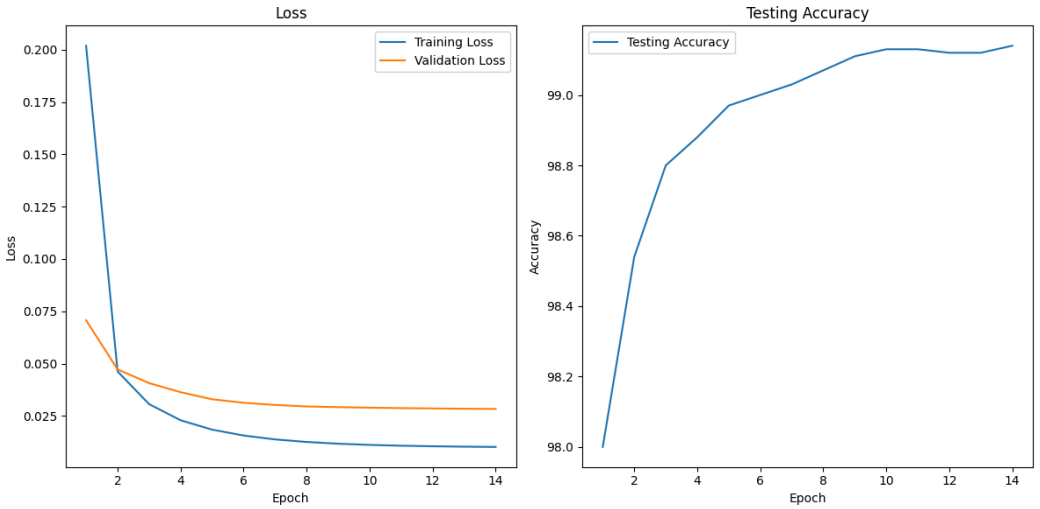
HW3.1



HW3.2

```
Epoch 1: Train Loss: 0.2019| Train Accuracy: 93.64% , Val Loss: 0.0707| Val Accuracy: 97.82% , Test Loss: 0.0603| Test Accuracy: 98.00%
Epoch 2: Train Loss: 0.0463| Train Accuracy: 98.61% , Val Loss: 0.0473| Val Accuracy: 98.52% , Test Loss: 0.0410| Test Accuracy: 98.54%
Epoch 3: Train Loss: 0.0307| Train Accuracy: 99.12% , Val Loss: 0.0407| Val Accuracy: 98.75% , Test Loss: 0.0359| Test Accuracy: 98.80%
Epoch 4: Train Loss: 0.0229| Train Accuracy: 99.34% , Val Loss: 0.0364| Val Accuracy: 98.90% , Test Loss: 0.0324| Test Accuracy: 98.88%
Epoch 5: Train Loss: 0.0185| Train Accuracy: 99.48% , Val Loss: 0.0330| Val Accuracy: 98.95% , Test Loss: 0.0291| Test Accuracy: 98.97%
Epoch 6: Train Loss: 0.0156| Train Accuracy: 99.56% , Val Loss: 0.0313| Val Accuracy: 99.03% , Test Loss: 0.0279| Test Accuracy: 99.00%
Epoch 7: Train Loss: 0.0138| Train Accuracy: 99.63% , Val Loss: 0.0302| Val Accuracy: 99.02% , Test Loss: 0.0271| Test Accuracy: 99.03%
Epoch 8: Train Loss: 0.0126| Train Accuracy: 99.67% , Val Loss: 0.0295| Val Accuracy: 99.05% , Test Loss: 0.0266| Test Accuracy: 99.07%
Epoch 9: Train Loss: 0.0117| Train Accuracy: 99.69% , Val Loss: 0.0292| Val Accuracy: 99.08% , Test Loss: 0.0262| Test Accuracy: 99.11%
Epoch 10: Train Loss: 0.0112| Train Accuracy: 99.71% , Val Loss: 0.0290| Val Accuracy: 99.10% , Test Loss: 0.0259| Test Accuracy: 99.13%
Epoch 11: Train Loss: 0.0108| Train Accuracy: 99.73% , Val Loss: 0.0287| Val Accuracy: 99.10% , Test Loss: 0.0257| Test Accuracy: 99.13%
Epoch 12: Train Loss: 0.0105| Train Accuracy: 99.73% , Val Loss: 0.0286| Val Accuracy: 99.10% , Test Loss: 0.0255| Test Accuracy: 99.12%
Epoch 13: Train Loss: 0.0103| Train Accuracy: 99.73% , Val Loss: 0.0285| Val Accuracy: 99.12% , Test Loss: 0.0254| Test Accuracy: 99.12%
Epoch 14: Train Loss: 0.0102| Train Accuracy: 99.73% , Val Loss: 0.0284| Val Accuracy: 99.08% , Test Loss: 0.0254| Test Accuracy: 99.14%
```

test accuracy = 99.14%

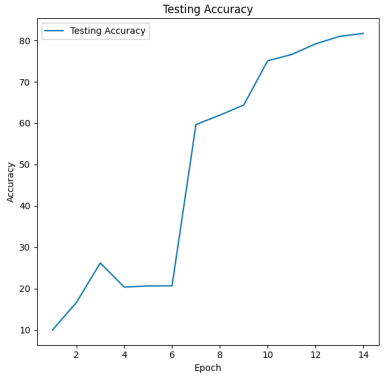
資料分割策略:

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。並且使用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。

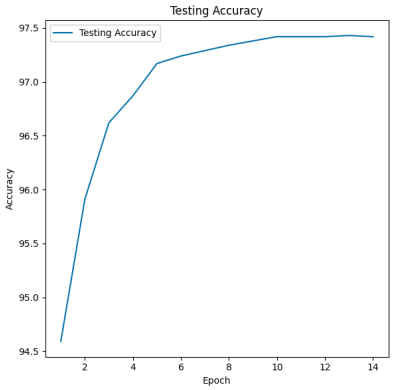
HW3.3

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

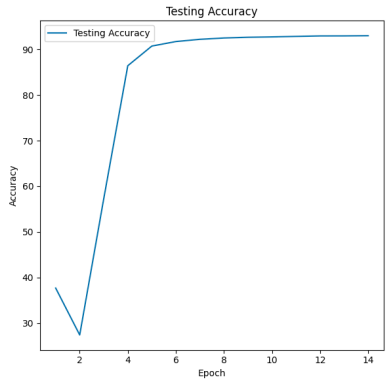
batch size變化:



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。

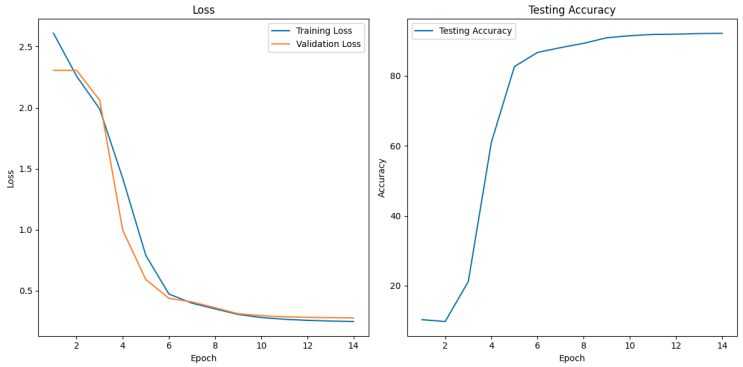


這是**batch size = 128**的結果，在前兩個epoch從37%掉到28%，隨後逐漸上升，到第四個epoch時準確度大約86%，並且最終在90%左右收斂，準確度相較**batch size = 32**的低。

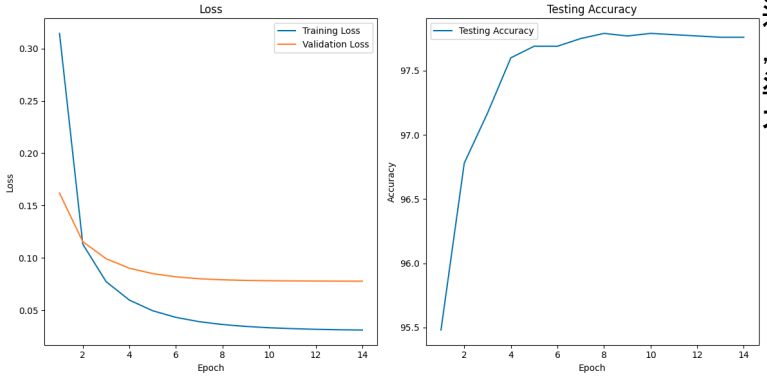
從這項實驗中可以發現：

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。

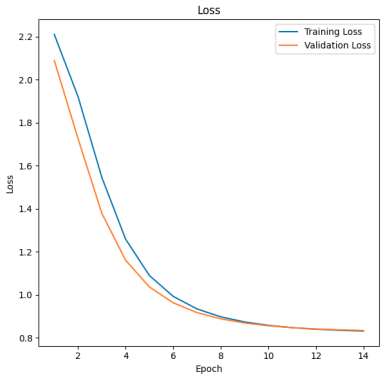
learning rate變化：

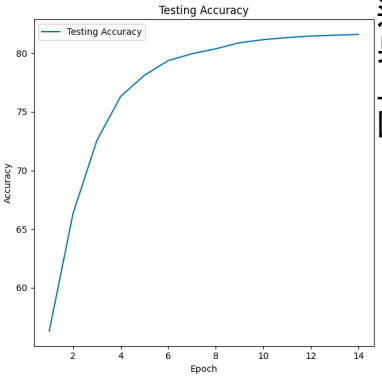


左圖是**learning rate=1**的結果，可以發現loss很大、且下降的速度也較慢，準確度不高。



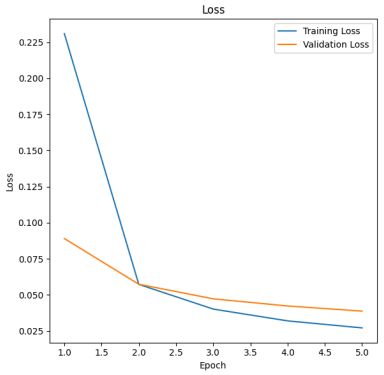
這是**learning rate=0.1**的結果，不論是loss的下降速度、抑或準確度都有效提升，且得到了不錯的成績。

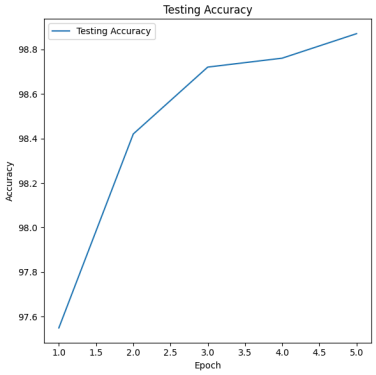




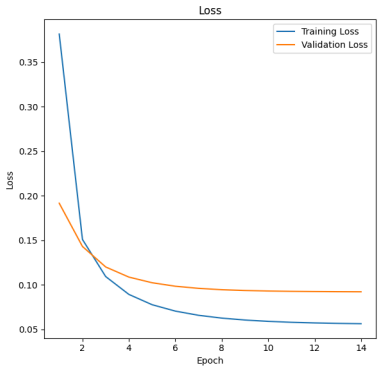
這是**learning rate=0.001**的結果，發現loss顯著增加了很多，且準確度也不高，猜測可能是因為梯度消失導致的狀況。

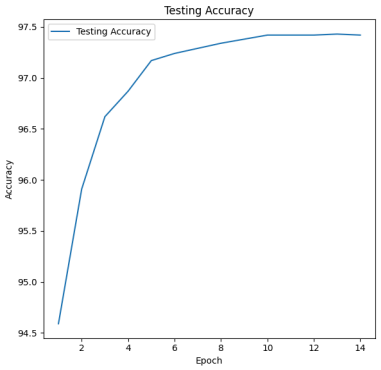
epoch變化：



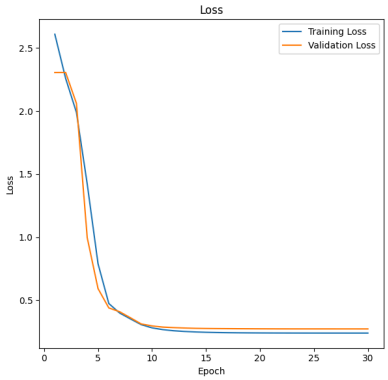


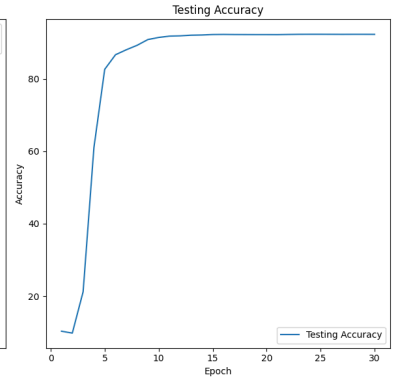
epoch = 5



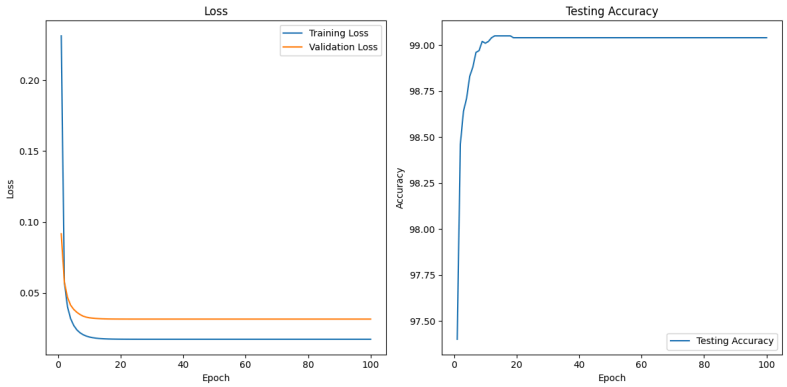


epoch = 14





epoch = 30



epoch = 100

我用epoch = 5、14、30及100進行測試，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。

gamma變化：
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，猜測如果再設得更大就會發生過擬合的狀況。
而0.5則是收斂非常快速，training loss和validation 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：

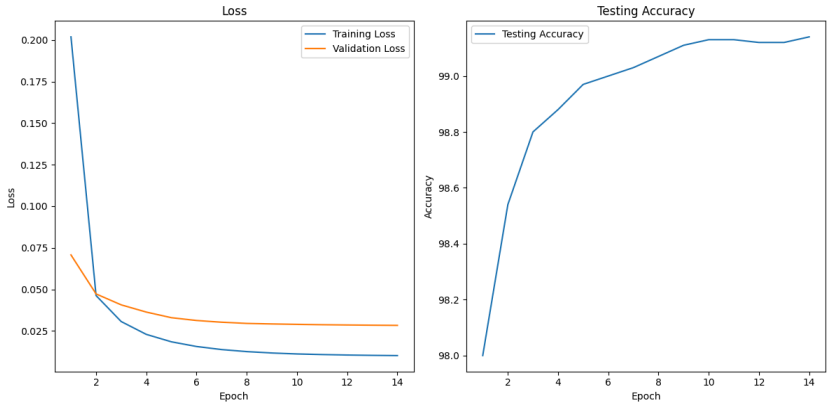
- 第一層是輸入層，會接受一張28*28的影像，輸出128個neuron。
- 第二層是隱藏層，包含64個neuron。
- 第三層是輸出層，會輸出10個類別數字的分類機率。

會選擇使用128和64個neuron而非範例程式的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。
在每一層之間，我用ReLU作為激活函數，避免梯度消失、並且加速收斂。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很高，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。

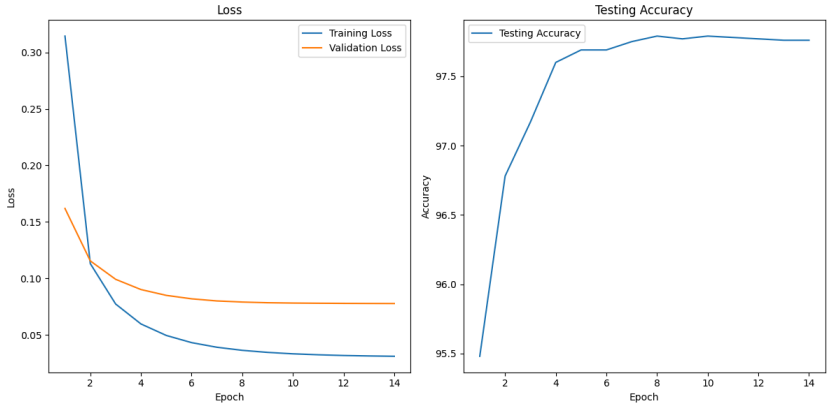
Bonus

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。所以由訓練過程和結果總結出優缺點各自如下：

	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



使用CNN



使用NN

如上面兩張圖片的比較，CNN的結果明顯比NN的高。