Week 2 Programming assignment

1 問題描述

本次作業要求使用一個神經網路，使其能同時近似 Runge 函數本身以及其導數。

$$f(x) = \frac{1}{1+25x^2}, \quad x \in [-1, 1] \tag{1}$$

$$f'(x) = \frac{-50x}{(1+25x^2)^2} \tag{2}$$

報告需包含所使用的方法、實驗結果與討論，並附上以下圖表與數據：

- 將真實函數與神經網路預測結果一同繪製。
- 展示訓練與驗證的損失曲線。
- 計算並報告誤差值 (MSE 與 Max Error)。

2 研究方法

2.1 數據集生成

我們在定義域 $[-1, 1]$ 之間均勻採樣生成 1000 個數據點。對於每個點 x_i，我們同時計算其對應的函數值 $y_i = f(x_i)$ 以及導數值 $y_i' = f'(x_i)$。數據集同樣以 8:2 的比例隨機劃分為訓練集和驗證集。

2.2 神經網路架構

我們沿用上次作業的網路架構，並針對激活函數進行對照實驗：

- 輸入層： 1 個神經元 (x)。
- 隱藏層 **1**： 64 個神經元。
- 隱藏層 **2**： 64 個神經元。
- 輸出層： 1 個神經元 $(\hat{y})$。

我們將分別測試在隱藏層使用 **ReLU** 和 **Tanh** 作為激活函數時，模型的性能表現。

2.3 訓練過程

- **複合損失函數 (Composite Loss Function)**：為了讓網路同時學習函數及其導數，我們定義了一個由兩部分組成的總損失函數：

$$\mathcal{L}_{\text{total}} = \mathcal{L}_f + \mathcal{L}_{f'} \tag{3}$$

其中，$\mathcal{L}_f$ 是預測函數值 $\hat{y}$ 與真實值 y 之間的均方誤差 (MSE)，而 $\mathcal{L}_{f'}$ 是預測導數值 $\hat{y}'$ 與真實導數 y' 之間的均方誤差。預測導數 $\hat{y}' = \dfrac{d\hat{y}}{dx}$ 是透過 PyTorch 的自動微分機制計算得出。

- **優化器 (Optimizer)**：使用 Adam 優化器，學習率設定為 0.001。
- **訓練週期 (Epochs)**：模型共訓練 1000 個週期。

3 實驗結果與分析

3.1 實驗一：使用 ReLU 激活函數

3.1.1 函數與導數近似結果

下圖展示了使用 ReLU 激活函數的模型，其對 Runge 函數及導數的擬合結果。

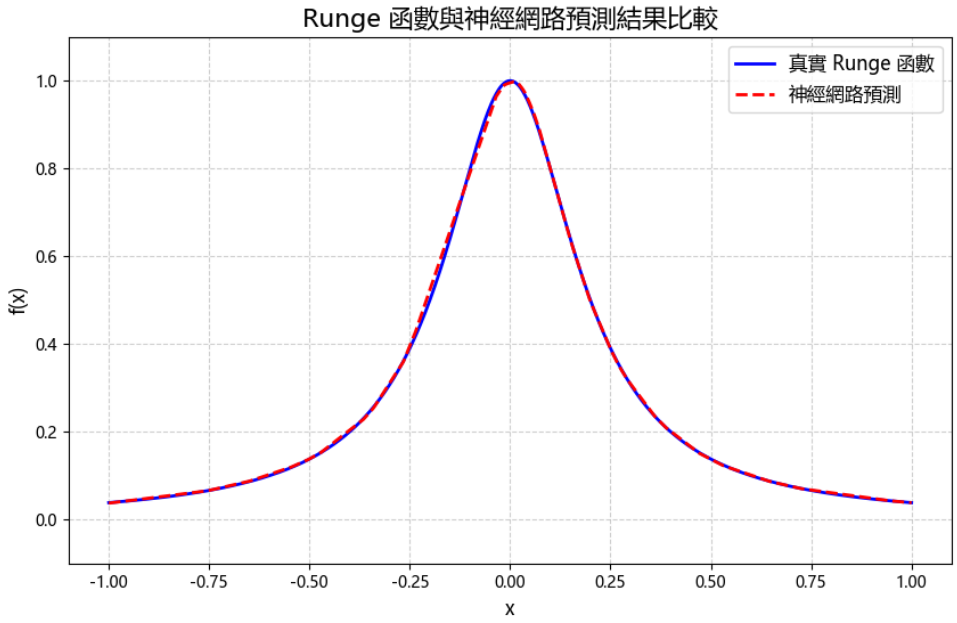

Figure 1: ReLU 模型對 Runge 函數的擬合結果

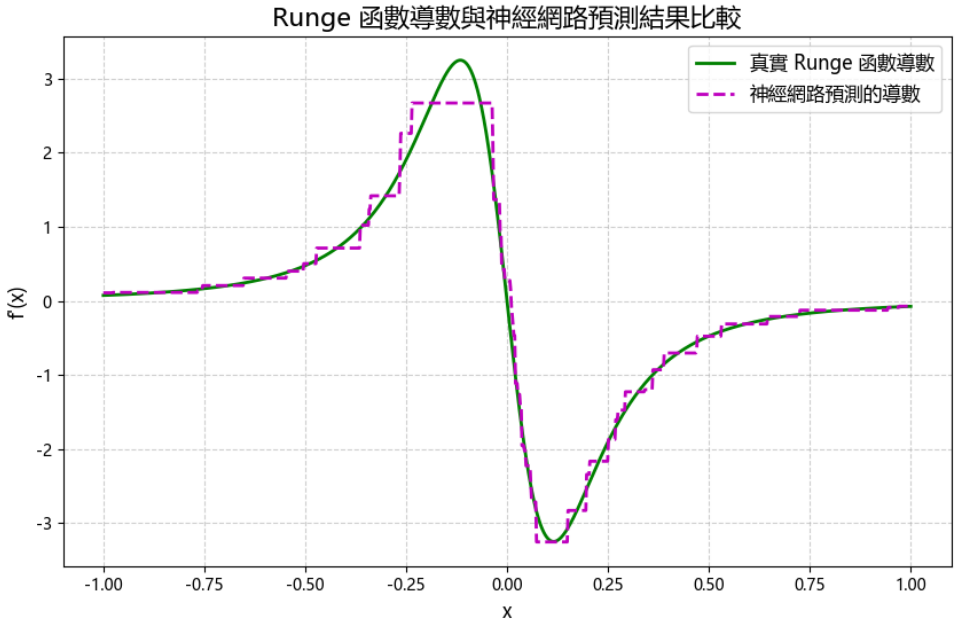

Figure 2: ReLU 模型對 Runge 函數導數的擬合結果

3.1.2 損失曲線與誤差

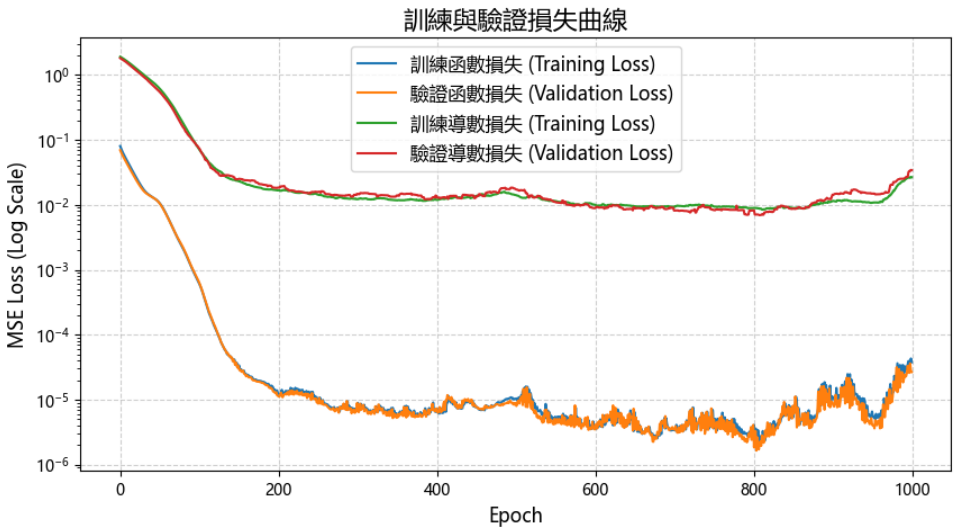

Figure 3: ReLU 模型的總損失曲線 (Log Scale)

評估指標 (ReLU)	均方誤差 (MSE)	最大誤差 (Max Error)
函數 $f(x)$	2.65×10^{-5}	2.46×10^{-2}
導數 $f'(x)$	3.43×10^{-2}	9.38×10^{-1}

Table 1: ReLU 模型在驗證集上的誤差評估

3.2 實驗二：使用 Tanh 激活函數

3.2.1 函數與導數近似結果

下圖展示了使用 Tanh 激活函數的模型，其對 Runge 函數及導數的擬合結果。

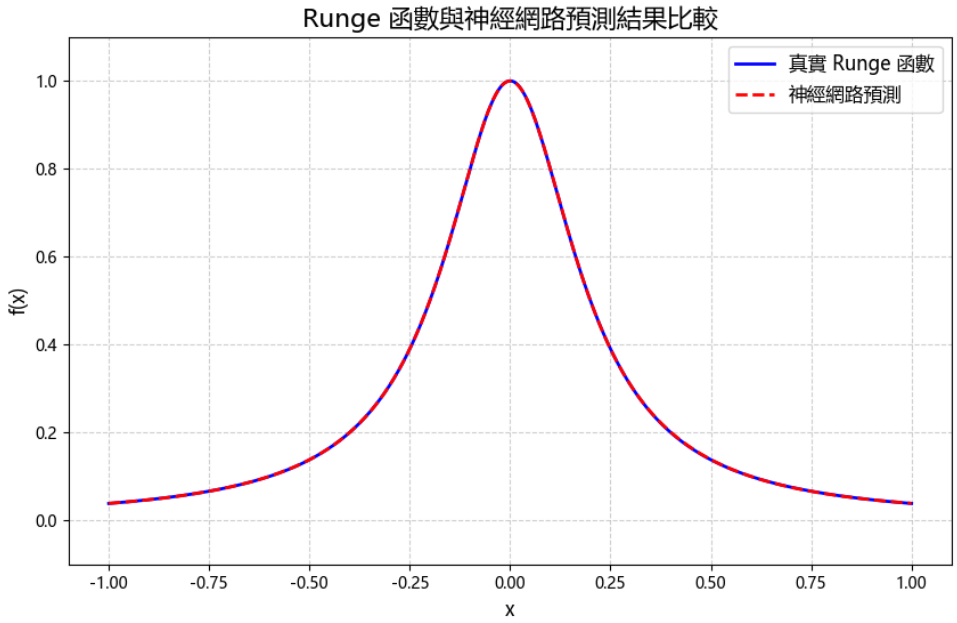

Figure 4: Tanh 模型對 Runge 函數的擬合結果

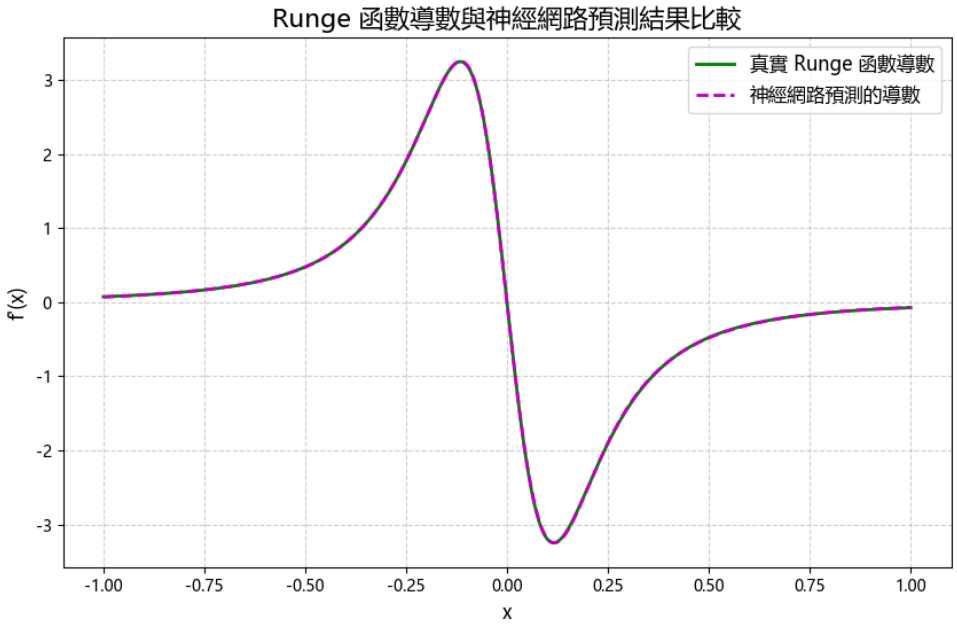

Figure 5: Tanh 模型對 Runge 函數導數的擬合結果

3.2.2 損失曲線與誤差

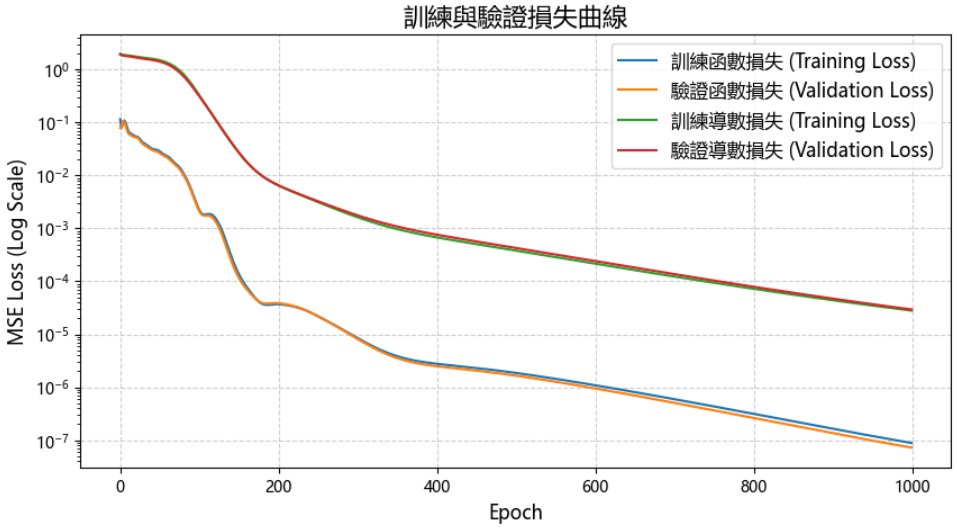

Figure 6: Tanh 模型的總損失曲線 (Log Scale)

評估指標 (Tanh)	均方誤差 (MSE)	最大誤差 (Max Error)
函數 $f(x)$	7.24×10^{-8}	8.13×10^{-4}
導數 $f'(x)$	2.94×10^{-5}	1.70×10^{-2}

Table 2: Tanh 模型在驗證集上的誤差評估

4 討論

從兩個實驗的結果對比中，可以得出非常清晰的結論：使用 Tanh 激活函數的模型在本次任務中的表現遠優於使用 ReLU 的模型。

1. 誤差大小：從表 1 和表 2 的數據來看，Tanh 模型的 MSE 和最大誤差，無論是在函數還是導數的擬合上，都比 ReLU 模型低了數個數量級。Tanh 模型的擬合精度非常高。

2. 擬合平滑度： Runge 函數及其導數都是無限次可導的平滑函數。Tanh 函數本身也是一個平滑函數，而 ReLU 函數則在 $x = 0$ 處存在一個不可導的尖點。理論上，由平滑激活函數（如 Tanh）構成的神經網路更容易擬合出平滑的目標函數。從圖 2 中可以看到，ReLU 模型擬合的導數曲線存在一些不自然的線性區段和轉折，而 Tanh 模型的擬合結果（圖 5）則非常平滑，與真實曲線高度吻合。

3. 導數學習：由於 ReLU 的導數是分段常數（0 或 1），這使得網路在學習目標函數的平滑導數時遇到了困難。Tanh 的導數是連續變化的，這為網路的反向傳播提供了更豐富的梯度資訊，從而能更精確地學習目標函數的導數。

5 結論

本次實驗成功地建構了一個能夠同時近似 Runge 函數及其導數的神經網路。透過引入導數的損失項，模型被迫學習函數的局部變化趨勢，從而能更精準地捕捉其形狀。

在激活函數的選擇上，實驗證明對於擬合像 Runge 函數這樣的平滑函數，使用平滑的 Tanh 激活函數比使用分段線性的 ReLU 激活函數能達到顯著更優的性能和更高的精度。