COMP9444 Neural Networks and Deep Learning

Assignment 1

Term 2, 2024

Submitted by

zID: z5504665

Name: Zilong Xia

I declare that:

This assessment item is entirely my own original work, except where I have acknowledged use of source material [such as books, journal articles, other published material, the Internet, and the work of other student/s or any other person/s.

This assessment item has not been submitted for assessment for academic credit in this, or any other course, at UNSW or elsewhere.

I understand that:

The assessor of this assessment item may, for the purpose of assessing this item, reproduce this assessment item and provide a copy to another member of the University.

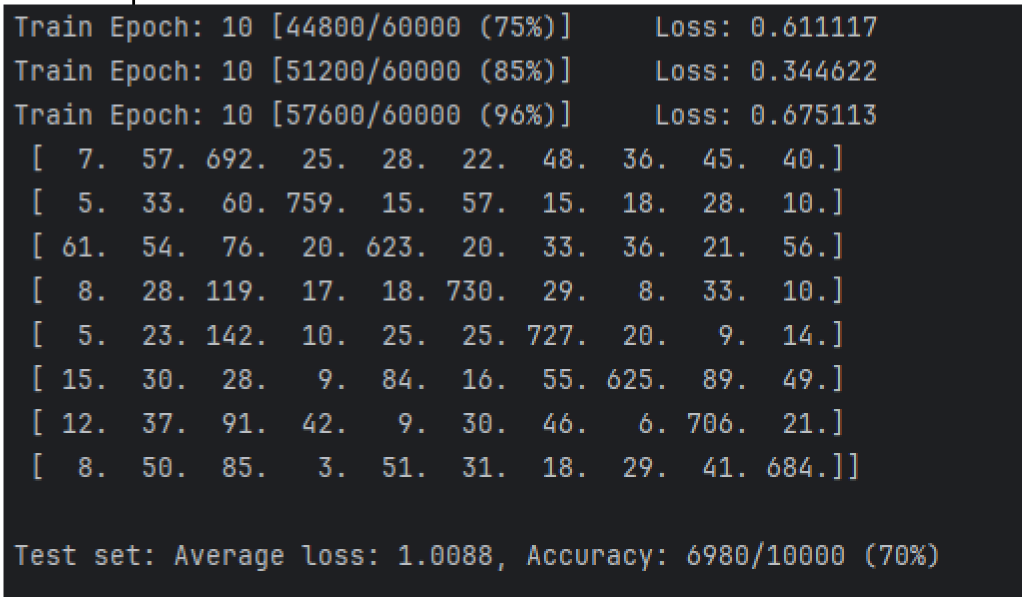
The assessor may communicate a copy of this assessment item to a plagiarism checking service (which may then retain a copy of the assessment item on its database for the purpose of future plagiarism checking).

I certify that I have read and understood the University Rules in respect of Student Academic Misconduct.

Name, student id, signature and date

Part 1: Japanese Character Recognition

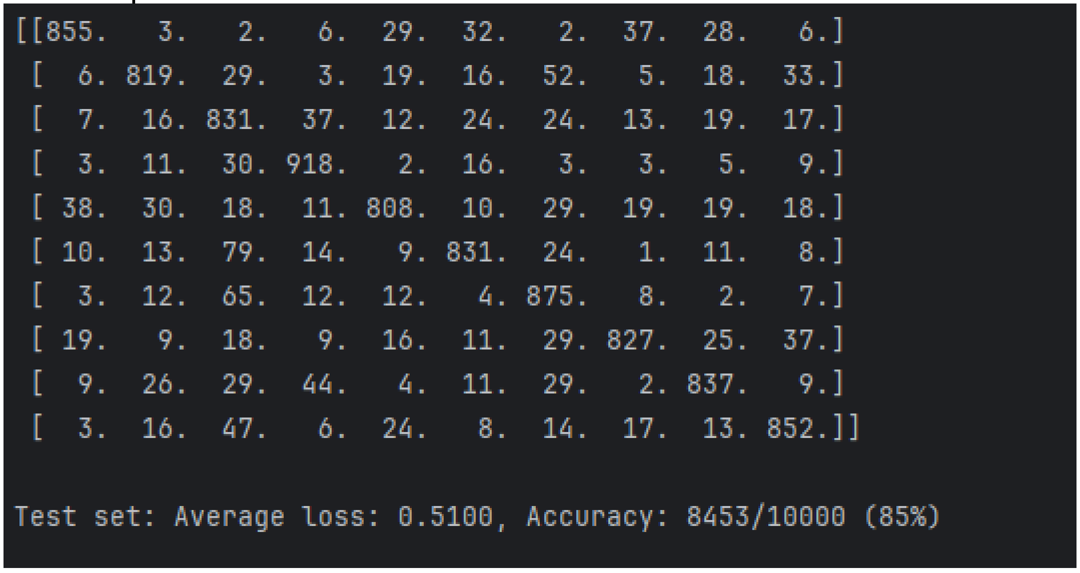
1. Answer question 1



```
Train Epoch: 10 [44800/60000 (75%)]     Loss: 0.611117
Train Epoch: 10 [51200/60000 (85%)]     Loss: 0.344622
Train Epoch: 10 [57600/60000 (96%)]     Loss: 0.675113
 [  7.  57. 692.  25.  28.  22.  48.  36.  45.  40.]
 [  5.  33.  60. 759.  15.  57.  15.  18.  28.  10.]
 [ 61.  54.  76.  20. 623.  20.  33.  36.  21.  56.]
 [  8.  28. 119.  17.  18. 730.  29.   8.  33.  10.]
 [  5.  23. 142.  10.  25.  25. 727.  20.   9.  14.]
 [ 15.  30.  28.   9.  84.  16.  55. 625.  89.  49.]
 [ 12.  37.  91.  42.   9.  30.  46.   6. 706.  21.]
 [  8.  50.  85.   3.  51.  31.  18.  29.  41. 684.]]

Test set: Average loss: 1.0088, Accuracy: 6980/10000 (70%)
```

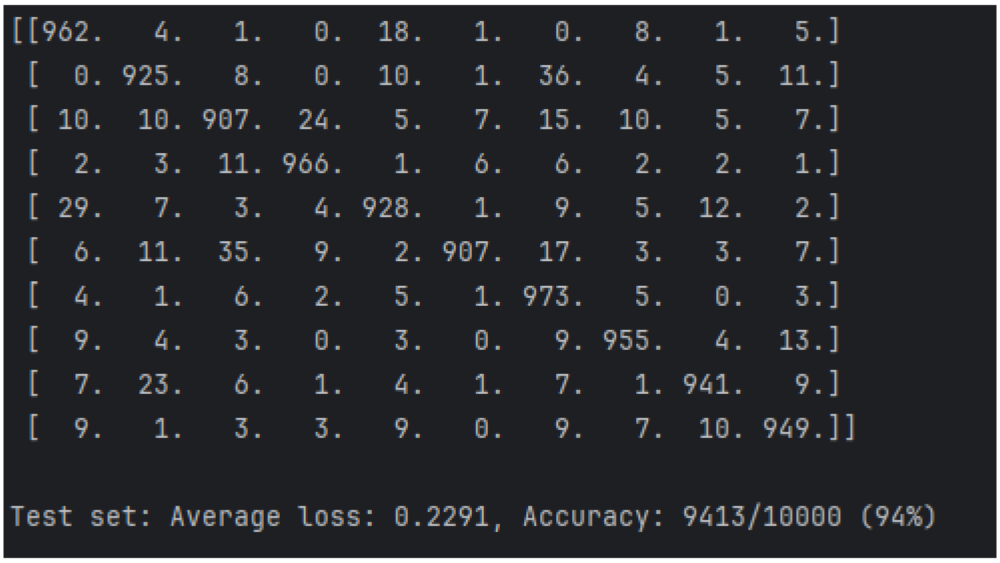
2. Answer question 2



```
[[855.   3.   2.   6.  29.  32.   2.  37.  28.   6.]
 [  6. 819.  29.   3.  19.  16.  52.   5.  18.  33.]
 [  7.  16. 831.  37.  12.  24.  24.  13.  19.  17.]
 [  3.  11.  30. 918.   2.  16.   3.   3.   5.   9.]
 [ 38.  30.  18.  11. 808.  10.  29.  19.  19.  18.]
 [ 10.  13.  79.  14.   9. 831.  24.   1.  11.   8.]
 [  3.  12.  65.  12.  12.   4. 875.   8.   2.   7.]
 [ 19.   9.  18.   9.  16.  11.  29. 827.  25.  37.]
 [  9.  26.  29.  44.   4.  11.  29.   2. 837.   9.]
 [  3.  16.  47.   6.  24.   8.  14.  17.  13. 852.]]

Test set: Average loss: 0.5100, Accuracy: 8453/10000 (85%)
```

3. Answer question 3



```
[[962.   4.   1.   0.  18.   1.   0.   8.   1.   5.]
 [  0. 925.   8.   0.  10.   1.  36.   4.   5.  11.]
 [ 10.  10. 907.  24.   5.   7.  15.  10.   5.   7.]
 [  2.   3.  11. 966.   1.   6.   6.   2.   2.   1.]
 [ 29.   7.   3.   4. 928.   1.   9.   5.  12.   2.]
 [  6.  11.  35.   9.   2. 907.  17.   3.   3.   7.]
 [  4.   1.   6.   2.   5.   1. 973.   5.   0.   3.]
 [  9.   4.   3.   0.   3.   0.   9. 955.   4.  13.]
 [  7.  23.   6.   1.   4.   1.   7.   1. 941.   9.]
 [  9.   1.   3.   3.   9.   0.   9.   7.  10. 949.]]

Test set: Average loss: 0.2291, Accuracy: 9413/10000 (94%)
```

4. Answer question 4

   a)

      NetLin: Around 70%
      NetFull: Around 85%
      NetConv: Around 94%

   b)

   NetLin:
   Parameters:
   Calculation:
   - Weights: (28×28)×10=7840(28 \times 28) \times 10 = 7840(28×28)×10=7840
   - Biases: 101010

   Total: 7840+10=78507840 + 10 = 78507840+10=7850

   NetFull:
   Parameters:
   Input to Hidden Layer: (28×28)×128+128(28 \times 28) \times 128 + 128(28×28)×128+128
   Hidden to Output Layer: 128×10+10128 \times 10 + 10128×10+10
   Calculation:
   Weights (Input to Hidden): 28×28×128=10035228 \times 28 \times 128 = 10035228×28×128=100352
   Biases (Hidden Layer): 128128128
   Weights (Hidden to Output): 128×10=1280128 \times 10 = 1280128×10=1280
   Biases (Output Layer): 101010
   Total: 100352+128+1280+10=101770100352 + 128 + 1280 + 10 = 101770100352+128+1280+10=101770

   NetConv:
   Parameters:
   First Convolutional Layer: 1×32×3×3+321 \times 32 \times 3 \times 3 + 321×32×3×3+32
   Second Convolutional Layer: 32×64×3×3+6432 \times 64 \times 3 \times 3 + 6432×64×3×3+64

Fully Connected Layer: 64×7×7×128+12864 \times 7 \times 7 \times 128 + 12864×7×7×128+128
Output Layer: 128×10+10128 \times 10 + 10128×10+10
Calculation:
Weights (First Conv Layer): 1×32×3×3=2881 \times 32 \times 3 \times 3 = 2881×32×3×3=288
Biases (First Conv Layer): 323232
Weights (Second Conv Layer): 32×64×3×3=1843232 \times 64 \times 3 \times 3 = 1843232×64×3×3=18432
Biases (Second Conv Layer): 646464
Weights (Fully Connected): 64×7×7×128=40140864 \times 7 \times 7 \times 128 = 40140864×7×7×128=401408
Biases (Fully Connected): 128128128
Weights (Output Layer): 128×10=1280128 \times 10 = 1280128×10=1280
Biases (Output Layer): 101010

Total: 288+32+18432+64+401408+128+1280+10=421642288 + 32 + 18432 + 64 + 401408 + 128 + 1280 + 10 = 421642288+32+18432+64+401408+128+1280+10=421642

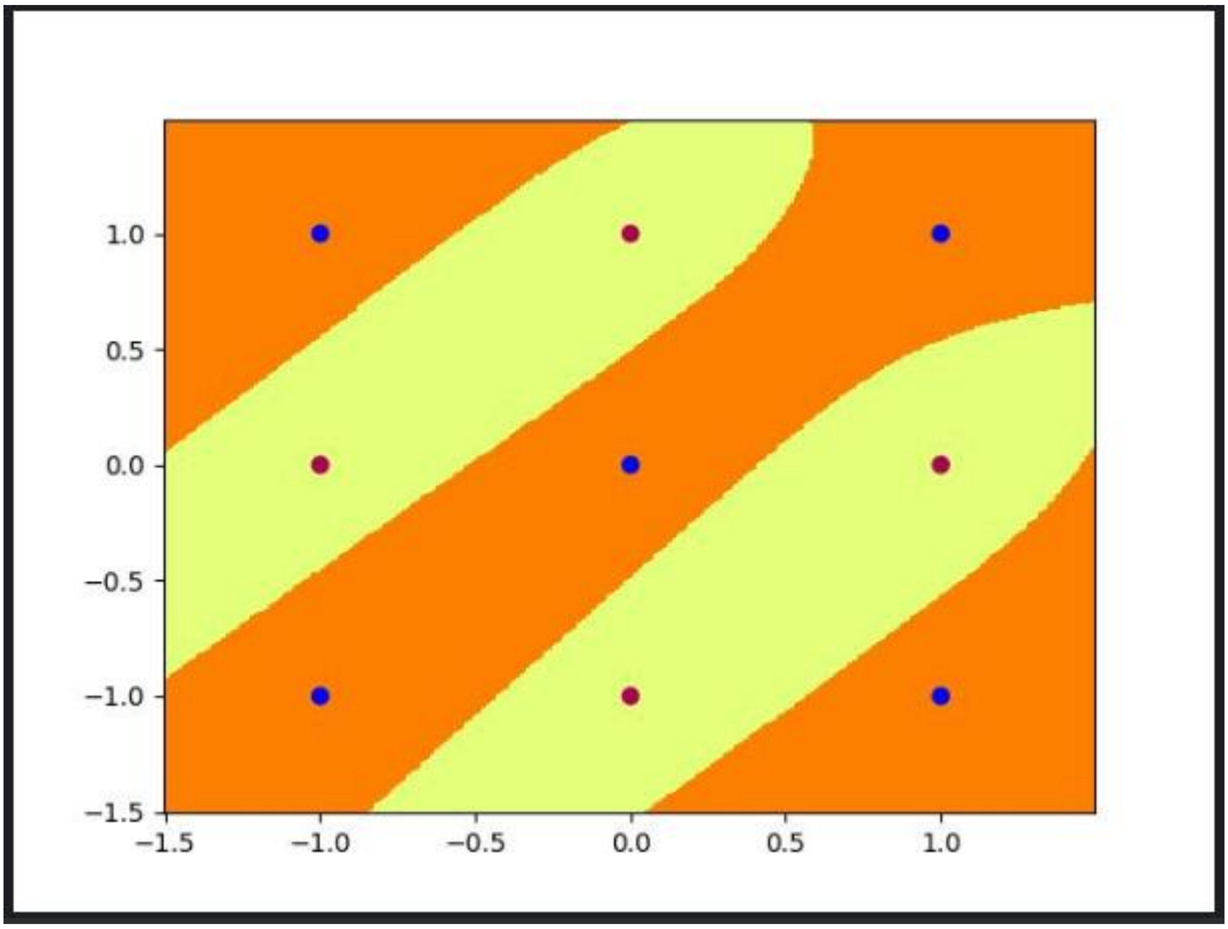
c)
NetLin:
Characters with similar shapes or strokes, such as "tsu" (3) and "su" (2), or "na" (4) and "re" (8).
The model's simplicity makes it difficult to capture subtle differences, leading to higher misclassification rates for visually similar characters.

NetFull:
Despite improved performance, characters like "ki" (1) and "ha" (5) might still be confused due to their structural similarities.
Characters with complex or less distinct features compared to others might be misclassified.
The hidden layer allows for better pattern recognition but still has limitations in distinguishing very similar characters.

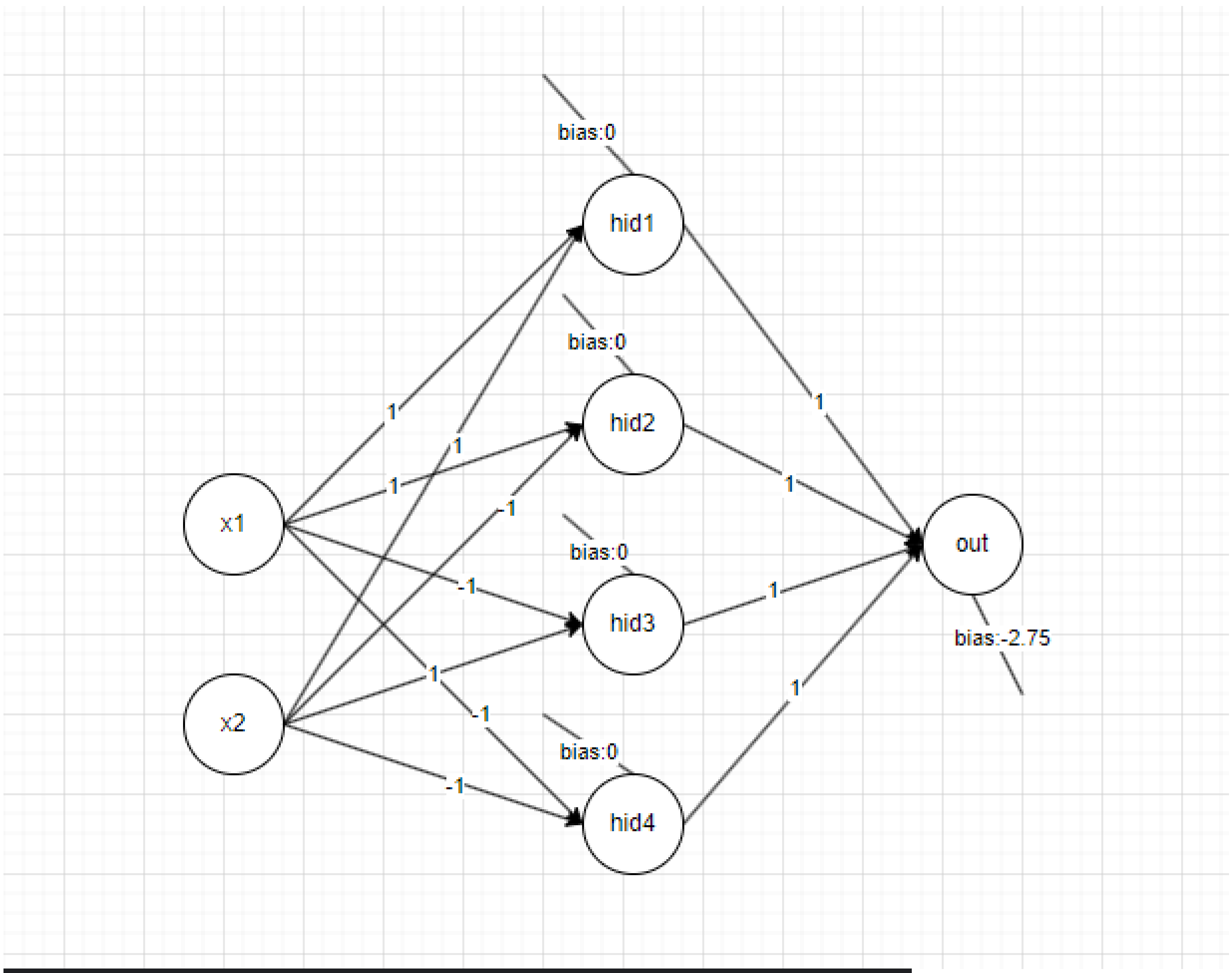
NetConv:
Even with high accuracy, some characters may still be confused due to very slight variations or noise in the dataset.
Characters like "ma" (6) and "ha" (5) might still have occasional misclassifications.
The convolutional layers significantly reduce errors by capturing spatial hierarchies, but perfect classification is challenging due to inherent data noise or ambiguities.

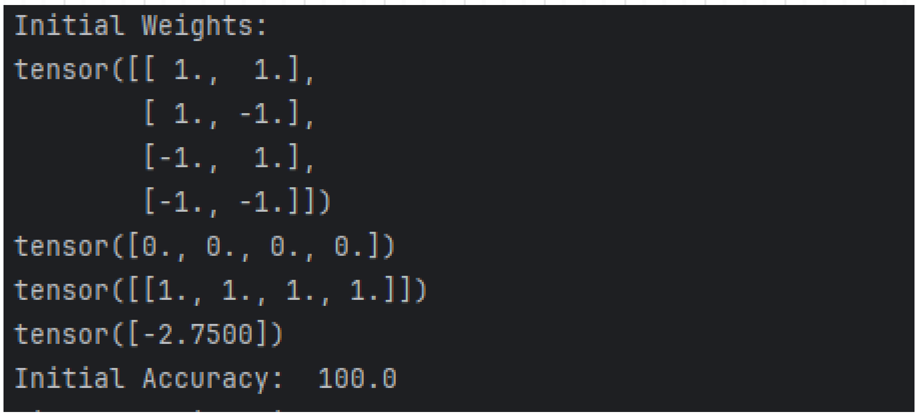
Part 2: Multi-Layer Perceptron

1. Answer question 1



2. Answer question 2





```
Initial Weights:
tensor([[ 1.,  1.],
        [ 1., -1.],
        [-1.,  1.],
        [-1., -1.]])
tensor([0., 0., 0., 0.])
tensor([[1., 1., 1., 1.]])
tensor([-2.7500])
Initial Accuracy:  100.0
```

for each hidden node iii can be represented as:

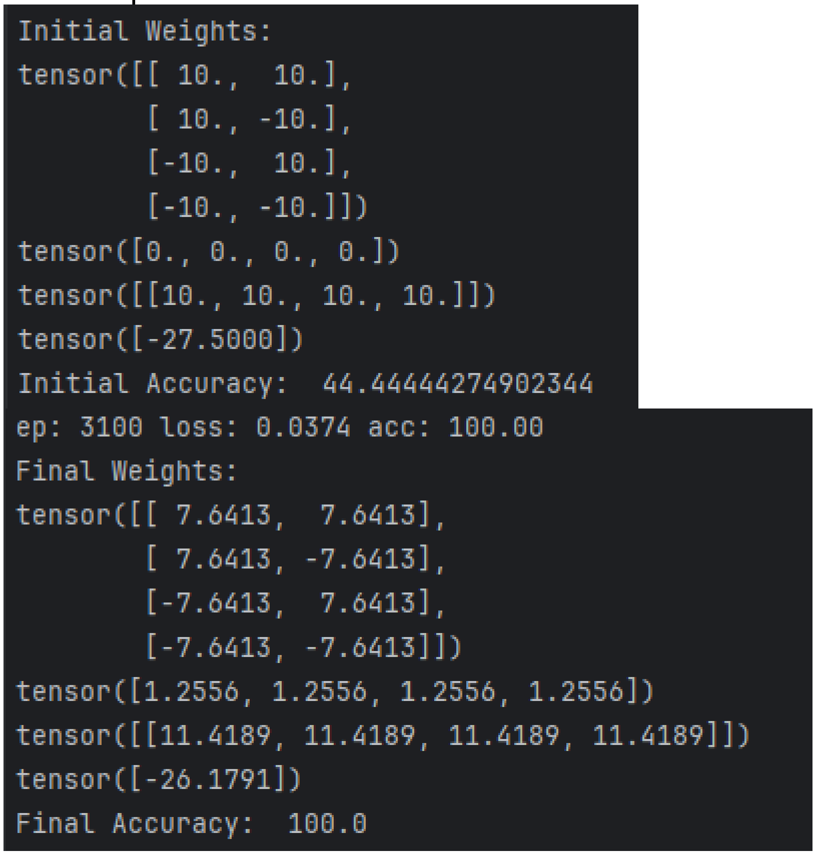
$$W_{in\_hid}[i] \cdot \text{inputs} + b_{hid}[i] = 0$$

This equation defines the line where the output of the corresponding hidden node transitions from 0 to 1.

NO	X1	X2	hid1	hid2	hid3	hid4	Y out	
1	1	1	1	0	0	0	0	
2	1	-1	1	1	0	1	1	

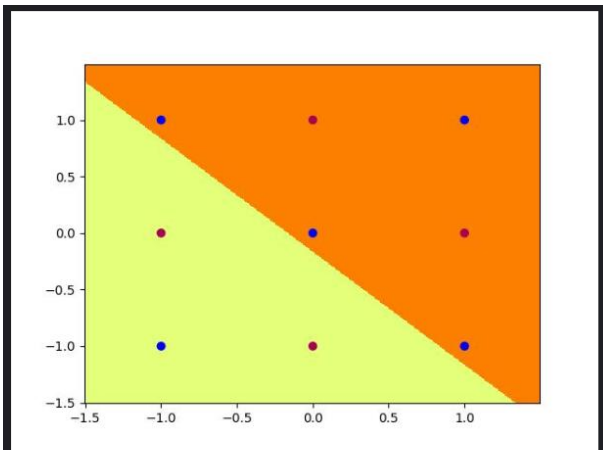
3	-1	1	0	1	1	0	1	
4	-1	-1	0	0	0	0	0	
5	1	0	1	0	0	0	0	
6	0	1	0	1	0	0	0	
7	-1	0	0	0	1	0	1	
8	0	-1	0	0	0	1	1	
9	0	0	0	0	0	0	0	

3. Answer question 3

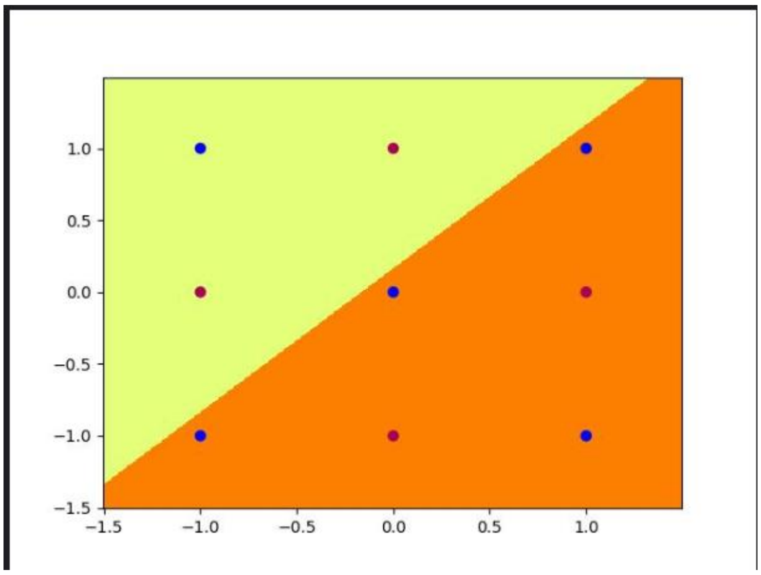


```
Initial Weights:
tensor([[ 10.,  10.],
        [ 10., -10.],
        [-10.,  10.],
        [-10., -10.]])
tensor([0., 0., 0., 0.])
tensor([[10., 10., 10., 10.]])
tensor([-27.5000])
Initial Accuracy:  44.44444274902344
ep: 3100 loss: 0.0374 acc: 100.00
Final Weights:
tensor([[ 7.6413,  7.6413],
        [ 7.6413, -7.6413],
        [-7.6413,  7.6413],
        [-7.6413, -7.6413]])
tensor([1.2556, 1.2556, 1.2556, 1.2556])
tensor([[11.4189, 11.4189, 11.4189, 11.4189]])
tensor([-26.1791])
Final Accuracy:  100.0
```

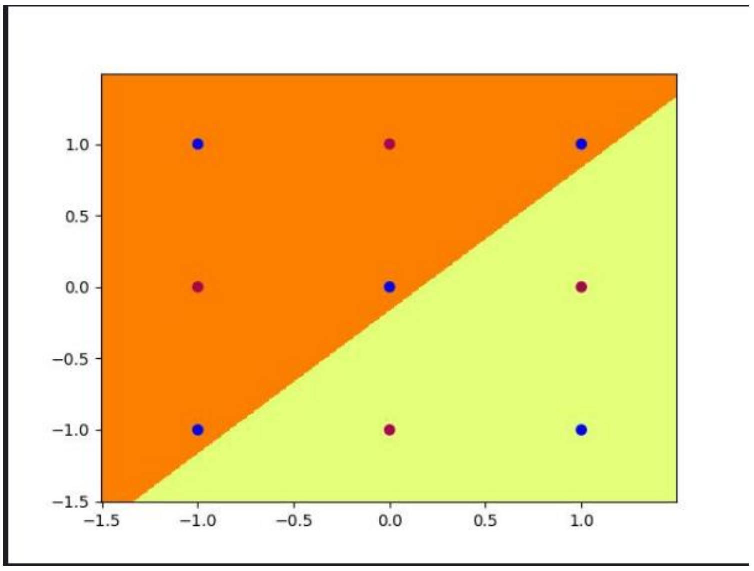
hid4-0:



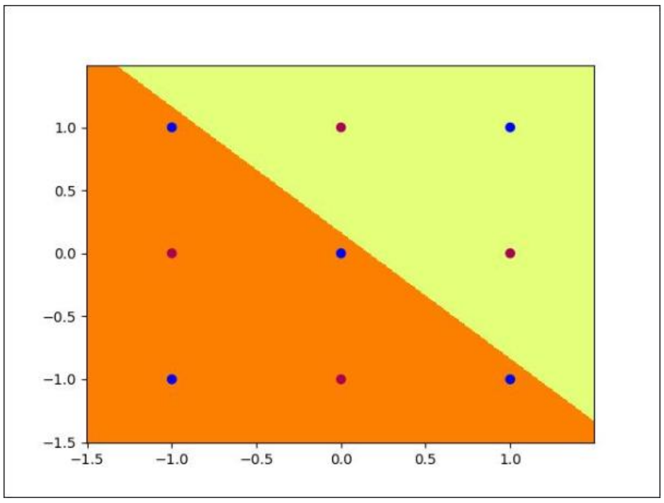
hid4-1:



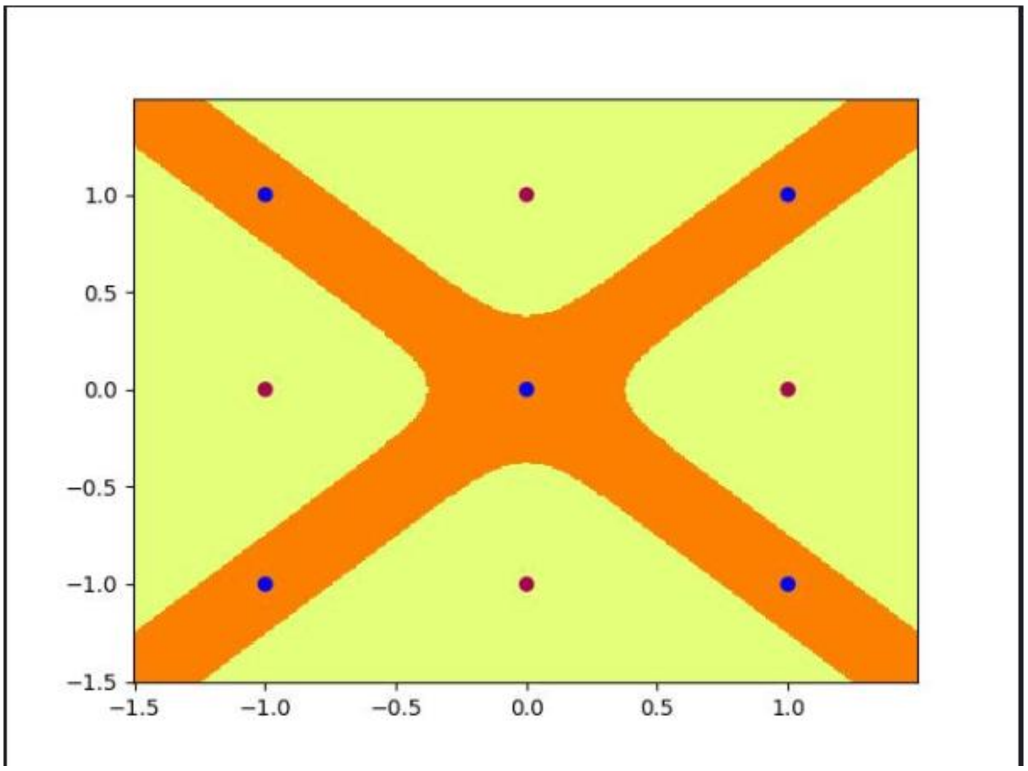
hid4-2:



hid4-3:

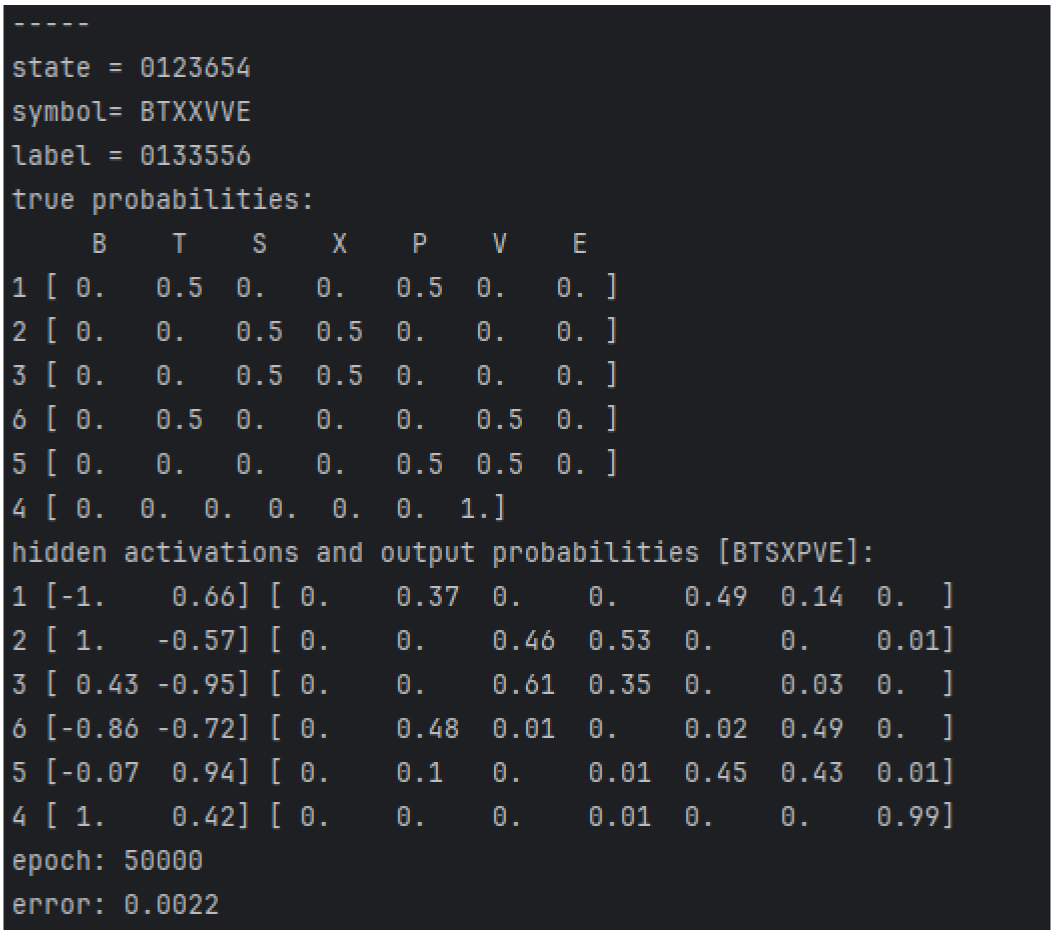


out_4:

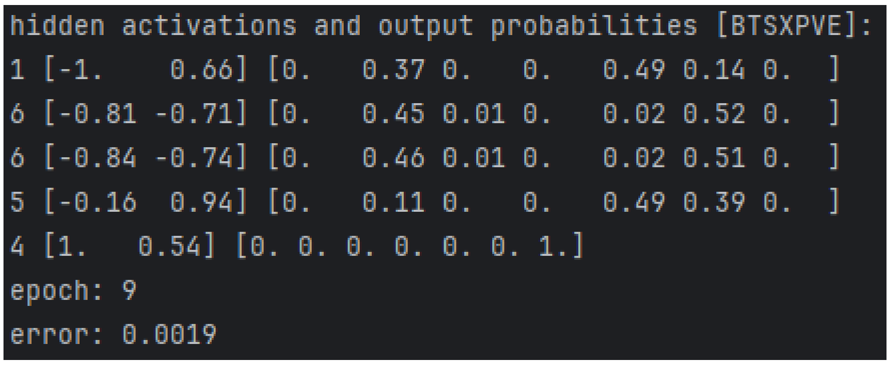


Part 3: Hidden Unit Dynamics for Recurrent Networks

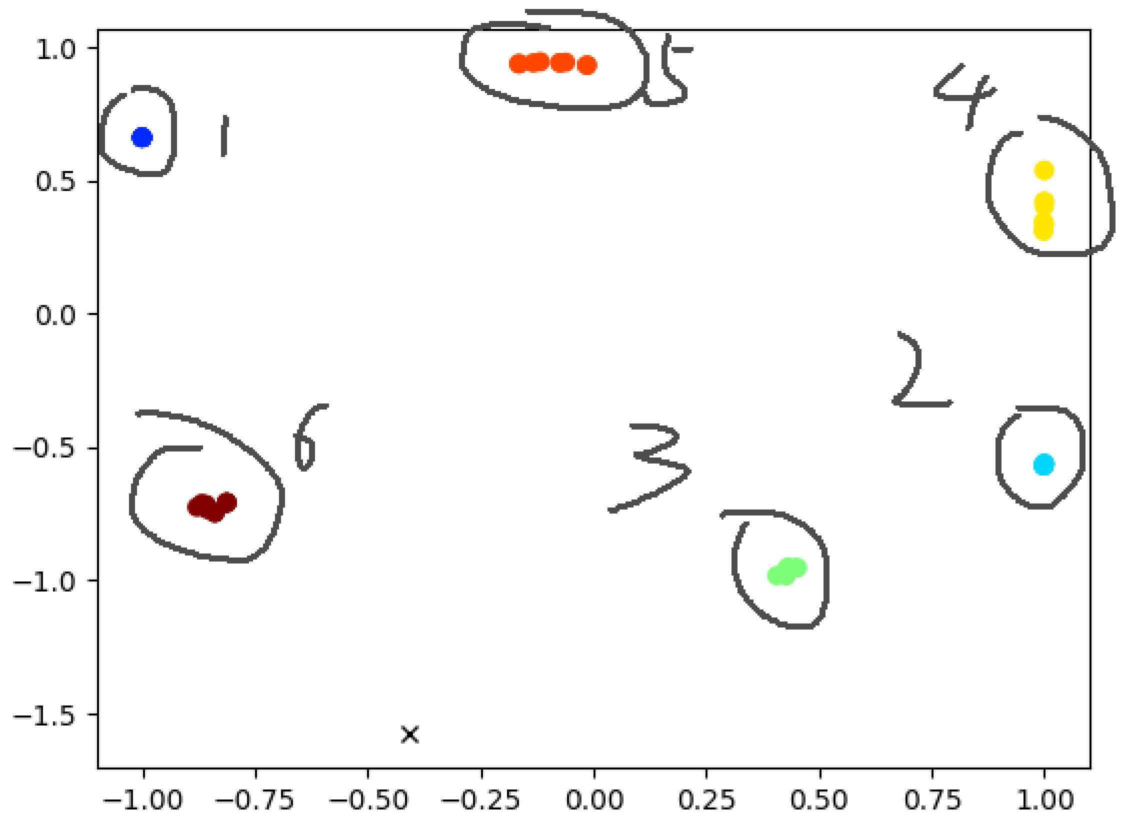
1. Answer question 1



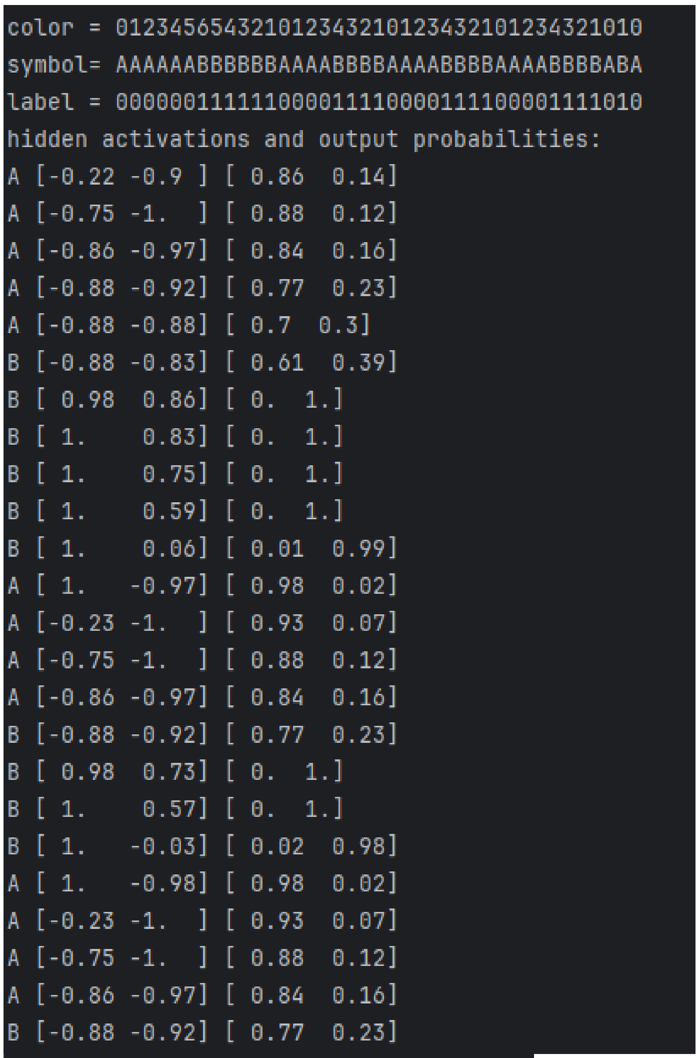
```
-----
state = 0123654
symbol= BTXXVVE
label = 0133556
true probabilities:
     B    T    S    X    P    V    E
1 [ 0.   0.5  0.   0.   0.5  0.   0. ]
2 [ 0.   0.   0.5  0.5  0.   0.   0. ]
3 [ 0.   0.   0.5  0.5  0.   0.   0. ]
6 [ 0.   0.5  0.   0.   0.   0.5  0. ]
5 [ 0.   0.   0.   0.   0.5  0.5  0. ]
4 [ 0.  0.  0.  0.  0.  0.  1.]
hidden activations and output probabilities [BTSXPVE]:
1 [-1.    0.66] [ 0.    0.37  0.    0.    0.49  0.14  0.  ]
2 [ 1.   -0.57] [ 0.    0.    0.46  0.53  0.    0.    0.01]
3 [ 0.43 -0.95] [ 0.    0.    0.61  0.35  0.    0.03  0.  ]
6 [-0.86 -0.72] [ 0.    0.48  0.01  0.    0.02  0.49  0.  ]
5 [-0.07  0.94] [ 0.    0.1   0.    0.01  0.45  0.43  0.01]
4 [ 1.    0.42] [ 0.    0.    0.    0.01  0.    0.    0.99]
epoch: 50000
error: 0.0022
```



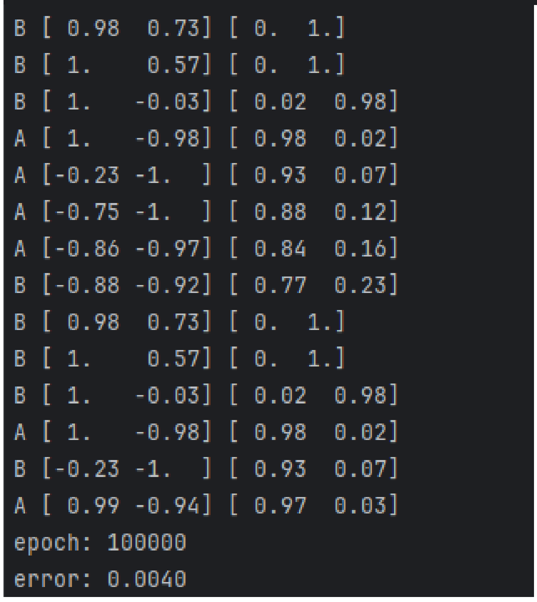
```
hidden activations and output probabilities [BTSXPVE]:
1 [-1.    0.66] [0.   0.37 0.   0.   0.49 0.14 0.  ]
6 [-0.81 -0.71] [0.   0.45 0.01 0.   0.02 0.52 0.  ]
6 [-0.84 -0.74] [0.   0.46 0.01 0.   0.02 0.51 0.  ]
5 [-0.16  0.94] [0.   0.11 0.   0.   0.49 0.39 0.  ]
4 [1.   0.54] [0. 0. 0. 0. 0. 0. 1.]
epoch: 9
error: 0.0019
```



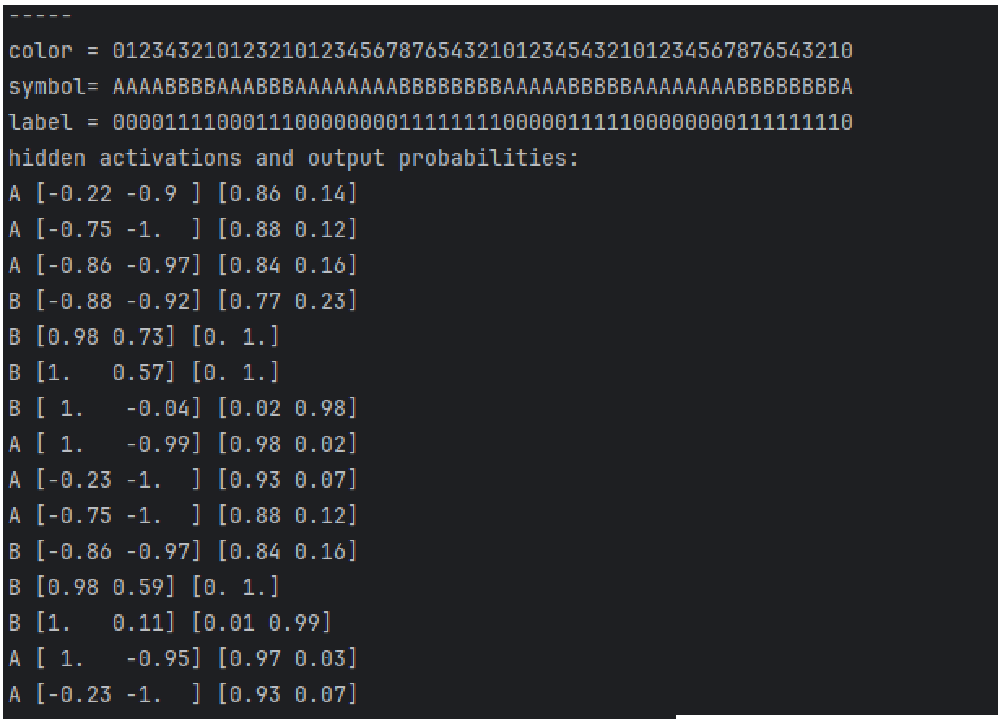
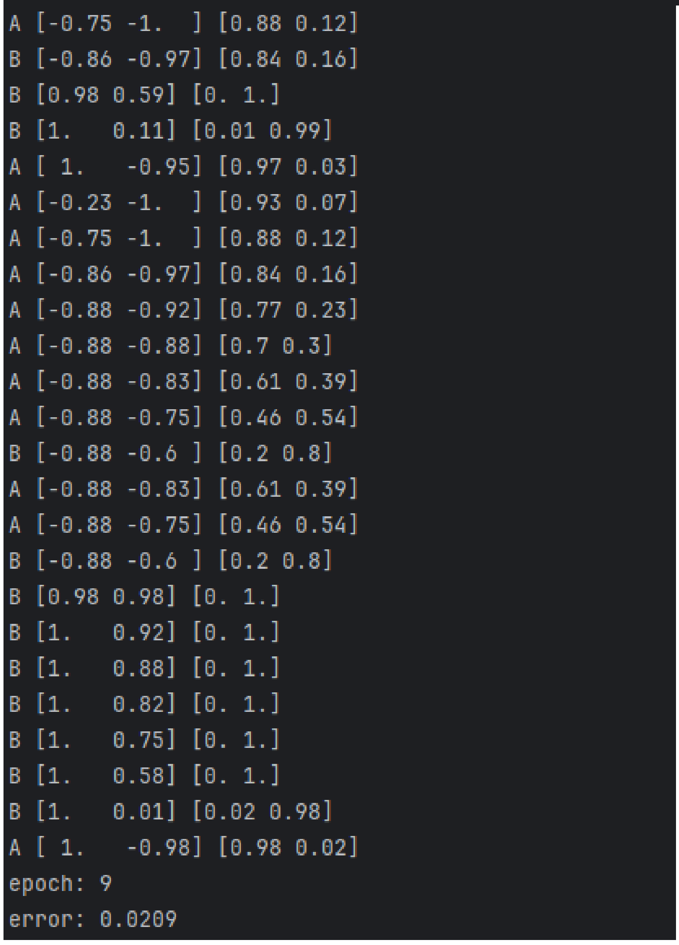
2. Answer question 2



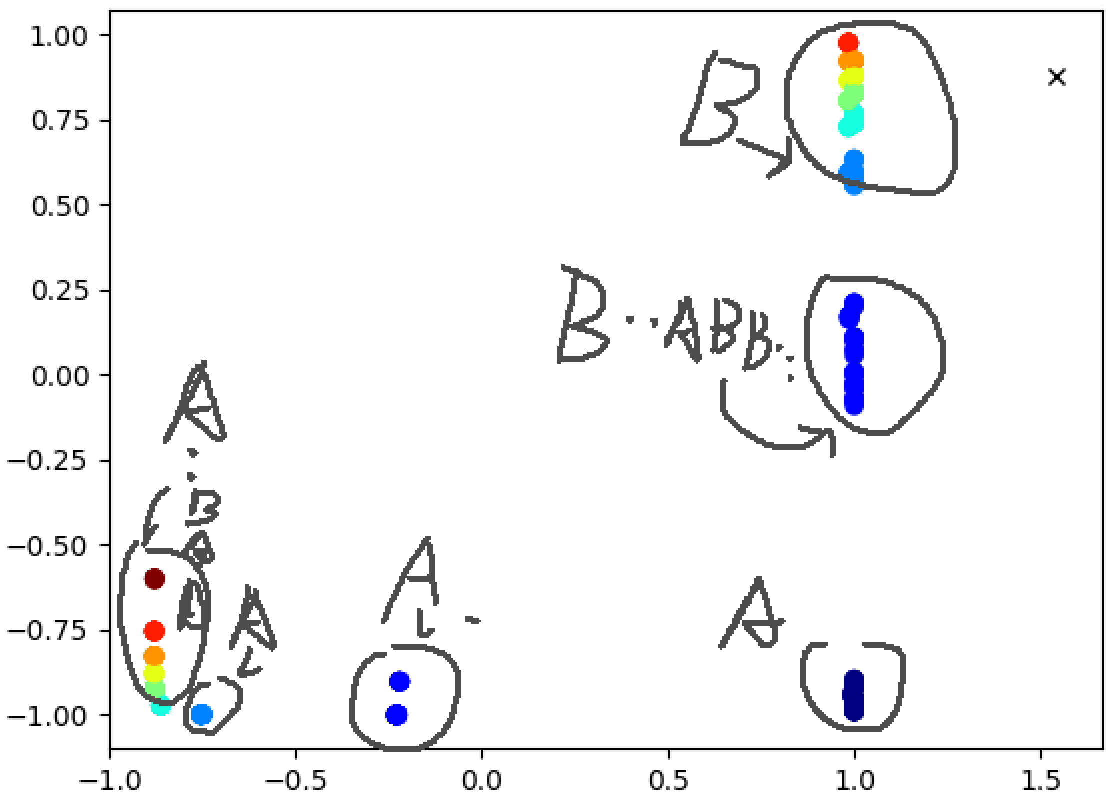
```
color = 012345654321012343210123432101234321010
symbol= AAAAAABBBBBBAAAABBBBAAAABBBBAAAABBBBABA
label = 000000111111000011110000111100001111010
hidden activations and output probabilities:
A [-0.22 -0.9 ] [ 0.86  0.14]
A [-0.75 -1.  ] [ 0.88  0.12]
A [-0.86 -0.97] [ 0.84  0.16]
A [-0.88 -0.92] [ 0.77  0.23]
A [-0.88 -0.88] [ 0.7  0.3]
B [-0.88 -0.83] [ 0.61  0.39]
B [ 0.98  0.86] [ 0.  1.]
B [ 1.    0.83] [ 0.  1.]
B [ 1.    0.75] [ 0.  1.]
B [ 1.    0.59] [ 0.  1.]
B [ 1.    0.06] [ 0.01  0.99]
A [ 1.   -0.97] [ 0.98  0.02]
A [-0.23 -1.  ] [ 0.93  0.07]
A [-0.75 -1.  ] [ 0.88  0.12]
A [-0.86 -0.97] [ 0.84  0.16]
B [-0.88 -0.92] [ 0.77  0.23]
B [ 0.98  0.73] [ 0.  1.]
B [ 1.    0.57] [ 0.  1.]
B [ 1.   -0.03] [ 0.02  0.98]
A [ 1.   -0.98] [ 0.98  0.02]
A [-0.23 -1.  ] [ 0.93  0.07]
A [-0.75 -1.  ] [ 0.88  0.12]
A [-0.86 -0.97] [ 0.84  0.16]
B [-0.88 -0.92] [ 0.77  0.23]
```



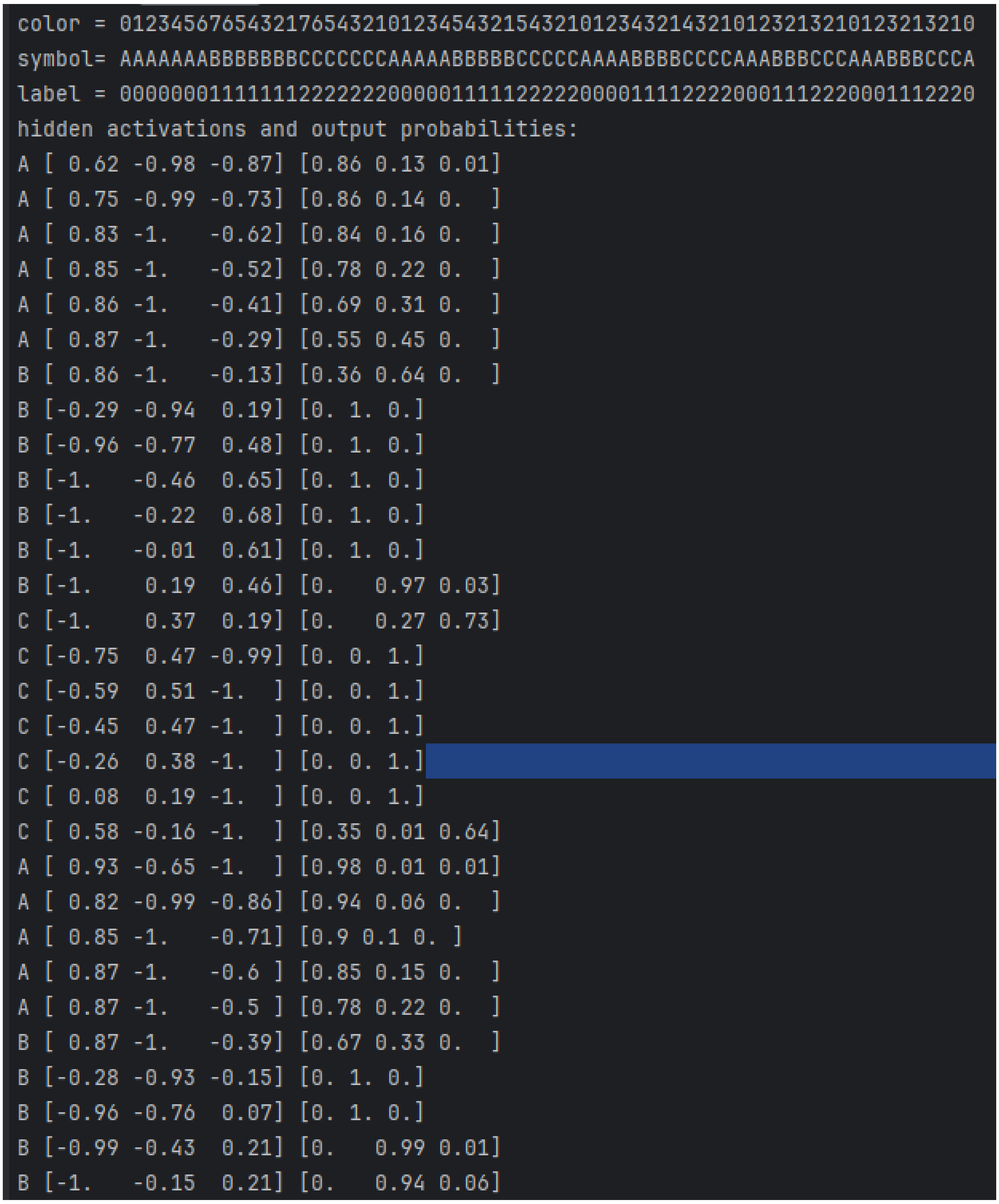
```
B [ 0.98  0.73] [ 0.  1.]
B [ 1.    0.57] [ 0.  1.]
B [ 1.   -0.03] [ 0.02  0.98]
A [ 1.   -0.98] [ 0.98  0.02]
A [-0.23 -1.  ] [ 0.93  0.07]
A [-0.75 -1.  ] [ 0.88  0.12]
A [-0.86 -0.97] [ 0.84  0.16]
B [-0.88 -0.92] [ 0.77  0.23]
B [ 0.98  0.73] [ 0.  1.]
B [ 1.    0.57] [ 0.  1.]
B [ 1.   -0.03] [ 0.02  0.98]
A [ 1.   -0.98] [ 0.98  0.02]
B [-0.23 -1.  ] [ 0.93  0.07]
A [ 0.99 -0.94] [ 0.97  0.03]
epoch: 100000
error: 0.0040
```

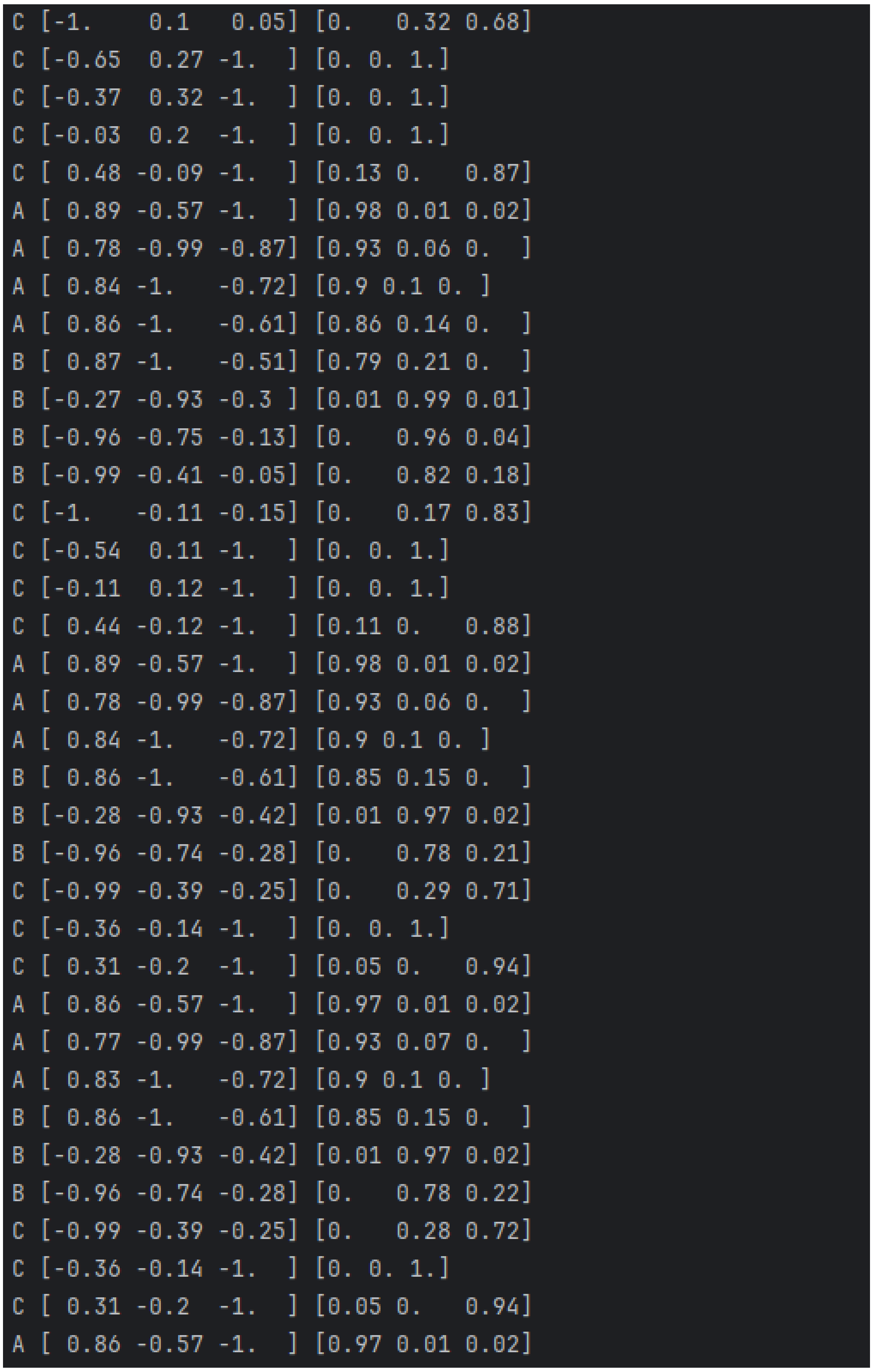
```
-----
color = 012343210123210123456787654321012345432101234567876543210
symbol= AAAABBBBAAABBBAAAAAAAAABBBBBBBBAAAAABBBBBAAAAAAAAABBBBBBBBA
label = 00001111000111000000000111111110000011111000000000111111110
hidden activations and output probabilities:
A [-0.22 -0.9 ] [0.86 0.14]
A [-0.75 -1.  ] [0.88 0.12]
A [-0.86 -0.97] [0.84 0.16]
B [-0.88 -0.92] [0.77 0.23]
B [0.98 0.73] [0. 1.]
B [1.   0.57] [0. 1.]
B [ 1.   -0.04] [0.02 0.98]
A [ 1.   -0.99] [0.98 0.02]
A [-0.23 -1.  ] [0.93 0.07]
A [-0.75 -1.  ] [0.88 0.12]
B [-0.86 -0.97] [0.84 0.16]
B [0.98 0.59] [0. 1.]
B [1.   0.11] [0.01 0.99]
A [ 1.   -0.95] [0.97 0.03]
A [-0.23 -1.  ] [0.93 0.07]
A [-0.75 -1.  ] [0.88 0.12]
B [-0.86 -0.97] [0.84 0.16]
B [0.98 0.59] [0. 1.]
B [1.   0.11] [0.01 0.99]
A [ 1.   -0.95] [0.97 0.03]
A [-0.23 -1.  ] [0.93 0.07]
A [-0.75 -1.  ] [0.88 0.12]
A [-0.86 -0.97] [0.84 0.16]
A [-0.88 -0.92] [0.77 0.23]
A [-0.88 -0.88] [0.7 0.3]
A [-0.88 -0.83] [0.61 0.39]
A [-0.88 -0.75] [0.46 0.54]
B [-0.88 -0.6 ] [0.2 0.8]
A [-0.88 -0.83] [0.61 0.39]
A [-0.88 -0.75] [0.46 0.54]
B [-0.88 -0.6 ] [0.2 0.8]
B [0.98 0.98] [0. 1.]
B [1.   0.92] [0. 1.]
B [1.   0.88] [0. 1.]
B [1.   0.82] [0. 1.]
B [1.   0.75] [0. 1.]
B [1.   0.58] [0. 1.]
B [1.   0.01] [0.02 0.98]
A [ 1.   -0.98] [0.98 0.02]
epoch: 9
error: 0.0209
```



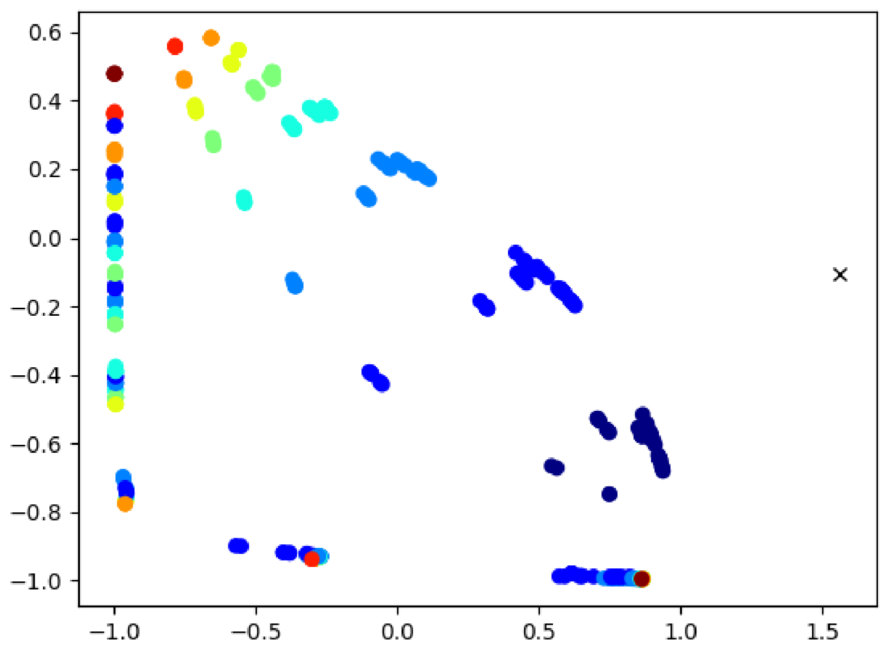
3. Answer question 3

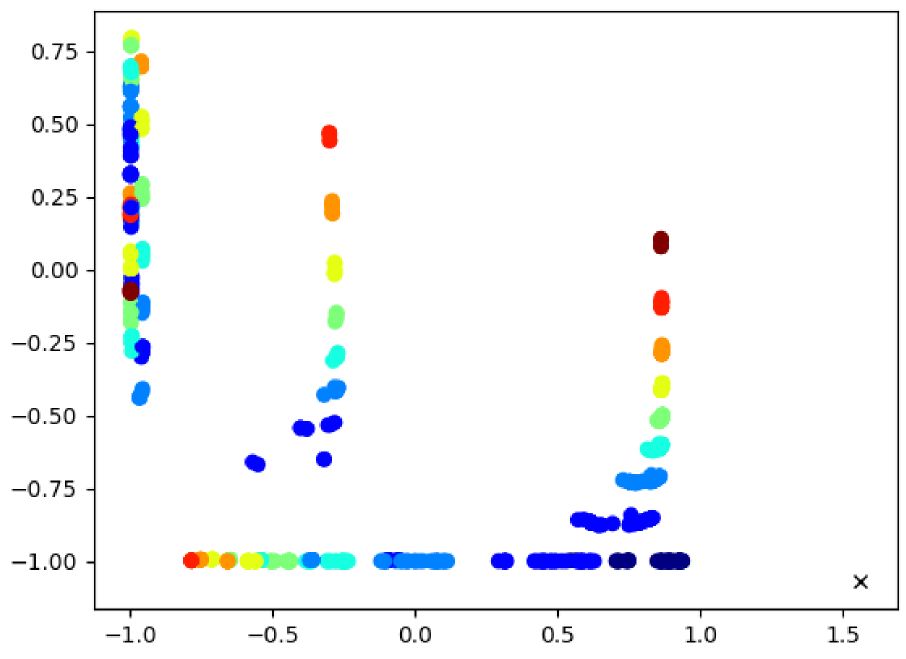


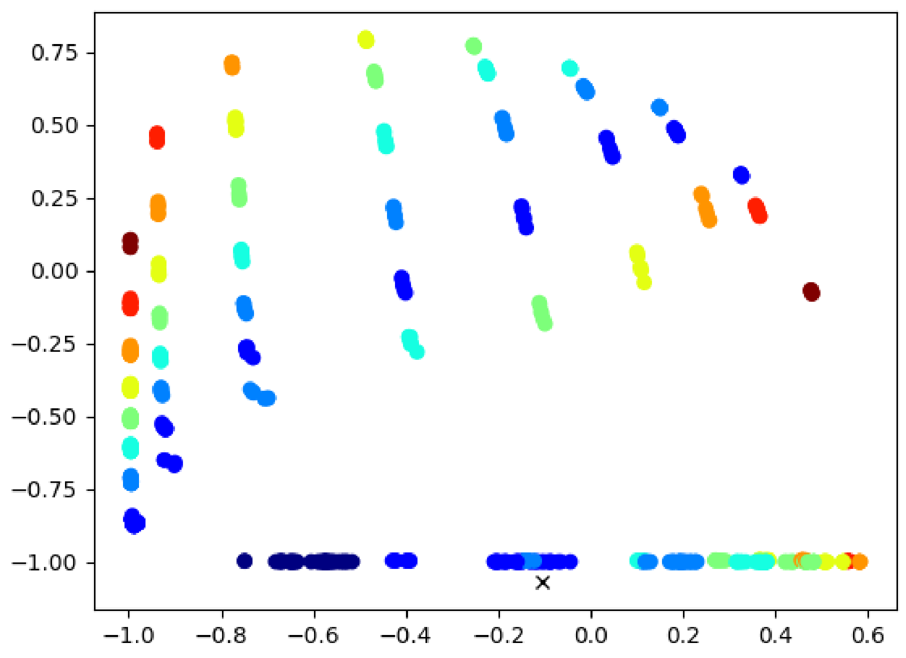
```
color = 0123456765432176543210123454321543210123432143210123213210123213210
symbol= AAAAAAABBBBBBBCCCCCCCAAAAABBBBBCCCCCAAAABBBBCCCCAAABBBCCCAAABBBCCCA
label = 0000000111111122222220000011111222220000111122220001112220001112220
hidden activations and output probabilities:
A [ 0.62 -0.98 -0.87] [0.86 0.13 0.01]
A [ 0.75 -0.99 -0.73] [0.86 0.14 0.  ]
A [ 0.83 -1.   -0.62] [0.84 0.16 0.  ]
A [ 0.85 -1.   -0.52] [0.78 0.22 0.  ]
A [ 0.86 -1.   -0.41] [0.69 0.31 0.  ]
A [ 0.87 -1.   -0.29] [0.55 0.45 0.  ]
B [ 0.86 -1.   -0.13] [0.36 0.64 0.  ]
B [-0.29 -0.94  0.19] [0.  1.  0.]
B [-0.96 -0.77  0.48] [0.  1.  0.]
B [-1.   -0.46  0.65] [0.  1.  0.]
B [-1.   -0.22  0.68] [0.  1.  0.]
B [-1.   -0.01  0.61] [0.  1.  0.]
B [-1.    0.19  0.46] [0.   0.97 0.03]
C [-1.    0.37  0.19] [0.   0.27 0.73]
C [-0.75  0.47 -0.99] [0.  0.  1.]
C [-0.59  0.51 -1.  ] [0.  0.  1.]
C [-0.45  0.47 -1.  ] [0.  0.  1.]
C [-0.26  0.38 -1.  ] [0.  0.  1.]
C [ 0.08  0.19 -1.  ] [0.  0.  1.]
C [ 0.58 -0.16 -1.  ] [0.35 0.01 0.64]
A [ 0.93 -0.65 -1.  ] [0.98 0.01 0.01]
A [ 0.82 -0.99 -0.86] [0.94 0.06 0.  ]
A [ 0.85 -1.   -0.71] [0.9 0.1 0. ]
A [ 0.87 -1.   -0.6 ] [0.85 0.15 0.  ]
A [ 0.87 -1.   -0.5 ] [0.78 0.22 0.  ]
B [ 0.87 -1.   -0.39] [0.67 0.33 0.  ]
B [-0.28 -0.93 -0.15] [0.  1.  0.]
B [-0.96 -0.76  0.07] [0.  1.  0.]
B [-0.99 -0.43  0.21] [0.   0.99 0.01]
B [-1.   -0.15  0.21] [0.   0.94 0.06]
```

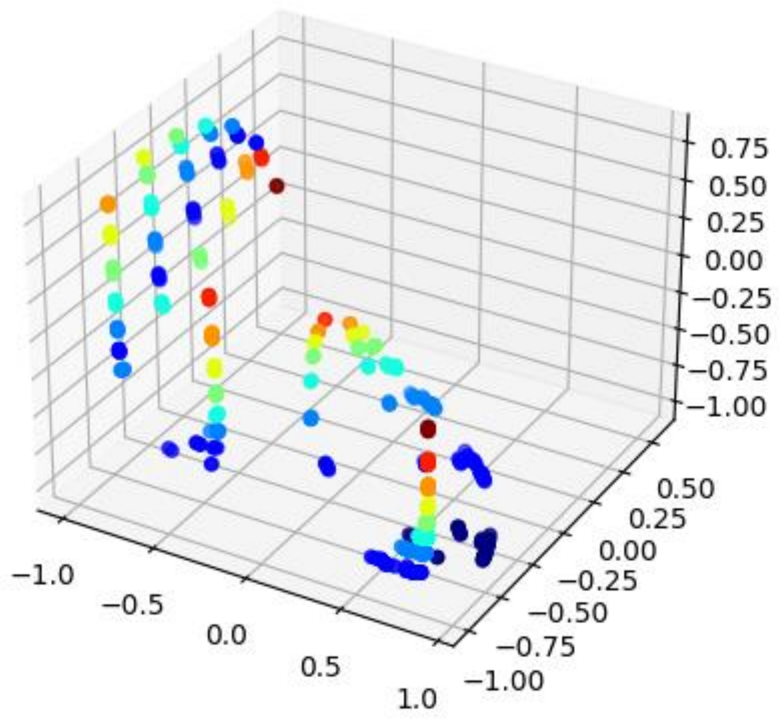


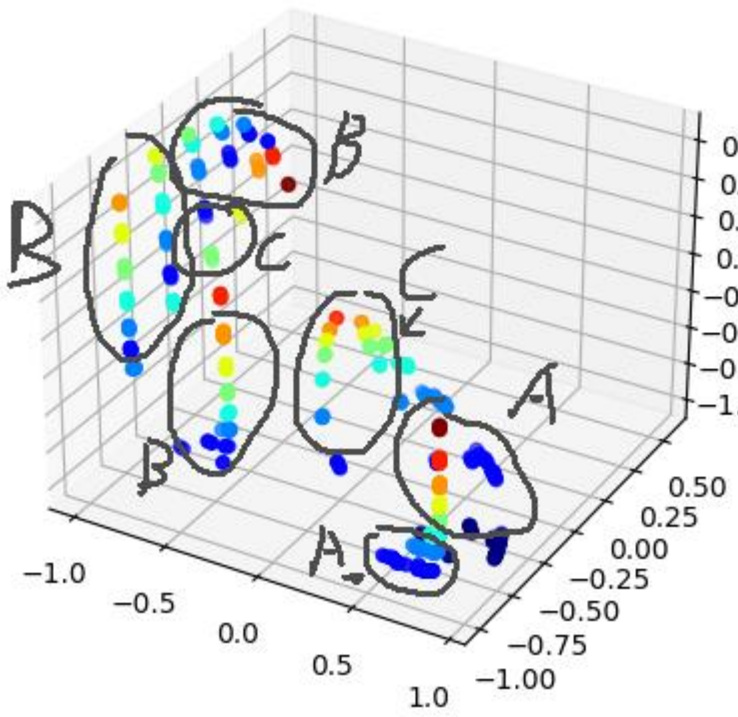
```
C [-1.    0.1   0.05] [0.   0.32 0.68]
C [-0.65  0.27 -1.  ] [0. 0. 1.]
C [-0.37  0.32 -1.  ] [0. 0. 1.]
C [-0.03  0.2  -1.  ] [0. 0. 1.]
C [ 0.48 -0.09 -1.  ] [0.13 0.   0.87]
A [ 0.89 -0.57 -1.  ] [0.98 0.01 0.02]
A [ 0.78 -0.99 -0.87] [0.93 0.06 0.  ]
A [ 0.84 -1.   -0.72] [0.9 0.1 0. ]
A [ 0.86 -1.   -0.61] [0.86 0.14 0.  ]
B [ 0.87 -1.   -0.51] [0.79 0.21 0.  ]
B [-0.27 -0.93 -0.3 ] [0.01 0.99 0.01]
B [-0.96 -0.75 -0.13] [0.   0.96 0.04]
B [-0.99 -0.41 -0.05] [0.   0.82 0.18]
C [-1.   -0.11 -0.15] [0.   0.17 0.83]
C [-0.54  0.11 -1.  ] [0. 0. 1.]
C [-0.11  0.12 -1.  ] [0. 0. 1.]
C [ 0.44 -0.12 -1.  ] [0.11 0.   0.88]
A [ 0.89 -0.57 -1.  ] [0.98 0.01 0.02]
A [ 0.78 -0.99 -0.87] [0.93 0.06 0.  ]
A [ 0.84 -1.   -0.72] [0.9 0.1 0. ]
B [ 0.86 -1.   -0.61] [0.85 0.15 0.  ]
B [-0.28 -0.93 -0.42] [0.01 0.97 0.02]
B [-0.96 -0.74 -0.28] [0.   0.78 0.21]
C [-0.99 -0.39 -0.25] [0.   0.29 0.71]
C [-0.36 -0.14 -1.  ] [0. 0. 1.]
C [ 0.31 -0.2  -1.  ] [0.05 0.   0.94]
A [ 0.86 -0.57 -1.  ] [0.97 0.01 0.02]
A [ 0.77 -0.99 -0.87] [0.93 0.07 0.  ]
A [ 0.83 -1.   -0.72] [0.9 0.1 0. ]
B [ 0.86 -1.   -0.61] [0.85 0.15 0.  ]
B [-0.28 -0.93 -0.42] [0.01 0.97 0.02]
B [-0.96 -0.74 -0.28] [0.   0.78 0.22]
C [-0.99 -0.39 -0.25] [0.   0.28 0.72]
C [-0.36 -0.14 -1.  ] [0. 0. 1.]
C [ 0.31 -0.2  -1.  ] [0.05 0.   0.94]
A [ 0.86 -0.57 -1.  ] [0.97 0.01 0.02]
```


0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0
1.5

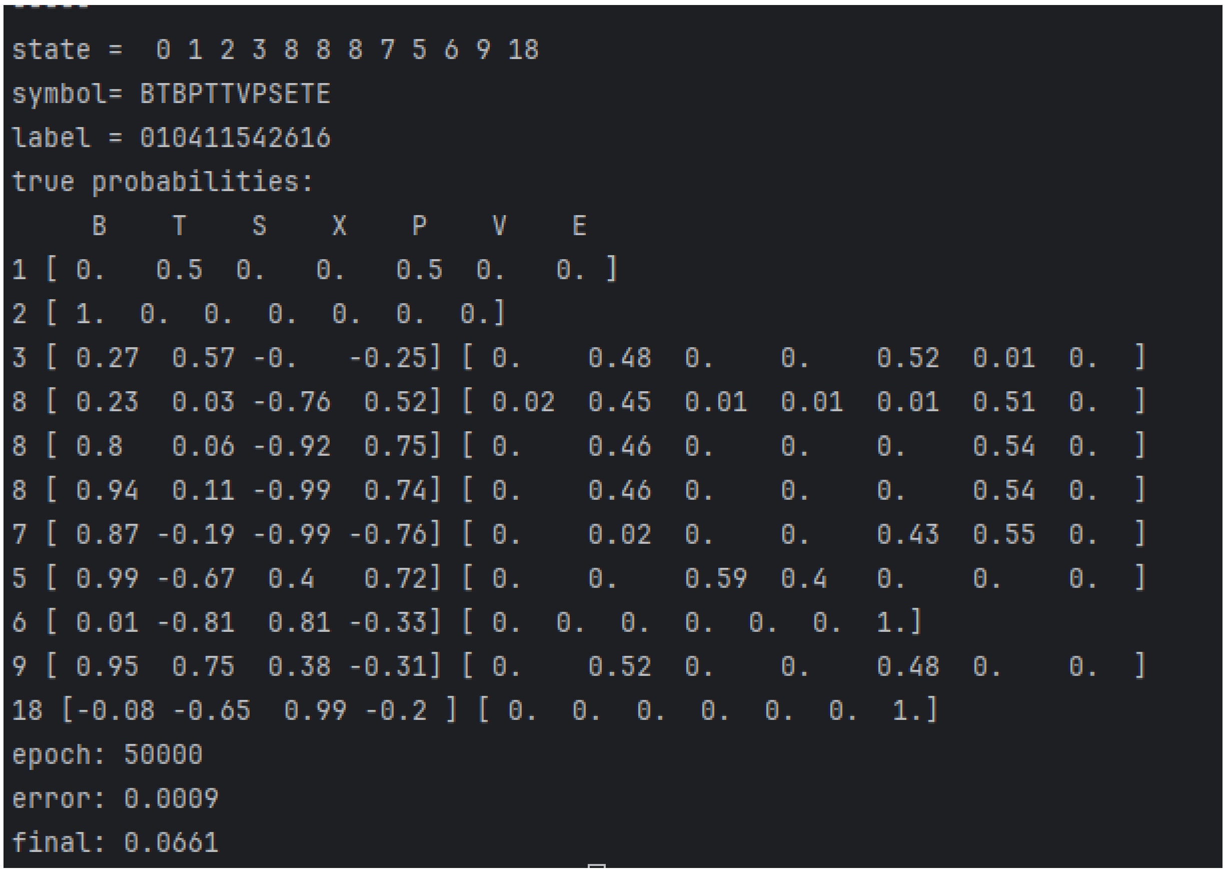

0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.0
−0.5
0.0
0.5
1.0
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00



In summary, the SRN achieves the anbncn prediction task by learning sequential dependencies between "a", "b", and "c" sequences. It updates its hidden state as it processes each character in the sequence, and its trained parameters enable it to predict the last "B" and subsequent "C"s accurately based on these learned dependencies.

4. Answer question 4



```
state =  0 1 2 3 8 8 8 7 5 6 9 18
symbol= BTBPTTVPSETE
label = 010411542616
true probabilities:
     B    T    S    X    P    V    E
1 [ 0.   0.5  0.   0.   0.5  0.   0. ]
2 [ 1.  0.  0.  0.  0.  0.  0.]
3 [ 0.27  0.57 -0.   -0.25] [ 0.    0.48  0.    0.    0.52  0.01  0.  ]
8 [ 0.23  0.03 -0.76  0.52] [ 0.02  0.45  0.01  0.01  0.01  0.51  0.  ]
8 [ 0.8   0.06 -0.92  0.75] [ 0.    0.46  0.    0.    0.    0.54  0.  ]
8 [ 0.94  0.11 -0.99  0.74] [ 0.    0.46  0.    0.    0.    0.54  0.  ]
7 [ 0.87 -0.19 -0.99 -0.76] [ 0.    0.02  0.    0.    0.43  0.55  0.  ]
5 [ 0.99 -0.67  0.4   0.72] [ 0.    0.    0.59  0.4   0.    0.    0.  ]
6 [ 0.01 -0.81  0.81 -0.33] [ 0.  0.  0.  0.  0.  0.  1.]
9 [ 0.95  0.75  0.38 -0.31] [ 0.    0.52  0.    0.    0.48  0.    0.  ]
18 [-0.08 -0.65  0.99 -0.2 ] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 50000
error: 0.0009
final: 0.0661
```

The LSTM's hidden state (h_t) evolves over time, capturing sequential patterns and dependencies in the input data.The cell state (c_t) evolves based on the information retained (f_t, i_t) and updated (g_t) at each time step.

The output at each time step (output) reflects the LSTM's prediction based on the current hidden state (h_t), projected through the output weights (V) and biases (out_bias).During training, the LSTM adjusts its parameters (W, U, V, biases) through backpropagation to minimize prediction errors across sequences, optimizing its ability to capture complex dependencies.

In summary, the LSTM achieves its task of sequence prediction by integrating information over time (c_t, h_t), selectively gating inputs (i_t, f_t), and producing sequential outputs (output). This architecture enables it to effectively model and predict sequences in various applications such as natural language processing, time series prediction, and more.