

資料結構 HW1

李承諺 109503520 通訊三

程式運作原理：

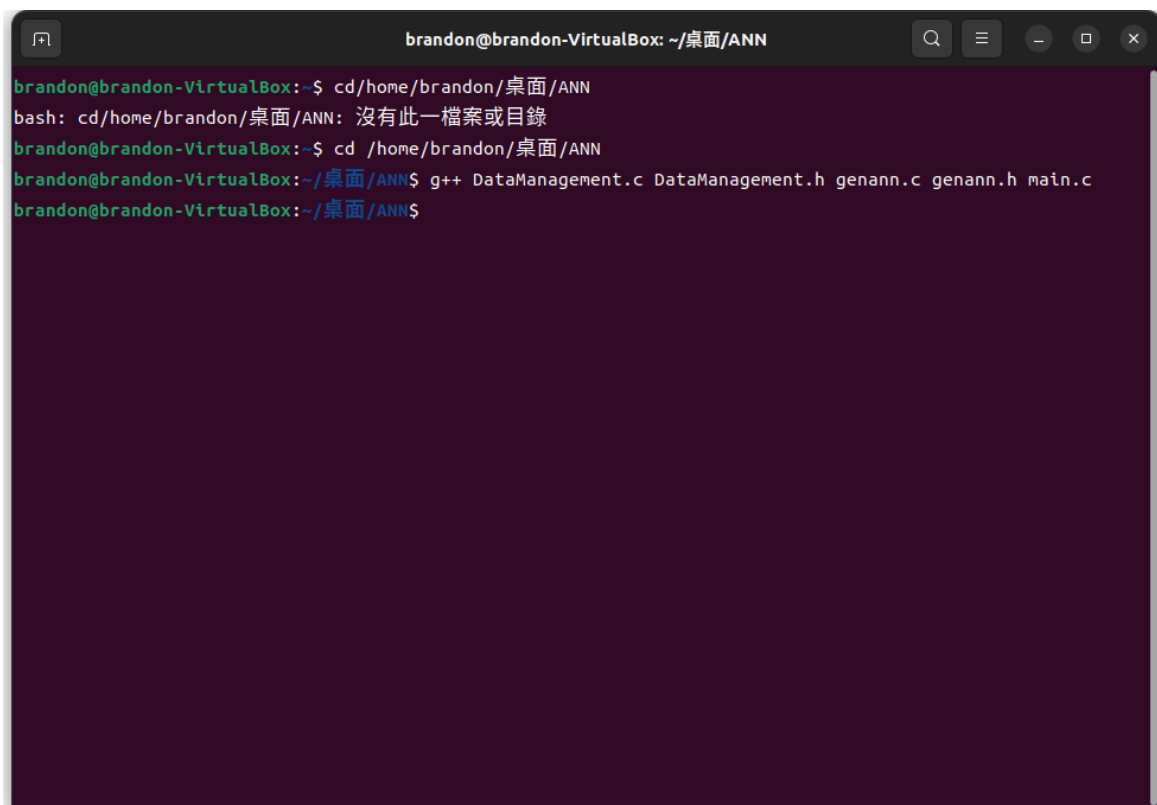
程式使用動態記憶體配置處理資料，首先產生 N bits 的所有情況(舉例來說，若輸入 4 bits 則生成 2^4 筆不同的資料)，並且將這些數據隨機排列，再將這些結果分為測試及和訓練集，其中測試集的資料數量佔 0.25%。在訓練神經網路時只會以訓練集進行訓練，在每次訓練完後會將不同的資料輸入模型(測試集、訓練集)並計算正確率

$$\text{正確率} = \frac{\text{模型輸出正確次數}}{\text{資料總數}}$$

1. 模型判斷訓練集資料的正確率 (train_accuracy)
2. 模型判斷測試集資料的正確率 (test_accuracy)
3. 模型判斷全部資料的正確率 (total_accuracy)

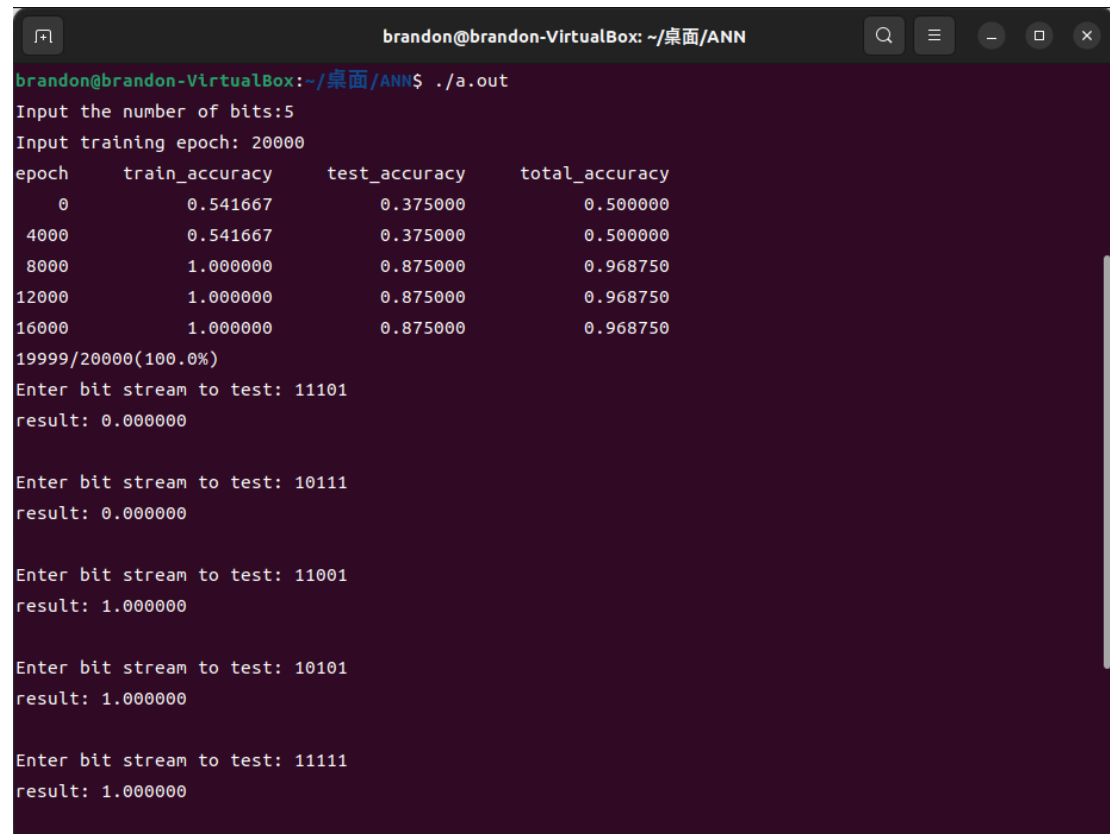
值得注意的是，在模型計算正確率時，輸入的資料不會影響其內部的權重數值。當程式運行完後，會將訓練過程輸出到 learning_accuracy.csv，之後可以透過呼叫 graph.py 來作圖

編譯結果：



```
brandon@brandon-VirtualBox: ~/桌面/ANN
brandon@brandon-VirtualBox:~$ cd/home/brandon/桌面/ANN
bash: cd/home/brandon/桌面/ANN: 沒有此一檔案或目錄
brandon@brandon-VirtualBox:~$ cd /home/brandon/桌面/ANN
brandon@brandon-VirtualBox:~/桌面/ANN$ g++ DataManagement.c DataManagement.h genann.c genann.h main.c
brandon@brandon-VirtualBox:~/桌面/ANN$
```

執行結果:



```
brandon@brandon-VirtualBox: ~/桌面/ANN
brandon@brandon-VirtualBox:~/桌面/ANN$ ./a.out
Input the number of bits:5
Input training epoch: 20000
epoch      train_accuracy  test_accuracy  total_accuracy
    0         0.541667      0.375000      0.500000
   4000         0.541667      0.375000      0.500000
   8000         1.000000      0.875000      0.968750
  12000         1.000000      0.875000      0.968750
  16000         1.000000      0.875000      0.968750
19999/20000(100.0%)
Enter bit stream to test: 11101
result: 0.000000

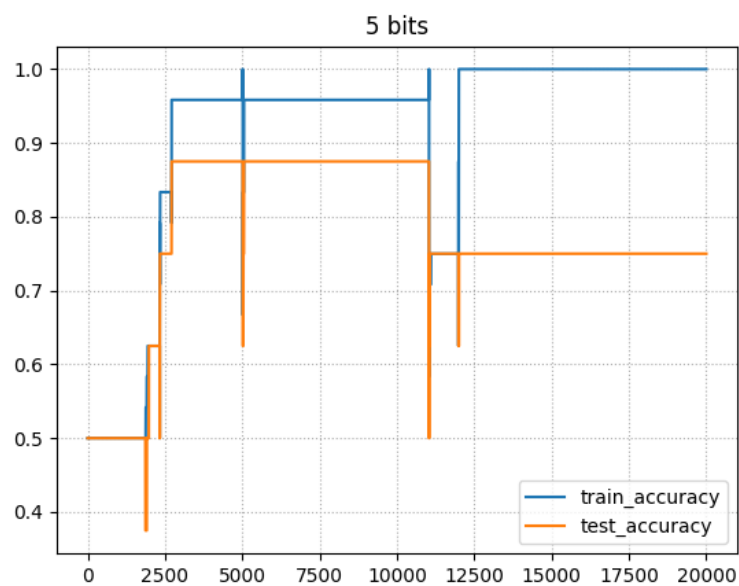
Enter bit stream to test: 10111
result: 0.000000

Enter bit stream to test: 11001
result: 1.000000

Enter bit stream to test: 10101
result: 1.000000

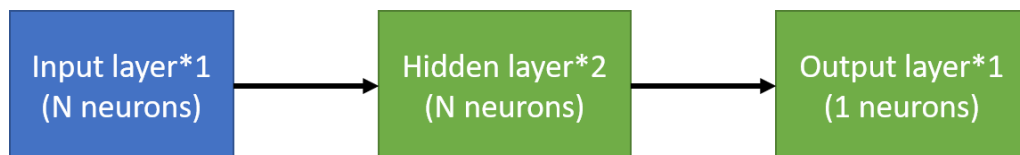
Enter bit stream to test: 11111
result: 1.000000
```

使用 python 作圖

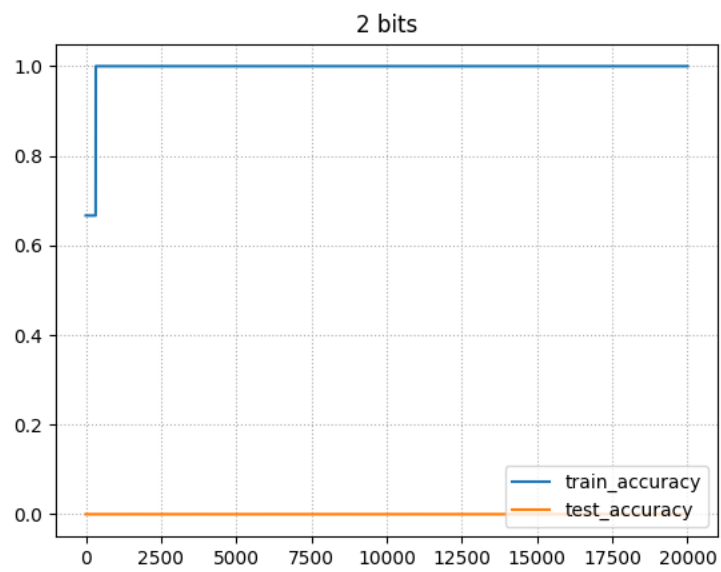


分析:

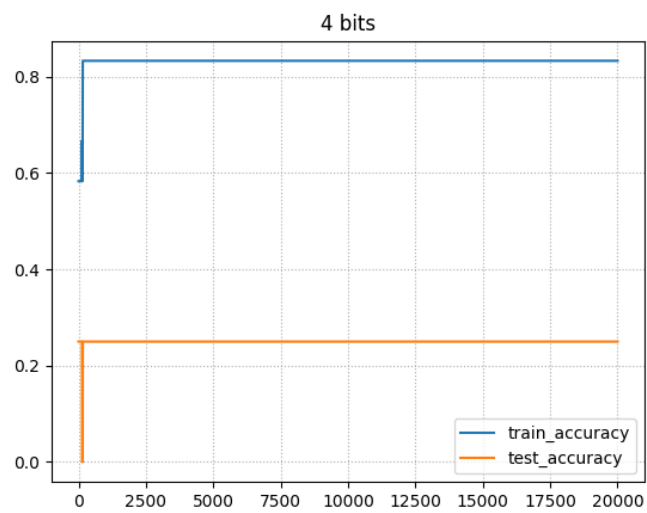
假設有 N bits 輸入，神經網路的結構為: 1 個 N 個神經元的 Input layer 以及 2 個 N 個神經元的 hidden layer，最後有一個 1 個神經元的 output layer 如圖:



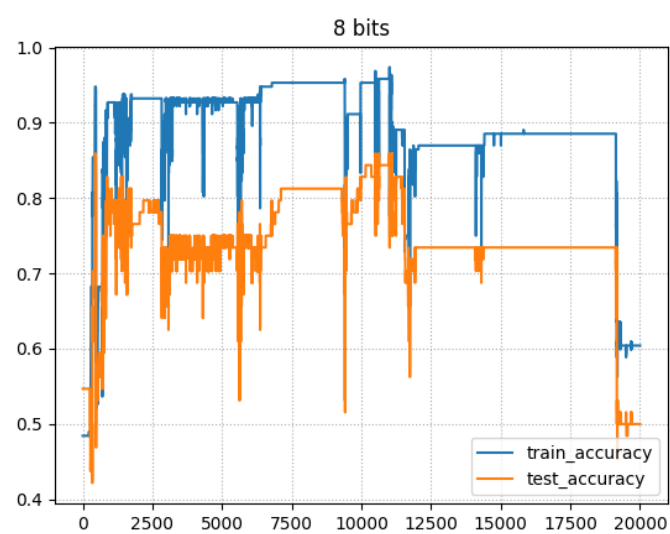
每次實驗都訓練 20000 次，訓練結果縱軸為準確率，橫軸為訓練次數(epoch)，結果如下:



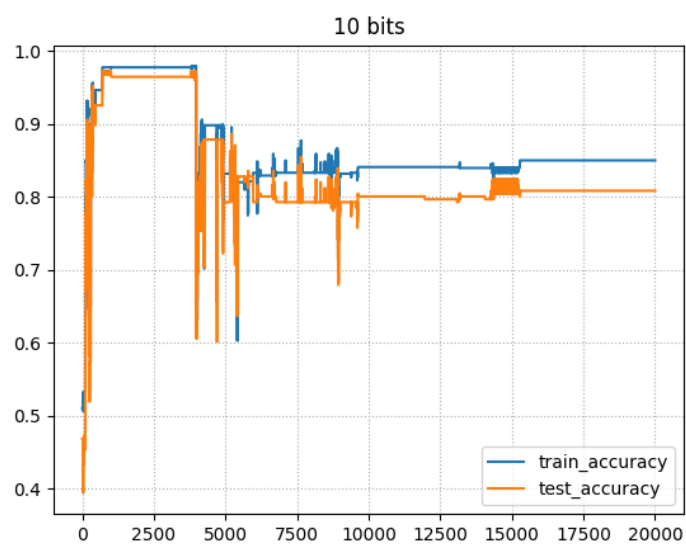
test_accuracy 最大值為 0 (0 epoch)



test_accuracy 最大值為 0.25 (0 epoch)

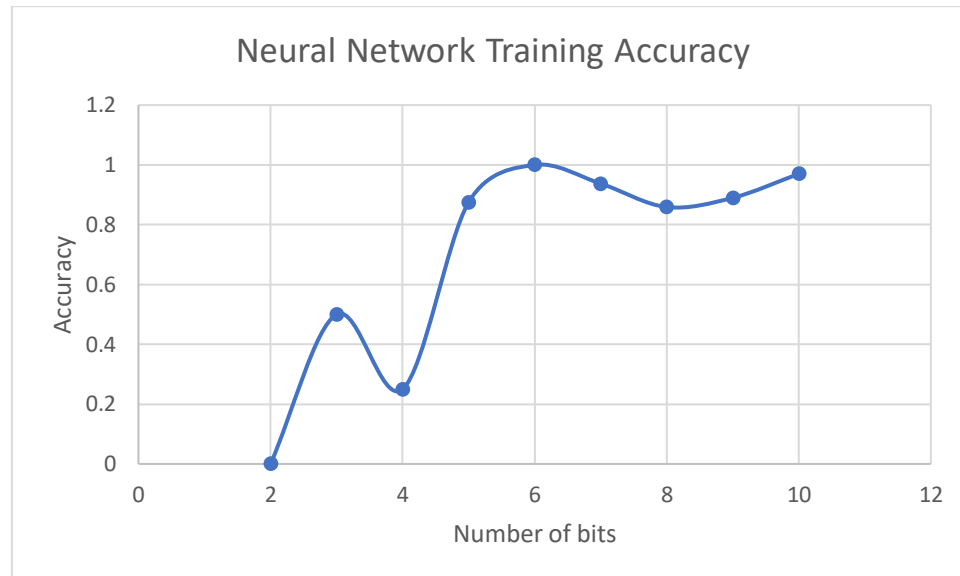


test_accuracy 最大値為 0.85 (442 epoch)



test_accuracy 最大値為 0.97 (700 epoch)

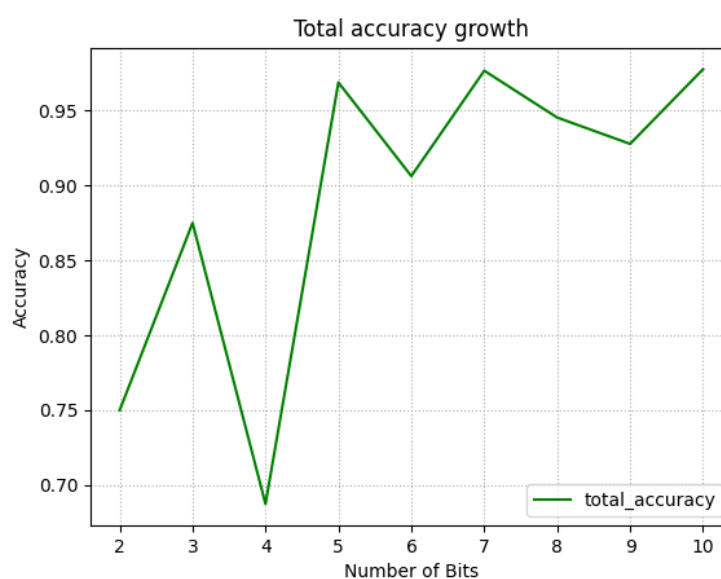
由實驗結果可以發現，在訓練之前準確率大約落在 0.5 左右，因為在訓練前模型沒有任何經驗，只能隨機猜測。在 bit 數量低的時候，由於測試集的資料種類很少，因此神經網路無法有效學習，導致預測的正確率較低；當 bit 數量越來越多時，因為可以學習的資料變多，因此神經網路的準確率會顯著上升，如圖：



(使用訓練過程中最大的 test_accuracy 作圖)

還有一點值得注意的是，訓練結果作圖會出現方形的稜角，這是因為實驗數據的種類較少，導致預測正確的機率數值種類也較少。

除了計算訓練過程正確率的變化之外，總正確率(total_accuracy)也是評估模型好壞的一個指標，因為它代表直接使用模型判斷 XOR 的正確率。由圖可以看出，bits 的數量越多，總正確率會越來越接近 100%。



(使用訓練過程中最大的 total_accuracy 作圖)