

Deep Multilayer Perceptrons

AI HW4

數據所 RE6121011 徐仁瓏

1.1

推導公式如附圖：

<p>1. Sum of Square:</p> <p>① Loss function:</p> $E_{SSE} = \frac{1}{2} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ <ul style="list-style-type: none">• y_i: true label• \hat{y}_i: softmax(z_i)• z_i: $w \cdot x$ <p>② Gradient:</p> $\frac{\partial E_{SSE}}{\partial w_k} = \frac{\partial E_{SSE}}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_k} \frac{\partial z_k}{\partial w_k}$ <ul style="list-style-type: none">• $\frac{\partial E_{SSE}}{\partial \hat{y}_i} = -\sum_{i=1}^n (y_i - \hat{y}_i)$• $\frac{\partial \hat{y}_i}{\partial z_k} = \hat{y}_i (1 - \hat{y}_i)$• $\frac{\partial z_k}{\partial w_k} = x_k$ $\Rightarrow \frac{\partial E_{SSE}}{\partial w_k} = -\sum_{i=1}^n (y_i - \hat{y}_i) \hat{y}_i (1 - \hat{y}_i) x_k$	<p>2. Cross Entropy: (Binary Classification for example)</p> <p>① Loss function:</p> $E_{CE} = -\sum_{i=1}^n [y_i \ln \hat{y}_i + (1 - y_i) \ln (1 - \hat{y}_i)]$ <p>② Gradient:</p> $\frac{\partial E_{CE}}{\partial w_k} = \frac{\partial E_{CE}}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial z_k} \frac{\partial z_k}{\partial w_k}$ <ul style="list-style-type: none">• $\frac{\partial E_{CE}}{\partial \hat{y}_i} = -\sum_{i=1}^n \left[\frac{y_i}{\hat{y}_i} - \frac{1 - y_i}{1 - \hat{y}_i} \right]$• $\frac{\partial \hat{y}_i}{\partial z_k} = \hat{y}_i (1 - \hat{y}_i)$• $\frac{\partial z_k}{\partial w_k} = x_k$ $\begin{aligned} \frac{\partial E_{CE}}{\partial w_k} &= -\sum_{i=1}^n \left[\frac{y_i}{\hat{y}_i} - \frac{1 - y_i}{1 - \hat{y}_i} \right] \hat{y}_i (1 - \hat{y}_i) x_k \\ &= -\sum_{i=1}^n \left[y_i (1 - \hat{y}_i) - \hat{y}_i (1 - y_i) \right] x_k \\ &= -\sum_{i=1}^n (y_i - \hat{y}_i) x_k \end{aligned}$
---	---

比較可以發現，平方和損失額外乘了一個 $\hat{y}_i(1 - \hat{y}_i)$ ，這是激活函數導數的影響。對於分類問題，當預測值 \hat{y}_i 接近 0 或 1 時， $\hat{y}_i(1 - \hat{y}_i)$ 會趨近於 0，使得梯度變得非常小，導致梯度下降的更新速度減慢。相較之下，交叉熵損失的梯度不會受到這種縮放效應的影響，從而加速訓練。

1.2

平方和損失的梯度在目標值接近時非常小，而交叉熵損失的梯度在接近正確分類時依然保持相對較大。這意味著，使用交叉熵損失時，模型能夠更穩定地調整權重，避免因過小的梯度而陷入局部極小值或停滯不前。

交叉熵損失函數直接針對分類概率分布進行優化，能更準確地將分類問題轉化為最大化似然估計的問題。而平方和損失函數則將分類問題轉化為迴歸問題來處理，這在數學上並不完全吻合，可能導致模型對輸入數據的區分能力較弱。

2.

在多類別分類問題中，Softmax 函數比單獨應用 Sigmoid 函數後正規化的方法更為有效，以下的特性使 Softmax 成為多類別分類問題的首選激活函數：

- Softmax 自然地生成一個歸一化的機率分布。
- 它鼓勵類別之間的競爭，符合多類別分類的語義。
- 梯度計算更簡單且穩定，與交叉熵損失的結合非常高效。

3.1

批次正規化的機制如下圖所示：

$$\bar{y}_B = \frac{1}{m} \sum_{i=1}^m x_i \quad , \quad \sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \bar{y}_B)^2$$

$$\hat{x}_i = \frac{x_i - \bar{y}_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta$$

其中 m 是批量大小， x_i 是當前批次中第 i 個樣本的輸入。 \hat{x}_i 為標準化後的結果，這使得每個批次的輸入具有零均值和單位方差，從而穩定輸入分布。接著，批次正規化引入可學習參數 γ 和 β ，進行縮放和平移，保留了模型學習複雜表示的能力。

這樣的動作使得模型通過穩定輸入分布，後續層不再需要適應不斷變化的輸入，從而加速訓練並提高收斂速度。減少梯度爆炸或梯度消失的問題，使得更高學習率也能穩定訓練。

3.2

如果移除 γ 和 β 參數，即變為標準化，雖使模型變得簡單，但也造成以下結果：

- 沒有 γ 和 β 參數，模型失去了對輸入的靈活調整能力。
- 對於某些層，需要將輸入縮放到特定範圍（例如激活函數的有效區域），移除這些參數可能導致模型無法學習到複雜的非線性特徵。
- 模型變得更加線性化，因為 γ 和 β 提供了對輸入特徵的再調整能力，移除它們可能導致模型的學習能力減弱，影響最終性能。

移除 γ 和 β 會使模型變得簡單，雖然能緩解一定程度的過擬合，但會降低模型擬合複雜函數的能力。

3.3

正則化作用機制

- 由於每個批次的均值 μ_B 和標準差 σ_B 是基於當前批次計算的，因此輸入的正規化具有隨機性。
- 這種隨機性類似於 Dropout 的效果，能讓模型在不同的訓練階段看到稍微不同的數據分布，從而提高泛化能力。
- 隨機性的正則化作用降低了模型對訓練數據的過度依賴，從而減少過擬合。

對泛化的影響

- 提高了模型在測試集上的表現，因為模型更能適應未見數據。
- 在小數據集上，Batch Normalization 可能取代部分正則化方法（如 Dropout）。

3.4

小批量的問題

- 當批量大小過小時，計算出的 μ_B 和 σ_B 可能具有較大波動，導致模型的正規化效果不穩定。
- 模型可能出現訓練不穩定或難以收斂的問題。
- 小批量中的數據不足以反映整體數據分布，這可能使模型過度擬合小批量內的特徵。

大批量的問題

- 當批量大小過大時， μ_B 和 σ_B 接近整體數據的均值與標準差，這減少了批次正規化帶來的隨機性。
- 正則化作用下降，可能導致模型更容易過擬合。

4.1

梯度消失問題通常發生在深層神經網路中，當梯度反向傳播時，梯度值可能因為連續的鏈式相乘而逐漸縮小，導致前面層的權重更新接近於零，從而影響網路的訓練。

Skip Connection 的機制為 $x_{out} = f(x_{in}) + x_{in}$ ，其中 $f(x_{in})$ 是中間層的變換。這種結構具有以下特性：

- 在反向傳播中，梯度可以直接通過 x_{in} 傳遞，不依賴於 $f(x_{in})$ 的梯度，從而避免梯度縮小。
- 如果某些層的學習變得困難，網路可以通過跳層連接將這些層的影響降到最低，確保網路至少保留輸入的主要特徵。

這使得深層網路（如 ResNet）能夠有效訓練，並解決由梯度消失引起的性能瓶頸。

4.2

加法 (Addition)

- 優點：

- 參數數量不變：輸入與輸出的維度相同，不增加額外的參數或計算成本。
- 數值穩定性：加法直接融合輸入和輸出的特徵，不會過分強調某一部分，保留了兩者的平衡。
- 簡單高效：加法計算簡單，適合深層模型快速學習。

- 缺點：

- 特徵結合的靈活性較低：僅支持相同維度的特徵融合，對於需要增強特徵多樣性的場合可能表現不足。

串接 (Concatenation)

- 優點：

- 特徵保留完整性：將輸入和輸出的特徵直接拼接，保留每一部分的全部信息。
- 適合多樣化特徵融合：提供更多的特徵學習空間，能讓後續層更靈活地選擇重要特徵。

- 缺點：

- 參數與計算量增加：拼接後維度增大，導致後續層的參數和計算成本顯著上升。
- 需要額外設計降維操作：若不進行降維處理，可能導致冗餘特徵影響模型性能。

適用場景

- 加法適合深層網路，如 ResNet，因其效率高且穩定性強。
- 串接適合需要強化多樣性特徵的場景，如多模態學習或需要結合多種特徵表徵的模型。

4.3

原因分析：

- BERT 的每層輸入和輸出的維度相同（例如，詞嵌入的維度），加法操作不改變維度，簡化了設計並減少了額外計算。
- 若使用串接，輸出維度將翻倍，這會顯著增加計算成本和參數量。
- 加法操作簡單高效，對於需要多層 Transformer 堆疊的 BERT 來說，加法有助於加速訓練和推理。
- 加法直接融合殘差特徵與主特徵，保持了特徵的平衡。串接可能使殘差特徵在高維空間中佔據較小權重，削弱其作用。

影響：

- 性能穩定性：採用加法使模型更易於訓練，避免冗餘特徵對學習的不良影響。
- 計算效率：減少參數數量和計算需求，使得 BERT 更適合大規模數據處理。

4.4

若改用串接，對 BERT 的影響：

- 串接後，維度增加，導致參數量和計算量顯著上升。例如，若輸入維度為 d ，串接後維度變為 $2d$ ，會使後續層的參數增加約 2 倍。

- 串接保留所有特徵，但這可能引入不必要的冗餘，增加訓練的難度，甚至可能導致過擬合。
- 增加的高維特徵可能需要更長的訓練時間來學習有效的權重，導致訓練效率降低。

4.5

替代方法：加權加法 (Weighted Addition)，公式： $x_{out} = \alpha \cdot f(x_{in}) + \beta \cdot x_{in}$ ，其中 α 和 β 是可學習參數，用於動態調整主輸出和殘差輸入的比例。

優點：

- 五靈活性更高：通過學習 α 和 β ，模型能動態調整主特徵和殘差特徵的權重。
- 維持低計算成本：與加法相似，維度不變，計算複雜度較低。
- 平衡信息權重：能更有效地根據輸入特性自適應地分配特徵的重要性。

挑戰：

- 增加了額外的參數 (α 和 β)，可能稍微增加訓練複雜度。
- 在某些極端情況下，可能導致學習不穩定，需要額外的正則化。

5.

推導過程如附圖：

<p>Multinomial:</p> $P(y_i x_i, \theta) = \prod_{k=1}^c \hat{y}_{i,k}^{y_{i,k}}$ <ul style="list-style-type: none"> • x: sample • \hat{y}: predict prob. • y: true label 	<p>MLE:</p> $L(\theta) = \prod_{i=1}^n \prod_{k=1}^c \hat{y}_{i,k}^{y_{i,k}}$ $\ell(\theta) = \ln L(\theta) = \sum_{i=1}^n \sum_{k=1}^c y_{i,k} \ln \hat{y}_{i,k}$ $\Rightarrow \text{Goal: } \max \ell(\theta) = \sum_{i=1}^n \sum_{k=1}^c y_{i,k} \ln \hat{y}_{i,k}$
<p>Cross Entropy:</p> $E_{CE} = - \sum_{i=1}^n \sum_{k=1}^c y_{i,k} \ln \hat{y}_{i,k}$ $\Rightarrow \text{Goal: } \min E_{CE} \equiv \max [-E_{CE}]$ $= \sum_{i=1}^n \sum_{k=1}^c y_{i,k} \ln \hat{y}_{i,k}$	

等价

統計方面，最大化觀察數據的生成機率，模型學習了數據的潛在分布。網路學習方面，在交叉熵損失下，模型調整權重，使得輸出機率對應的對數似然達到最大化。因此，最小化交叉熵損失等價於最大化觀測標籤在給定參數下的機率，這樣的學習目標與統計意圖完全一致。

6.1 Self-information :

$$P(A) = 0.4, P(B) = 0.3, P(C) = 0.2, P(D) = 0.1$$

$$\text{Self-information: } I(x) = -\log_2 P(x)$$

$$I(A) = -\log_2(0.4) = -\log_2(2^{-1.3219}) \approx 1.3219 \text{ bits}$$

$$I(B) = -\log_2(0.3) = -\log_2(2^{-1.7370}) \approx 1.7370 \text{ bits}$$

$$I(C) = -\log_2(0.2) = -\log_2(2^{-2.3219}) \approx 2.3219 \text{ bits}$$

$$I(D) = -\log_2(0.1) = -\log_2(2^{-3.3219}) \approx 3.3219 \text{ bits}$$

6.2 Entropy of classifier's predictions:

$$H(p) = - \sum_{x \in \{A, B, C, D\}} p(x) \log_2 p(x)$$

$$H(p) = -[0.4 \log_2 0.4 + 0.3 \log_2 0.3 + 0.2 \log_2 0.2 + 0.1 \log_2 0.1] = 1.8465 \text{ bits}$$

6.3 Cross-entropy:

$$Q(A) = 0.3, Q(B) = 0.3, Q(C) = 0.3, Q(D) = 0.1$$

$$H(Q, P) = - \sum_{x \in \{A, B, C, D\}} Q(x) \log_2 P(x)$$

$$H(Q, P) = -[0.3 \log_2 0.4 + 0.3 \log_2 0.3 + 0.3 \log_2 0.2 + 0.1 \log_2 0.1] = 1.9465 \text{ bits}$$

6.4 KL Divergence:

$$\begin{aligned}
 D_{KL}(Q||P) &= \sum_{x \in \{a, b, c, d\}} Q(x) \log_2 \frac{Q(x)}{P(x)} \\
 &= \sum_x Q(x) [\log_2 Q(x) - \log_2 P(x)] \\
 &= H(Q, P) - H(Q)
 \end{aligned}$$

$$H(Q) = -[0.3 \log_2 0.3 + 0.3 \log_2 0.3 + 0.3 \log_2 0.3 + 0.1 \log_2 0.1] = 1.8955 \text{ bits}$$

$$D_{KL}(Q||P) = 1.9465 - 1.8955 = 0.051 \text{ bits}$$

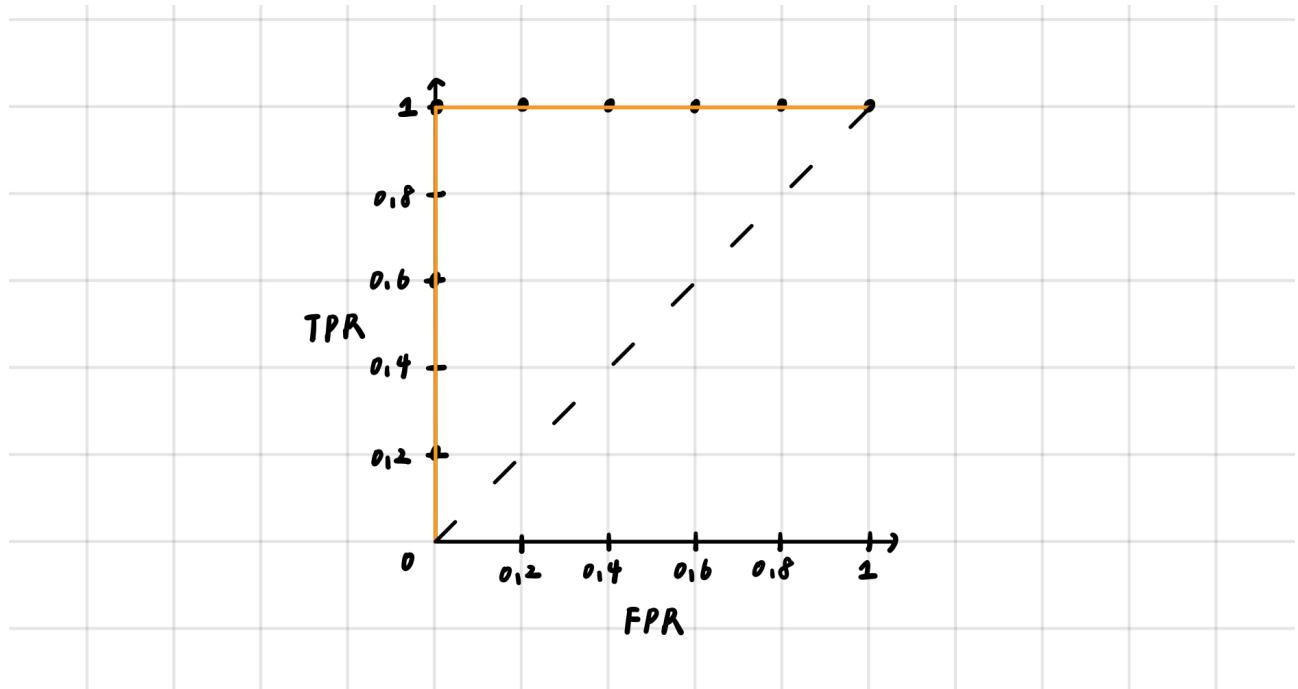
KL 散度衡量兩個分佈之間的差異。在這裡， $D_{KL}(Q||P)$ 表示分類器的預測分佈 P 與真實分佈 Q 的偏離程度。較小的 KL 散度（如 0.051 bits）表明分類器的預測接近真實分佈，但仍有改進空間。這對模型性能來說是重要的診斷指標，幫助理解分類器對不同類別的預測是否合理。

7.1

計算準則：對於每個閾值，將預測機率與閾值比較，大於或等於閾值的樣本視為正例，否則視為負例。

Threshold	True Positive	False Positive	TPR	FPR
0.1	5	5	1.0	1.0
0.2	5	5	1.0	1.0
0.3	5	4	1.0	0.8
0.4	5	3	1.0	0.6
0.5	5	2	1.0	0.4
0.6	5	1	1.0	0.2
0.7	5	0	1.0	0.0
0.8	3	0	0.6	0.0
0.9	1	0	0.2	0.0
1.0	0	0	0.0	0.0

7.2 ROC曲線



7.3 AUCROC

AUCROC 是 ROC 曲線下的面積，表示分類器的整體性能，計算結果為 1。這表明分類器的性能相當好，其預測能力遠高於隨機猜測。

8.1

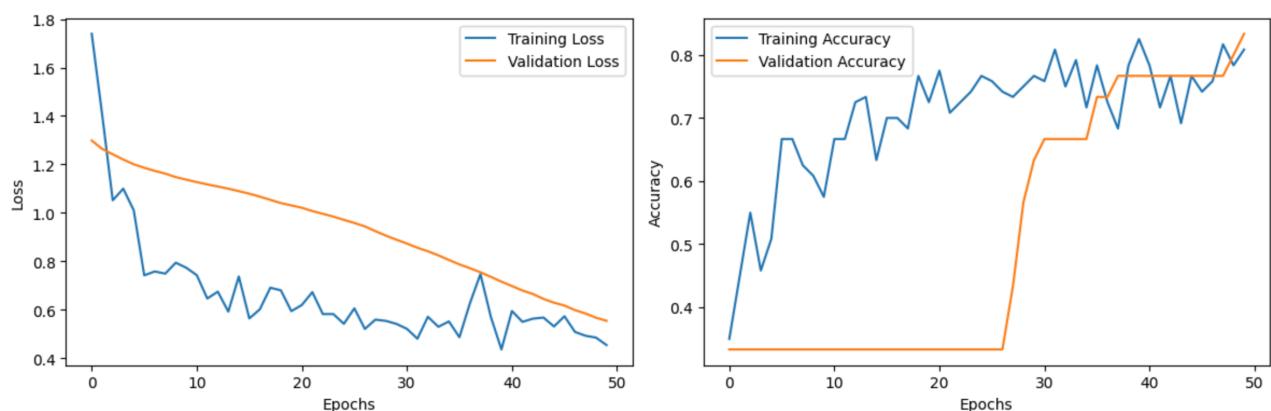
Colab URL: <https://colab.research.google.com/drive/1qZe4wVb6ejlIdXScalUAx9aw-fTZ2tMS#scrollTo=bXehAnGs9X1>

8.2

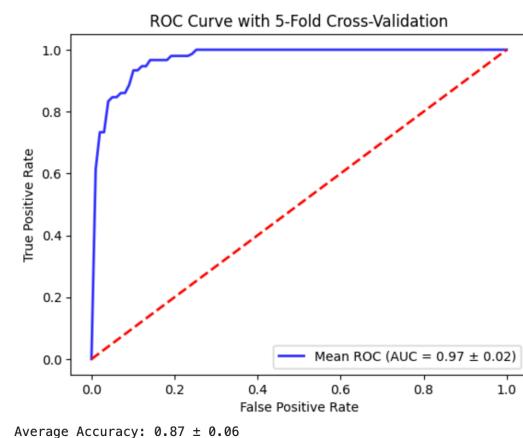
使用預設程式碼，除了最後一層激活函保持為 Softmax 外，比較其他激活函數改為 Sigmoid 和 ReLU 的結果，以下為實驗結果。

Sigmoid

訓練過程圖：



ROC曲線圖：

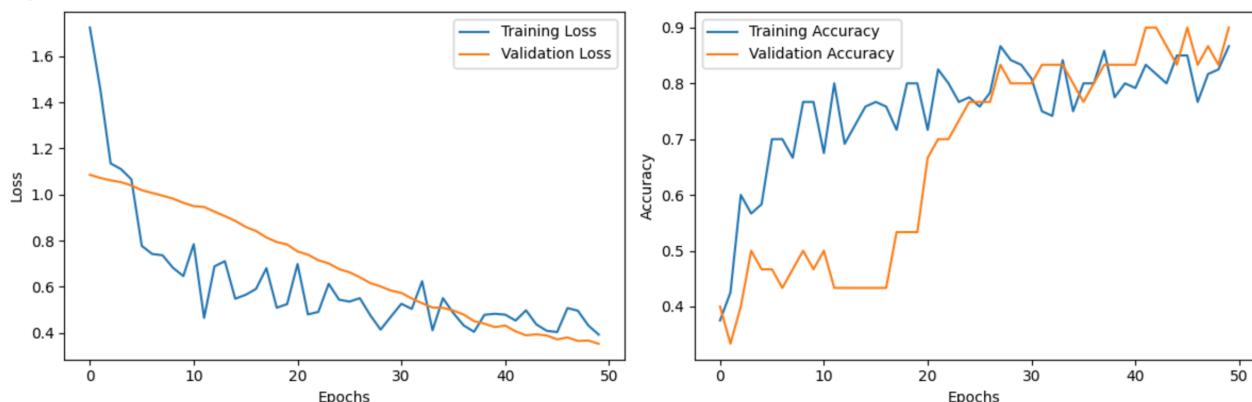


實驗結果：

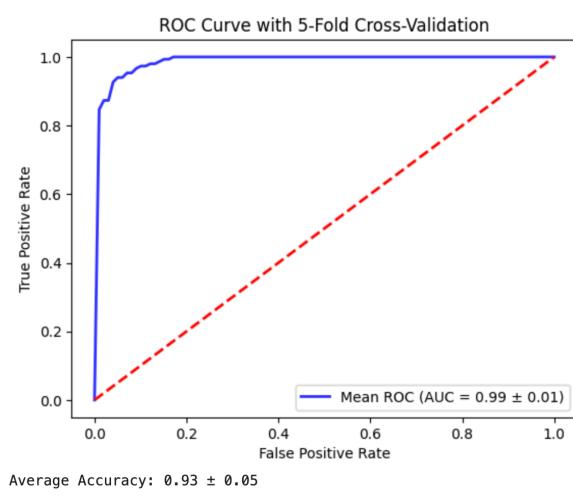
Activation	Test Average Accuracy	Test AUC
Sigmoid	0.87 ± 0.06	0.97 ± 0.02

ReLU

訓練過程圖：



ROC曲線圖：



實驗結果：

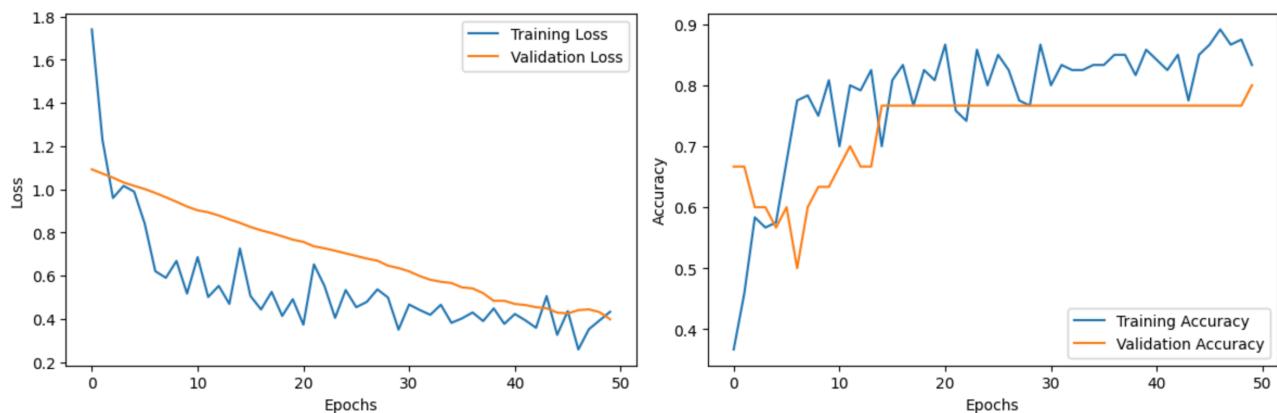
Activation	Test Average Accuracy	Test AUC
ReLU	0.93 ± 0.05	0.99 ± 0.01

8.3 綜合實驗

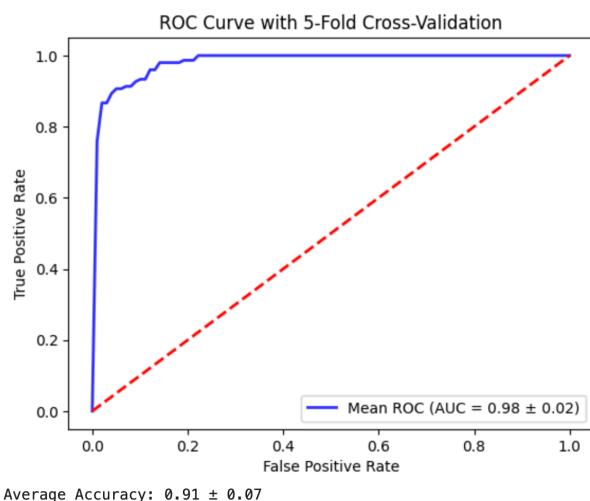
由於從上述可知 ReLU 實驗結果較好，因此後續實驗皆使用 ReLU，並針對其他不同配置行實驗，最後會再針對各個實驗做分析。

神經元數目從 16 改為 32

訓練過程圖：



ROC曲線圖：

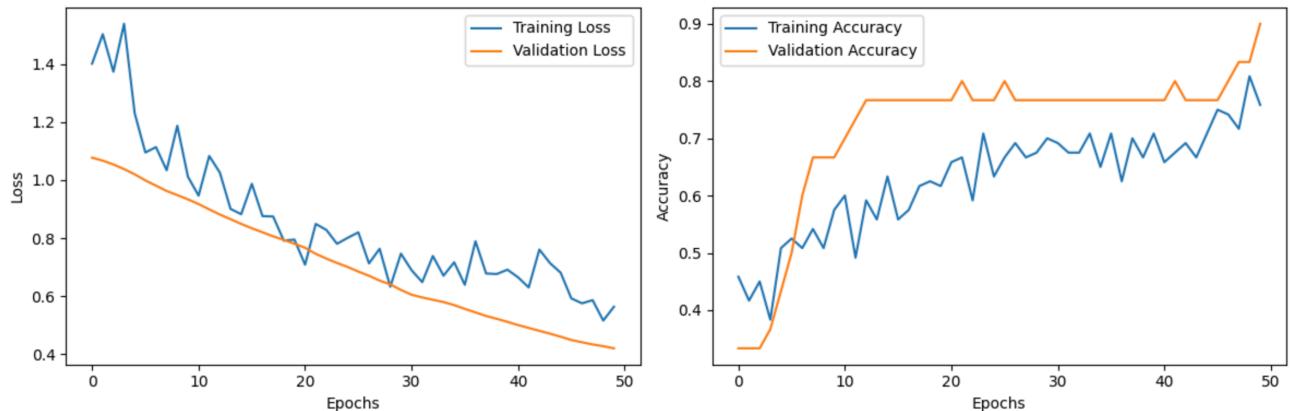


實驗結果：

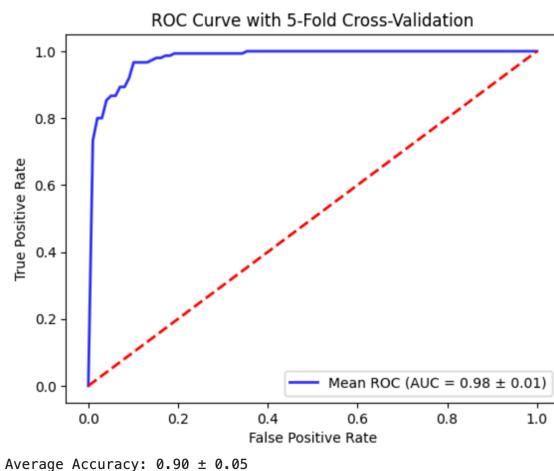
Number Of Neurons	Test Average Accuracy	Test AUC
32	0.91 ± 0.07	0.98 ± 0.02

優化器從 SGD 改成 ADAM

訓練過程圖：



ROC曲線圖：

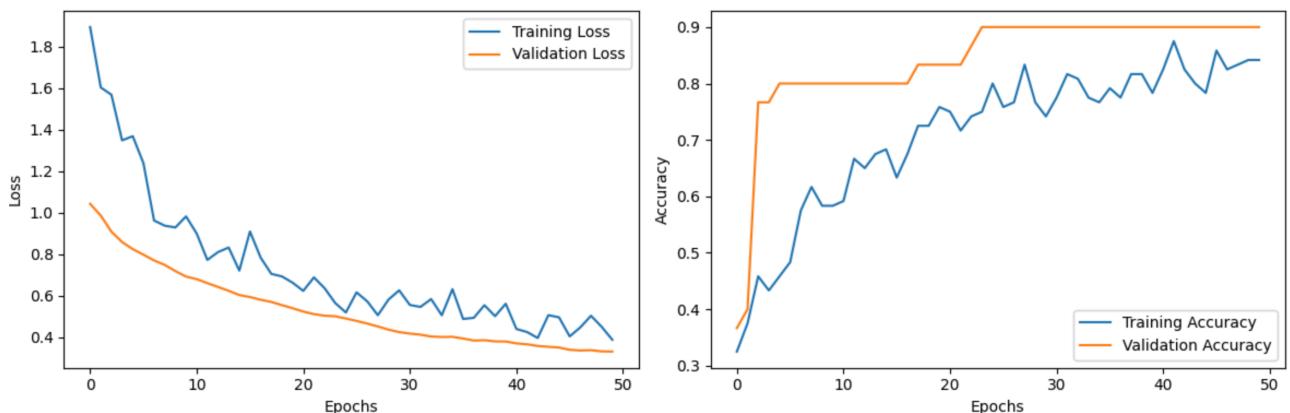


實驗結果：

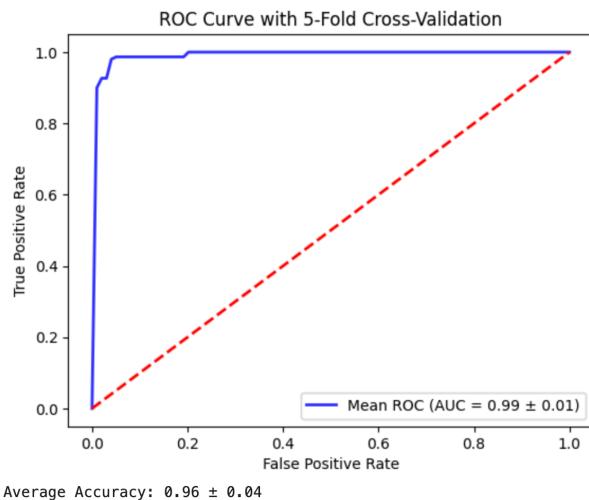
Optimizer	Test Average Accuracy	Test AUC
Adam	0.90 ± 0.05	0.98 ± 0.01

標準化從 MinMaxScalar 改成 StandScalar

訓練過程圖：



ROC曲線圖：



實驗結果：

Normalization	Test Average Accuracy	Test AUC
StandScalar	0.96 ± 0.04	0.99 ± 0.01

實驗總結

做完上述所有實驗，整理成表格後如下表所示：

Activation	Number Of Neurons	Optimizer	Normalization	Test Average Accuracy	Test AUC
Sigmoid	16	SGD	MinMax	0.87 ± 0.06	0.97 ± 0.02
ReLU	16	SGD	MinMax	0.93 ± 0.05	0.99 ± 0.01
ReLU	32	SGD	MinMax	0.91 ± 0.07	0.98 ± 0.02
ReLU	16	ADAM	MinMax	0.90 ± 0.05	0.98 ± 0.01
ReLU	16	SGD	Stand	0.96 ± 0.04	0.99 ± 0.01

從上述實驗中，可以發現在這個模型和資料集中，激活函數使用 ReLU 的結果比 Sigmoid 的結果好，可能是因為其收斂較快。神經元數量從 16 改為 32，表現結果並沒有比較好，說明在此資料集中，小的模型就可以有很好的表現。接著將優化器從 SGD 改為 ADAM，發現效果並沒有提升。再來將標準化方法從 MinMax 改為 Stand，發現結果大幅提升，因此最終選擇這個模型。