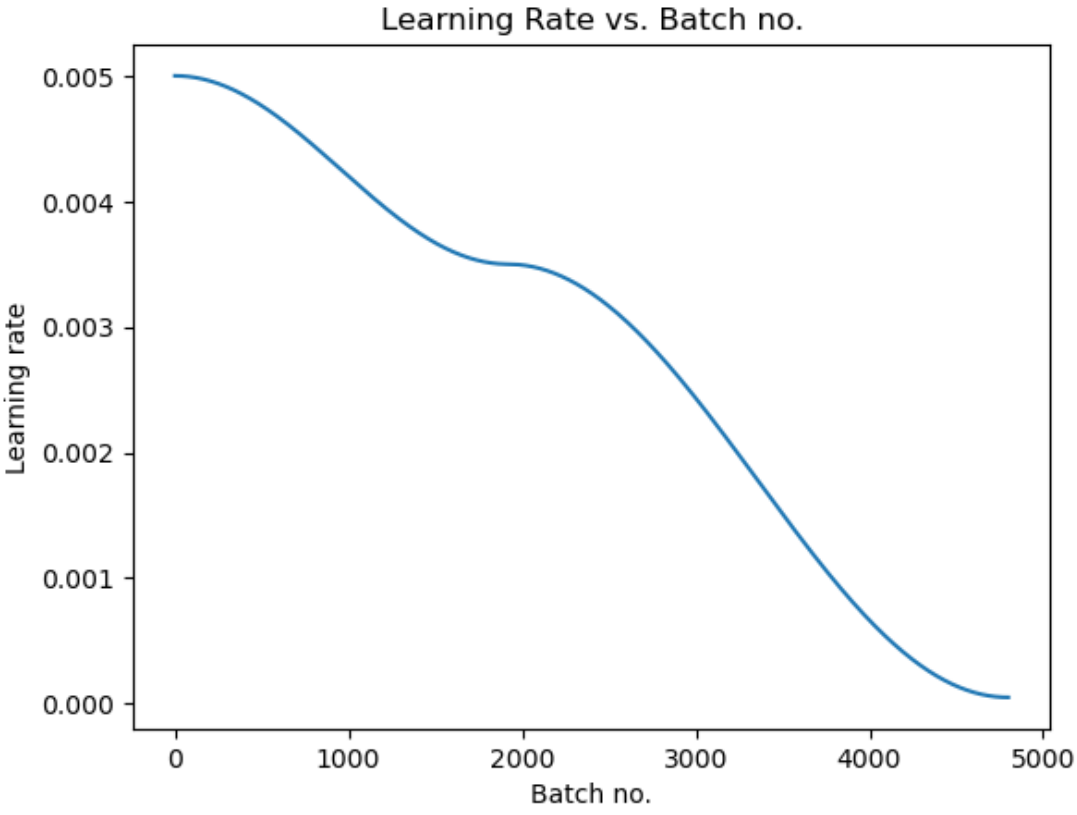
Optimizer: Adam

Scheduler: OneCycleLR(optimizer, max_lr=0.0035, steps_per_epoch=len(train_loader), epochs=n_epochs, div_factor=0.7, final_div_factor=100, pct_start=0.4)

其中我也有嘗試的調整 Adam 中的 betas，但最後測試出來還是預設的表現最佳，因此皆使用預設的參數。

Best result:


Loss vs. No. of epochs
Training
Validation
loss
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0
2
4
6
8
epoch


Learning Rate vs. Batch no.
Learning rate
0.005
0.004
0.003
0.002
0.001
0.000
0
1000
2000
3000
4000
5000
Batch no.

```
Test Loss: 0.336408

Test Accuracy of Class     0: 85.20% (852/1000)
Test Accuracy of Class     1: 98.00% (980/1000)
Test Accuracy of Class     2: 83.10% (831/1000)
Test Accuracy of Class     3: 91.10% (911/1000)
Test Accuracy of Class     4: 83.70% (837/1000)
Test Accuracy of Class     5: 96.70% (967/1000)
Test Accuracy of Class     6: 73.50% (735/1000)
Test Accuracy of Class     7: 95.90% (959/1000)
Test Accuracy of Class     8: 98.00% (980/1000)
Test Accuracy of Class     9: 95.90% (959/1000)

Test Accuracy (Overall): 90.11% (9011/10000)
```