類神經網路作業三

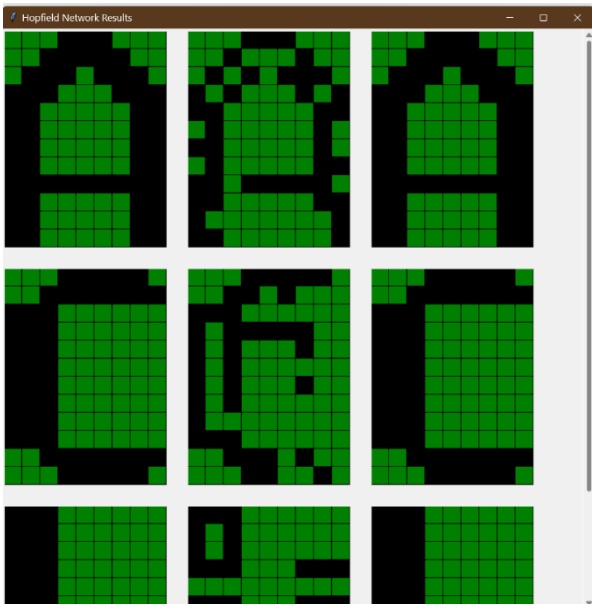
110502516 許尚軒

一、 執行說明

執行.exe 檔後就會跑出結果(exe 檔要與 data 放在同一層資料夾)

其中左側為訓練資料中間為測試資料右側為回想結果，最右側有 scrollbar

可以往下拉



basic.exe: basic data

bonus2.exe: bonus data

bonus.exe: bonus data but bias is 0

noice_basic.exe: basic data making noice

noice_nonus.exe: bonus data making noice

二、 程式簡介:

本次作業我是採用 hopfield 類神經網路架構再將結果用 tkinter 顯示出來

Hopfield:

$$x_j(n+1) = \text{sgn}(\sum_{i=1}^{p} w_{ji} x_i(n) - \theta_j) = \text{sgn}(u_j(n) - \theta_j)$$

$$= \begin{cases} 1 & \text{若} u_j(n) > \theta_j \\ x_j(n) & \text{若} u_j(n) = \theta_j \\ -1 & \text{若} u_j(n) < \theta_j \end{cases}$$

一開始先定義測資格式及初始化鍵結值和閥值為 0

```
7   class HopfieldNetwork:
8       def __init__(self, pattern_size):
9           self.pattern_size = pattern_size[0] * pattern_size[1]
10          self.weights = np.zeros((self.pattern_size, self.pattern_size))
11          self.bias = np.zeros((1, self.pattern_size))
```

訓練時依照公式

$$W = \begin{bmatrix} w_{11} & \cdots & w_{1p} \\ \vdots & \ddots & \vdots \\ w_{p1} & \cdots & w_{pp} \end{bmatrix}$$

$$= \frac{1}{p} \sum_{k=1}^{N} \underline{x}_k \underline{x}_k^T - \frac{N}{p} I \qquad \theta_j = \sum_{i=1}^{p} w_{ji}, i = 1, \cdots, p$$

改變閥值和鍵結值

```
13      def train(self, patterns):
14          for pattern in patterns:
15              pattern_flat = np.array(pattern).flatten()
16              weight_update = np.outer(pattern_flat, pattern_flat)
17              np.fill_diagonal(weight_update, 0)
18              self.weights += weight_update / self.pattern_size
19          self.bias = np.sum(self.weights, axis=0, keepdims=True)
20          self.bias = self.bias.flatten()
```

回想時根據鍵結值和閥值一步步回想直到達到最大回想次數(在此設為 100)或是達到穩定狀態

```python
    def recall(self, input_pattern, max_iterations=100):
        input_pattern_flat = np.array(input_pattern).flatten()
        for _ in range(max_iterations):
            output = np.sign(np.dot(self.weights, input_pattern_flat) - self.bias)

            for i in range(output.shape[0]):
                    if output[i] == 0:
                        output[i] = input_pattern_flat[i]

            if np.array_equal(output, input_pattern_flat):
                break
            input_pattern_flat = output
        return output.reshape(input_pattern.shape)
```

GUI:

一些基本設定

```python
def display_result(training_patterns, test_patterns, result_patterns, pattern_size):
    root = tk.Tk()
    root.title("Hopfield Network Results")
    bigger = 25
    width1 = bigger * (pattern_size[1] + 1)
    height1 = bigger * (pattern_size[0] + 1)

    canvas_frame = tk.Frame(root)
    canvas_frame.pack(side=tk.LEFT)

    scrollbar = Scrollbar(canvas_frame, orient="vertical")
    scrollbar.pack(side="right", fill="y")

    canvas = Canvas(canvas_frame, yscrollcommand=scrollbar.set, width=800, height=800)
    canvas.pack(side="left", expand=True)

    scrollbar.config(command=canvas.yview)

    frame_container = tk.Frame(canvas)
    canvas.create_window((0, 0), window=frame_container, anchor='nw')
```

將每一筆輸入顯示，出來包含 train data(canva0), test data(canva1),

recall(canva2)

```python
        canvas2 = Canvas(frame, width=width1, height=height1)
        canvas2.create_bitmap((150, 150), bitmap="gray12")
        for r in range(pattern_size[0]):
            for c in range(pattern_size[1]):
                if result[r, c] == 1:
                    canvas2.create_rectangle(c * bigger, r * bigger, (c + 1) * bigger, (r + 1) * bigger, fill="black")
                else:
                    canvas2.create_rectangle(c * bigger, r * bigger, (c + 1) * bigger, (r + 1) * bigger, fill="green")

        canvas0.grid(row=0, column=0)
        canvas1.grid(row=0, column=1)
        canvas2.grid(row=0, column=2)

    canvas.update_idletasks()
    canvas.config(scrollregion=canvas.bbox("all"))

    root.mainloop()
```

主程式:

獲取 basic trainning data 及 basic test data 並將 1 當成 1，空格當成-1，然後將上述資料變成 nparray 的 list 送去訓練及回想然後顯示

```python
if __name__ == "__main__":
    # Example training and testing patterns
    if getattr(sys, 'frozen', False):
        current_dir = os.path.dirname(sys.executable)
    else:
        current_dir = os.path.dirname(os.path.abspath(__file__))

    training_patterns = []
    training_pattern = []
    with open(os.path.join(current_dir, 'Basic_Training.txt'), 'r') as file:
        lines = file.readlines()
        for i, line in enumerate(lines):
            if not((i + 1) % 13):
                continue

            training_line = []
            for c in line:
                if c == ' ':
                    training_line.append(-1)
                elif c == '1':
                    training_line.append(1)
            training_pattern.append(training_line)

            if (i + 1) % 13 == 12:
                training_patterns.append(np.array(training_pattern))
                training_pattern = []
```

```python
    test_patterns = []
    test_pattern = []
    with open(os.path.join(current_dir, 'Basic_Testing.txt'), 'r') as file:
        lines = file.readlines()
        for i, line in enumerate(lines):
            if not((i + 1) % 13):
                continue

            test_line = []
            for c in line:
                if c == ' ':
                    test_line.append(-1)
                elif c == '1':
                    test_line.append(1)
            test_pattern.append(test_line)

            if (i + 1) % 13 == 12:
                test_patterns.append(np.array(test_pattern))
                test_pattern = []
    # Train the Hopfield network
    pattern_size = [12, 9]
    hopfield = HopfieldNetwork(pattern_size)
    hopfield.train(training_patterns)

    # Test the Hopfield network
    result_patterns = [hopfield.recall(test_pattern, max_iterations=100) for test_pattern in test_patterns]

    # Display results in GUI
    display_result(training_patterns, test_patterns, result_patterns, pattern_size)
```

Noice:

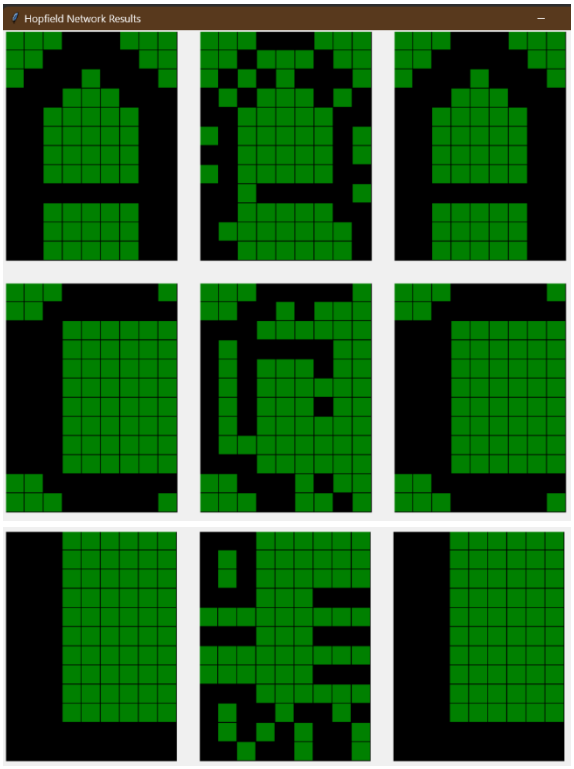
以 0.2 的機率隨機產生 noice

```
105     random_probabilities = 0.2
```

```
120                 if c == ' ':
121                     training_line.append(-1)
122                     test_line.append(-1)
123                     if random.random() < random_probabilities:
124                         test_line[-1] *= -1
125                 elif c == '1':
126                     training_line.append(1)
127                     test_line.append(1)
128                     if random.random() < random_probabilities:
129                         test_line[-1] *= -1
```

三、 實驗結果與分析及討論

Basic:



結果:

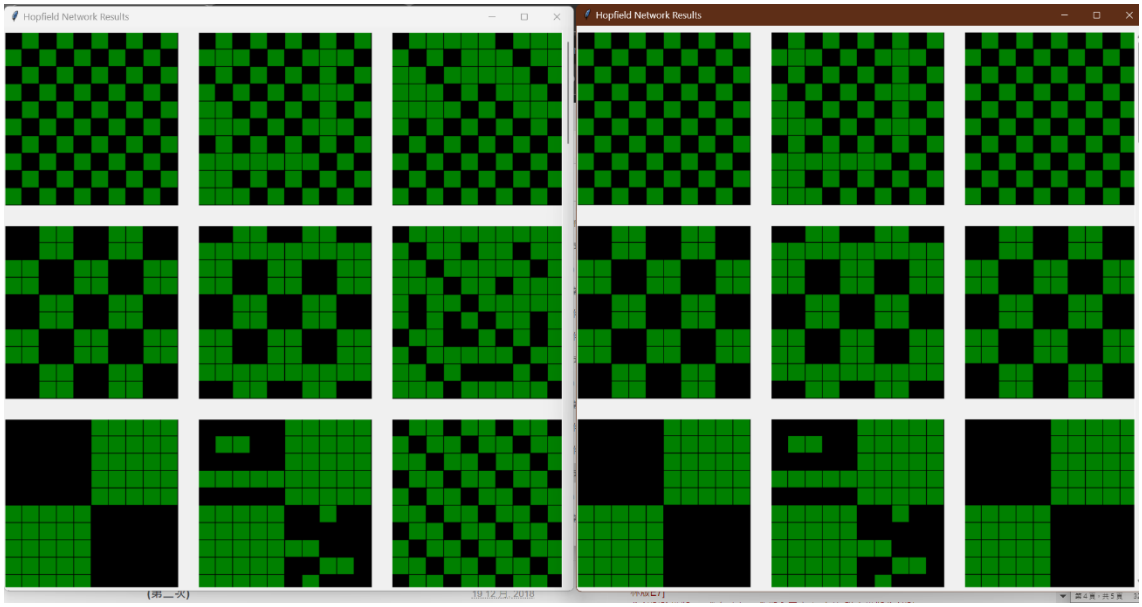
100%正確回想

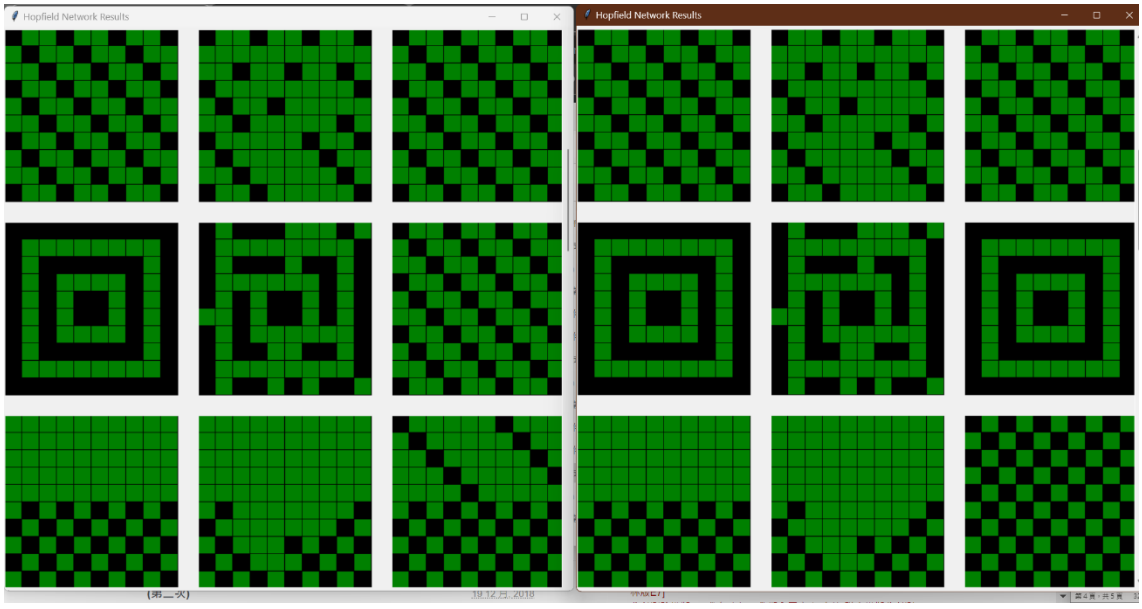
分析:

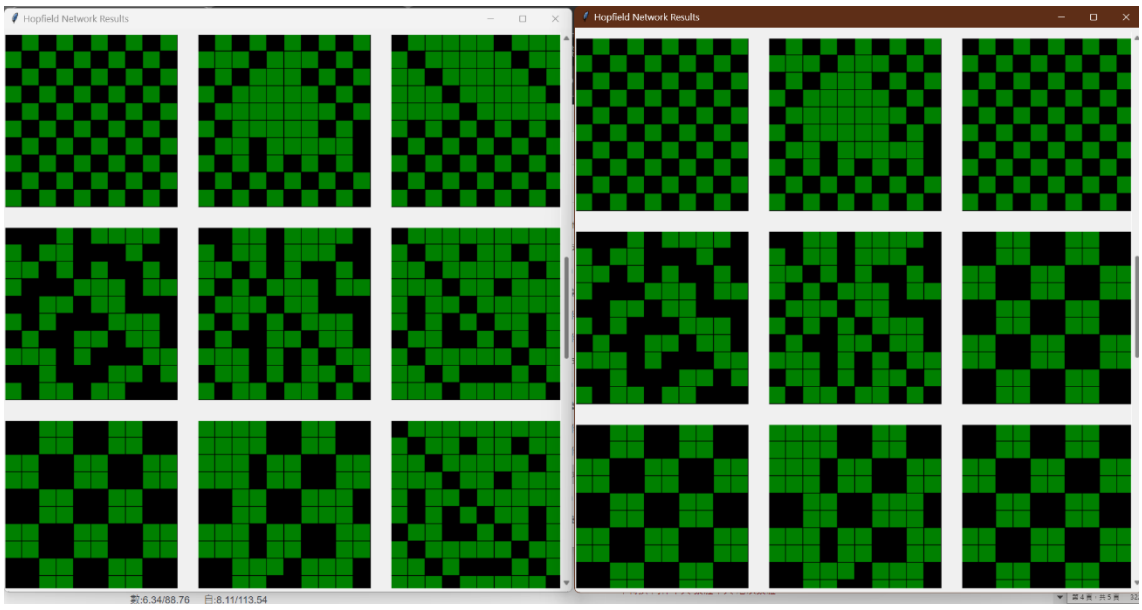
可以看到因為 $3 \leq \frac{p}{4\ln p} = 5.8$，因此三個測資都可以正常回想。

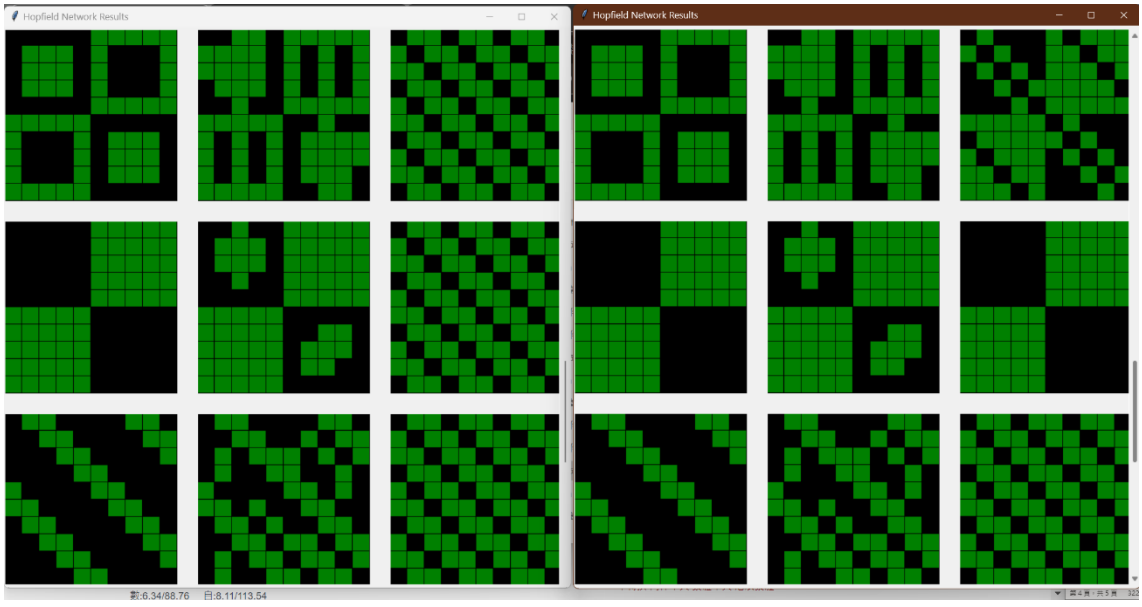
Bonus:

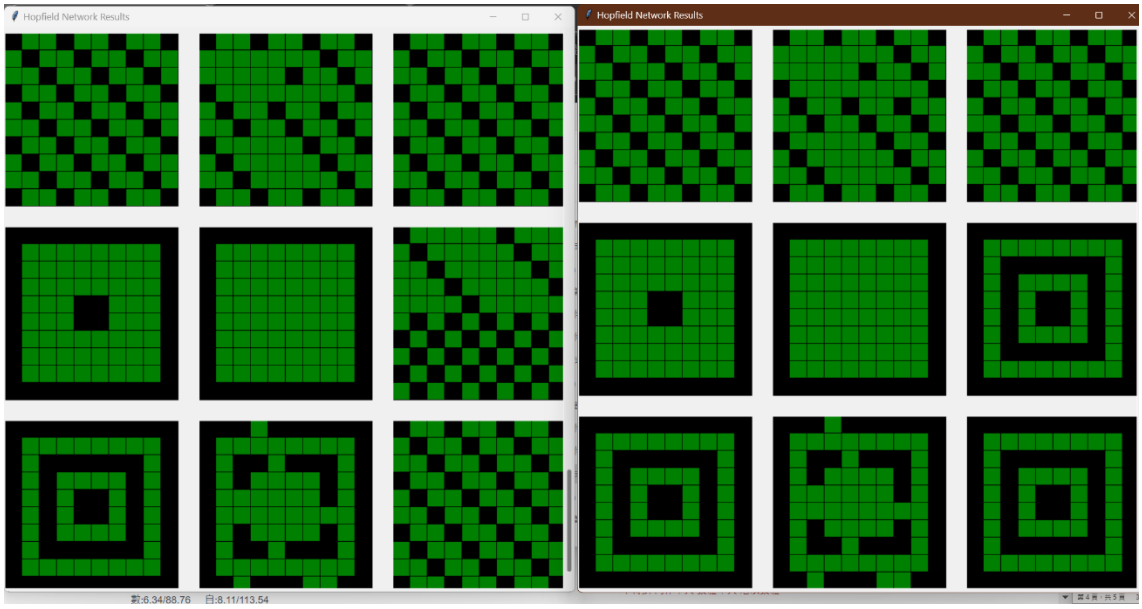
左側為 bonus(bias = 0)、右側為 bonus2(bias = wj)






Hopfield Network Results
Hopfield Network Results
數:6.34/88.76 日:8.11/113.54


Hopfield Network Results
Hopfield Network Results
數:6.34/88.76 日:8.11/113.54


Hopfield Network Results
Hopfield Network Results
數:6.34/88.76 日:8.11/113.54

分析:

Bonus: 13%正確回想

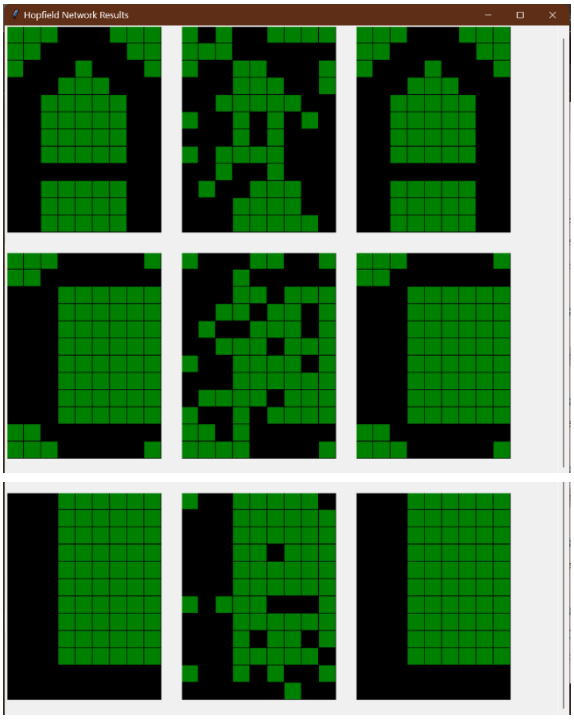
Bonus2: 66%正確回想

討論:

一開始測試 bonus2 時發現部分無法正常回想，就去測試 bonus，但發現更糟糕，所以我認為是因為 $15 \leq \frac{p}{4\ln p} = 5.4$，導致不論 bias 怎麼設都會有部分測資沒辦法正常回想。

Noice_basic:

0.2 機率產生 noice



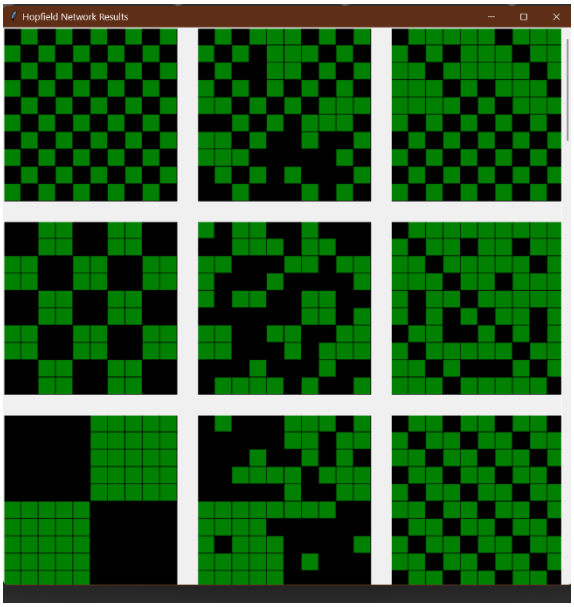
結果:

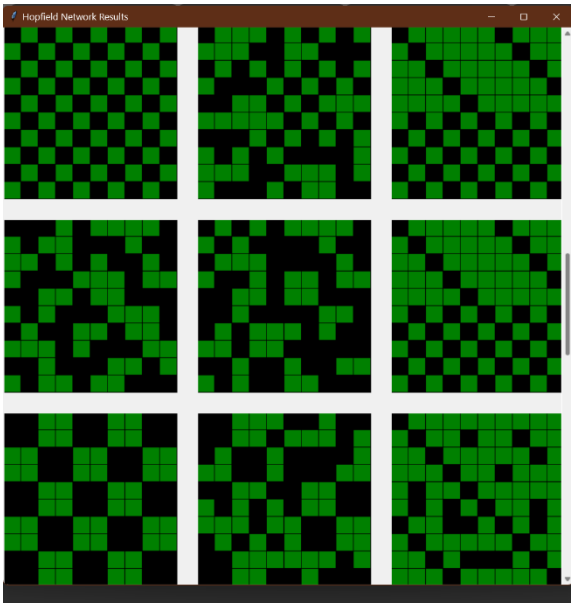
100%正確回想

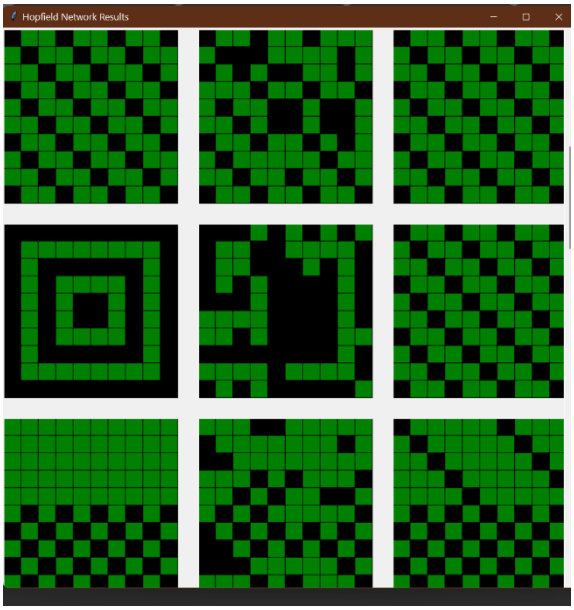
分析:

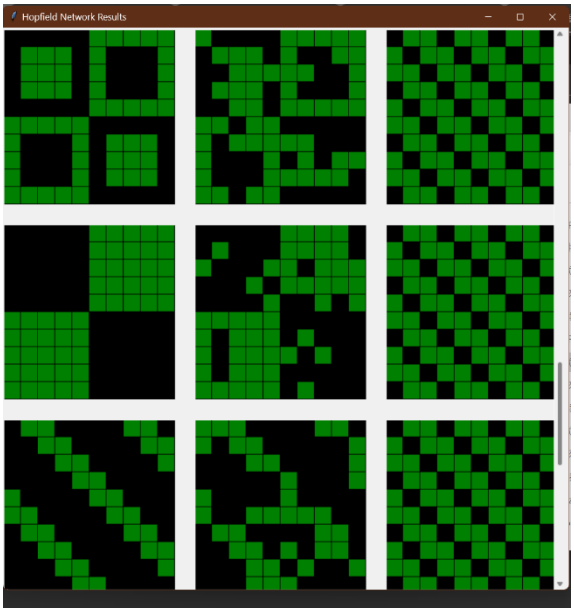
跟 bonus 一樣可以正常回想。

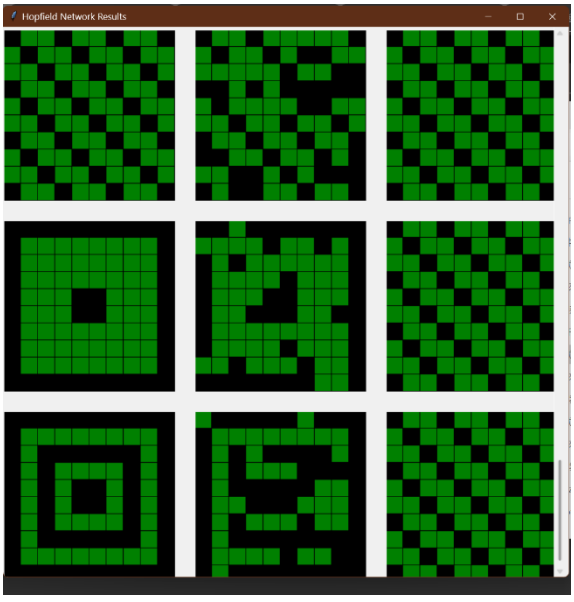
Noice_bonus:






Hopfield Network Results


Hopfield Network Results


Hopfield Network Results

結果:

7%正確回想

討論:

可以發現正確回想率非常低，甚至大部分都回想成了相似的圖形，記憶量遠低於要記憶的數量。