EAI lab 1

Understand NN and Training Process

數據所 RE6121011 徐仁瓏

1. Introduction

本次作業的目標是透過實作一個簡單的神經網絡來解決MNIST手寫數字分類問題，並在過程中理解模型的前向與反向傳播機制。模型將包含三個主要層次：內積層（Inner-product layer）、激活層（Activation layer，使用Sigmoid或ReLU函數）以及Softmax層。訓練過程中使用交叉熵損失函數（cross-entropy）來評估模型表現，並透過交叉驗證來提高模型的穩健性。預期模型將至少包含一層隱藏層，且在測試集上的準確度需要達到90%以上。最終將透過繪製訓練與驗證的準確度及損失曲線來評估模型的學習過程與效果。

2. Methodology

在本部分，我將介紹資料前處理的方式，說明前向傳播和反向傳播的實作步驟，並解釋我如何設計神經網絡模型以及各層的具體功能。

2.1 Data Preprocessing

MNIST資料集是一個28×28像素的灰階手寫數字圖像分類數據集，包含10個不同的類別（數字0~9）。訓練集共有60,000張圖片，測試集則包含10,000張圖片。為了進行正規化，我對輸入數據應用了標準化處理，使用訓練集的平均值和標準差來將每個像素值縮放至相同的尺度，這有助於模型更穩定地學習。我們選擇只使用訓練集的平均值和標準差，因為假設在真實情境中我們無法事先知道測試集的分佈情況。此外，將標籤從0~9轉換成 One-Hot Encoding 格式，這樣模型的輸出就可以對應到每一類別的機率預測。

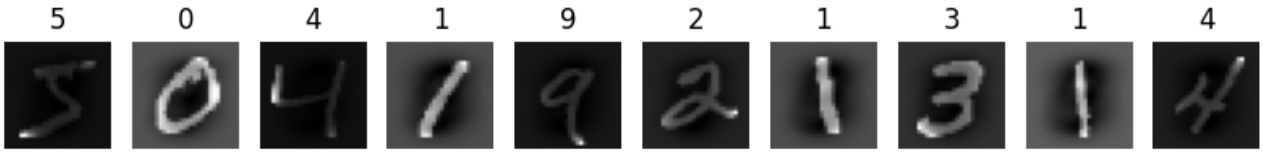

圖1 處理過後的資料集顯示前10個數字圖片

2.2 Forward Propagation

在前向傳播過程中，資料依序通過每一層網絡進行運算，並生成最終的輸出。實作的方式很簡單，直接將公式套入即可完成前向傳播。以下包含可能會使用到的網路：

- **Inner product**：全連接層，主要進行矩陣相乘，將輸入資料與權重進行內積，並加上偏移值。公式為 $Y = XW + B$ 。

- **Softmax**：激活函數，這一層的作用是在輸出層對各個類別進行歸一化，輸出每個類別的預測概率，確保所有概率加總為1。公式為 $Softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$ 。

- **Sigmoid**：激活函數，將輸出壓縮在(0,1)之間，適用於二元分類任務。公式為 $Sigmoid(x) = \frac{1}{1+e^{-x}}$ 。

- **ReLU**：激活函數，能解決梯度消失問題，保留正數，並將負數歸零。公式為 $ReLU(x) = max(0,x)$ 。

2.3 Backward Propagation

反向傳播是根據損失函數計算每個參數的梯度，並通過梯度下降法更新權重。雖然神經網路極其複雜，但是透過連鎖律可以很好的計算出當前目標的微分，以利於讓模型權重進行迭代更新，使損失函數越來越小，從而逐步優化模型。以下包含可能會使用到的網路：

- **Inner product**：由於我們設定的全連接層公式為 $Y = XW + B$ ，因此我們可以利用連鎖律計算權重的梯度，並將其傳遞至上一層。故在全連接層中，損失函數分別對 X、W、B 偏微分，結果如下表所示。

X	$\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial X} = \frac{\partial E}{\partial Y}W$
W	$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial W} = X\frac{\partial E}{\partial Y}$
B	$\frac{\partial E}{\partial B} = \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial B} = \frac{\partial E}{\partial Y}$

- **Softmax**：由於我們的 Softmax 層是最後一層，用於輸出每個類別的預測概率，因此 Softmax 的反向傳播需要計算預測值與真實值的誤差，再利用這些誤差來更新權重。故損失函數對 Softmax 的微分可被簡化為 $y - t$ ，其中 y 為通過 Softmax 層後輸出的結果，t 代表 One-Hot Encoding 形式的十個分類結果。

- **Sigmoid**：假設我們令 f 函數為 Sigmoid 函數，那麼它對 X 的微分為 $\frac{\partial f(x)}{\partial x} = f(x)\big(1 - f(x)\big)$ ，故在 Sigmoid 激活層中，損失函數對 X 的微分為 $\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial X} = \frac{\partial E}{\partial Y}f(X)\big(1 - f(X)\big)$ 。

- **ReLU**：假設我們令 f 函數為 ReLU 函數，那麼它對 X 的微分為 $\frac{\partial f(x)}{\partial x} = 1_{x>0}$，故在 ReLU 激活層中，損失函數對 X 的微分為 $\frac{\partial E}{\partial X} = \frac{\partial E}{\partial Y}\frac{\partial Y}{\partial X} = \frac{\partial E}{\partial Y}1_{x>0}$。

我認為 Backward Propagation 算是本次任務中的一大挑戰，除了要先了解其本身的意義外，還需要了解因為有連鎖律的存在，使其在計算上方便許多，可以分開計算各層中的微分。此外，雖然助教在講義皆有部分 Backward Propagation 的微分結果，但是對於矩陣的運算，在維度上還是需要特別注意，像我在這次任務中就多次因為兩個矩陣的維度不相符，而導致無法計算的問題出現。

2.4 Model Architecture

我設計的神經網絡模型包括三層結構：

- **輸入層 (Input Layer)**：該層接收 28×28 的圖像，並將其展平為 784 個神經元，對應每張圖像的像素點，作為全連接層的輸入。
- **隱藏層 (Hidden Layer)**：該層包含 256 個神經元，這是我自行設定的超參數，目的是增加網絡的表達能力。我使用 Sigmoid 作為激活函數，幫助模型學習非線性的表徵。
- **輸出層 (Output Layer)**：輸出層包含 10 個神經元，對應手寫數字 0~9 的分類結果。經過 Softmax 函數處理後，該層輸出每個類別的概率，確保總和為1。

在前向傳播過程中，資料依序通過這三層，最終生成分類結果。實作方面的話則是通過第一層 Inner Product Layer，參數 W_1 設計成 784×256 的矩陣，參數 B_1 設計成 batch_size×256 的矩陣，使形狀為 batch_size×784 的圖像資料 X 通過全連接層 $Z_1 = XW_1 + B_1$，得到 batch_size×256 的 Z_1，其後通過 Sigmoid 激活函數得到 A_1。接著再通過第二層 Inner Product Layer，參數 W_2 設計成 256×10 的矩陣，參數 B_2 設計成 batch_size×10 的矩陣，使形狀為 batch_size×256 的 A_1 通過全連接層 $Z_2 = A_1W_2 + B_2$，得到 batch_size×10 的 Z_2，最後通過 Softmax 層得到 Y，也就是各個類別的輸出概率。以數學公式可表示為下：

$$Y = Softmax\left(\left(Sigmoid\left(XW_1 + B_1\right)\right)W_2 + B_2\right)$$

在反向傳播過程中，基於預測結果與真實標籤之間的誤差，更新每一層的權重，逐步優化模型。實作方面則和前向傳播相反，先對 Softmax 層中的 Z_2 微分得到 $\frac{\partial E}{\partial Z_2}$，再分別對第二層全連接層中的 A_1、W_2、B_2 偏微分得到 $\frac{\partial E}{\partial A_1}$、$\frac{\partial E}{\partial W_2}$、$\frac{\partial E}{\partial B_2}$，接著再對 Sigmoid 層中的 Z_1 微分得到 $\frac{\partial E}{\partial Z_1}$，最後再分別對第一層全連接層中的 X、W_1、B_1 偏微分得到 $\frac{\partial E}{\partial X}$、$\frac{\partial E}{\partial W_1}$、$\frac{\partial E}{\partial B_1}$，完成反向傳播的計算。從中得到的 $\frac{\partial E}{\partial W_1}$、$\frac{\partial E}{\partial B_1}$、$\frac{\partial E}{\partial W_2}$、$\frac{\partial E}{\partial B_2}$，即可透過公式 $\theta' = \theta - \eta\frac{\partial E}{\partial \theta}$ 來更新權重，其中 θ 代表權重，η 代表學習率。

3. Experiments

3.1 Setting

本次實驗的參數設定如表所示。由於這是一個多分類問題，我選擇使用 cross-entropy 作為損失函數。模型進行了 50 次迭代，並使用梯度下降法 (gradient descent) 來更新權重，學習率 (learning rate) 設定為 0.05。批次大小 (batch size) 設定為 2000，以加速模型的訓練過程。

權重初始值的設定方式對訓練過程也十分重要。為了確保網絡能有效學習，我採用了Xavier初始化法來隨機生成權重初值，公式如下：

$$W \sim N(0, \frac{1}{n_{in}})$$

其中 n_{in} 為輸入神經元的數量。這種方法可以根據輸入和輸出神經元的數量動態調整初始權重範圍，從而減少梯度消失的風險。

在驗證方法方面，我選用了 K-fold Cross Validation，並將 K 設為 3。將訓練集劃分為 3 個子集，輪流將其中一個作為驗證集，其餘兩個作為訓練集。這樣可以在不同的驗證集上評估模型的表現，從而避免模型只在單一驗證集上表現良好，而在其他數據上表現不佳。最終，我在每個fold的驗證結果中挑選出最好的權重，並從這些最佳權重中再選出一個最佳的作為測試集的最終權重，這樣的過程能夠有效提升模型的泛化能力。

3.2 Result

在結果部分，我們使用以下的 Accuracy 曲線和 Loss 曲線來檢視模型的訓練過程與表現。

3.2.1 Accuracy Curve Analysis

如圖 2 所示，從 Accuracy 曲線可以觀察到，模型的準確率在訓練的初期階段訊速提升，這表明模型很快學習到了數據中的基本特徵。在隨後的訓練過程中，模型逐漸適應數據的複雜性，準確率呈現穩定上升的趨勢。曲線的上升表明模型能夠捕捉到更多資料中的模式和特徵，不斷提升其分類能力。

然而，從曲線中也可以看到類似鋸齒狀的急劇變化，特別是在一些區段，曲線並不平滑。這種現象可能由於學習率設置過高導致，因為較大的學習率會使得權重的更新步伐過大，從而引發模型在不同批次數據上的波動。如果降低學習率並增加迭代次數，模型的權重更新會更加緩和，可能有助於平滑準確率的上升曲線，使模型收斂得更穩定。此外，訓練集和驗證集的準確率曲線基本上是同步上升的，並且兩者之間的差距相對較小，這表明模型並沒有發生過擬合現象。一般來說，過擬合會導致訓練準確率顯著高於驗證準確率，但在這裡，這樣的現象並未出現。因此，可以推斷目前的模型在訓練過程中具有良好的泛化能力。

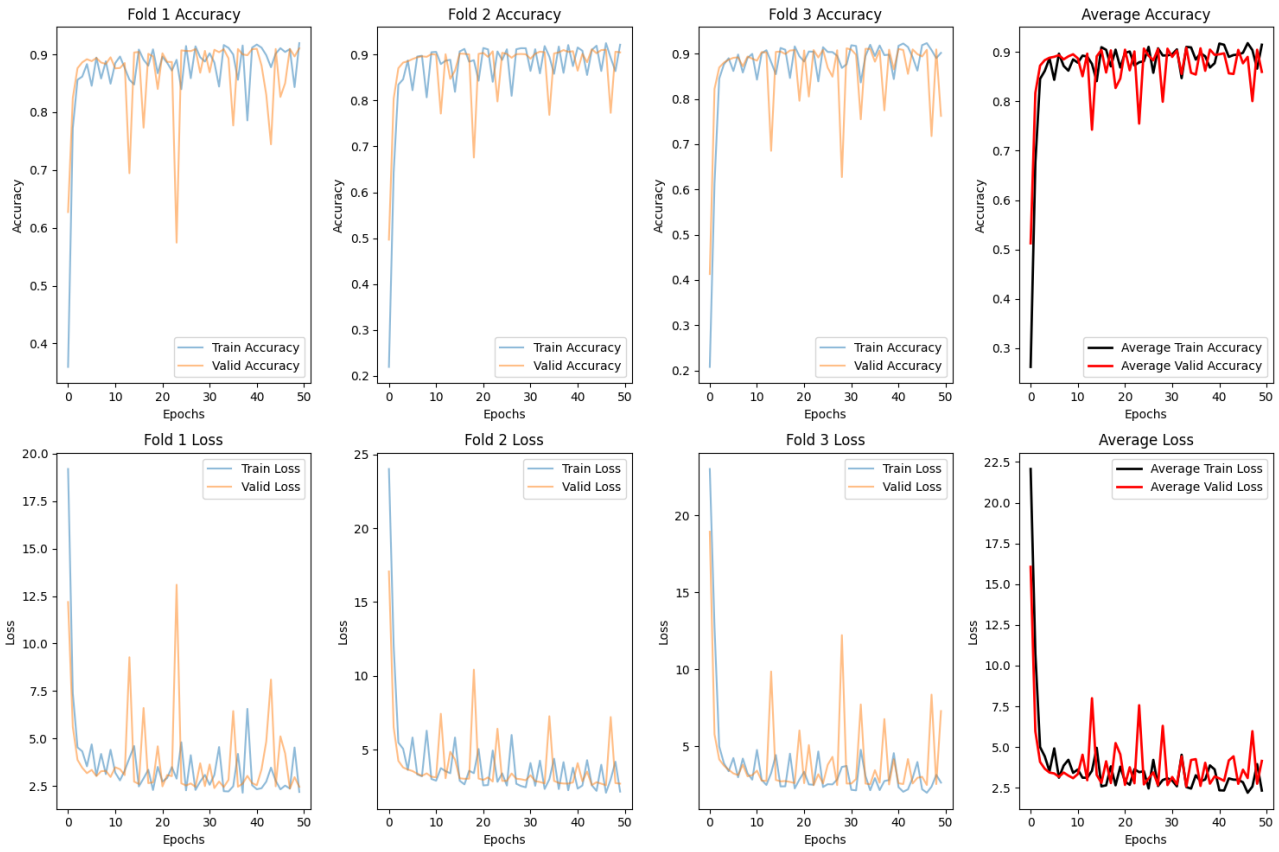

圖2 模型訓練過程的Accuracy 曲線和 Loss 曲線

未來，可以考慮對學習率進行進一步調整，適當降低學習率並增大訓練的迭代次數。這不僅可能會解決曲線的不穩定性，還可能提高模型在更複雜任務上的表現。更長的訓練時間也有助於模型更深入地學習數據中的模式和結構。

3.2.2 Loss Curve Analysis

從 Loss 曲線可以明顯看出損失值的變化趨勢。在訓練初期，損失值相對較高，這是因為模型在初期尚未充分學習數據中的特徵，因此分類錯誤較多。隨著訓練進行，損失值逐步下降，顯示模型的不斷學習和改進，並且分類錯誤逐漸減少。這與 Accuracy 曲線的上升趨勢相呼應，表明模型的學習效果正變得更加精確。

然而，和 Accuracy 曲線相似，損失曲線中也存在不規則的鋸齒狀波動，這可能表明在一些小批次的數據上，模型的性能並不穩定。這種不穩定現象可能是由於過大的學習率導致每次更新的變化幅度過大，導致損失值在不同批次之間急劇變化。這再度強調了對學習率進行進一步調整的重要性，可能降低學習率會使損失曲線變得更加平滑，並有助於模型更穩定地收斂。

針對這些曲線的表現，未來可以進行更多實驗，例如進行學習率調整、選擇不同的優化器或增強數據預處理技術，來觀察是否能夠改善模型的損失收斂表現及其穩定性。

3.2.3 K-fold Cross Validation Results

在進行 K-fold 交叉驗證的過程中，我們針對模型的泛化能力進行了多次測試，並從中選擇了每個驗證集上表現最好的模型。根據每個驗證集的準確率評估，最終選出了表現最佳的權重。在此過程中，我們評估了各個驗證集的準確率，以下是選出的最佳模型在各個驗證集上的表現：

Fold	Train Accuracy	Valid Accuracy
1	0.9195	0.9106
2	0.9101	0.9113
3	**0.8767**	**0.9120**

如上表所示，每次交叉驗證結果的準確率相對穩定且接近，這進一步表明模型具備良好的泛化能力。最後，我們從這三個模型中，選擇了在驗證集上準確率最高的一個作為最終的模型，並將其應用於測試集上進行測試。

	Accuracy	Loss
Test dataset	**0.9139**	2.5575

測試結果顯示 Test Accuracy 為 0.9139，比各驗證集的 Accuracy 都還要高，證明模型並沒有過度配適，反而展現了良好的泛化能力。

4. Challenges and Solutions

- **Challenges in Backward Propagation**

反向傳播是神經網路訓練中的核心部分，但也是許多新手面臨的一大挑戰。理解反向傳播的原理需要熟悉微分、連鎖律等數學概念。連鎖律的應用可以讓梯度的計算變得更加有效率，但在實際寫程式時，常常因為計算過程中的繁瑣細節而犯錯。

- **Matrix Dimension Alignment**

在反向傳播的公式推導及運算中，矩陣的維度匹配是非常關鍵的一部分，因為維度不正確會導致計算錯誤。模型的每一層都有特定的權重和輸出張量，這些張量必須在形狀上正確匹配才能進行正確的矩陣運算。因此，在寫程式的過程中，需要特別注意每個張量的維度，確保在前向和反向傳播時的矩陣運算能順利進行。

- **Axis Confusion in NumPy**

在訓練的初期，對於 NumPy 中的軸參數容易產生混淆，經常導致預期之外的結果。理解如何正確操作陣列至關重要，尤其是在決定是否對 row（axis=0）或 column（axis=1）進行加總、平均或其他操作時。這方面的誤解可能導致模型訓練失敗或計算錯誤。在更加注意陣列的維度和軸的選擇後，結果的一致性得到了顯著改善。

- **Activation Function Selection: ReLU vs. Sigmoid**

在模型的激活函數選擇上，ReLU 和 Sigmoid 各有優劣。ReLU 是目前深度學習中常用的激活函數，因為它能夠有效解決梯度消失問題。然而，ReLU 在某些情況下容易導致資料發生 NaN 的現象，特別是在梯度更新過程中出現極端值時。而 Sigmoid 函數雖然相對較為穩定，但當輸入的 x 值太大時，會導致 overflow 問題，使得輸出過於極端，無法有效傳遞梯度。因此，在選擇激活函數時需要根據具體情況做出平衡，並可能需要考慮正規化技術來緩解相關問題。

- **Learning Rate and Iteration Adjustments**

若算力及時間允許，可以考慮降低學習率並提高迭代次數，來達到更好的模型表現。因為過大的學習率會導致模型在訓練過程中出現不穩定的損失曲線，進而難以收斂。相對地，較低的學習率可以讓模型更細緻地學習資料中的表徵，減少損失值的波動。此外，增加迭代次數也能幫助模型逐漸趨向穩定，並有機會達到更高的準確率。

5. Conclusion

在這次作業中，我實作了一個三層的神經網路解決 MNIST 手寫數字分類問題，並在測試集上達到了 91.39% 的準確率。過程中我深入理解前向與反向傳播的原理，尤其是在梯度計算和權重更新方面所面臨的挑戰。通過對矩陣維度的仔細檢查和對NumPy操作的熟悉，我克服了多次出現的維度不匹配問題，並選擇了合適的激活函數以增強模型的表現。雖然在訓練過程中遇到了一些困難，但這些挑戰反而促進了我對模型運作的深入思考和理解。