COMP9444 Neural Networks and Deep Learning

Assignment 1

Term 1, 2025

Submitted by

zID: z5527498

Name: Zihan Chen

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.

Zihan Chen, 5527498, 16/03/2025

Part 1: Japanese Character Recognition

1. Confusion matrix:

```
[[765.   6.   8.  14.  30.  64.   2.  62.  32.  17.]
 [  6. 669. 107.  19.  30.  24.  55.  14.  25.  51.]
 [  7.  60. 694.  26.  27.  20.  47.  39.  43.  37.]
 [  5.  34.  56. 763.  15.  56.  14.  18.  28.  11.]
 [ 62.  55.  79.  20. 618.  19.  32.  38.  21.  56.]
 [  8.  29. 124.  18.  19. 725.  28.   8.  33.   8.]
 [  5.  21. 144.  10.  24.  25. 728.  21.  10.  12.]
 [ 16.  28.  26.  12.  84.  17.  57. 623.  89.  48.]
 [ 10.  37.  92.  43.   5.  30.  46.   8. 707.  22.]
 [  9.  53.  82.   4.  51.  31.  19.  32.  39. 680.]]
```

   Final Accuracy: 6972/10000 (70%)

2. Confusion matrix:

```
[[858.   4.   2.   5.  26.  30.   3.  40.  29.   3.]
 [  4. 816.  35.   4.  19.   8.  60.   4.  20.  30.]
 [  7.  13. 838.  35.  13.  21.  25.  10.  22.  16.]
 [  3.   9.  29. 919.   3.  15.   5.   4.   4.   9.]
 [ 39.  30.  19.   7. 804.   9.  30.  17.  24.  21.]
 [ 11.  13.  77.  12.   8. 824.  26.   1.  18.  10.]
 [  3.  17.  43.  12.  10.   5. 892.   7.   2.   9.]
 [ 24.  12.  15.   7.  15.   6.  34. 828.  22.  37.]
 [ 12.  28.  30.  44.   2.   8.  27.   3. 838.   8.]
 [  3.  17.  46.   4.  31.   5.  26.  14.  15. 839.]]
```

   Final Accuracy: 8456/10000 (85%)
   Total number of independent parameters: 159010

3. Confusion matrix:

```
[[949.   2.   4.   0.  14.   4.   1.  22.   1.   3.]
 [  0. 937.   2.   0.  11.   2.  33.   3.   2.  10.]
 [  9.   7. 857.  41.  12.  15.  38.  10.   5.   6.]
 [  1.   1.  16. 958.   5.   5.   5.   3.   3.   3.]
 [ 23.   2.   5.   8. 919.   3.  16.   9.  11.   4.]
 [  4.  11.  23.   7.   6. 915.  21.   5.   4.   4.]
 [  3.   1.   6.   3.   3.   2. 976.   3.   2.   1.]
 [  1.   6.   0.   0.   6.   1.   9. 952.  10.  15.]
 [  3.  13.   0.   4.   5.   4.   9.   7. 952.   3.]
 [  7.   5.   2.   2.  11.   1.   0.   3.  10. 959.]]
```

   Final accuracy: 9374/10000 (94%)
   Total number of independent parameters: 137610

4. Answer:
   a. NetLin achieves only **70%** accuracy, showing that simple linear transformations cannot effectively distinguish complex characters, NetFull improves the accuracy to **86%** by using a hidden layer, and NetConv relies on convolutional feature extraction to achieve **94%** accuracy, which is significantly better than the other models.
   b. The number of parameters among the three models:
      NetLin: （28×28）×10+10=**7850**.

NetFull:

Number of Layer 1 Parameters: （28×28）×200+200=157000

Number of Layer 2 Parameters: 200×10+10=2010

The number of parameters for NetFull is 157000+2010=**159010**.

NetConv:

Number of Layer 1 Parameters: 1×64×3×3+64+64×2=768

Number of Layer 2 Parameters: 64×128×3×3+128+128×2=74112

Number of parameters for the full connection layer: 128×7×7×10+10=62730

The number of parameters for NetConv is 768+74112+62730=**137610**.

c. In NetLin, “ま”, “は”, “な”, “き”, etc. are confused as **“す”** many times, among which **“ま”** is confused the most often, probably because under the Linear model, only the coarse overall strength can be learned, which is not enough to distinguish the fine structure.

In NetFull, although many characters can be separated, there are still many confused characters, among which **“は”** and **“す”** are confused the most times, probably because some of their strokes and local shapes are similar, and they can not be completely distinguished.

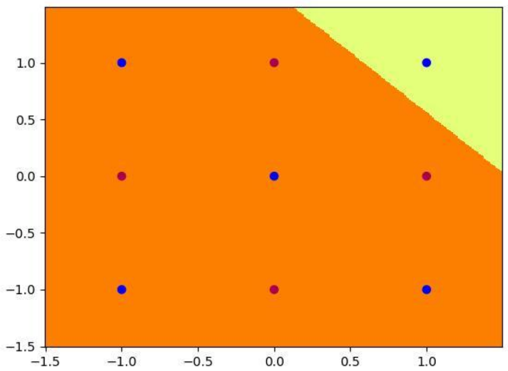
In NetConv, the classification error rate is significantly lower, but there are still a few pairs of characters that can be confused, including **“す”** and **“つ”**, which may be difficult to distinguish because of the similarity of the curves in the handwritten form and the small differences.

Part 2: Multi-Layer Perceptron

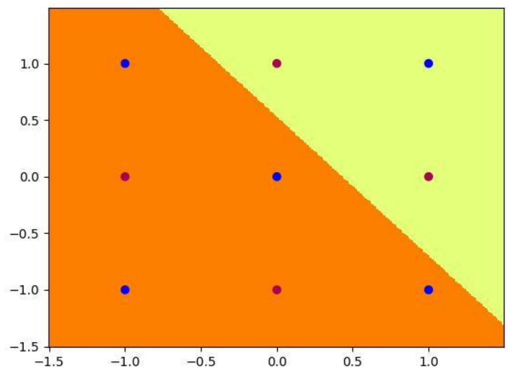
1. The final weights and biases are:

```
Final Weights:
tensor([[-3.9584, -3.7360],
        [-5.9626, -4.8528],
        [ 4.3274,  5.4683],
        [ 4.8521,  5.0717],
        [ 4.0252, -4.2475],
        [-1.6475, -2.0342]])
tensor([ 6.0587,  2.5703,  2.1522,  7.5707, -7.3568,  3.2602])
tensor([[-7.0006,  7.5703,  7.1637, -7.3403,  6.0856, -2.2648]])
tensor([5.1673])
Final Accuracy:  100.0
```

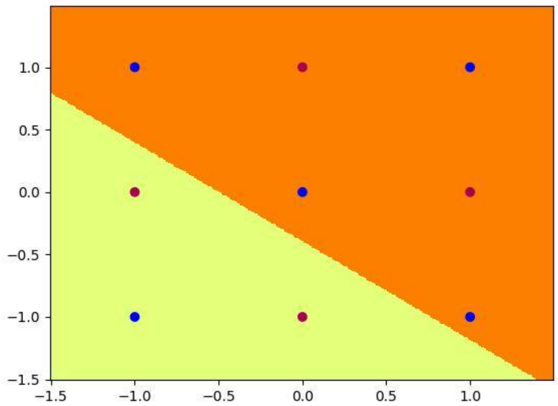
Images:



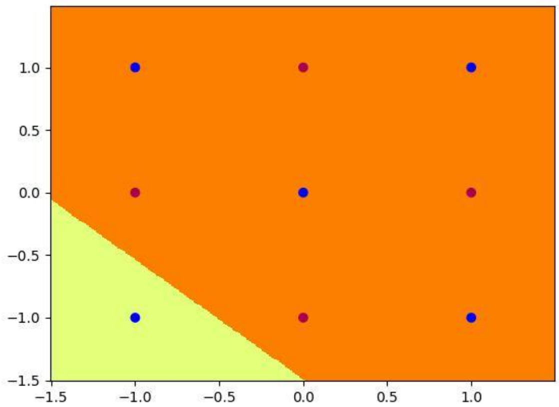
hid_6_0.jpg



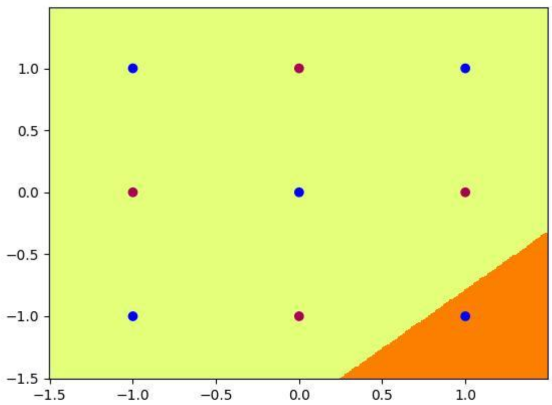
hid_6_1.jpg



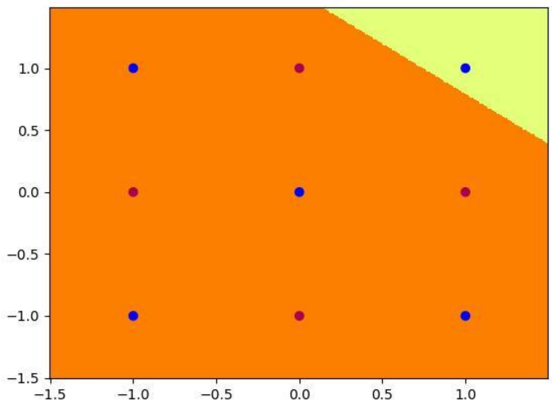
hid_6_2.jpg



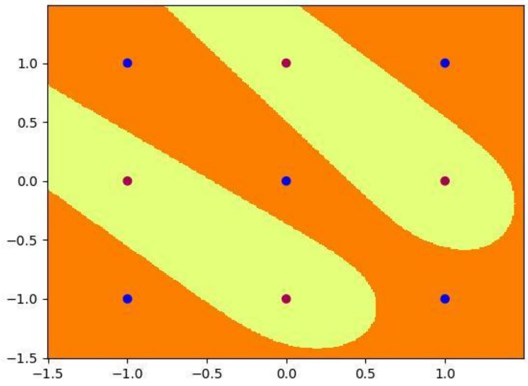
hid_6_3.jpg



hid_6_4.jpg

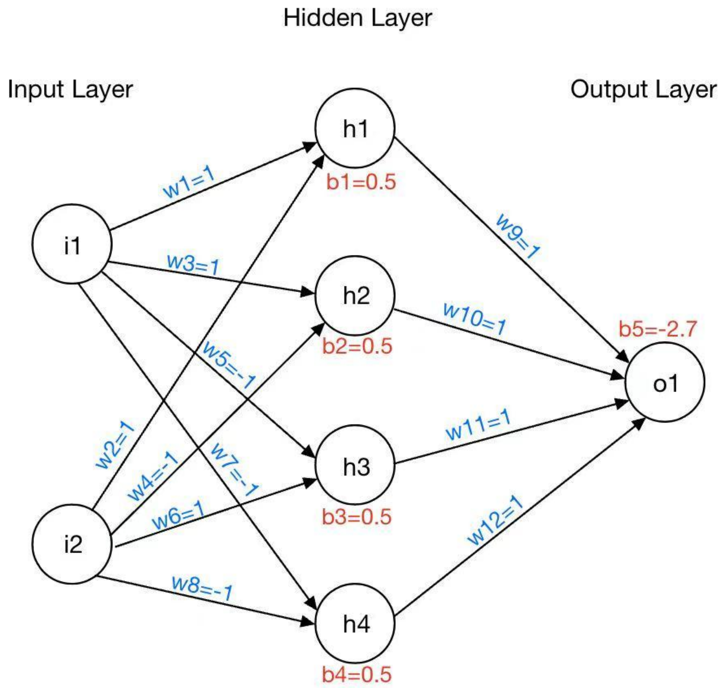


hid_6_5.jpg



out_6.jpg

2. The value of all the weights and biases:



The equations are:

x + y = -0.5

x – y = -0.5

-x + y = -0.5

-x - y = -0.5

Activation table:

(x,y)	Activation in Node1	Activation in Node2	Activation in Node3	Activation in Node4
(1.0,1.0)	1	1	1	0
(1.0,-1.0)	1	1	0	1
(-1.0,1.0)	1	0	1	1
(-1.0,-1.0)	0	1	1	1
(0.0,1.0)	1	0	1	0

(1.0,0.0)	1	1	0	0
(0.0,-1.0)	0	1	0	1
(-1.0,0.0)	0	0	1	1
(0.0,0.0)	1	1	1	1

3. Rescaled weights:

    Initial Weights:

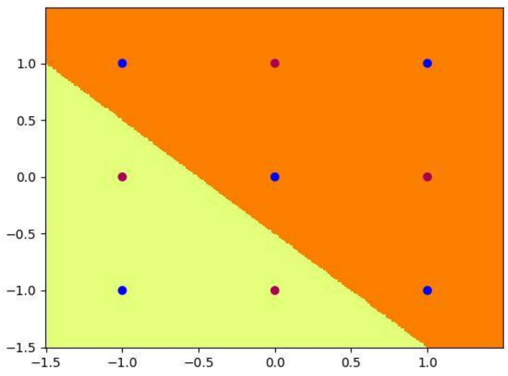
    tensor([[10., 10.],
            [10., -10.],
            [-10., 10.],
            [-10., -10.]])

    tensor([5., 5., 5., 5.])

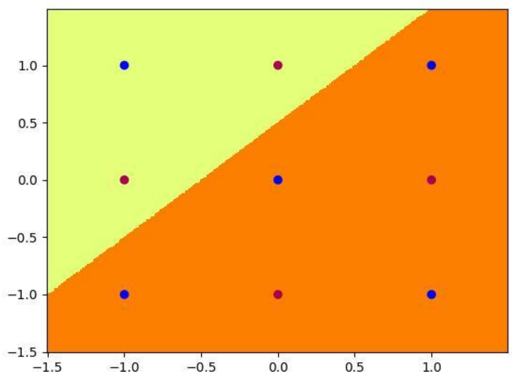
    tensor([[10., 10., 10., 10.]])

    tensor([-27.])

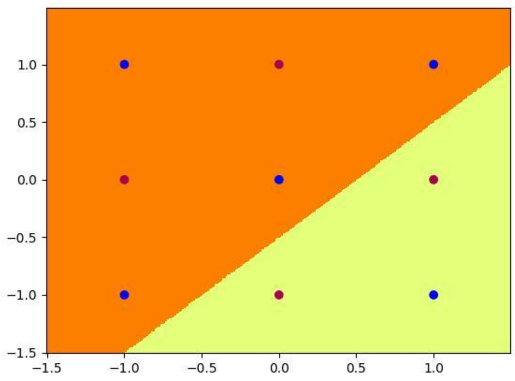
Image:



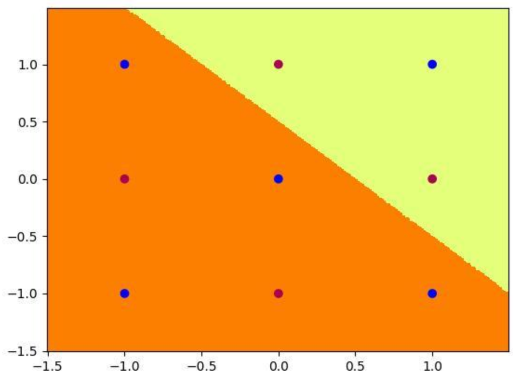
hid_4_0.jpg



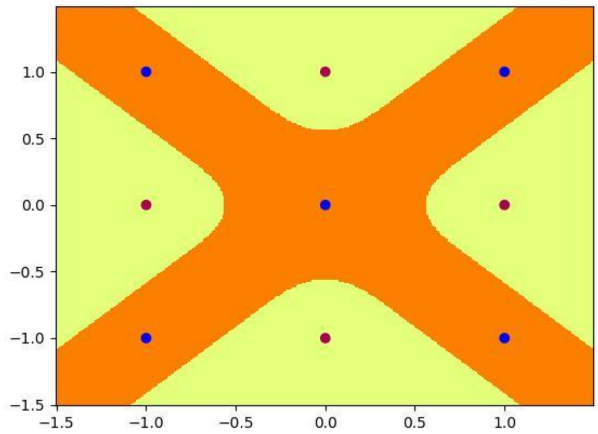
hid_4_1.jpg



hid_4_2.jpg



hid_4_3.jpg

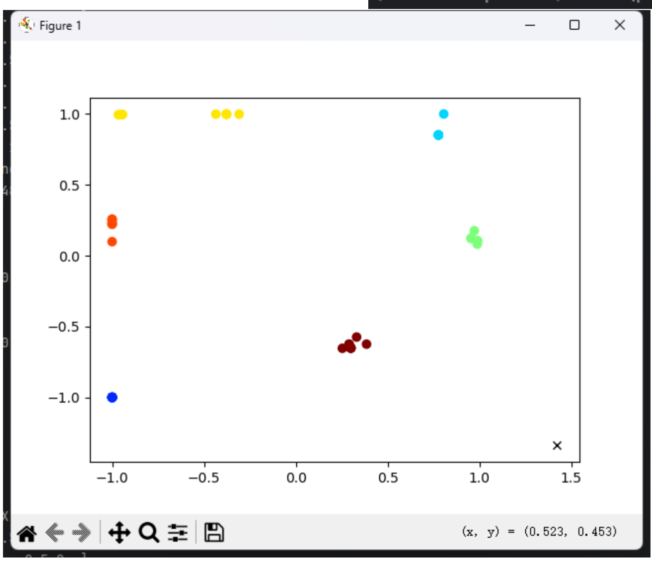


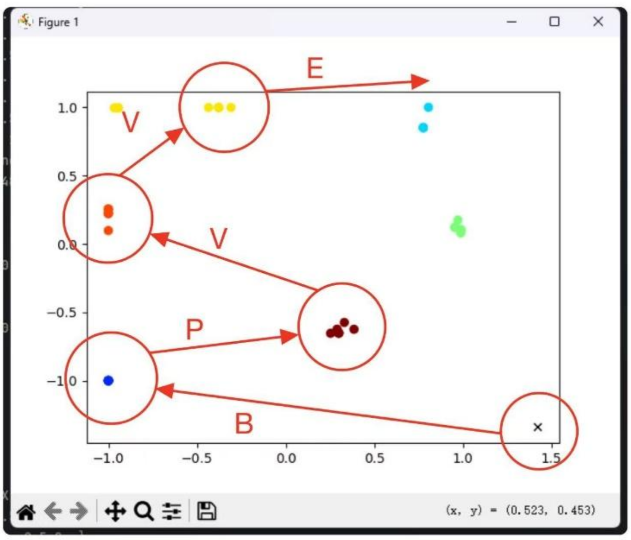
out_4.jpg

Part 3: Hidden Unit Dynamics for Recurrent Networks

1. Image:

```
-----
state = 01654
symbol= BPVVE
label = 04556
true probabilities:
     B    T    S    X    P    V    E
1 [0.  0.5 0.  0.  0.5 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. -1.] [0.   0.48 0.   0.   0.49 0.04 0.  ]
6 [ 0.3  -0.65] [0.   0.47 0.01 0.01 0.01 0.5  0.  ]
5 [-1.    0.22] [0.   0.02 0.   0.   0.42 0.55 0.  ]
4 [-0.95  1.  ] [0. 0. 0. 0. 0. 0. 1.]
epoch: 9
error: 0.0005
(sessionInspection) PS D:\project\COMP9444\hw1>
```



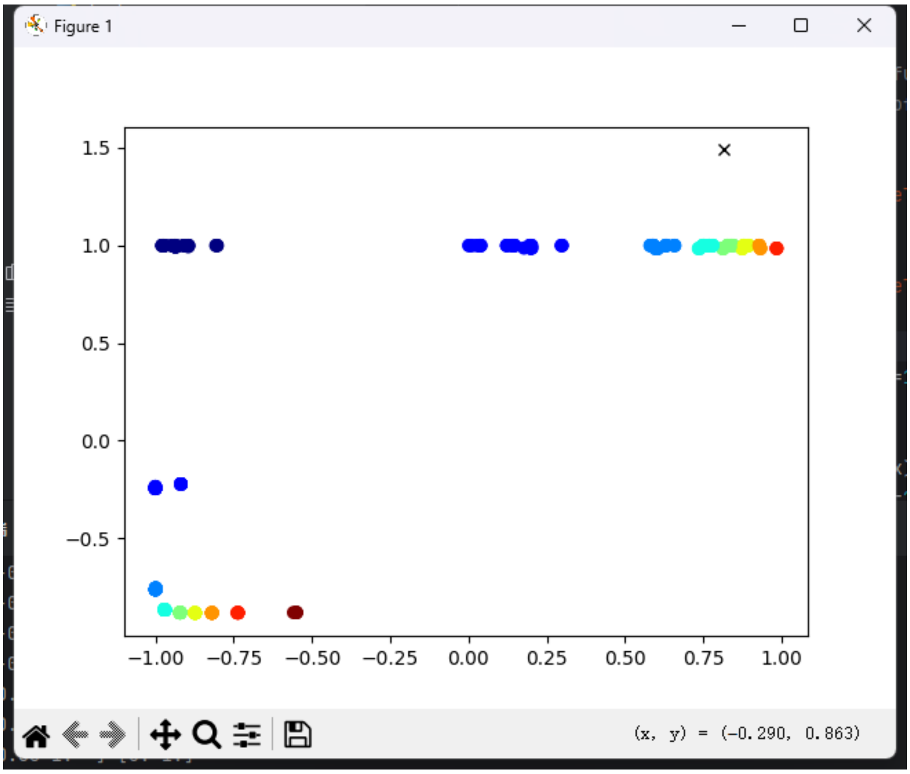


2. Explanation:

The network learns the anbn structure through hidden state changes. When reading A, the hidden layer follows a trajectory that encodes the counts of A and predicts A. Once all Bs have been read, the hidden state resets, predicting A to start a new sequence. This transition is shown in the 2D/3D visualization as a transition from A trajectory to B trajectory to B trajectory, reflecting the structure of the sequence learned by the network.

The hidden layer keeps track of the remaining Bs through a counting mechanism, and when there is only one B left, the network still predicts B with high probability to ensure the correct output. Once the count of Bs is zero, the hidden state resets and predicts A with high probability, implying a new sequence. This segmented counting mechanism achieves an accurate sequence transition and ensures the integrity of the anbn structure.

Image:

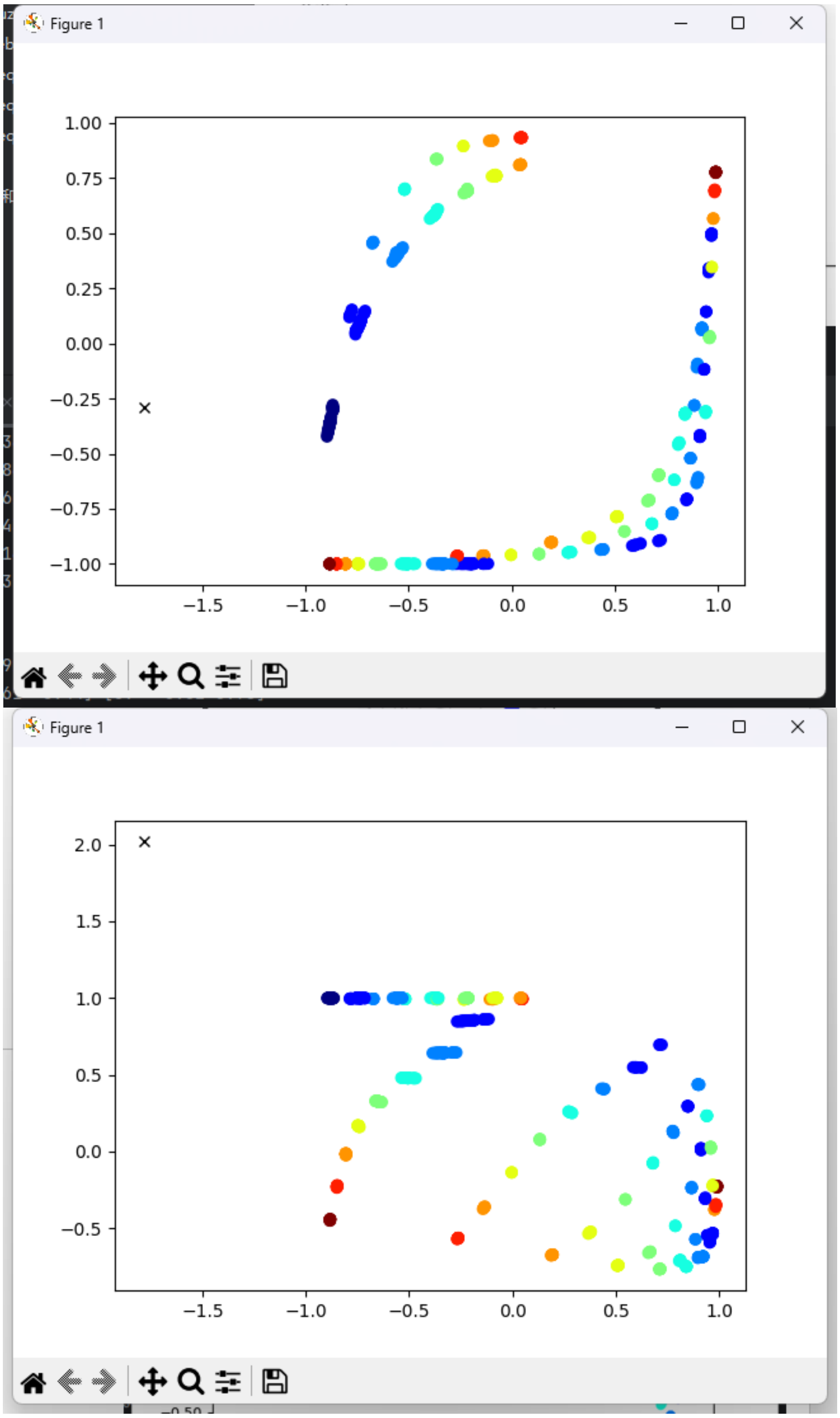


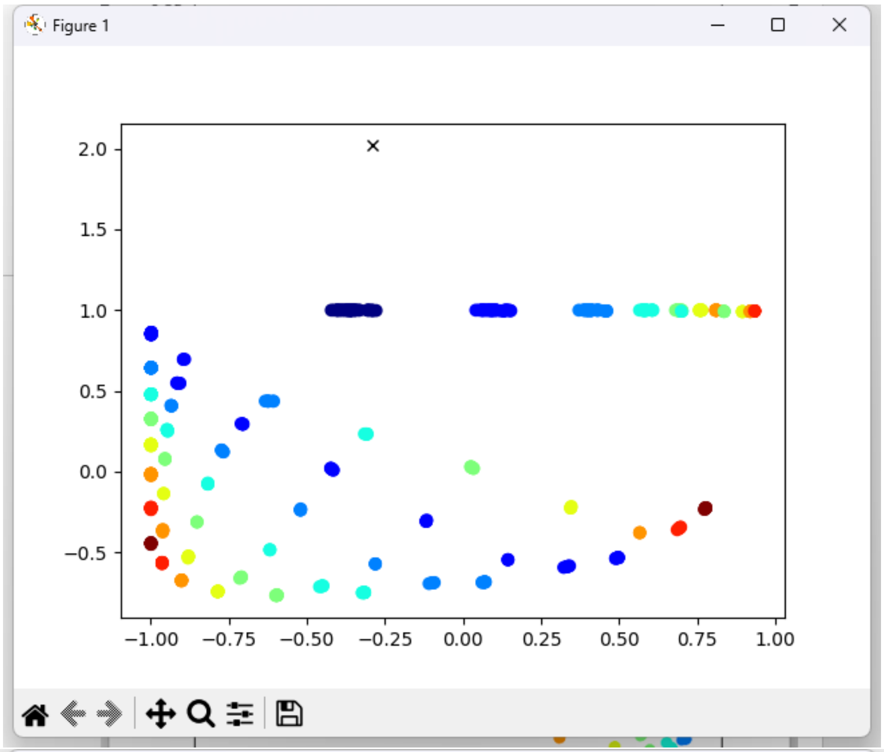
3. Explanation:

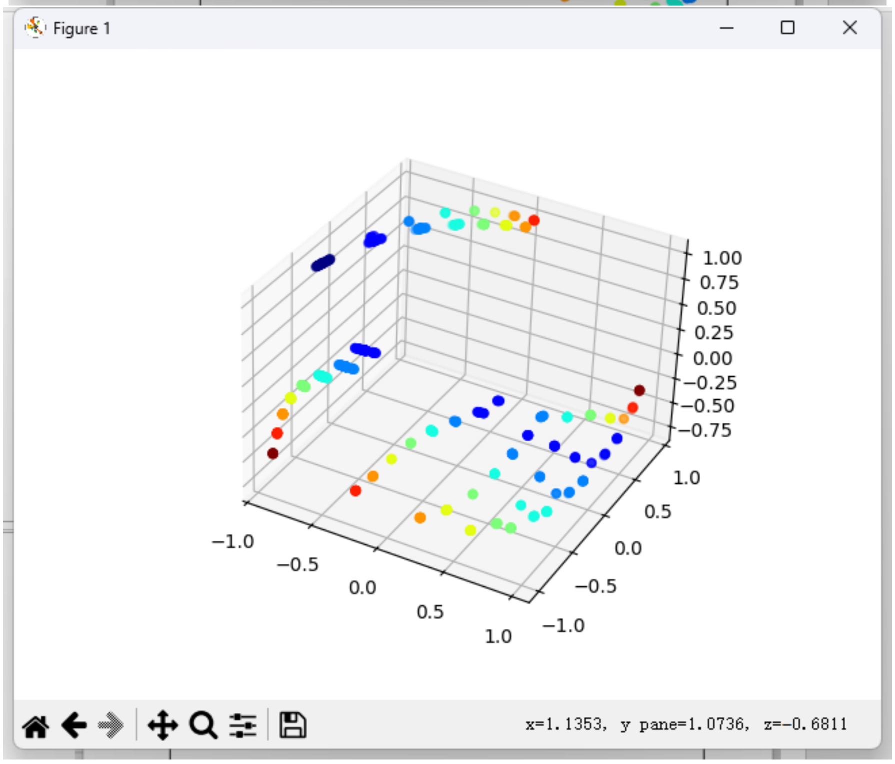
The network learns the anbncn structure through hidden state changes. As A’s increase, activations follow a trajectory, predicting A until B appears. Upon B, the hidden state shifts, tracking remaining B’s and predicting them until switching to C. The same process follows for C’s. When C’s reach zero, the hidden state resets, allowing correct prediction of the next A and the new sequence.

The network predicts the last B, all C’s and the next A by tracking character order and count. As B’s decrease, it maintains high B probability, ensuring transition to C. Similarly, during C’s, the count decreases, predicting C until reset. This segmented counting mechanism enables learning the anbncn structure.

Image:







4. Analysis: As LSTM introduces a gating mechanism on top of RNN, thus retaining or forgetting information, it enables the network to effectively track the long range dependencies of Embedded Reber Grammar. As seen by outputting the hidden layer and prediction probabilities at each moment, LSTM is able to distinguish between

different branches and accurately predict the next character, thus performing better on this task.

Image:

```
-----
state =  0 1 10 11 16 15 13 16 16 15 14 17 18
symbol= BPBPVPXTVVEPE
label = 0404543155646
true probabilities:
     B    T    S    X    P    V    E
1 [ 0.   0.5  0.   0.   0.5  0.   0. ]
10 [ 1.  0.  0.  0.  0.  0.  0.]
11 [ 0.   0.5  0.   0.   0.5  0.   0. ]
16 [ 0.   0.5  0.   0.   0.   0.5  0. ]
15 [ 0.   0.   0.   0.   0.5  0.5  0. ]
13 [ 0.   0.   0.5  0.5  0.   0.   0. ]
16 [ 0.   0.5  0.   0.   0.   0.5  0. ]
16 [ 0.   0.5  0.   0.   0.   0.5  0. ]
15 [ 0.   0.   0.   0.   0.5  0.5  0. ]
14 [ 0.  0.  0.  0.  0.  0.  1.]
17 [ 0.  0.  0.  0.  1.  0.  0.]
18 [ 0.  0.  0.  0.  0.  0.  1.]
hidden activations and output probabilities [BTSXPVE]:
1 [ 0.1  -0.74  0.75 -0.71] [ 0.    0.48  0.    0.    0.51  0.    0.  ]
10 [ 0.64  0.4  -0.6   0.69] [ 1.  0.  0.  0.  0.  0.  0.]
11 [-0.63 -0.43  0.26 -0.68] [ 0.    0.51  0.    0.    0.49  0.    0.  ]
16 [-0.92 -0.79 -0.63  0.53] [ 0.    0.49  0.    0.    0.    0.5   0.  ]
15 [-0.99  0.71  0.75  0.72] [ 0.    0.01  0.    0.    0.42  0.58  0.  ]
13 [-0.01 -0.08 -0.74 -0.04] [ 0.    0.02  0.55  0.42  0.    0.01  0.  ]
16 [-0.99 -0.74 -0.63  0.53] [ 0.    0.44  0.    0.    0.    0.55  0.  ]
16 [-0.99 -0.93 -0.31  0.72] [ 0.    0.46  0.    0.    0.    0.54  0.  ]
15 [-1.    0.72  0.75  0.72] [ 0.    0.01  0.    0.    0.42  0.57  0.  ]
14 [-0.53  0.58 -0.51 -0.57] [ 0.  0.  0.  0.  0.  0.  1.]
17 [-0.99 -0.69  0.73 -0.91] [ 0.    0.47  0.    0.    0.53  0.    0.  ]
18 [-1.    0.38 -0.69  0.02] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 50000
error: 0.0009
final: 0.0620
```