Deep learning lab 01

Task 1:

```
Task1 | Epoch: 43 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2581 |Val Acc:96.5200
Task1 | Epoch: 44 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2592 |Val Acc:96.5200
Task1 | Epoch: 45 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2602 |Val Acc:96.5200
Task1 | Epoch: 46 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2611 |Val Acc:96.5200
Task1 | Epoch: 47 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2620 |Val Acc:96.5100
Task1 | Epoch: 48 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2629 |Val Acc:96.5100
Task1 | Epoch: 49 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2637 |Val Acc:96.5200
Task1 | Epoch: 50 |Train Loss:  0.0000 |Train Acc:100.0000 |Val Loss:  0.2644 |Val Acc:96.5300
```

Task 2:

```
Task2 | Epoch: 44 |Train Loss:  0.0350 |Train Acc:99.3780 |Val Loss:  0.3967 |Val Acc:95.7800
Task2 | Epoch: 45 |Train Loss:  0.0276 |Train Acc:99.4700 |Val Loss:  0.4444 |Val Acc:95.5200
Task2 | Epoch: 46 |Train Loss:  0.0353 |Train Acc:99.3520 |Val Loss:  0.4657 |Val Acc:95.2900
Task2 | Epoch: 47 |Train Loss:  0.0294 |Train Acc:99.4320 |Val Loss:  0.4642 |Val Acc:95.6700
Task2 | Epoch: 48 |Train Loss:  0.0295 |Train Acc:99.4320 |Val Loss:  0.5104 |Val Acc:95.7100
Task2 | Epoch: 49 |Train Loss:  0.0284 |Train Acc:99.4980 |Val Loss:  0.4419 |Val Acc:95.8200
Task2 | Epoch: 50 |Train Loss:  0.0361 |Train Acc:99.3540 |Val Loss:  0.4207 |Val Acc:95.5300
```

Network	Epoch	Batch size	Learning rate	T_ACC	V_ACC	Activation function	Loss function
784-60-30-15-10	75	500	0.001	93.6840	92.090	ReLu	Softmax and cross-entropy loss
784-256-128-64-10	75	500	0.001	93.7080	91.6500	ReLu	Softmax and cross-entropy loss
發現 784-256-128-64-10 的架構成長空間還很大，因此再針對他的 hyperparameter 去修改 主要是發現第一個架構已經不多飽和了(batch_size 下降不會更好，learning rate 及 epoch 往上調也沒太大進步)							
784-256-128-64-10	75	500	0.01	100	95.030	ReLu	Softmax and cross-entropy loss
784-256-128-64-10	50	50	0.01	100	96.3100	ReLu	Softmax and cross-entropy loss
784-256-128-64-10	50	25	0.01	100	96.5300	ReLu	Softmax and cross-entropy loss

How to improve

Design method:

一開始是設計兩層 NN，但發現跟 kaggle 上的 accuracy(約飽和在 92%)還是有一段差距。因此把架構設計成三層，發現 accuracy 有上升的趨勢。之後又發生飽和，因此最終設定四層。

Hyperparameter:

Epoch: 通常會選大一點(>=50)，如有發現 validation accuracy 有飽和趨勢，再慢

慢往下調整。Validation accuracy 會飽和的可能原因在於 model 開始 overfitting，所以此時把 Epoch 往上調也無濟於事。

Learning rate: 通常會從 0.0001 開始調，但發現在某些狀況下 converge 速度過慢，因此調成 0.001。而如果 0.01 的話，容易發現 validation accuracy 跳來跳去，不容易往最小值移動。

Batch_size: 剛開始設為 1000，但後來發現 Batch_size 的縮小可以使 accuracy 上升，但會有一個極值。多次嘗試之後，發現 Batch_size 為 50 的效果最好。缺點是，隨著 batch_size 的下降，意味者一個 epoch 要更新的次數增加，導致了一個 epoch 訓練的時間比大 batch_size 還要長，但因為訓練效果比較快到飽和，所以 epoch 不用這麼大。不過後來使用其他 hidden layer 的參數，發現會有不同適合的 hyperparameter。在某一個嘗試(784-60-30-15-10)中，最適合的 batch_size 為 500，learning rate 則為 0.001。

What differences do you find between the results of Task1 and Task2?

發現同一組超參數套進去 Task2 (架構一樣)，但其 validation accuracy 差異很大。Task 1 與 Task2 雖然架構一樣，適合的超參數卻不一樣，可能的原因是 pytorch 通過 optimizer.step()，自動進行 back propogation 和 weight 更新。Adam optimizer 有內置的最佳化策略(例如動量，學習率調整等)，這些因素可能會影響結果。而另外一個可能的原因為 initialization，pytorch 通常會自動為 network 採用較好的初始化方式，有助於改善訓練效果。但沒有使用 framework 的情況下，可能會使用不佳的初始化方式，進而導致收斂速度過慢或陷入 local minimum。因此，個別適合兩個架構的參數會不一樣。

層數	epoch	Batch_size	Learning rate	T. ACC	V.ACC	Activation function	Loss function
TASK1: 784-256-128-64-10	50	25	0.01	100	96.5300	ReLu	Softmax and cross-entropy loss
TASK2: 784-256-128-64-10	50	100	0.001	99.5580	95.5400	ReLu	Softmax and cross-entropy loss
TASK2: 784-256-128-64-10	50	50	0.001	99.3540	95.5300	ReLu	Softmax and cross-entropy loss