Table of Contents

Part1

Q2

The network with **21 hidden layer nodes** successfully trained the model in around **149300 epochs**, with 0.0090 loss.

number of independent parameters from input to hidden layer 1 is: (2+1) * 21 = 63

number of independent parameters from hidden layer 1 to hidden layer 2 is: (21 + 1) * 21 = 462

number of independent parameters from hidden layer 2 to output layer is: 21 + 1 = 22

Total number of independent parameters: 63 + 462 + 22 = 547

The output image of Full3Net is shown below:

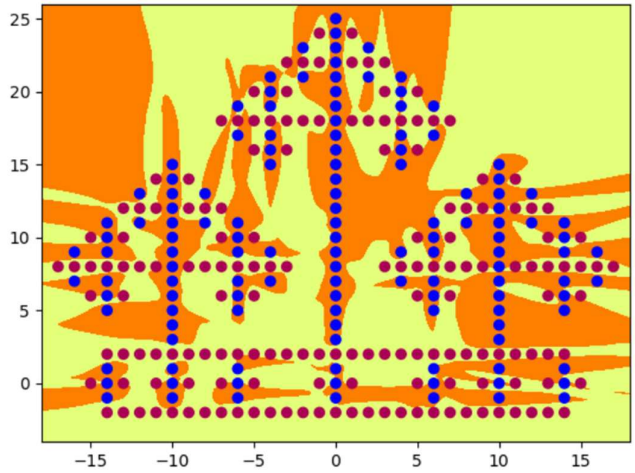


Figure 1: output image of Full3Net

Q4

The network with **19 hidden layer nodes** successfully trained the model in around **125400 epochs**, with 0.0116 loss.

number of independent parameters from input to hidden layer 1 is: (2+1) * 19 = 57

number of independent parameters from hidden layer 1 to hidden layer 2 is: (19 + 1) * 19 = 380

number of independent parameters from hidden layer 2 to hidden layer 3 is: (19 + 1) * 19 = 380

number of independent parameters from hidden layer 3 to output layer is: 19 + 1 = 20

Total number of independent parameters: 57 + 380*2 + 20 = 837

The output image of Full4Net is shown below:

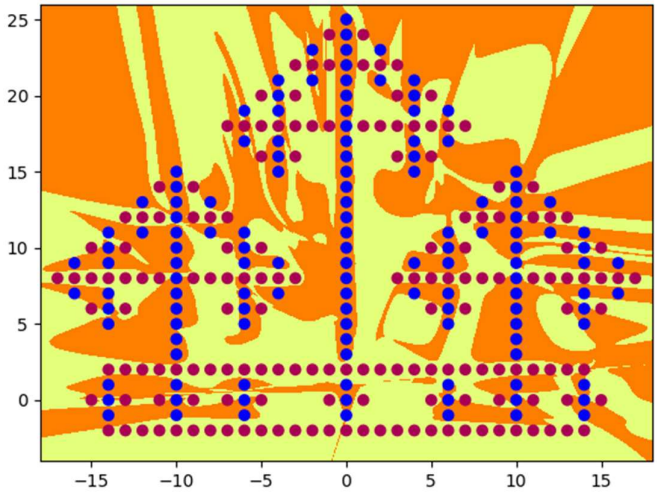


Figure 2: output image of Full4Net

The images of hidden layers of Full4Net are shown below:

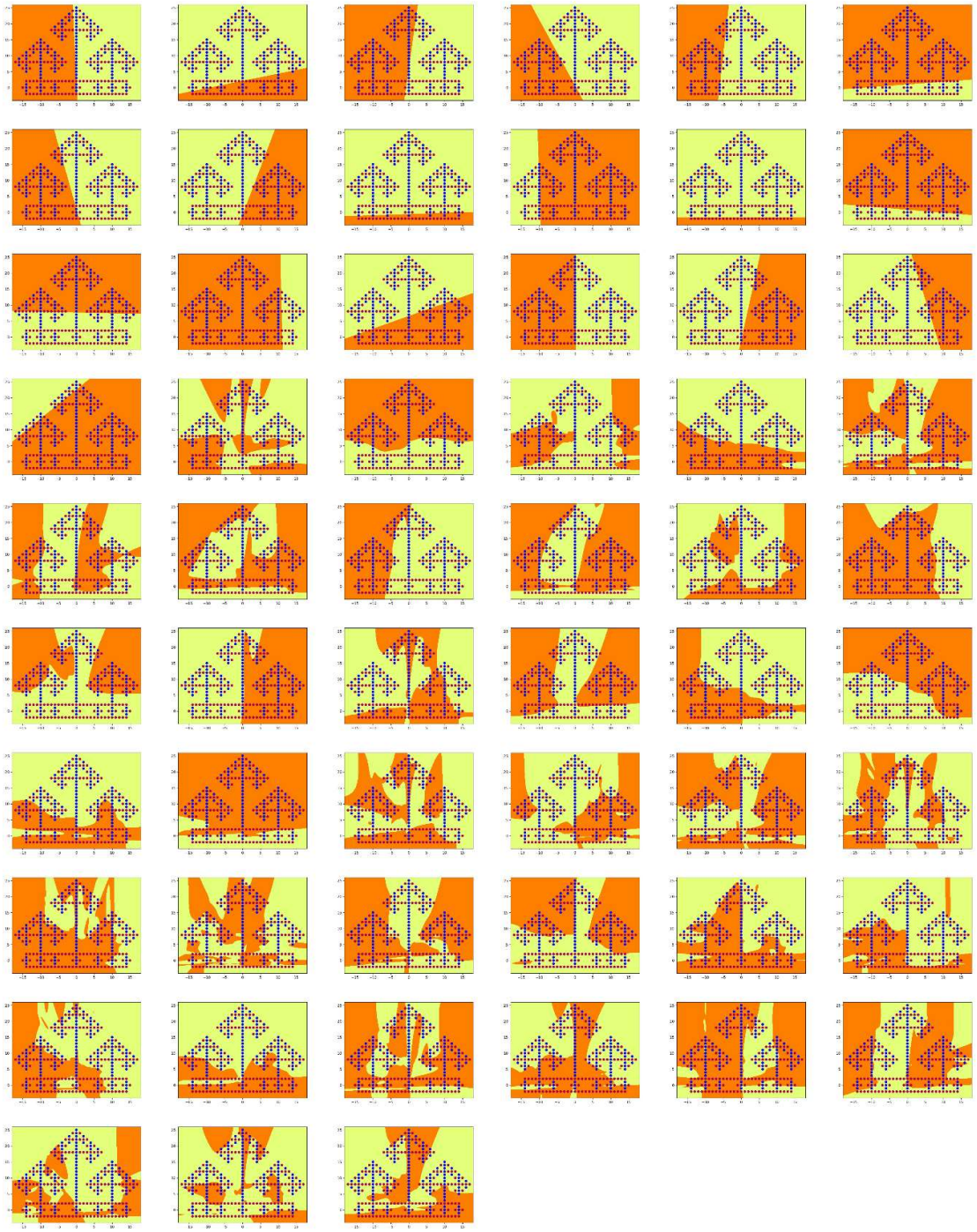


Figure 3: hidden layer nodes images of Full4Net

Q6

The network with **17 hidden layer nodes** successfully trained the model in around **76400 epochs**, with 0.0135 loss.

number of independent parameters from input to hidden layer 1 is: (2+1) * 17 = 51

number of independent parameters to hidden layer 2 is: (17 + 1) * 17 + 2*17 = 340

number of independent parameters to output layer is: 1 + 2 + 17*2 = 37

Total number of independent parameters: 51 + 340 + 37 = 428

The output image of Full3Net is shown below:

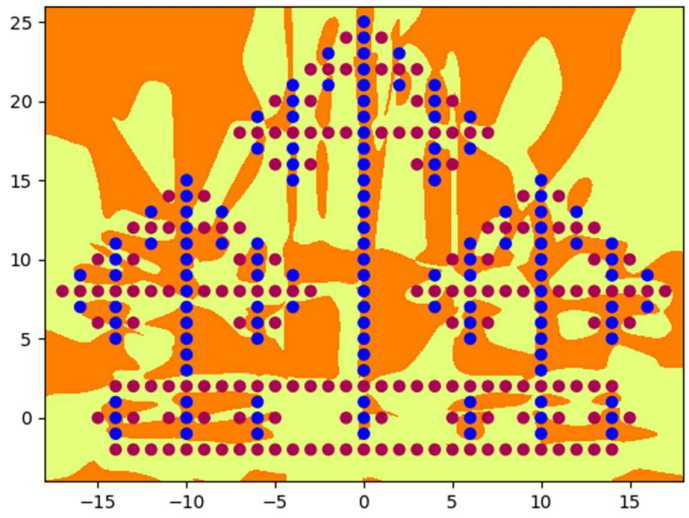


Figure 4: image of output layer of DenseNet

The images of the hidden layer nodes are shown below:

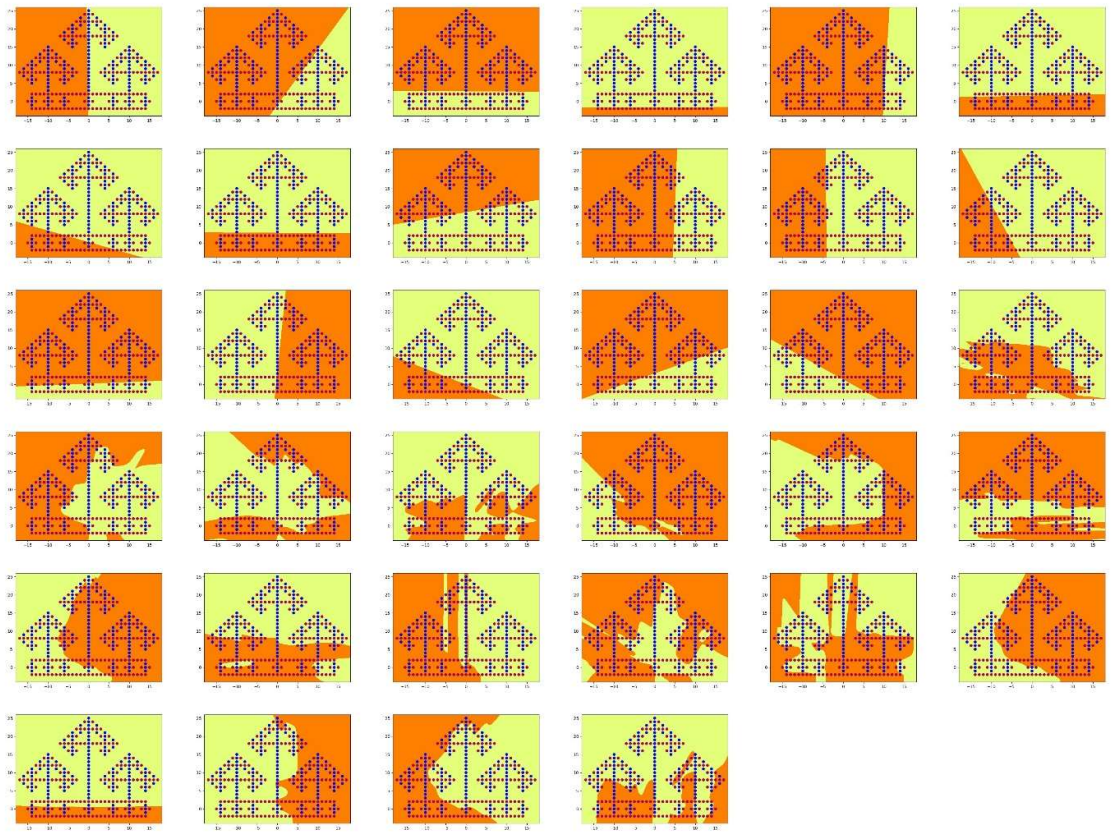


Figure 5: images of hidden layer nodes of DenseNet

Q7

(a).

The comparison table of the total number of independent parameters and the approximate number of minimum epochs for training among the three networks is shown below:

Table 1: Comparison table

	Number of Independent parameters	Number of minimum epochs
Full3Net	547	149500
Full4Net	837	125400
DenseNet	428	76400

Full4Net has the largest number of independent parameters, because of its extra hidden layer. While Full3Net has the largest number of the minimum training epochs due to its low training speed. DenseNet has the least number of these two for it uses skip connections, because it uses features learned from earlier layers, and thus can learn mapping with fewer parameters effectively.

(b).

Full4Net:

Based on the functions, for each layer, Full4Net uses features learned from the last layer linearly. The first hidden layer image was divided by one linear line. Curves separation lines occur from the second hidden layer and the more precise separated sections are divided in the third hidden layer.

DenseNet:

According to the functions, each layer learns features from all the previous layers rather than just the last layer. From the images, the first hidden layer is similar to Full4Net, where the image is divided by a linear line. As for the images of the second hidden layer, compared to the second hidden layer of Full4Net, they are more precise and accurate, while compared to the third layer of Full4Net, the separation sections are more concentrated.

(c).

Full4Net contains one more hidden layer than Full3Net. The output layers of Full4Net and Full3Net take in only the features of the third hidden layer, while the output layer of DenseNet takes in features of all the previous hidden layers and the inputs directly. This speeds up the training speed of DenseNet and reduces the number of independent parameters for the feature reuse.

Comparing the output images of these three networks, although they all identified the dots successfully, the separation patterns of Full4Net are more scattered than the other two networks.

Part 2

The final image is shown below, which is topologically identical to the provided image:

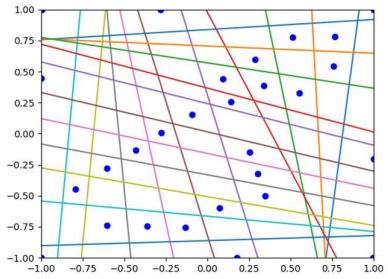


Figure 6: Final image of malta28 dataset

After flipping the image, it looks similar to the provided image shown below:

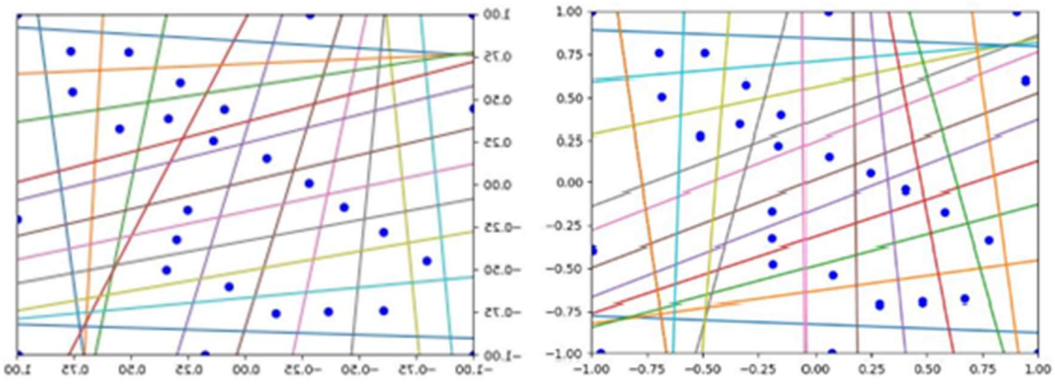


Figure 7: Comparison between the flipped image and the original image

Part 3

Q1

The annotated figure is shown below:

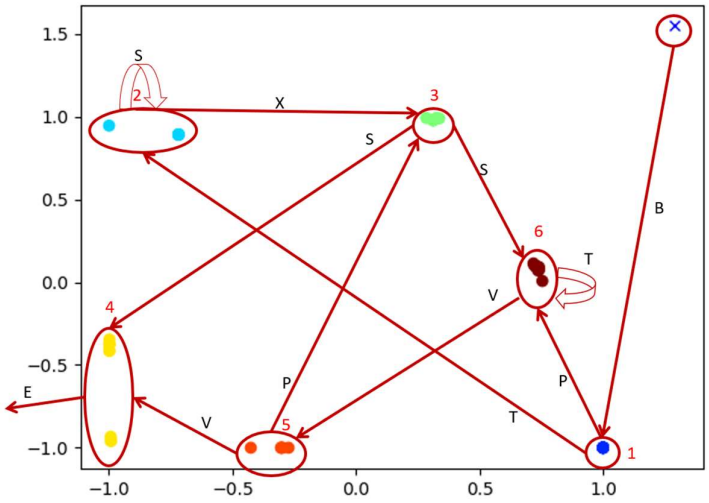


Figure 8: Annotated figure of SRN dynamics

Q2

The resulting figure is shown below

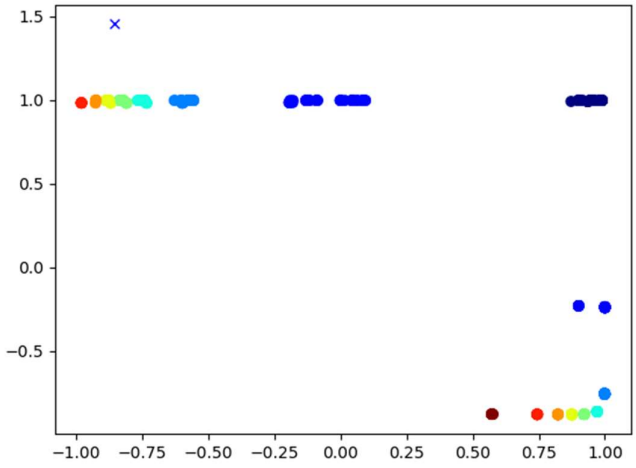


Figure 9: anbn resulting figure

Q3

Overall description

Based on the resulting figure shown in Q2 and the printed-out probabilities, most B dots are on the left as annotated in the figure below and most A dots are located on the area, while some B and A dots are mixed. This is because the first B and the A dots that are after the initial A are not predictable. Thus, only the successfully predicted B and A dots are located at value 1 in y axis.

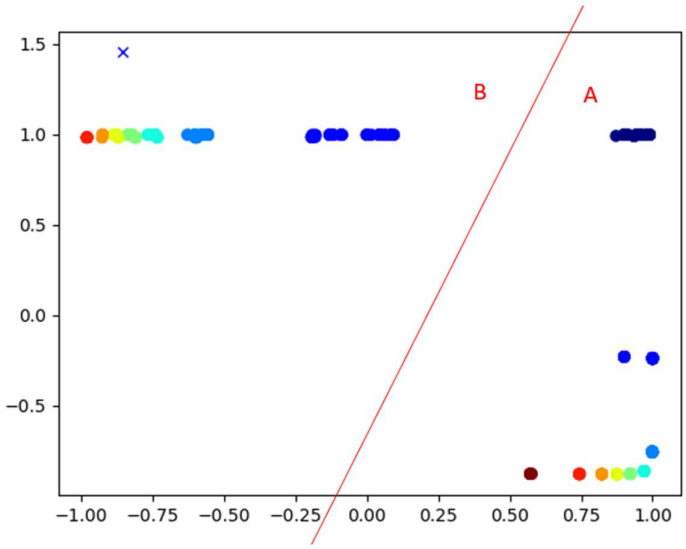


Figure 10: Resulting figure of anbn by SRN with annotation

Hidden unit activation change

Comparing the resulting image with the 10 epochs figure below, it shows that the predicted probabilities of B and A dots, circled by the green rectangle, are moving towards 1 in y axis as the epochs grows, which means the training is improving the prediction accuracy for both B and A characters.

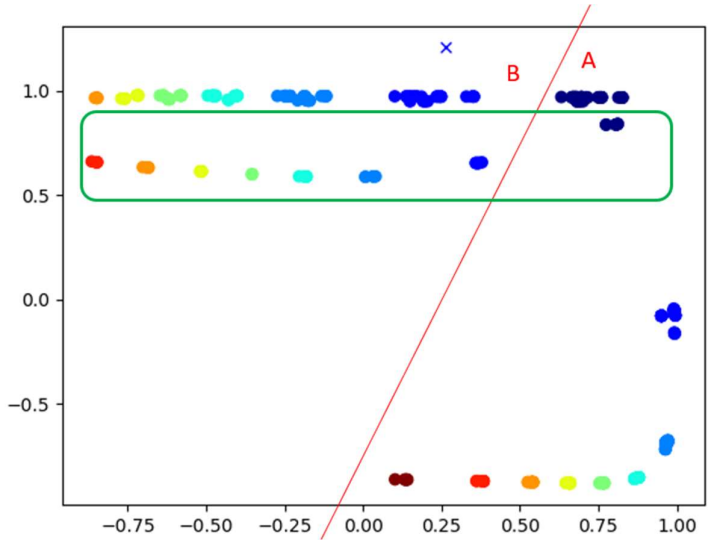


Figure 11: 10 epochs figure of anbn predicted by SRN with annotation

Explanation of character prediction

SRN takes in the sequential characters and predicts the probability based on the previous states. The previous state is then combined with the input and together output the new state. In this case, the states are memorized as numbers. SRN can count the number of given A from 0 to a certain number and predict the number of B by counting down the number till 1, where the first A of the next sequence is then predicted as 0.

Take the figure below for example, state sequences are marked with red lines. For each sequence, the number is counted from 0 till a certain number when B appears, then the number starts to

decrease as predicted to 0. This is how the character B can be predicted after the first B appears and how the first A in the next sequence can be predicted.

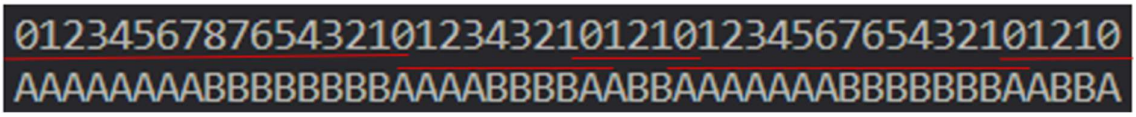
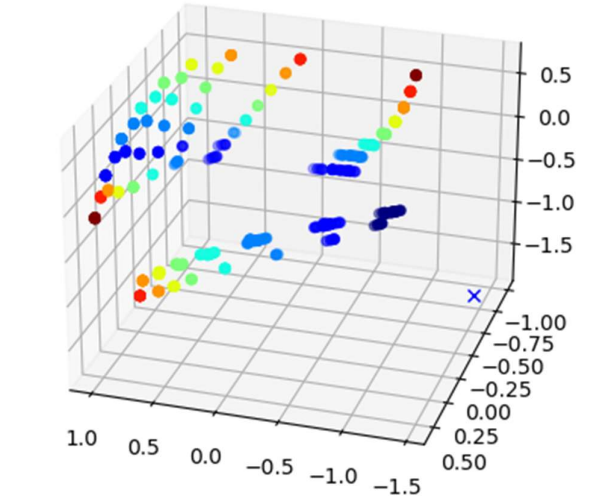
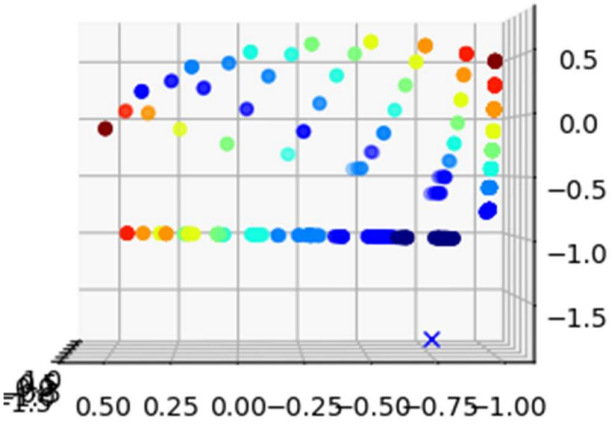


Figure 12: Example of prediction of character B and A

Q4

The points in hidden unit spaces of different views are shown below:





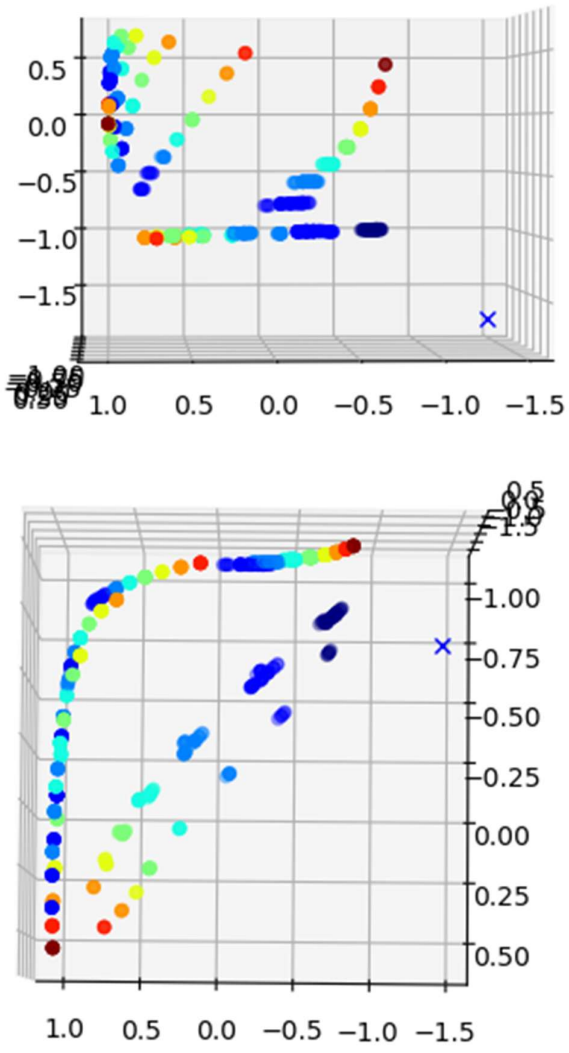


Figure 13: resulting figures in different view of anbncn by SRN

Q5

Hidden unit activation changes

The general concepts are the same as described in Q3 for the abnb prediction model. The successfully predicted probabilities are at the value -1 in z-axis, marked in the figures below.

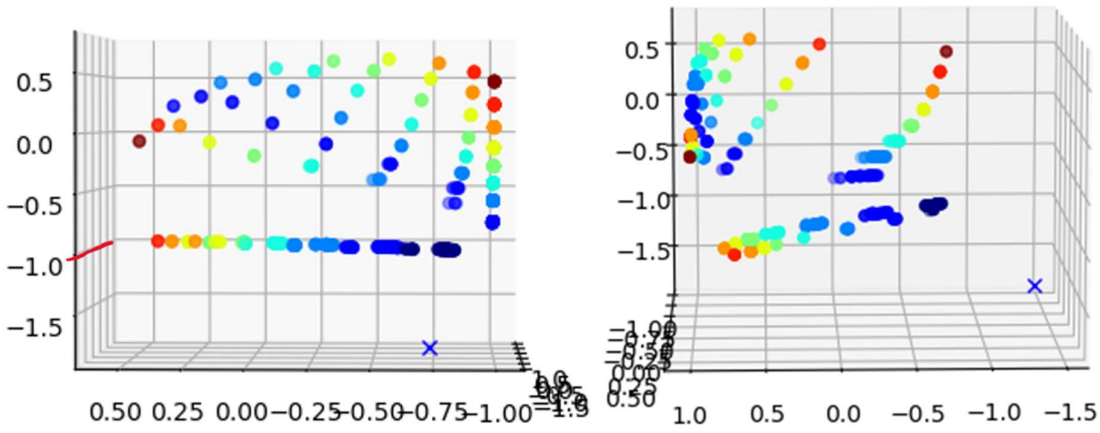


Figure 14: resulting fiture of anbncn by SRN with annotation

It can also be observed that only C, as marked by red line mentioned before, are mostly landed on -1 value in z-axis, while A and B have deviations. This is because only C is deterministic, while most A and B are unpredictable despite the last B and the first A of the next sequence are predictable.

For example, the figure below shows the dynamic change during the training. It indicates that the probability of B is increasing to near 1, then C appears and its probability keeps around 1/-1, which continues to the first A of the next sequence.

```
A [-0.73 -1.    0.05] [0.57 0.43 0.  ]
B [-0.79 -1.    0.25] [0.36 0.64 0.  ]
B [ 0.22 -0.92  0.36] [0. 1. 0.]
B [ 0.7  -0.74  0.49] [0. 1. 0.]
B [ 0.87 -0.49  0.56] [0. 1. 0.]
B [ 0.94 -0.22  0.56] [0. 1. 0.]
B [0.97 0.04 0.49] [0. 1. 0.]
B [0.99 0.28 0.33] [0. 1. 0.]
C [0.99 0.47 0.07] [0.   0.03 0.97]
C [ 0.59  0.39 -0.99] [0. 0. 1.]
C [ 0.71  0.18 -1.  ] [0. 0. 1.]
C [ 0.58  0.07 -1.  ] [0. 0. 1.]
C [ 0.41 -0.1  -1.  ] [0. 0. 1.]
C [ 0.1  -0.33 -1.  ] [0. 0. 1.]
C [-0.38 -0.61 -1.  ] [0.22 0.   0.77]
A [-0.79 -0.84 -1.  ] [1. 0. 0.]
```

Figure 15: example figure showing the hidden activation dynamics of anbncn

Explanation of character prediction

Besides the general concept mentioned in the previous section “Q3. Explanation of character prediction”, the difference here is that only the last B, all C and first A of the next sequence can be predicted. Most B is not deterministic is because the first B is unpredictable, while as the number of B increases, the predicted probability increases. Once the number of B counts down to 1, C occurs. This is why the predicted probability of the last B is high. Then, the number of C decreases from the

memorized maximum till 1. Finally, when it comes to 0, A appears. Thus, all C is predictable, as well as the first A of the next sequence.

Q6

The context layer of SRN is only combined with the input layer, while the context layer of LSTM is modulated by forget gate, input gate and output gate. The forget gate receives input at a specific time and the previous hidden gate. Then they are multiplied with the weight, the results of which are used to determine whether to forget (close to 1 --- keep, close to 0 --- forget). This property helps keep the useful information and removes the information that is no longer useful.

The input gate updates the LSTM unit based on the prior hidden state and current input. The output gate determines the current hidden state and passes it to the following LSTM unit.

By printing out the hidden nodes values as below, it shows that the result shown on the top [0.89 0.95 0.98 -0.36] is similar to the value of hid, which is based on the value of the context and the output gate. The larger the context is, the closer the value of hid is to the value of o_t. This property shows how LSTM controls the forget feature to extract useful information.

```
18 [ 0.89  0.95  0.98 -0.36] [0. 0. 0. 0. 0. 0. 1.]
epoch: 9
error: 0.0017
final: 0.0717
context
tensor([[ 1.4189,  1.9133,  2.8472, -2.5306]])
hid
tensor([[ 0.8882,  0.9508,  0.9812, -0.3630]])
i_t
tensor([[0.9991, 0.9958, 0.9741, 0.9942]])
f_t
tensor([[0.9865, 0.3947, 0.9821, 0.9605]])
g_t
tensor([[-0.9031,  0.9920,  0.8857, -0.9899]])
o_t
tensor([[0.9987, 0.9931, 0.9878, 0.3677]])
```

Figure 16: example LSTM context value