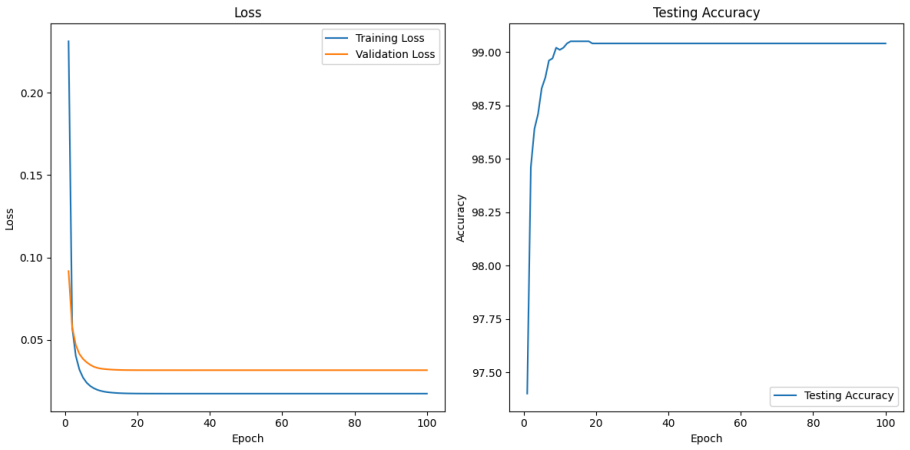
HW3.1



HW3.2

```
Epoch 1: Train Loss: 0.2311| Train Accuracy: 92.91% , Val Loss: 0.0917| Val Accuracy: 97.07% , Test Loss: 0.0764| Test Accuracy: 97.40%
Epoch 2: Train Loss: 0.0575| Train Accuracy: 98.29% , Val Loss: 0.0577| Val Accuracy: 98.22% , Test Loss: 0.0472| Test Accuracy: 98.46%
Epoch 3: Train Loss: 0.0402| Train Accuracy: 98.82% , Val Loss: 0.0469| Val Accuracy: 98.57% , Test Loss: 0.0392| Test Accuracy: 98.64%
Epoch 4: Train Loss: 0.0319| Train Accuracy: 99.05% , Val Loss: 0.0414| Val Accuracy: 98.83% , Test Loss: 0.0346| Test Accuracy: 98.71%
Epoch 5: Train Loss: 0.0270| Train Accuracy: 99.21% , Val Loss: 0.0384| Val Accuracy: 98.87% , Test Loss: 0.0324| Test Accuracy: 98.83%
Epoch 6: Train Loss: 0.0240| Train Accuracy: 99.29% , Val Loss: 0.0363| Val Accuracy: 98.93% , Test Loss: 0.0309| Test Accuracy: 98.88%
Epoch 7: Train Loss: 0.0219| Train Accuracy: 99.37% , Val Loss: 0.0348| Val Accuracy: 98.97% , Test Loss: 0.0298| Test Accuracy: 98.96%
Epoch 8: Train Loss: 0.0206| Train Accuracy: 99.41% , Val Loss: 0.0336| Val Accuracy: 98.97% , Test Loss: 0.0291| Test Accuracy: 98.97%
Epoch 9: Train Loss: 0.0196| Train Accuracy: 99.44% , Val Loss: 0.0329| Val Accuracy: 98.98% , Test Loss: 0.0286| Test Accuracy: 99.02%
Epoch 10: Train Loss: 0.0189| Train Accuracy: 99.47% , Val Loss: 0.0325| Val Accuracy: 98.98% , Test Loss: 0.0282| Test Accuracy: 99.01%
Epoch 11: Train Loss: 0.0185| Train Accuracy: 99.47% , Val Loss: 0.0322| Val Accuracy: 98.97% , Test Loss: 0.0279| Test Accuracy: 99.02%
Epoch 12: Train Loss: 0.0181| Train Accuracy: 99.49% , Val Loss: 0.0320| Val Accuracy: 98.97% , Test Loss: 0.0278| Test Accuracy: 99.04%
Epoch 13: Train Loss: 0.0179| Train Accuracy: 99.49% , Val Loss: 0.0319| Val Accuracy: 98.97% , Test Loss: 0.0277| Test Accuracy: 99.05%
Epoch 14: Train Loss: 0.0177| Train Accuracy: 99.50% , Val Loss: 0.0318| Val Accuracy: 99.00% , Test Loss: 0.0276| Test Accuracy: 99.05%
Epoch 15: Train Loss: 0.0176| Train Accuracy: 99.49% , Val Loss: 0.0317| Val Accuracy: 99.00% , Test Loss: 0.0276| Test Accuracy: 99.05%
```

```
Epoch 92: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 93: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 94: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 95: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 96: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 97: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 98: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 99: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
Epoch 100: Train Loss: 0.0173| Train Accuracy: 99.51% , Val Loss: 0.0316| Val Accuracy: 99.00% , Test Loss: 0.0275| Test Accuracy: 99.04%
```

final test accuracy = 99.04%

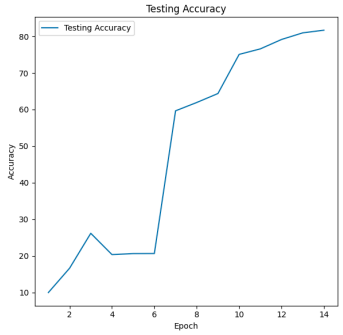
資料分割策略:

我利用torch.utils.data.random_split函數將完整的MNIST資料集的訓練集劃分成90%的訓練集和10%的驗證集，確保訓練過程資料的多樣性，以及使用驗證集檢測模型性能時可以確保模型的泛化能力、避免過擬合的狀況，並用於調整模型超參數和檢查模型在未知資料上的表現。還有使用MNIST資料集的測試集，這部分資料完全未參與模型的訓練或驗證，用於最終模型效能的評估。

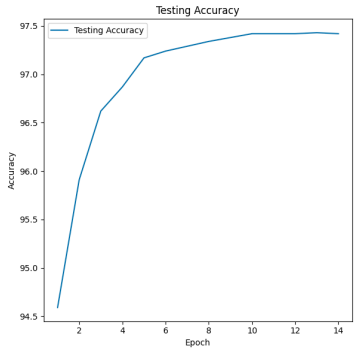
HW3.3

本次作業做了許多實驗，測試了不同的batch size、learning rate和epoch及gamma變化，以下是實驗結果的重點整理:

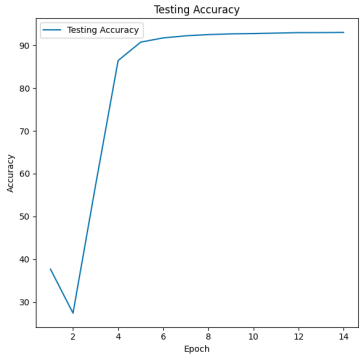
batch size變化:



左圖是**batch size = 32**的結果，模型在訓練過程中的準確率波動較大，且收斂速度較慢。



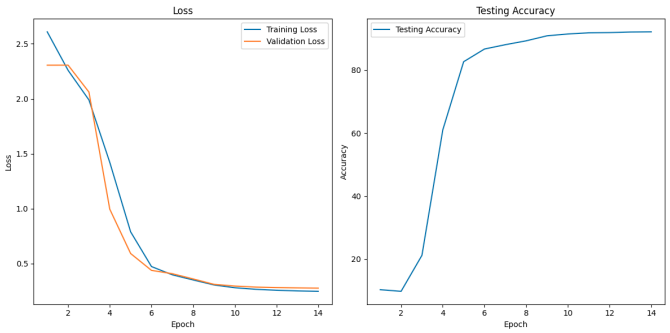
這是**batch size = 64** 的結果，模型在初期就達到94%以上的準確率，後面也隨著訓練過程穩定上升，直到97.5%左右逐漸收斂，這個設定的表現最佳。



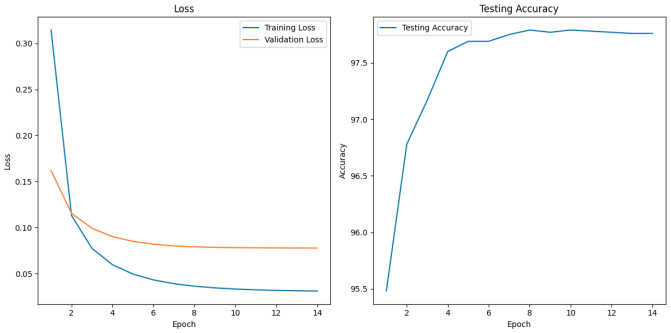
這是**batch size = 128**的結果，在前兩個epoch從37%掉到28%，隨後逐漸上升，到第四個epoch時準確度大約86%，並且最終在90%左右收斂，準確度相較**batch size = 32**的低。

從這項實驗中可以發現：
batch size太小或太大可能造成模型表現不穩定或表現不佳，適中的batch size（例如實驗中設定為64的結果）可以在訓練的穩定性及準確度取得更好的平衡。

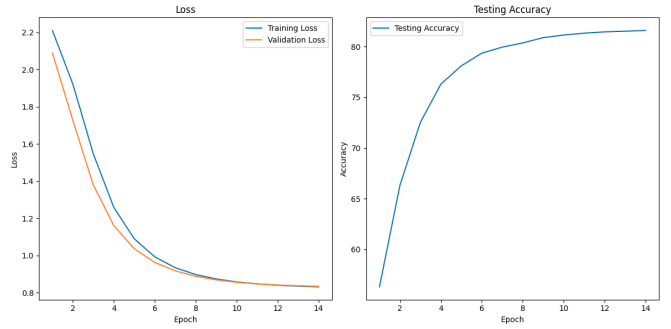
learning rate變化：



左圖是**learning rate=1**的結果，可以發現loss很大、且下降的速度也較慢，準確度不高。

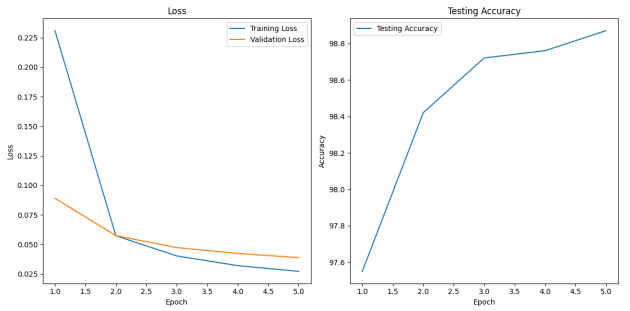


這是**learning rate=0.1**的結果，不論是loss的下降速度、抑或準確度都有效提升，且得到了不錯的成績。

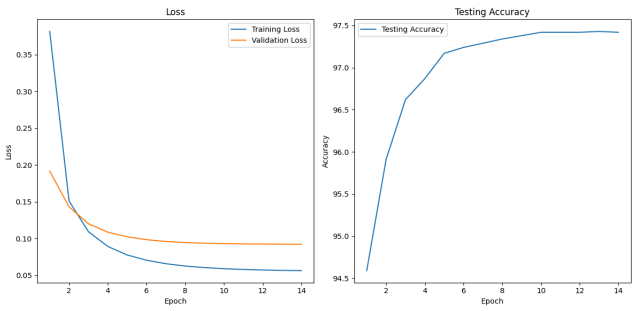


這是**learning rate=0.001**的結果，發現loss顯著增加了很多，且準確度也不高，這個結果也跟之前作業的實驗中得到的結論相似，也就是learning rate太高或太低都不好。

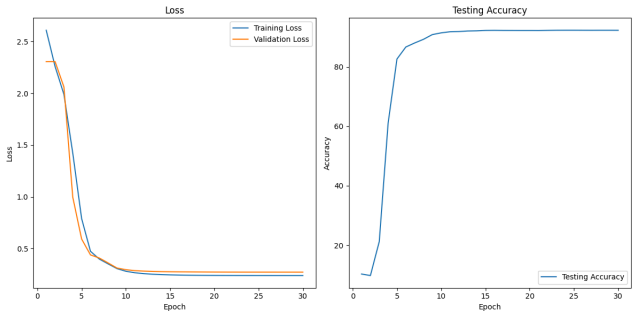
epoch變化：



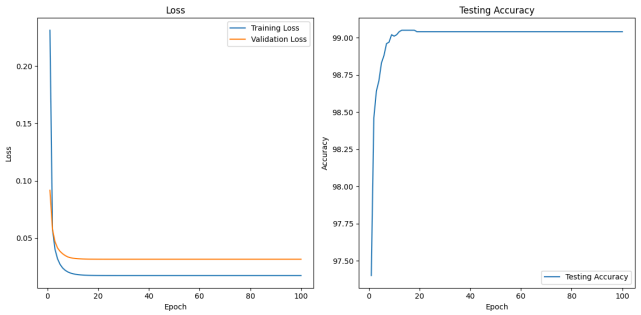
epoch = 5



epoch = 14



epoch = 30

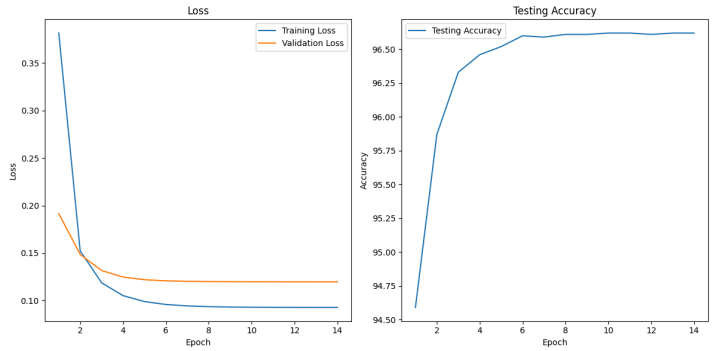


epoch = 100

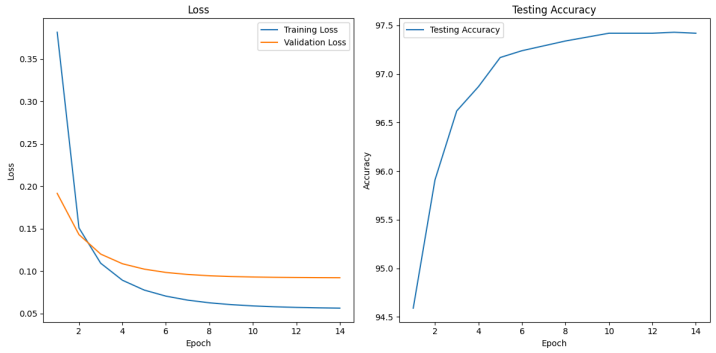
上面實驗是epoch = 5、14、30及100的結果，實驗下來發現只有在前14步左右會影響較多，超過15步以後就收斂得差不多了，並且當epoch設為100時會在第20步左右有準確度稍微下降的情況。因此在實驗結果中得出以下結論：

1. epoch如果太小，模型會來不及收斂到最好的解，導致訓練結果較差。
2. epoch如果太大，會浪費計算資源、訓練時間會很久，並且可能會過擬合。
3. 使用early stopping strategy可以防止過擬合導致的準確度下降。

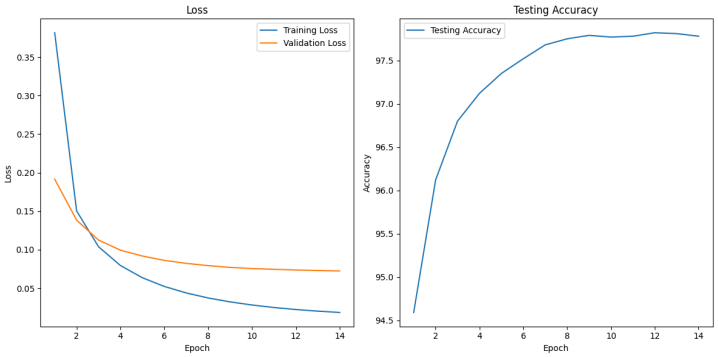
gamma變化：



gamma = 0.5，收斂速度較快，但accuracy與其他兩個實驗相比不太好。會有這個結果應該是因為模型在訓練初期就快速接近局部最佳解，並且在那附近就停止探索，導致沒辦法更充分尋找全局的最佳解。



gamma = 0.7，相較0.5的收斂速度慢一些，但可以發現他的表現有提升，證明了前述觀點。



gamma = 0.9，收斂速度較慢，但accuracy最高。我認為會有這個結果是因為模型需要更多的迭代來逐步收斂到最佳解，也因此讓模型能夠進行更充分的探索，有更大機會找到全局最佳解。

這個實驗我用gamma = 0.5、0.7和0.9進行測試。0.9收斂速度較慢，但表現最好，但在第10~14個epoch時，accuracy發生兩次稍微下降再回升到原位的震盪，猜測應該是有一點過擬合的狀況。
而0.5則是收斂非常快速，training loss和val loss都大約在epoch=8時就收斂完了。
0.7同時具有快速收斂和優異表現的特性，但因為他的最終結果還是沒有像0.9那麼好，所以還是覺得0.9是最好的。

HW3.4

我設計了三層Fully connected layers來處理影像分類任務：

- 第一層是輸入層，會接受一張28*28的影像，輸出128個neuron。
- 第二層是隱藏層，包含64個neuron。
- 第三層是輸出層，會輸出10個類別數字的分類機率。

會選擇使用128和64個neuron而不是投影片中範例程式提供的512，是因為MNIST dataset的圖片解析度很低，不需要用到太高的特徵維度，我使用的數量就足夠表達手寫數字的特徵了。

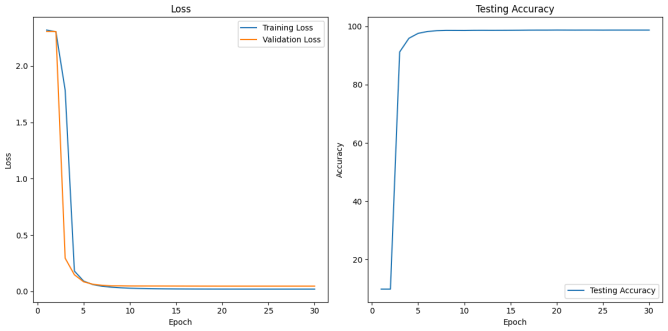
如果用太多的neuron反而可能會浪費計算資源、訓練時間變長，模型效能也不太會有顯著提升。甚至在進行實驗後發現，neuron設得太高反而分數還比較低。
在每一層之間，我用ReLU作為激活函數，加速收斂。在最終的輸出層沒有用激活函數是因為我們是分類任務，直接對每個類別的得分計算Cross-entropy loss。
這種簡單的Fully connected layer NN足以讓訓練的準確度夠高，而且因為結構簡單，訓練過程會比較穩定，不容易出現過擬合的狀況。
而因為我用了訓練集、驗證集跟測試集的分割，也能夠確保模型能很好地泛化到沒見過的資料上。
另外，我後來有嘗試使用early stopping strategy，讓模型訓練時間提前結束，避免後續過擬合導致accuracy降低的可能性。這個策略有成功讓test accuracy提升，但因為原本就已經99.04%了，所以並沒有提升非常多，只有升到99.05%。

Bonus

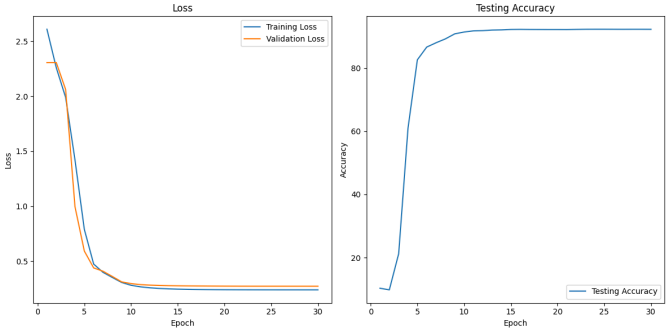
CNN的結構有包含Convolution layers、Pooling layers和Fully connected layer，相較NN的來得複雜，但他在捕捉影像的局部特徵(手寫字元)的能力上比NN還要好。
下面是我從訓練過程和結果總結出的優缺點:

	優點	缺點
NN	實作簡單	訓練速度較CNN慢、無法有效處理空間結構特徵
CNN	計算效率較高、圖像處理表現優異	設計架構較複雜

下面為NN和CNN的實驗結果比較，由於其他次實驗也大多都是CNN表現較佳，因此這邊僅放其中一項實驗結果作展示。兩張圖表皆在超參數設定皆為epoch = 30，learning rate = 0.06的限制下運作，其餘超參數與範例程式預設的相同。



左圖是使用CNN的結果，實驗結果的accuracy是98.75%，而training loss和val loss分別降到0.0198和0.0464。



左圖是使用NN的結果，實驗結果的accuracy是92.27%，而training loss和val loss則分別是0.2394和0.2726。

如上面實驗所示，CNN的表現明顯比NN的好。