z5232961

Shuwan Guo

Assignment 1

Part 1: Fractal Classification Task

2. [1 mark]

20 is close to the minimum number of hidden nodes required for Full3Net to be trained successfully. The picture of the function computed by my network is shown below in Figure 1. Total number of independent parameters is 2*20+20*20+20+20*2+1=501.

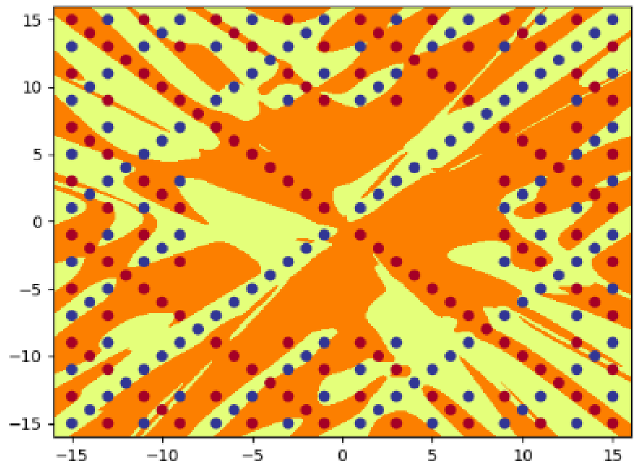


Figure 1: out_full3_20.png

4. [1 mark]

19 is close to the minimum number of hidden nodes required for Full4Net to be trained successfully. The picture of the function computed by my network is shown below in Figure 2. The plots of the hidden units of all 3 layers are shown in Figure 3, Figure 4 and Figure 5 below. Total number of independent parameters is 2*19+19*19*2+19+3*19+1=837.

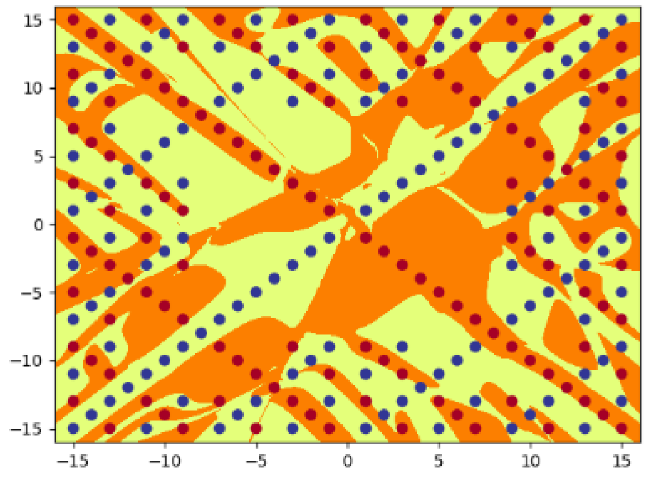


Figure 2: out_full4_19.png

Figure 3: Plots of First Hidden Layer Nodes of Full4Net

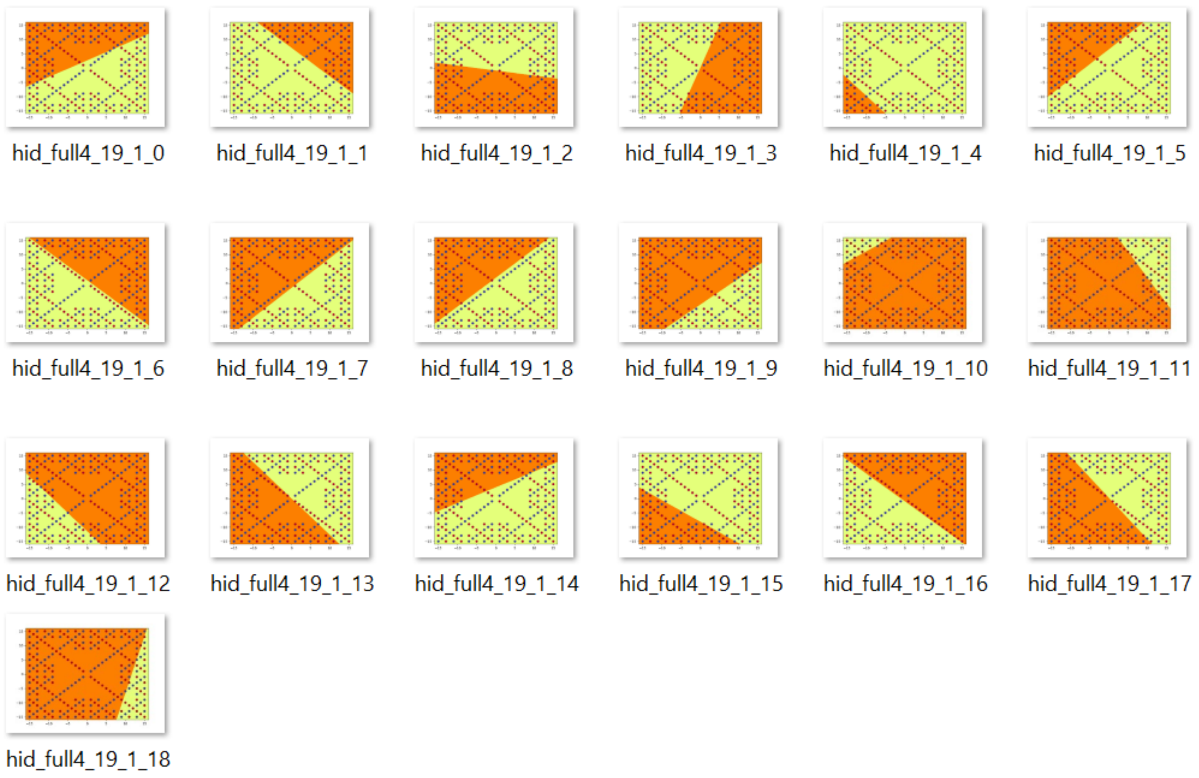


Figure 4: Plots of Second Hidden Layer Nodes of Full4Net

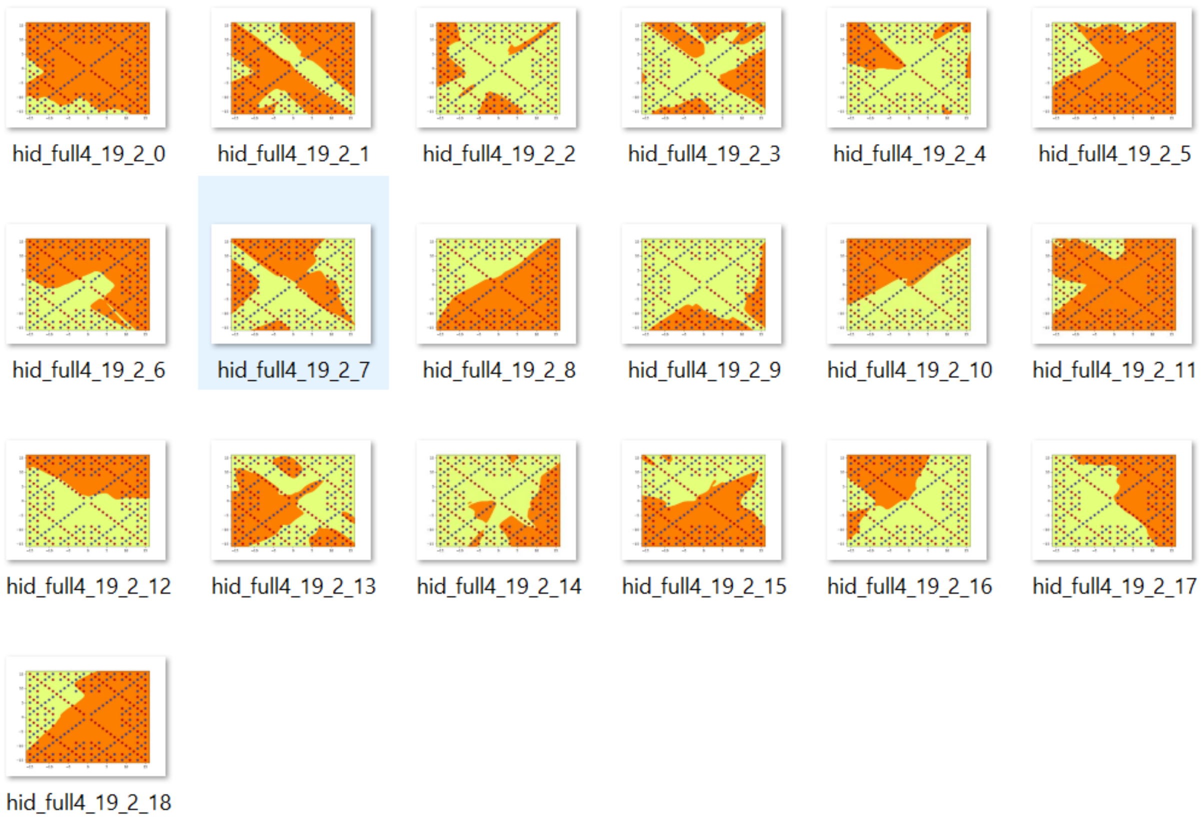
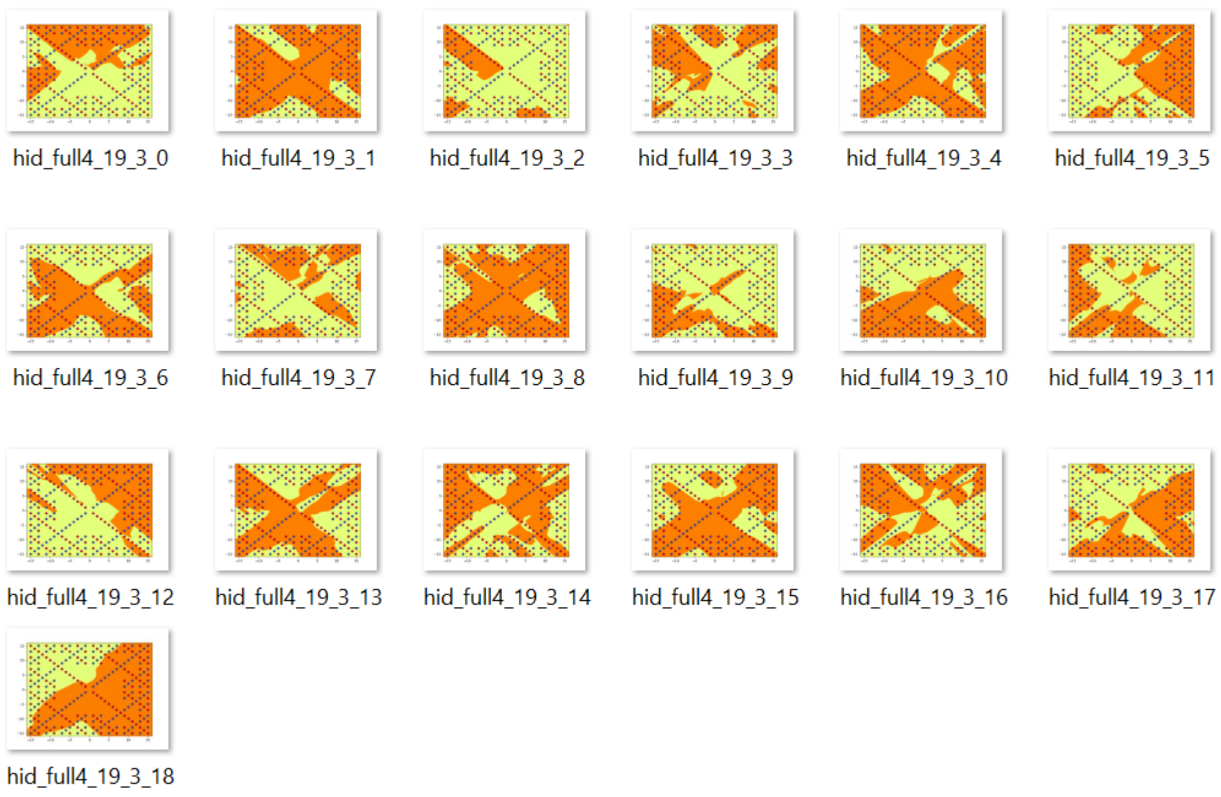


Figure 5: Plots of Third Hidden Layer Nodes of Full4Net



6. [1 mark]

15 is close to the minimum number of hidden nodes required for DenseNet to be trained successfully. The picture of the function computed by my network is shown below in Figure 6. The plots of the hidden units of both layers are shown in Figure 7 and Figure 8 below. Total number of independent parameters is 1+2*15+1+2*15+15*15+1+2+15+15=320.

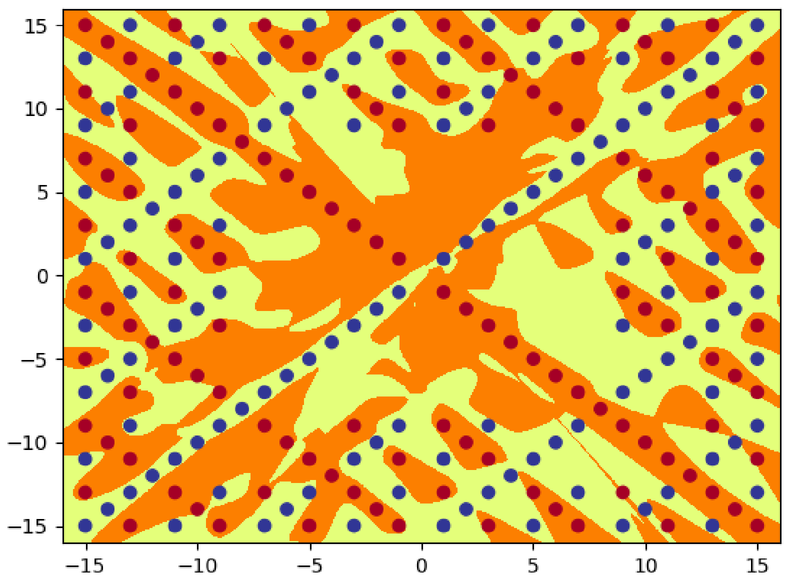


Figure 6: out_dense_15.png

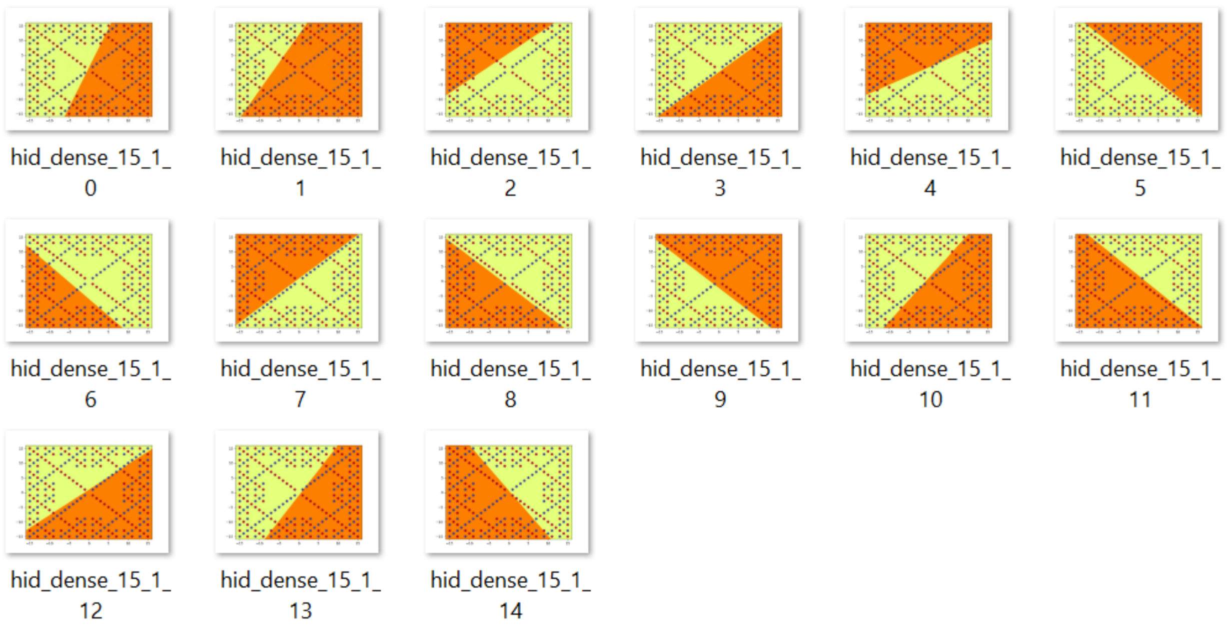


Figure 7: Plots of First Hidden Layer Nodes of DenseNet

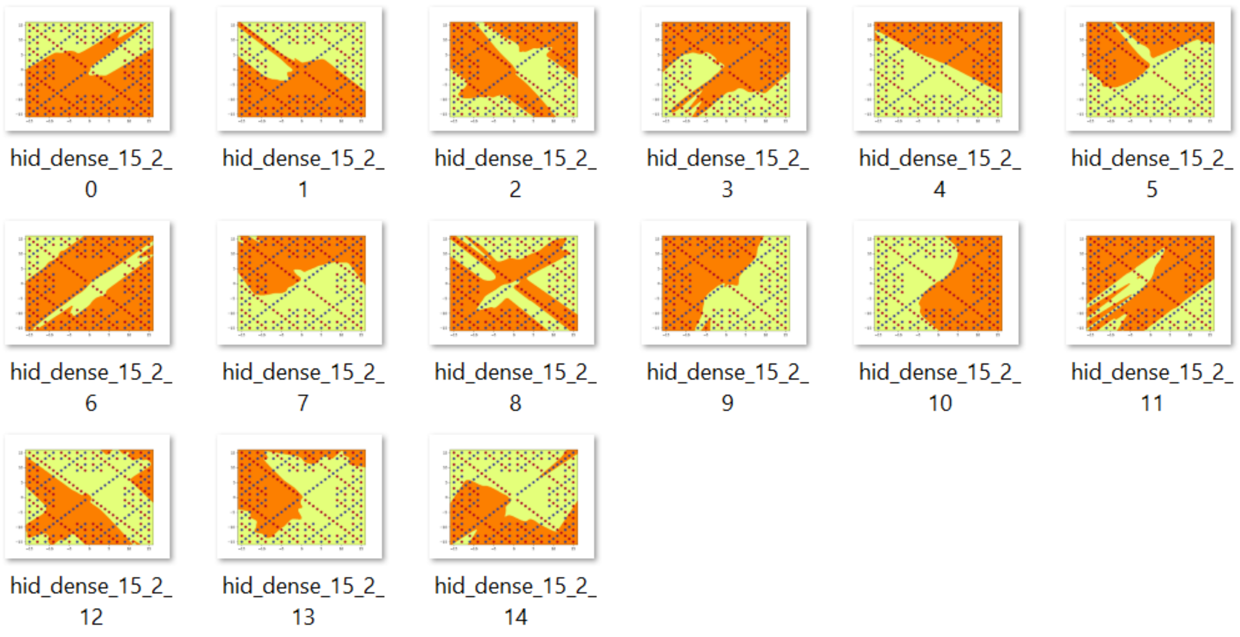


Figure 8: Plots of Second Hidden Layer Nodes of DenseNet

7. [3 marks]

a.

Model	Total Parameter Number	Epoch Number
Full3Net	501	124700
Full4Net	837	196800
DenseNet	320	87800

b.

Full4Net multiplies input with weights of each first hidden layer unit and applies tanh on the summation plus a bias. Then Full4Net multiplies the previous transferred summations with weights of each second hidden layer unit and applies tanh on the new summation plus a bias. After that, Full4Net multiplies the new transferred summations with weights of each third hidden layer unit and applies tanh on each newer summation plus a bias. Finally, Full4Net multiplies the newer transferred summations with weights of the output node and applies sigmoid on the final summation plus a bias to get the output.

DenseNet multiplies input with weights of each first hidden layer unit and applies tanh on the summation plus a bias. Then DenseNet multiplies the transferred summations with weights of each second hidden layer unit and forms a new summation, which is added to the summation of the multiplications between input and weights of this unit plus a bias. Then DenseNet applies tanh on the total and multiplies the transferred totals with weights of the output node and forms a new summation, which is added to the summation of the multiplications between input and weights of the output node, as well as summation of the multiplications between the transferred summations in the first hidden layer with weights of the output node plus a bias. Finally DenseNet applies sigmoid on the new total to get the output.

c.

Compared to Full3Net, Full4Net adds one more hidden layer, which leads to more complexity and non-linearity of the overall function, as well as more detailed separation of the two classes of red and blue dots as shown in Figure 1 and Figure 2 above. Compared to Full3Net and Full4Net, DenseNet has shortcut connections from each layer to all its subsequent layers, which leads to an overall function with more complexity and fewer parameters. The second layer of DenseNet can achieve segmentation effect close to that achieved by the third layer of Full4Net as shown in Figure 5 and Figure 8 above.

Part 2: Encoder Networks

1. [2 marks]

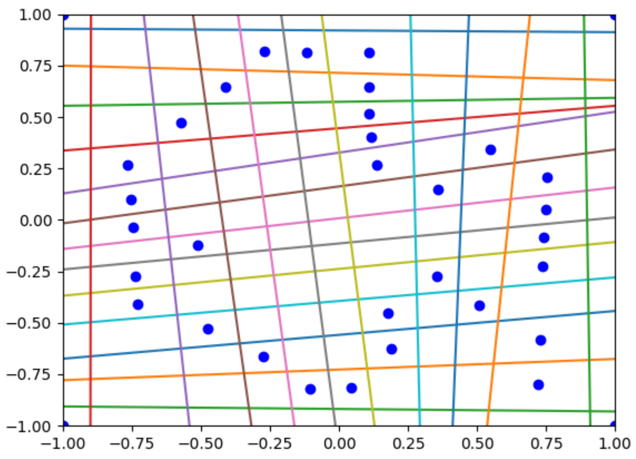
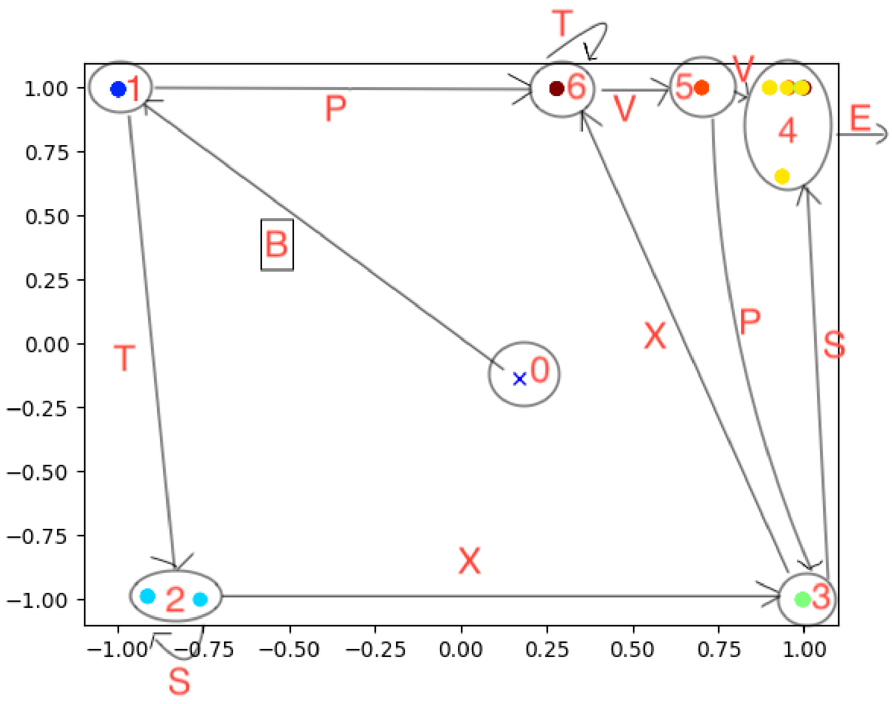


Figure 9: ant35 Image

Part 3: Hidden Unit Dynamics for Recurrent Networks

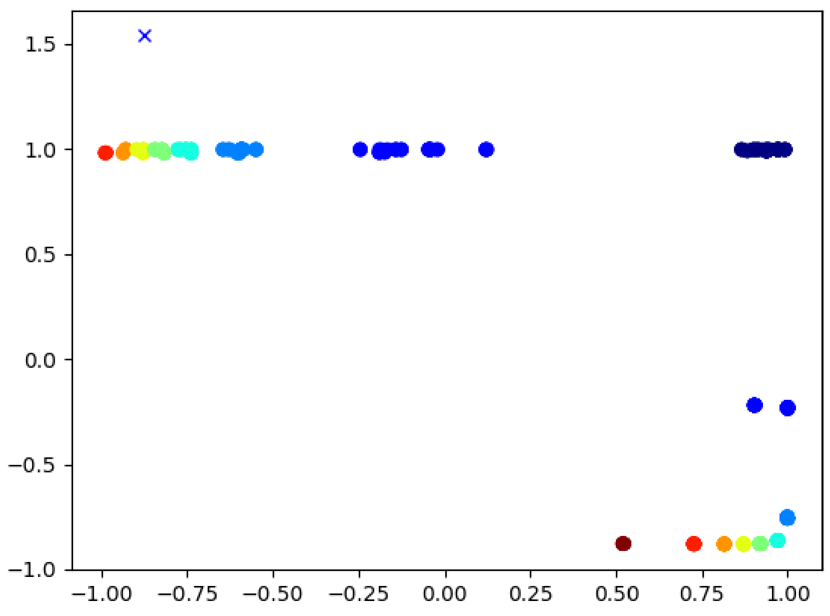
1. [2 marks]

Figure 10: Hidden Unit Activations for Reber SRN



2. [1 mark]

Figure 11: Hidden Unit Activations for anbn SRN

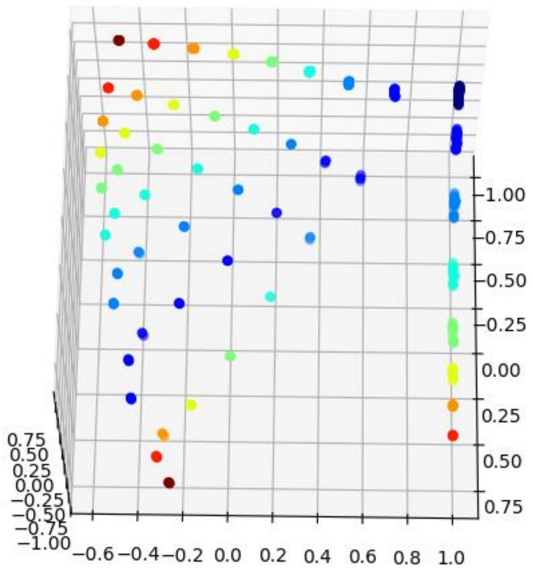


3. [1 mark]

In Figure 11 above, two hidden unit values for one input character are represented as x, y coordinates of a dot. Each color represents one state with the number of A's we have seen or the number of B's we are still expecting to see. Since the longest string consists of 8 A's, we can see there are altogether 8 color states, with one initial state shown as a cross. Before occurrence of the first B, activities are going on at the top part of Figure 11. As the string is processed, we move from state 0 to state 1, the dark blue state, then to state 2, the blue state…, until we reach the red state with all 8 A's seen. Then with the occurrence of the first B, we start to move to state 8, the brown state. After that, activities are going on at the bottom part of Figure 11. Then we can predict the subsequent B's in the sequence and are moving back one state with the processing of one B. When we finish processing all B's, we move back to state 0 again with the predicted appearance of the first A in the next sequence and then to state 1 at the top.

4. [1 mark]

Figure 12: Hidden Unit Activations for anbncn SRN



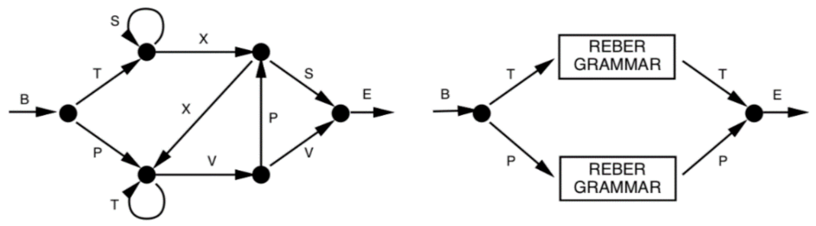
5. [1 mark]

In Figure 12 above, three hidden unit values for one input character are represented as x, y, z coordinates of a dot. The color state is used to count up the A's and count down the B's and C's. Since the longest string consists of 8 A's, we can see there are altogether 8 color states divided into three regions. Before occurrence of the first B, activities are going on at the right part of Figure 12. As the string is processed, we move from state 0 to state 1, the dark blue state, then to state 2, the blue state…, until we reach the red state, state 7, with all 8 A's seen. Then with the occurrence of the first B, we start to move to state 8, the brown state at the left bottom corner. After that, activities are going on in the middle part of Figure 12. Then we can predict the subsequent B's and C's in the sequence and are moving back one state with the processing of one B. When we finish processing all B's, we move to the brown state at the top left corner with the first appearance of C. After that, we are moving back one state with the processing of one C. When we finish processing all C's, we move back to state 0 again with the predicted appearance of the first A in the next sequence and then to state 1 at the top right corner.

6. [3 marks]

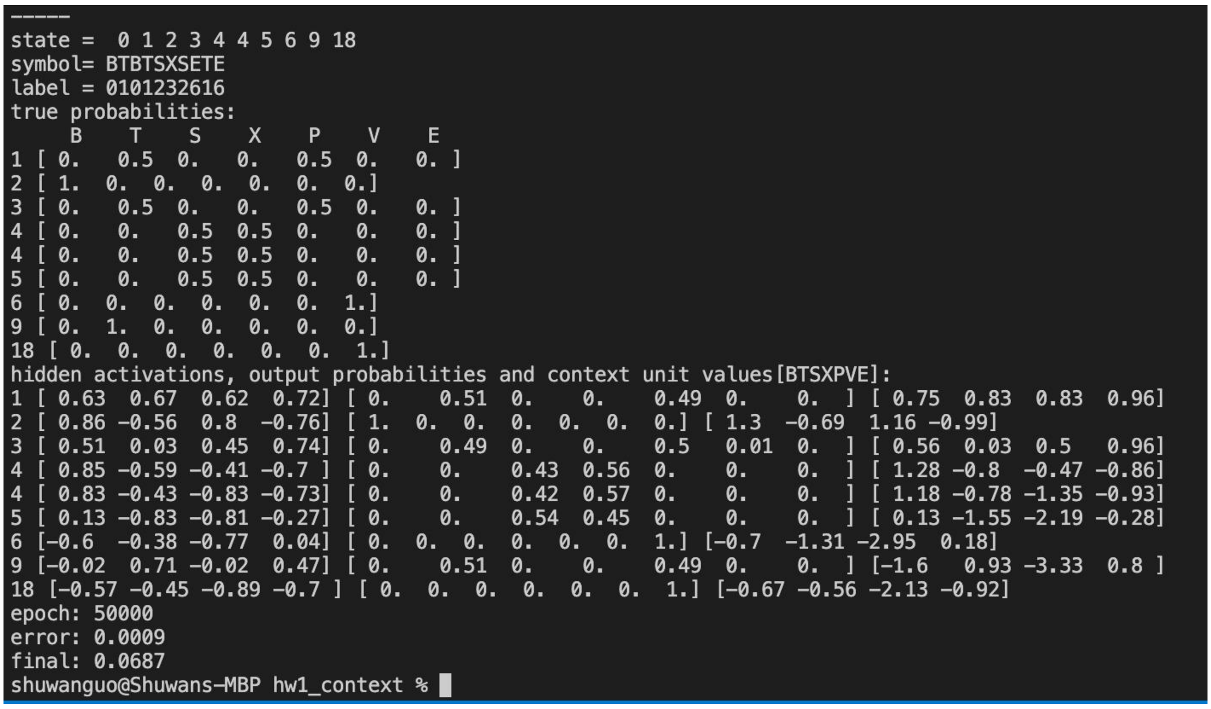
To predict the Embedded Reber Grammar, we need LSTM that can learn long range dependencies using a combination of forget, input and output gates.

Figure 13: Embedded Reber Grammar



The LSTM maintains a context layer which is distinct from the hidden layer but contains the same number of units. The hidden unit value is based on the output gate value and context unit value. The current context unit value is based on the previous context unit value, forget gate value and input value. The forget gate determines how much previous context to keep for the current context unit while the input gate determines how much current input information to keep. The output gate determines how much current context information to be passed to the current hidden unit value.

I modify the code to print out the context unit values as follows:



```
-----
state =  0 1 2 3 4 4 5 6 9 18
symbol= BTBTSXSETE
label = 0101232616
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.   0.5  0.   0.   0.5  0.   0. ]
4 [ 0.   0.   0.5  0.5  0.   0.   0. ]
4 [ 0.   0.   0.5  0.5  0.   0.   0. ]
5 [ 0.   0.   0.5  0.5  0.   0.   0. ]
6 [ 0.  0.  0.  0.  0.  0.  1.]
9 [ 0.  1.  0.  0.  0.  0.  0.]
18 [ 0.  0.  0.  0.  0.  0.  1.]
hidden activations, output probabilities and context unit values[BTSXPVE]:
1 [ 0.63  0.67  0.62  0.72] [ 0.    0.51  0.    0.    0.49  0.    0.  ] [ 0.75  0.83  0.83  0.96]
2 [ 0.86 -0.56  0.8  -0.76] [ 1.  0.  0.  0.  0.  0.  0.] [ 1.3  -0.69  1.16 -0.99]
3 [ 0.51  0.03  0.45  0.74] [ 0.    0.49  0.    0.    0.5   0.01  0.  ] [ 0.56  0.03  0.5   0.96]
4 [ 0.85 -0.59 -0.41 -0.7 ] [ 0.    0.    0.43  0.56  0.    0.    0.  ] [ 1.28 -0.8  -0.47 -0.86]
4 [ 0.83 -0.43 -0.83 -0.73] [ 0.    0.    0.42  0.57  0.    0.    0.  ] [ 1.18 -0.78 -1.35 -0.93]
5 [ 0.13 -0.83 -0.81 -0.27] [ 0.    0.    0.54  0.45  0.    0.    0.  ] [ 0.13 -1.55 -2.19 -0.28]
6 [-0.6  -0.38 -0.77  0.04] [ 0.  0.  0.  0.  0.  0.  1.] [-0.7  -1.31 -2.95  0.18]
9 [-0.02  0.71 -0.02  0.47] [ 0.    0.51  0.    0.    0.49  0.    0.  ] [-1.6   0.93 -3.33  0.8 ]
18 [-0.57 -0.45 -0.89 -0.7 ] [ 0.  0.  0.  0.  0.  0.  1.] [-0.67 -0.56 -2.13 -0.92]
epoch: 50000
error: 0.0009
final: 0.0687
shuwanguo@Shuwans-MBP hw1_context %
```

As the string is processed, values of the four context units behave differently since each unit may perverse context information for a different category. Some values of a unit are increasing slightly (e.g. from -0.8 to -0.78), meaning adding in a small amount of previous context in the category. Some are increasing to a greater extent (e.g. from -0.99 to 0.96), meaning adding in a larger amount of previous context in the category. Similarly, some values may decrease to some extent to move out a certain amount of irrelevant previous context. This context unit mechanism makes LSTM able to learn long range dependencies.