COMP9444 Neural Networks and Deep Learning

Assignment 1

Term 2, 2025

zID: z5567992

Name: Yi Ren

Part 1: Japanese Character Recognition

Question1:

```
Train Epoch: 10 [57600/60000 (96%)]          Loss: 0.670037
<class 'numpy.ndarray'>
[[764.    5.    8.   13.   31.   63.    2.   63.   31.   20.]
 [  7. 665. 105.   19.   31.   24.   57.   15.   24.   53.]
 [  8.   62. 691.   25.   26.   21.   48.   35.   46.   38.]
 [  5.   37.   57. 757.   15.   56.   14.   19.   30.   10.]
 [ 59.   51.   80.   19. 622.   22.   31.   39.   20.   57.]
 [  8.   27. 122.   17.   20. 727.   27.    8.   33.   11.]
 [  5.   24. 146.   10.   28.   23. 719.   21.   10.   14.]
 [ 16.   29.   28.   11.   87.   17.   54. 622.   89.   47.]
 [ 11.   37.   95.   42.    7.   30.   45.    7. 705.   21.]
 [  7.   53.   90.    3.   51.   29.   19.   30.   40. 678.]]

Test set: Average loss: 1.0099, Accuracy: 6950/10000 (70%)
```

Question2:

```
Train Epoch: 10 [57600/60000 (96%)]          Loss: 0.268608
<class 'numpy.ndarray'>
[[853.    2.    1.    6.   31.   30.    6.   38.   29.    4.]
 [  3. 813.   30.    3.   20.   12.   60.    6.   21.   32.]
 [  8.   13. 849.   35.   11.   18.   25.   11.   15.   15.]
 [  3.   10.   35. 912.    2.   11.    6.    5.    7.    9.]
 [ 38.   31.   17.    6. 814.    8.   30.   22.   18.   16.]
 [  9.    9.   93.    6.   11. 829.   22.    3.   10.    8.]
 [  3.    9.   49.   11.   21.    5. 891.    5.    1.    5.]
 [ 16.   17.   24.    5.   22.    8.   35. 814.   26.   33.]
 [ 10.   24.   29.   45.    6.    8.   30.    3. 836.    9.]
 [  2.   18.   48.    6.   27.    4.   28.   16.   13. 838.]]

Test set: Average loss: 0.5097, Accuracy: 8449/10000 (84%)
```

28*28 is the size of the input image, and 150 is the number of hidden nodes. So the number of weights in the first fully connected layer should be 28*28*150. The number of weights in the second layer is 150*10, where 10 is the size of the output image. So The total number of independent parameters in the network is:(28*28*150+150)+(150*10+10) = 119260

Question3

Train Epoch: 10 [57600/60000 (96%)] Loss: 0.019439

```
<class 'numpy.ndarray'>
[[939.   3.   0.   1.  33.   1.   3.  14.   1.   5.]
 [  3. 935.   2.   0.   6.   0.  38.   3.   2.  11.]
 [ 11.  10. 876.  32.   5.   4.  31.   7.   7.  17.]
 [  3.   0.  14. 960.   2.   2.   4.   4.   1.  10.]
 [ 10.  10.   3.   4. 936.   2.  15.   8.   5.   7.]
 [  6.  16.  61.   7.   3. 862.  30.   4.   4.   7.]
 [  1.  10.  13.   2.   3.   1. 964.   5.   0.   1.]
 [  6.   9.   2.   0.   8.   0.  15. 938.   3.  19.]
 [  3.  21.   9.   4.  15.   1.  12.   1. 929.   5.]
 [  4.   8.   4.   2.   8.   0.   3.   5.   2. 964.]]
```

Test set: Average loss: 0.2696, Accuracy: 9303/10000 (93%)

First convolutional layer

Input channel: 1
Output channel: 20
The size of each kernel = 5×5
bias: 20
In total: 1*20*5*5 + 20 = 520

Second convolutional layer

Input channel: 20
Output channel: 40
The size of each kernel = 5×5
bias: 40
In total: 20*40*5*5 + 40 = 20040

fc1 = nn.Linear(40*4*4, 64)
Input = 40 × 4 × 4 = 640
Output = 64
In total: 64*640 + 64 = 41024

Fc2 = nn.Linear(40*4*4, 64)
Input = 64
Output = 10
In total: 64*10 + 10 = 650

Hence,the total independent parameters of this network is
520 + 20040 + 41024 + 650 = 62234

Question4

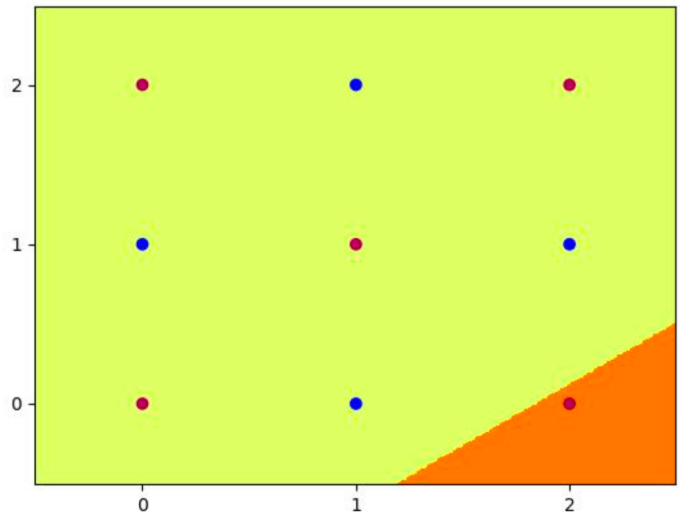
(a).NetConv performs best in terms of accuracy, outperforming the two models that only use fully connected layers, because CNN can better extract spatial local features in images, thereby improving classification performance. NetLin 70%, NetFull 84%, NetConv 93%. NetLin is the model with the worst generalization ability among the three, while NetConv not only has the highest accuracy, but also has the least overfitting.

(b). NetLin: 784 × 10 + 10 = 7850
NetFull: 119260 (see question 2)
NetConv: 62234 (see question 3)

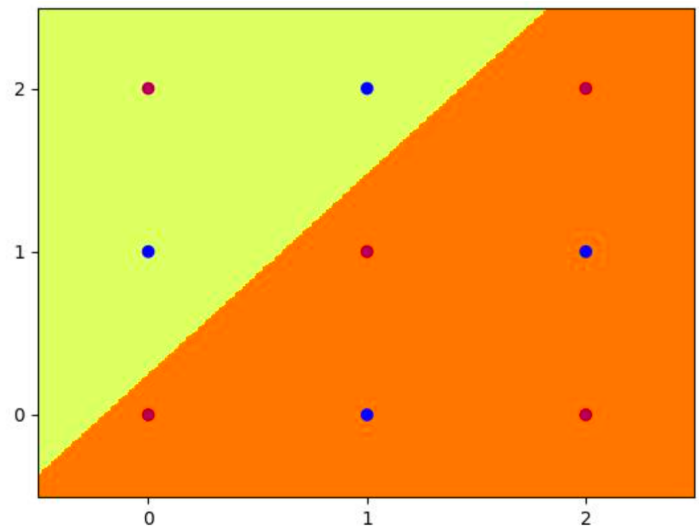
(c).From the confusion matrices of the three models, we can observe that some characters are easily confused during recognition, mainly because their stroke structures are similar. For example, in the NetLin model, su was misjudged as su up to 25 times, indicating that the model lacks the ability to recognize the boundaries between these characters. The linear model cannot capture the local spatial structural features of the image, so it is easy to confuse kana with similar strokes. The same problem occurred up to 35 times in the NetFull model, and the reason is similar to Netlin. The NetConv model has the strongest feature extraction capability, with an accuracy rate of 93%, and significantly reduced confusion. CNN can handle the translation and distortion of the image, so it also has better fault tolerance for handwritten characters of different styles. The CNN method can extract more from the image and achieve good accuracy.

Part 2: Multi-Layer Perceptron

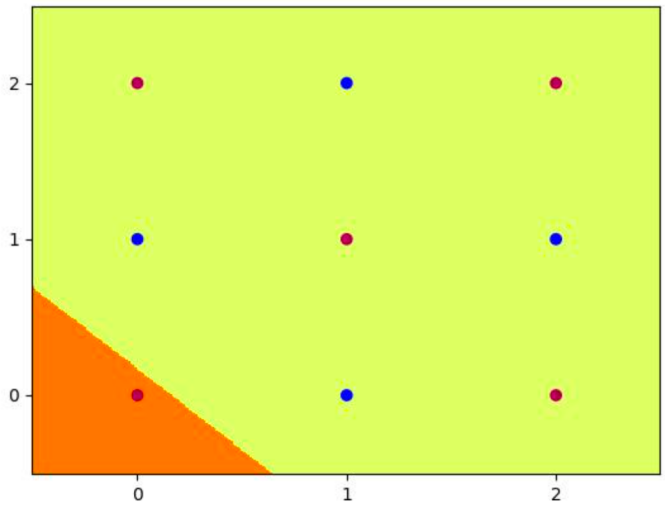
Question1



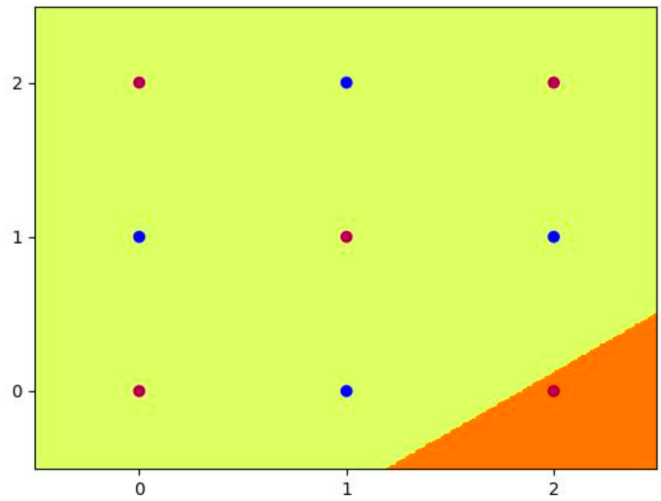
hid_5_0



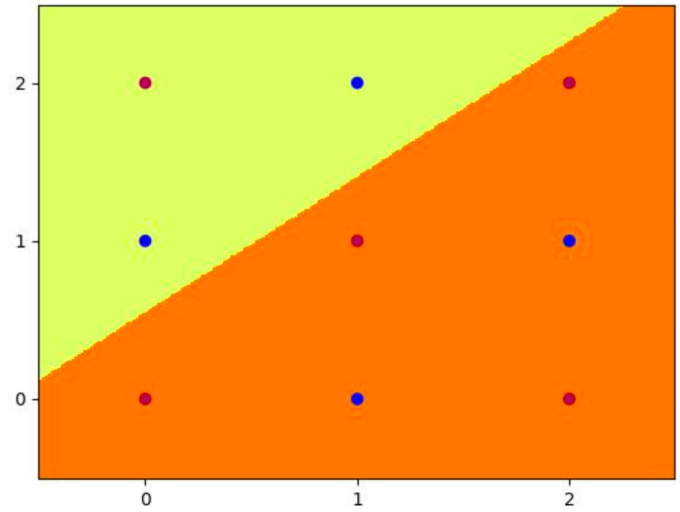
hid_5_1



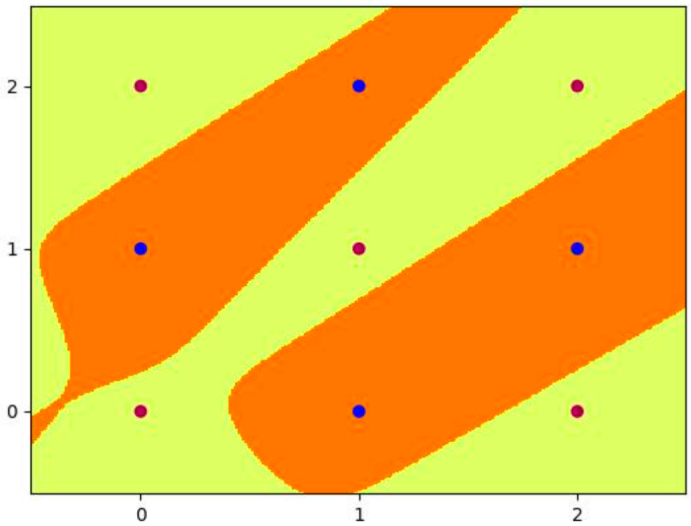
hid_5_2



hid_5_3

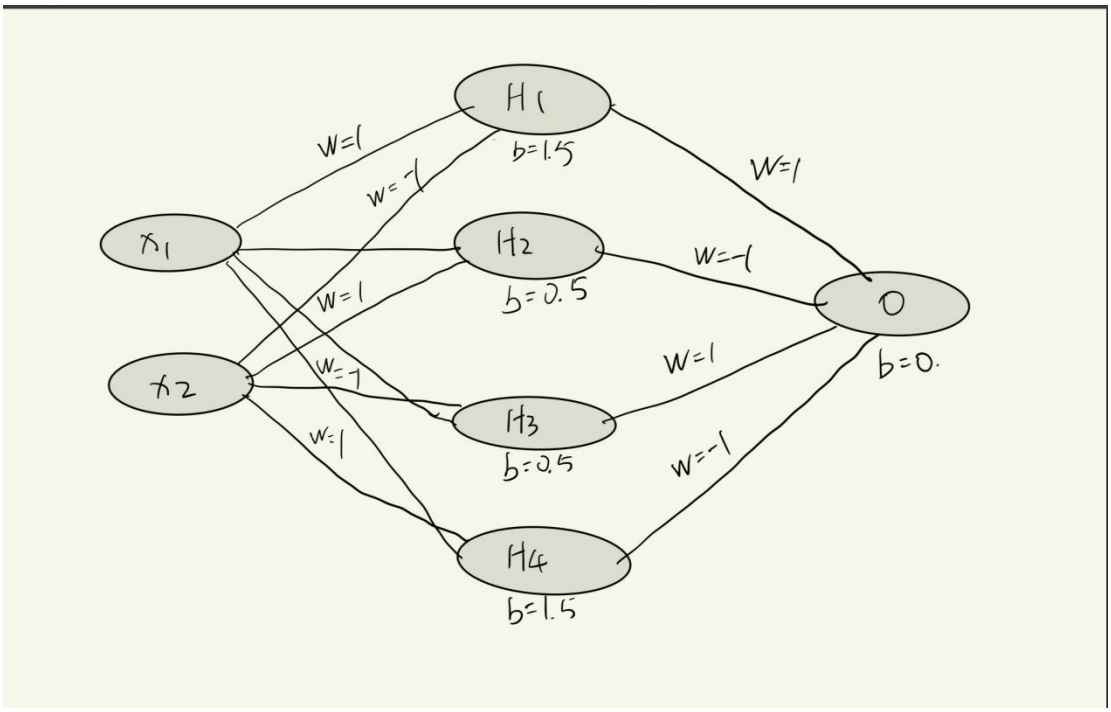


hid_5_4



final output

Question2



(x1,x2)	H1	H2	H3	H4	Output
(0,2)	0	1	0	1	0
(1,2)	1	1	0	1	1
(2,2)	1	0	0	1	0
(0,1)	1	1	0	1	1
(1,1)	1	0	0	1	0
(2,1)	1	0	1	1	1
(0,0)	1	0	0	1	0
(1,0)	1	0	1	1	1
(2,0)	1	0	1	0	0

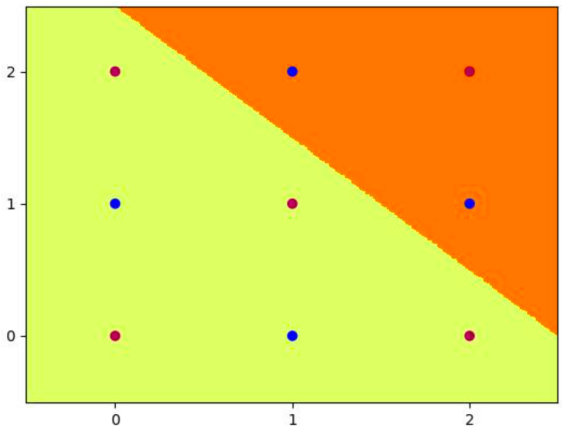
Output

in_hid_weight = [[1, 1], [1, 1], [-1, 1], [-1, 1]]
hid_bias = [-2.5, -1.5, -0.5, 0.5]
hid_out_weight = [[1, -1, 1, -1]]
out_bias = [0]

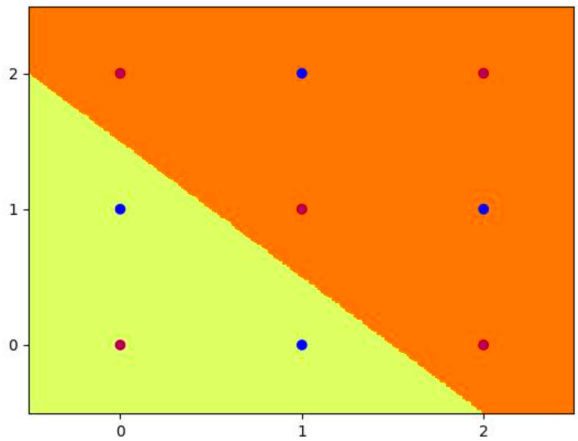
Initial Weights:
tensor([[1., 1.],
[1., 1.],
[-1., 1.],
[-1., 1.]])
tensor([-2.5000, -1.5000, -0.5000, 0.5000])

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

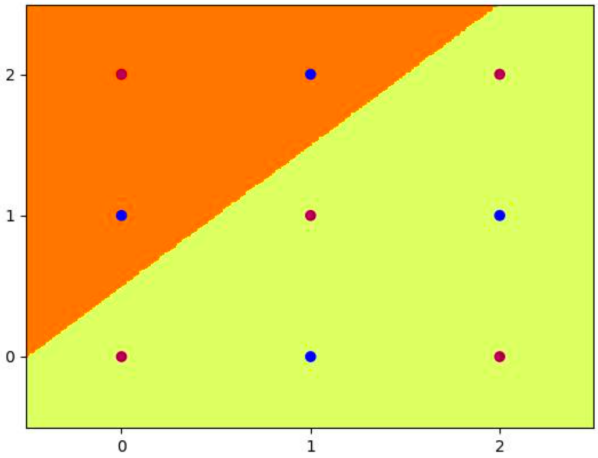
Question3



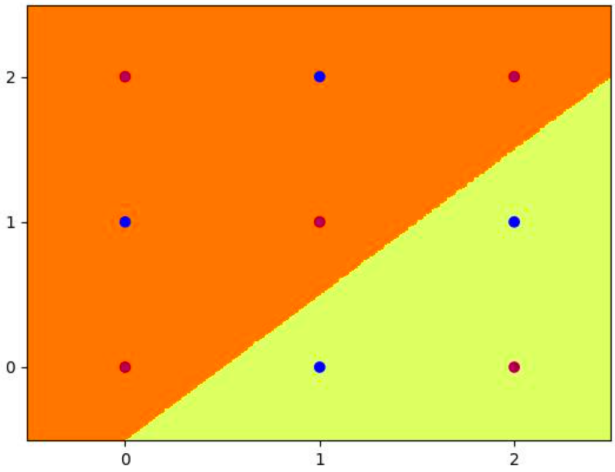
hid_4_0



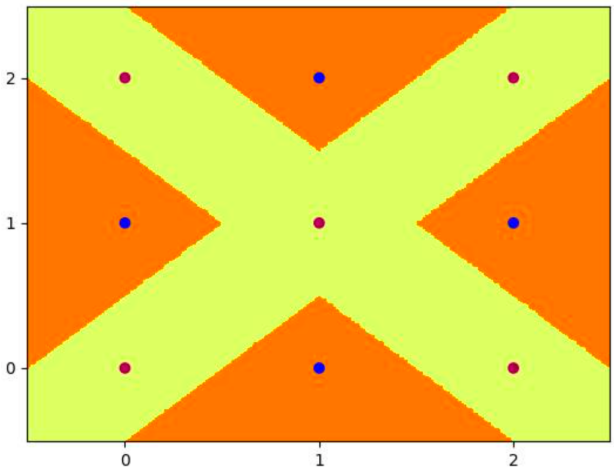
hid_4_1



hid_4_2



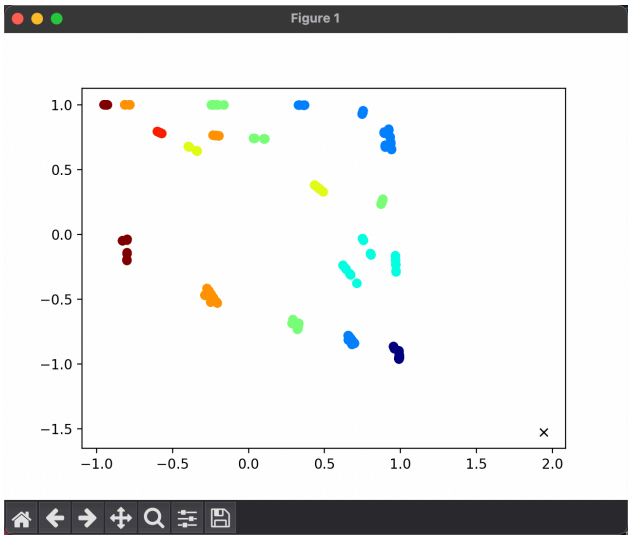
hid_4_3



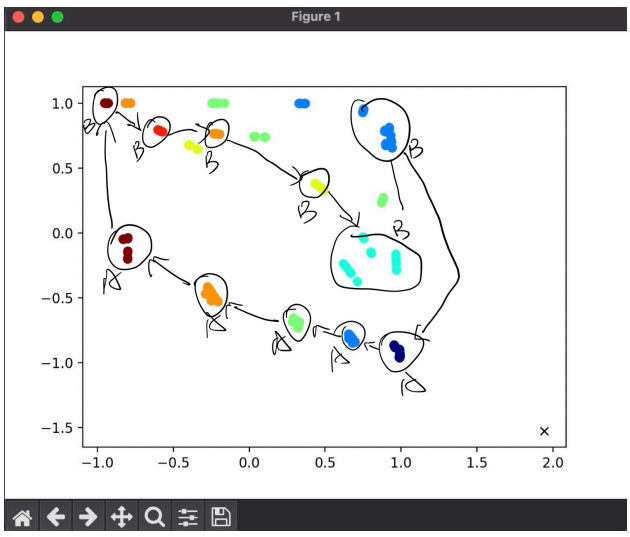
final output

Part 3: Hidden Unit Dynamics for Recurrent Networks

Question1

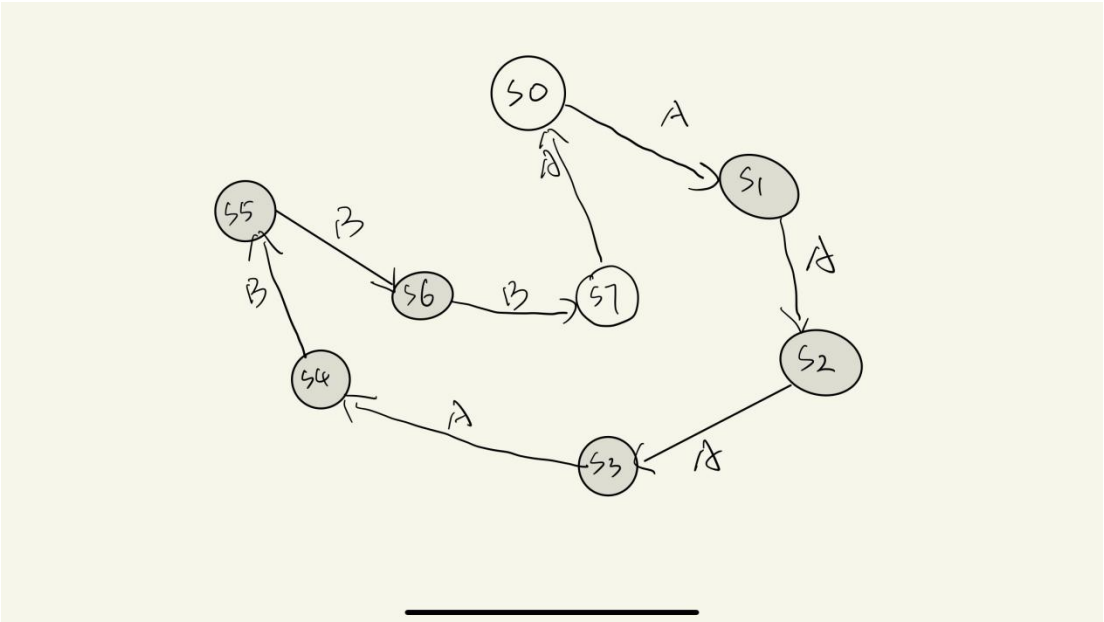


anb2n



anb2n

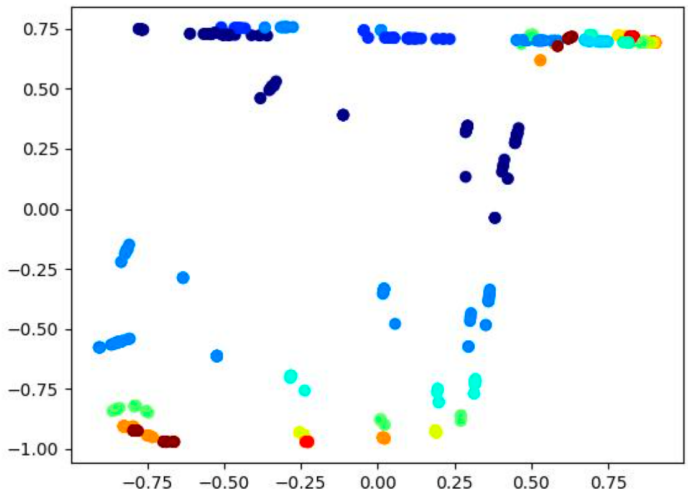
Question2



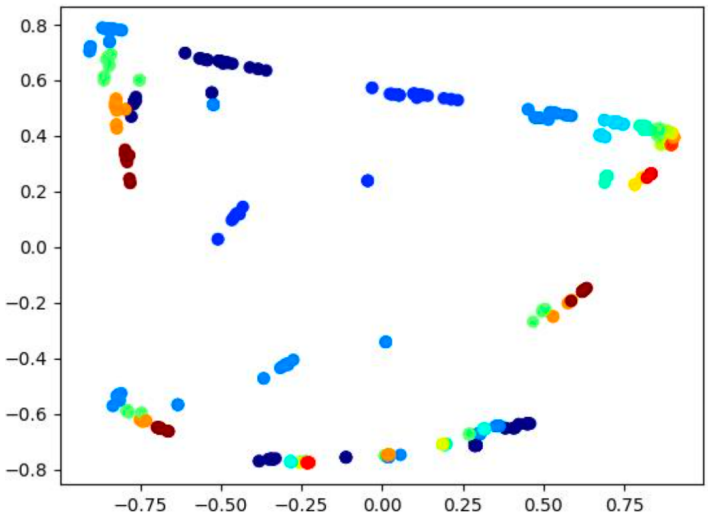
Question3

At the beginning of the sequence, the network reads the hidden vector position of A. Then when the first B appears, the hidden state corresponds to the previous number of A n, so n is known. Then after entering the B area, the hidden vector moves to the right in the lower half of the circle, outputting a B every step, until 2n. After the last B, the vector returns to the starting position, so the probability of outputting A is 1, so the network 100% predicts the starting A of the next line and recycles.

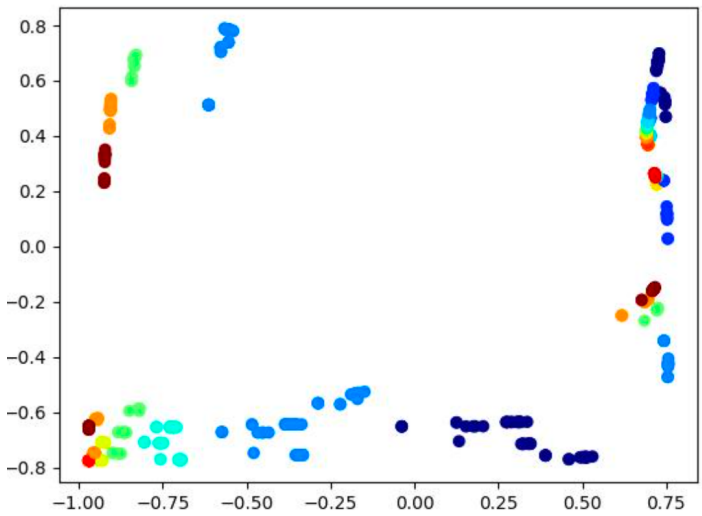
Question4



anb2nc3n_lstm3_01



anb2nc3n_lstm3_02



anb2nc3n_lstm3_12

Question5

The reason why LSTM successfully completed the anb2nc3n prediction task is that LSTM relies on memory cells and three gates as counters. The input gate, forget gate, and output gate of LSTM allow cell state c to act as different counters at different stages, and automatically clear useless counts at the end of the stage. Ordinary RNNs do not have memory cells and cannot store cell states of information for a long time, which will be diluted step by step. In this way, LSTM can remember long-distance and long-sequence information and can retain or discard content in time, so it can easily predict anb2nc3n.