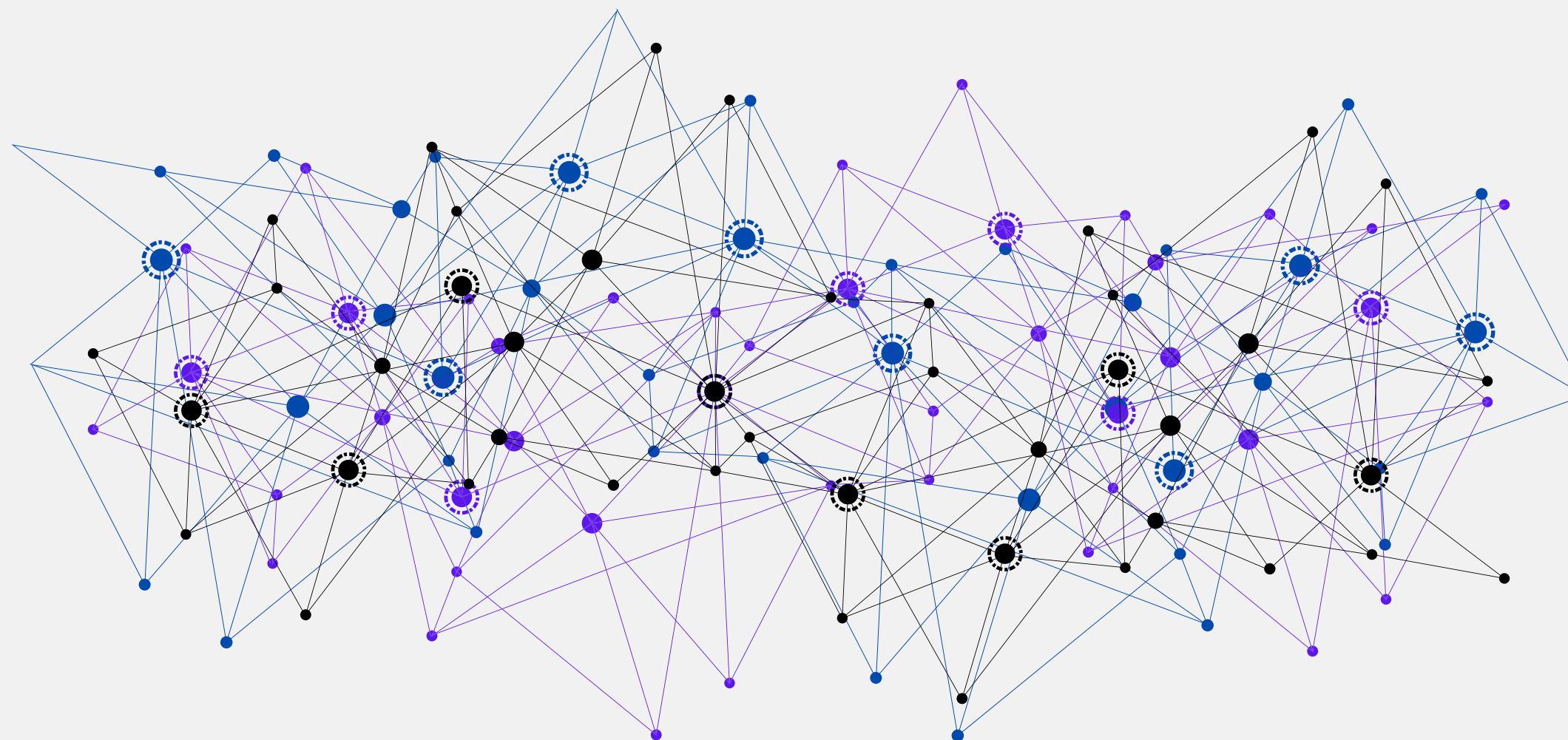


阿米巴神經網絡說明書



作者

李翱汶博士

$$\mathbf{x}_{k+1} = (\mathcal{W}_k \sigma_k + \beta_k) \odot (W_k \mathbf{x}_k + \mathbf{b}_k)$$

$$\varphi(\mathbf{x}_1) := \mathbf{x}_p$$

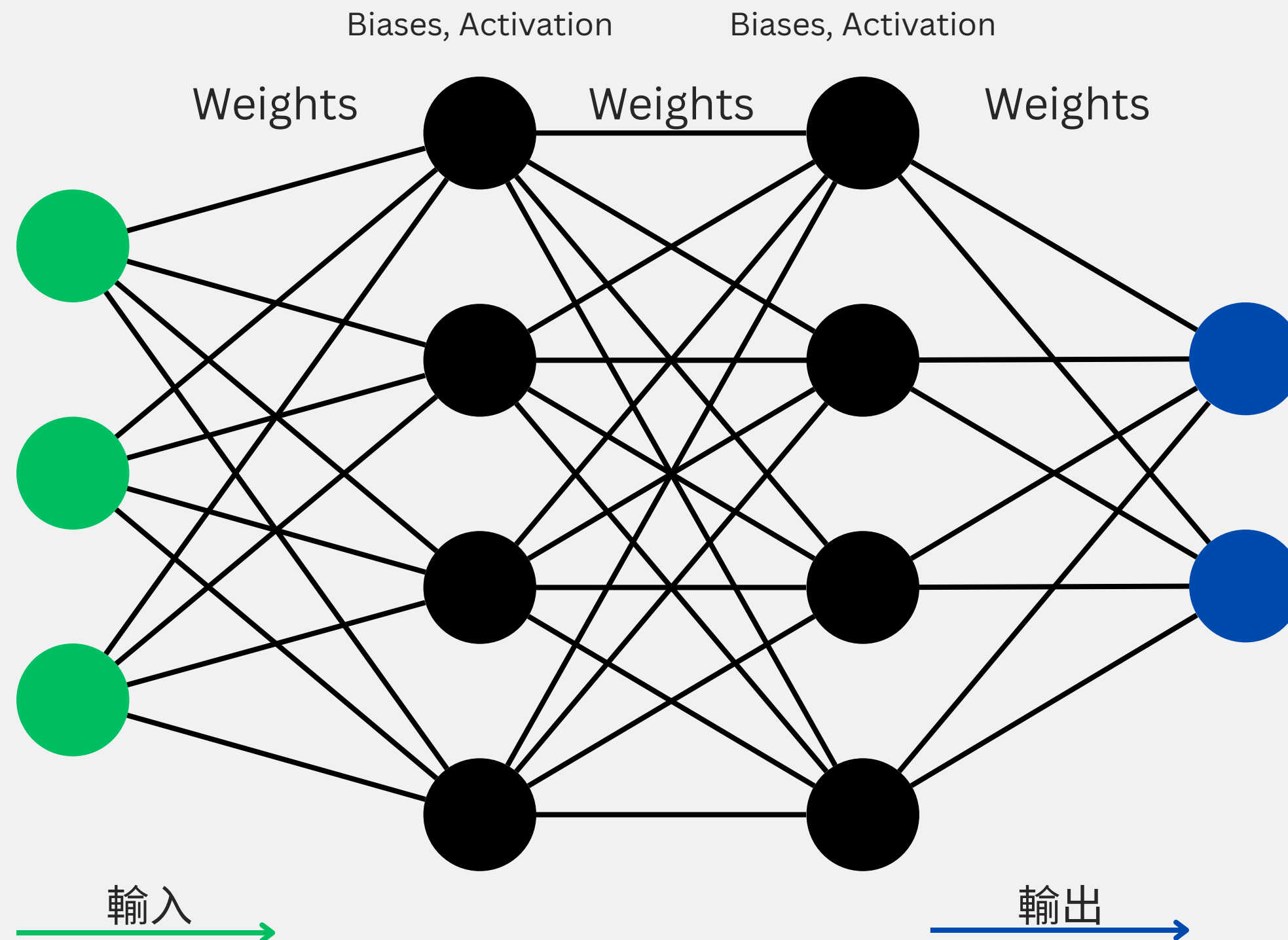
引言

- 我們目標是要研發一個更有效率的人工神經網絡 (*Artificial Neural Network (ANN)*) 架構
- 主要參考是 *Self-Expanding Neural Network (SENN)* 的架構
- 模型名字叫**阿米巴神經網絡 (*Amoeba Neural Network (ANN)*)**, 因為它可以自我擴張和變型
- 研發成功將會是一個革命性創新
- 可以令**語言大模型 (LLM)** 更加有效率, 訓練更便宜
- 由於我們還在研究和測試階段, 所以會在短時間內有大量改動
- 要有心理準備

計劃

- 我們先從最簡單的**圖像識別 (OCR) 人工智能 (Artificial Intelligence (AI))** 開始
- 原因是 OCR 已經有大量的 AI 訓練數據, 可以為我們定下標準, 作出模型表現上的比較
- 如果初步確定阿米巴神經網絡是比傳統人工神經網絡有效率的話, 我們可以再測試在其他地方的應用
- 例如: 預測附隨機性的事件和金融市場價格等等
- 如研究經費許可, 我們更可以測試於模型較小的**已知知識蒸餾 (Knowledge Distillation)** 的 LLM 上
- 例如: **ChatGPT, DeepSeek**

人工神經網絡架構概念圖



詞彙及定義

- **矢量 (Vector):** 縱排及橫排的單一排數例

- 例子:
 $(0 \quad 1 \quad 2)$ $\begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$

- **矩陣 (Matrix):** 縱橫二維的數表

- 例子:
 $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

- 矢量其實都歸類為 (一維的) 矩陣

激活函數

- 激活函數 (*Activation Function*) 是負責控制神經元輸出的函數
- 我們需要做測試的目的就是想找出最優化的激活函數
- 代碼為 `activation_function(value, degree)`
- 定義: `activation_function(value, 0)=value`
- 例子: `power(value, degree)=value**degree`

Vandermonde 矩陣

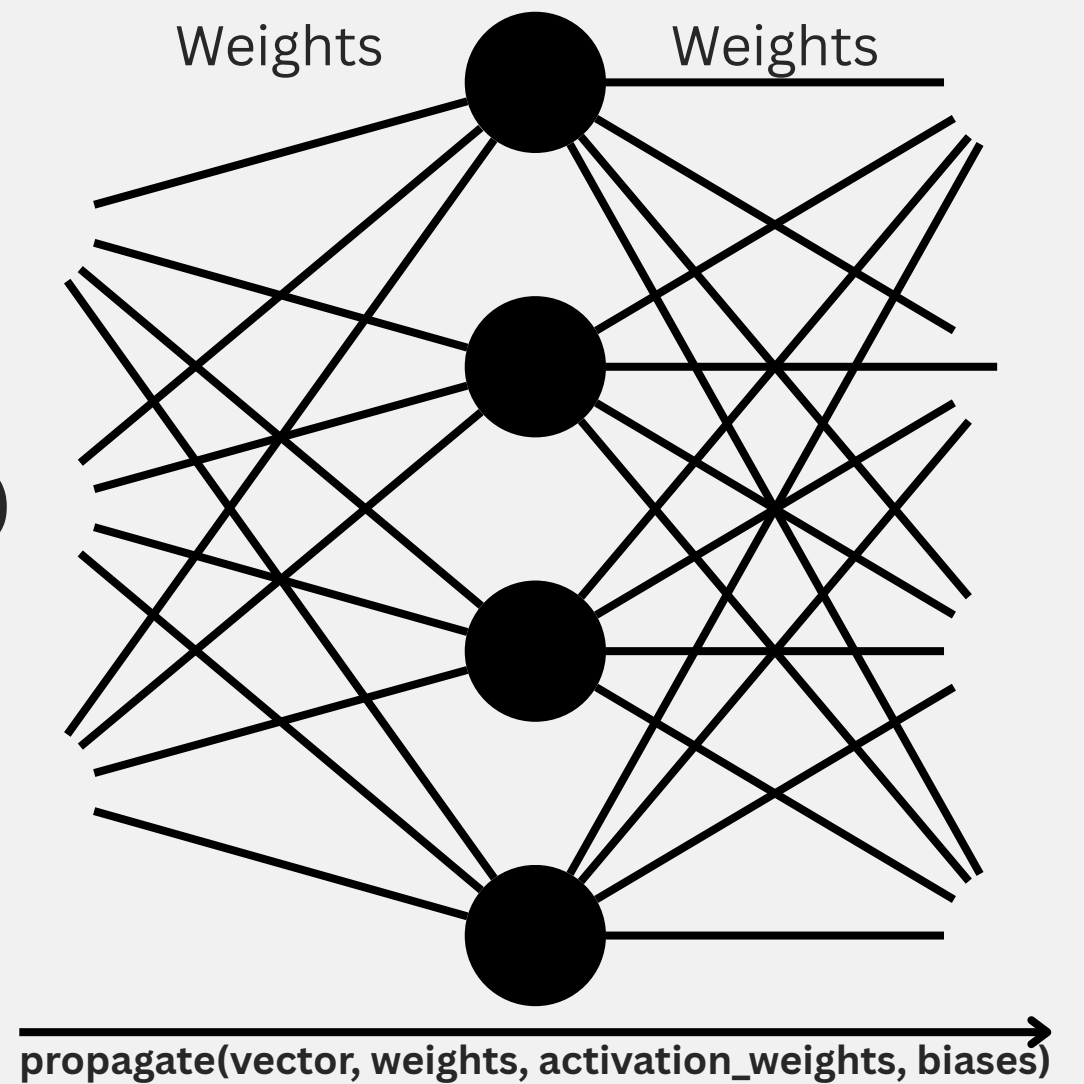
- **Vandermonde 矩陣 (*Vandermonde Matrix*)** 是激活函數應用於一個矢量而得出的矩陣
- 虛擬碼:

$$\text{vandermonde_matrix}(\text{vector}) = \begin{pmatrix} \text{activation_function}(\text{vector}[1], 0) & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \text{activation_function}(\text{vector}[m], n) \end{pmatrix}$$

- 例子: **`vector_power(vector, max_power)`**

Propagate 函數

- **Propagate 函數**是一層神經元對輸入操作後傳遞給下一層的輸出
- 虛擬碼: `propagate(vector, weights, activation_weights, biases)`



Softmax 函數

- **Softmax** 是一個矢量函數
- 虛擬碼: **softmax(vector, temperature)**
- 輸出是 AI 為每個可能性判斷出來是正確答案的偽概率
- 它「軟性」地提取矢量中的最大值
- **Temperature** 是溫度, 這裏的溫度只是個參數名字, 跟物理溫度沒有關係
- 溫度愈低, 矢量裏的最大值會愈接近 1, 其他值會接近 0
- 溫度愈高, 矢量裏的數值會較為平均分佈
- 溫度參數的設定是為了解 LLM 的輸出附有隨機性更人性化
- 溫度低 LLM 的輸出會用詞較生硬和機器化, 但每次對輸入的回答較為一致
- 溫度太高 LLM 的輸出會變得語無倫次, 胡說八道
- 在做圖像識別時我們可以把溫度設成 1, 因為我們永遠選最大值作為 AI 的答案輸出, 所以溫度參數對輸出沒有任何影響

神經網絡

- **神經網絡 (*Neural Network*)** 就是我們想構建的最終函數, 這函數輸入是我們最初問題的輸入, 輸出就是神經網絡思考後給我們的答案
- 未訓練好的神經網絡給出來的答案基本上是亂碼, 不能夠直接使用
- 虛擬碼: **Neural_Network(vector, weights, activation_weights, biases, temperature)**
- 首先我們輸入 **vector**
- 虛擬碼: **next_vector=vector**
- 然後, 每一層神經元的輸出傳遞給下一層
- 虛擬碼: **next_vector=propagate(next_vector, weights[k], activation_weights[k], biases[k])**
- 最後一層的 **next_vector** 輸出我們給他做個 **softmax**
- 虛擬碼: **output_vector=softmax(next_vector, temperature)**
- **output_vector** 就是神經網絡給出來的答案輸出

圖像識別

- 我們會使用網上開源的**用戶介面 (*User Interface* (UI))**以及 AI 訓練資料以節省時間
- 而矩陣數據我們用 Excel 儲存
- UI 需要客製化, 在圖片處理視窗裏加設「用家反饋」以及顯示“Loss”值



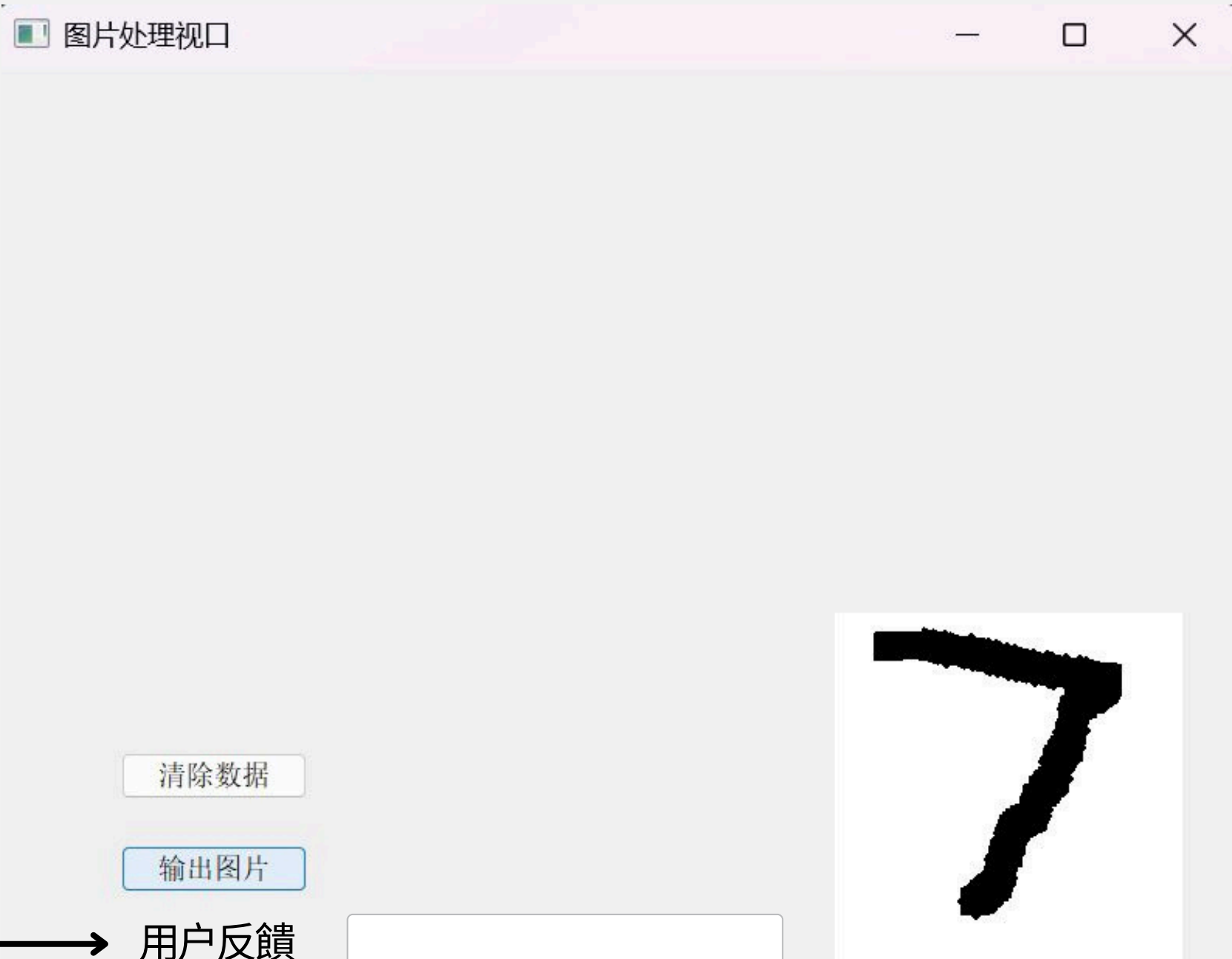
矩陣數據儲存

- 矩陣數據我們用 Excel 儲存
- 程式必須可以從 Excel 檔取出矩陣數據
- 訓練後再把訓練後的矩陣另存新 Excel 檔

用戶介面客製化

在 UI 裏加入;

- 用戶反饋: 用戶可以輸入正確答案反饋以助 AI 學習
- 顯示 Loss 值: Loss 值可以方便我們進行實時數據研究



數據格式轉換

- 先把圖片轉換為黑白值矩陣
- 再把矩陣用 **reshape** 轉成矢量
- 矩陣轉成矢量沒有格式上的要求
- 所以挑選最簡單的方式即可
- 示例:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix} \xrightarrow{\text{reshape}} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{pmatrix}$$