COMP9444 Assignment

Z5309467, Minseok (Joshua) Kim

Term 2 2022

Contents

Fractal Classification Task 3
Full3Net 3
Full4Net 5
DenseNet 7
Discussion 9
Encoder Networks 11
Hidden Unit Dynamics for Recurrent Networks 12
Annotated Reber plot 12
$anbn$ plot 13
$anbncn$ Plot 14
LSTM 15
APPENDIX 18
Hidden Full4Net Plots 18
Layer 1 18
Layer 2 21
Layer 3 24
Hidden DenseNet Plots 27
Layer 1 27
Layer 2 29

Fractal Classification Task

All plots of the hidden layers are located in the Appendix.

Full3Net

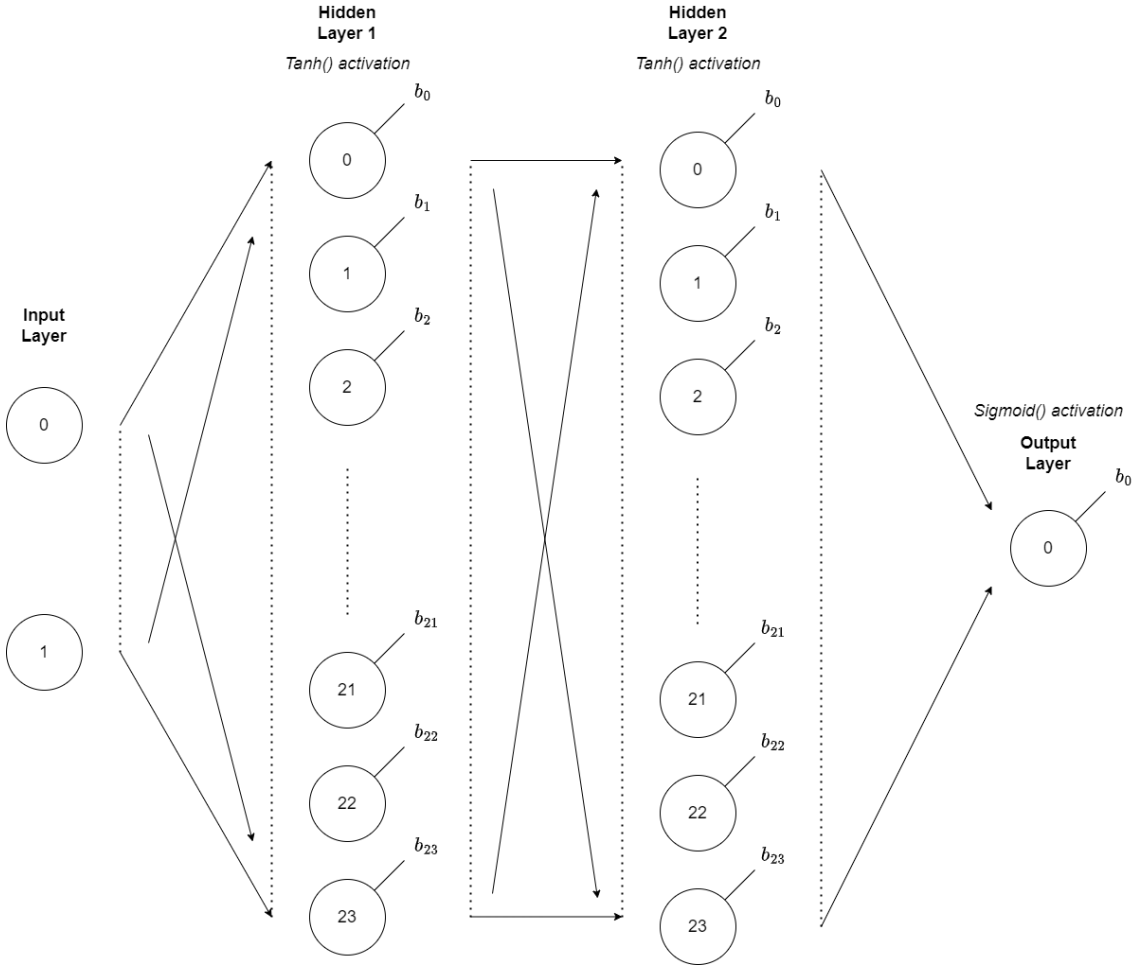


Figure 1. Diagram of the Full3Net Network

Due to the stochastic nature of weight initialization, it cannot be said with certainty that any one value is the minimum number of hidden nodes required to converge to 100% accuracy due to the chance that the network will stall in a local minimum or simply not enough epochs. Due to this, if the ratio of test runs that do not converge against those that do is *determined to be within a reasonable range*, it will be assumed that that number of nodes converges. As the condition *within a reasonable range* is subjective, other people's definition of consistent convergence, and hence also the minimum number of nodes required will differ.

For full3Net, this value was determined to be 24 hidden nodes, with 5 test runs done on this value shown in Table 1. It should be noted that although some runs converged for hidden nodes as low as 22, it was only occasionally so and a value of 24 was a more conservative estimate of the minimum nodes required.

To determine the approximate number of epochs needed to train the Full3 Network, the network was trained 5 times at 24 hidden nodes and the average epochs taken to train the network was used. This is shown in Table 1

Full3Net Training @ hid = 24

Testing run	Epoch
1	91000
2	189400
3	94900
4	33000
5	112000
Average	104100

Table 1. Calculation of approximate epochs required to train Full3Net at 24 hid

The approximate number of epochs required to fully train the Full3Net data is around 104100 epochs.

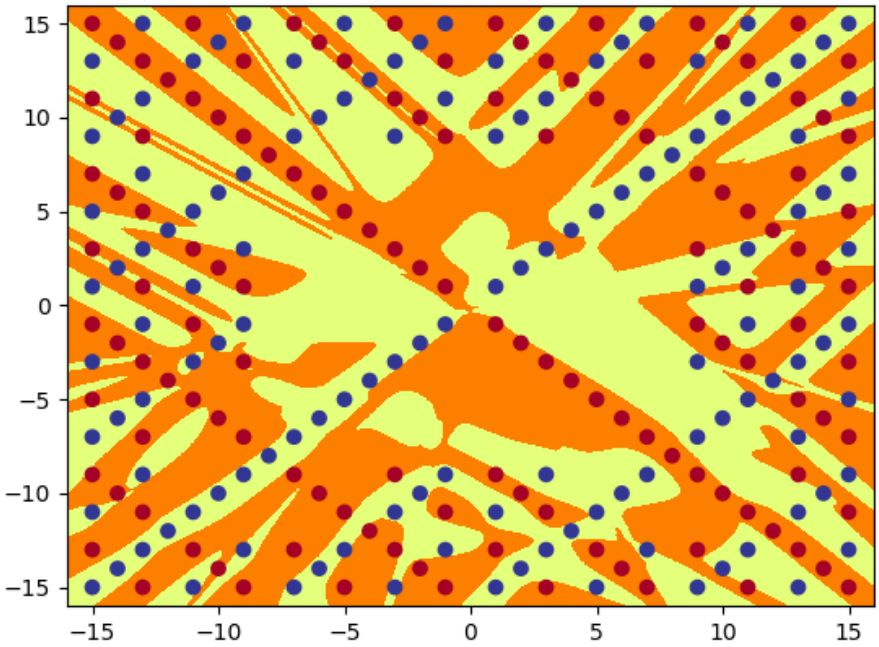


Figure 2. Full3Net Fractal Classification output when trained at 24 hid, fully trained at 91000 epochs.

The number of independent parameters in Full3Net can be calculated by summing the total amounts of weights and bias. In figure 1, it can be seen that Full3Net is fully connected without any other connections and bias on all nodes except for the input nodes. Hence, the total number of independent parameters is

$$\text{independent parameters} = \sum w_{in\to hid1} + \sum w_{hid1\to hid2} + \sum w_{hid2\to out} + \sum bias$$

$$= \sum_{i=0}^{1}\sum_{j=0}^{23} w_{ij} + \sum_{i=0}^{23}\sum_{j=0}^{23} w_{ij} + \sum_{i=0}^{23} w_i + \ (24 + 24 + 1)$$

$$= (24 + 24) + (24 * 24) + 24 + (49) \quad = \quad 697$$

Full4Net

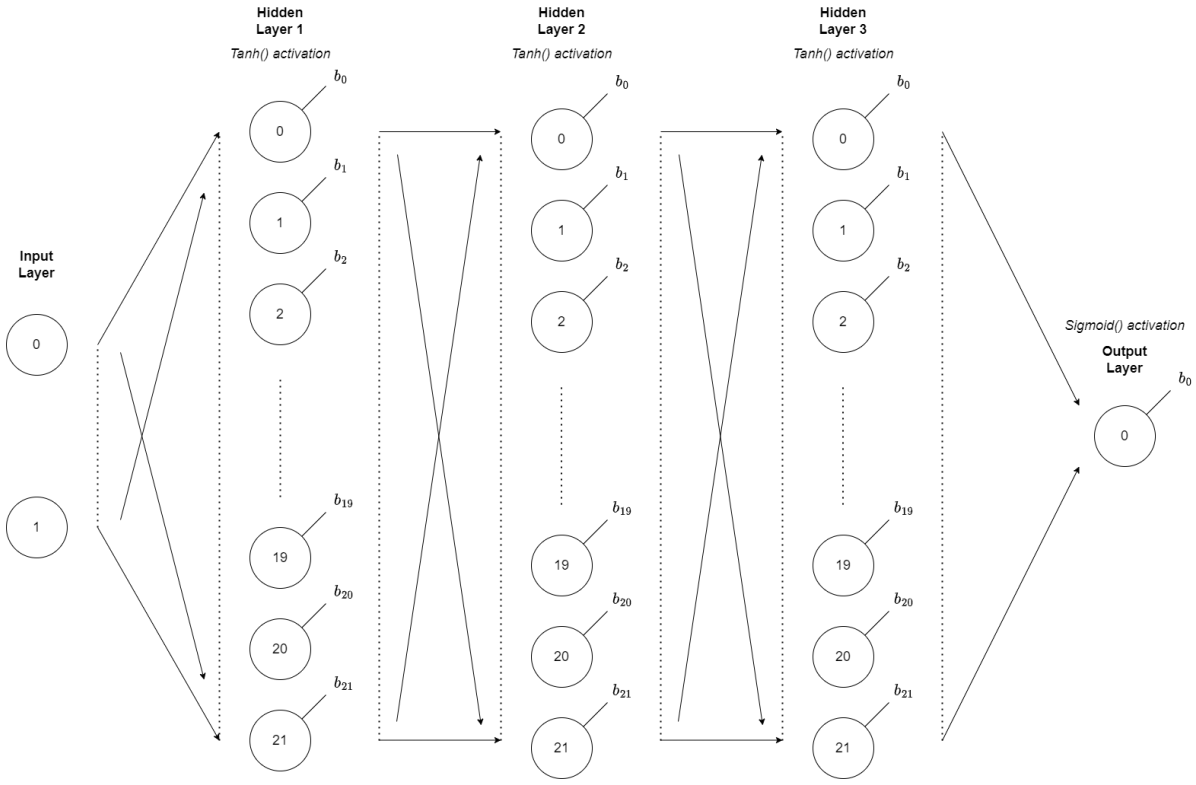


Figure 3. Diagram of the Full4Net Network

Using a similar methodology to Full3Net, the minimum number of hidden nodes for the Full4Net was determined to be 22. Again, although number of nodes as low as 19, it only occasionally converged and was determined that any number below 22 did not converge consistently. The approximate number of epochs to train is calculated by finding the average of 5 successful runs as shown in Table 2.

Full4Net Training @ hid = 22

Testing run	Epoch
1	50000
2	99400
3	89500
4	70000
5	181600
Average	98100

Table 2. Calculation of approximate epochs required to train Full4Net at 22 hid

Hence the approximate number of epochs for Full4Net is 98100 for convergence.

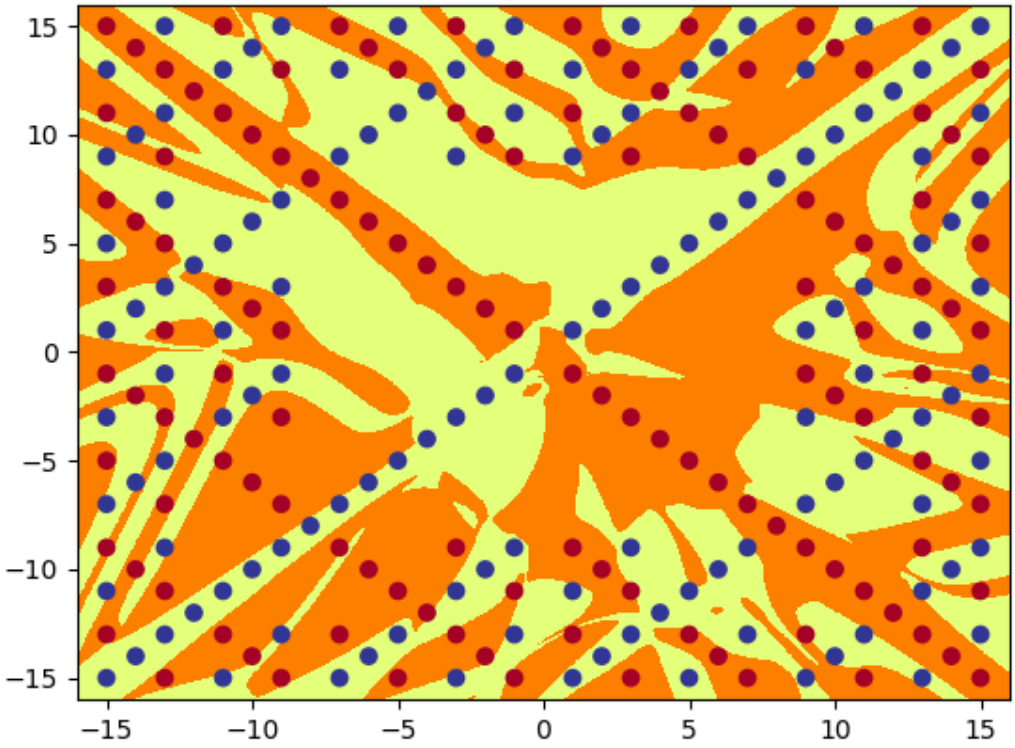


Figure 2. Full4Net Fractal Classification output when trained at 22 hid, fully trained at 70000 epochs.

To calculate the independent parameters, using the same equation from Full3Net,

$$\text{independent parameters} = \sum w_{in\to hid1} + \sum w_{hid1\to hid2} + \sum w_{hid2\to hid3} + \sum w_{hid3\to out} + \sum bias$$

$$= \sum_{i=0}^{1}\sum_{j=0}^{21} w_{ij} + \sum_{i=0}^{21}\sum_{j=0}^{21} w_{ij} + \sum_{i=0}^{21}\sum_{j=0}^{21} w_{ij} + \sum_{i=0}^{21} w_i + \ (22 + 22 + 22 + 1)$$

$$= (22 + 22) + (22 * 22) + (22 * 22) + 22 + 67$$

$$= 1101$$

Hence the number of independent parameters for Full4Net is 1101.

DenseNet

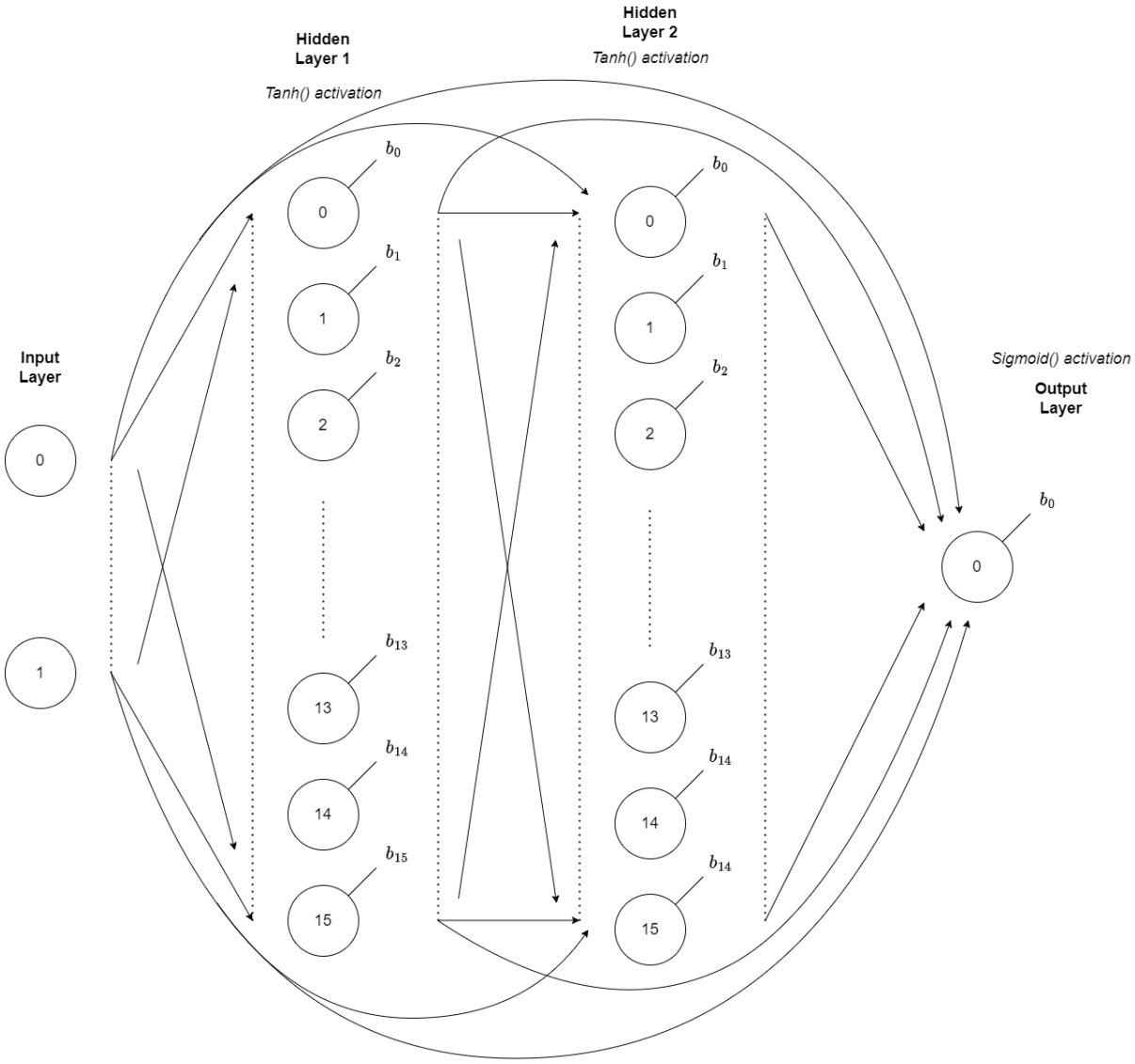


Figure 3. Diagram of the DenseNet Network

The minimum number of hidden nodes required for consistent convergence for DenseNet was 16 nodes. Again, hid as low as 15 converged but simply not consistent enough. The approximate epochs required to converge is calculated in Table 3 by calculating the average of 5 runs that converged.

DenseNet Training @ hid = 16

Testing run	Epoch
1	32000
2	86600
3	71500
4	123000
5	99400
Average	82500

Table 3. Calculation of approximate epochs required to train DenseNet at 16 hid

The approximate number of epochs required to converge at 16 hid is 82500.

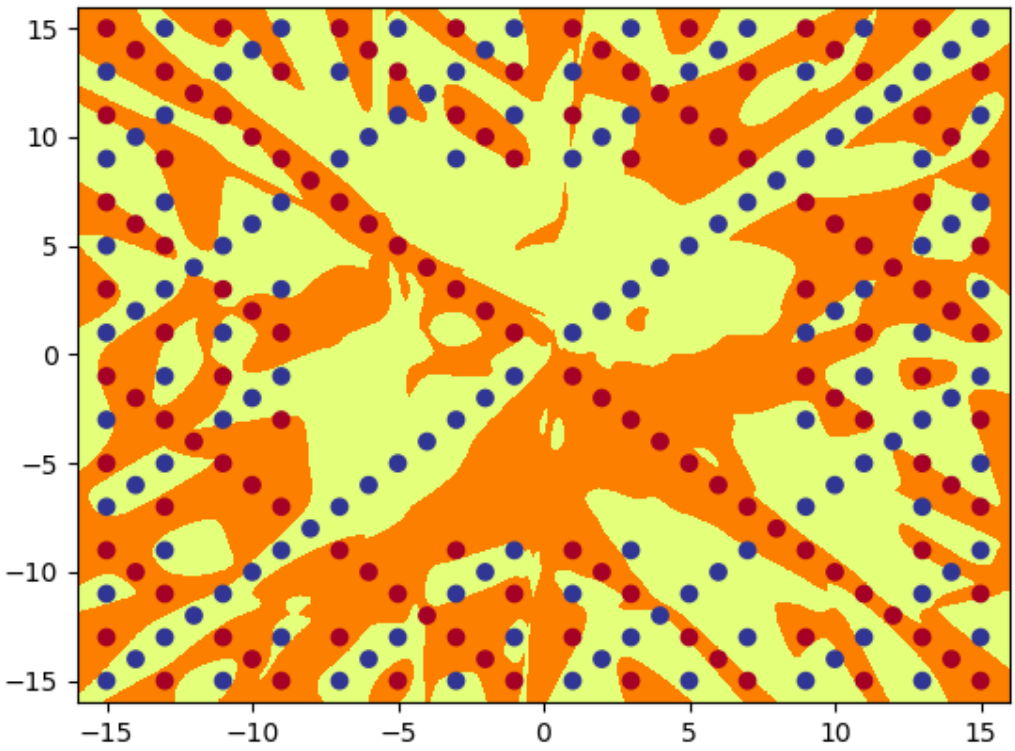


Figure 2. DenseNet Fractal Classification output when trained at 16 hid, fully trained at 86600 epochs.

The number of independent parameters in DenseNet is calculated by

$$\text{independent parameters} = \sum w_{in\to hid1} + \sum w_{hid1\to hid2} + \sum w_{hid2\to out}$$

$$+ \left(\sum w_{in\to hid2} + \sum w_{in\to out} + \sum w_{hid1\to out}\right) + \sum bias$$

$$= \sum_{i=0}^{1}\sum_{j=0}^{15} w_{ij} + \sum_{i=0}^{15}\sum_{j=0}^{15} w_{ij} + \sum_{i=0}^{15} w_i + \left(\sum_{i=0}^{1}\sum_{j=0}^{15} w_{ij} + \sum_{i=0}^{1} w_i + \sum_{i=0}^{15} w_i\right) + (16+16+1)$$

$$= (16+16) + (16*16) + (16) + \left((2*16) + (2) + (16)\right) + (33)$$

$$= 387$$

Discussion

a)

Network	Independent Parameters	Epochs
Full3Net	697	104100
Full4Net	1101	98100
DenseNet	378	82500

Table 4. Comparison of Independent Parameters against the required epochs for the 3 different networks

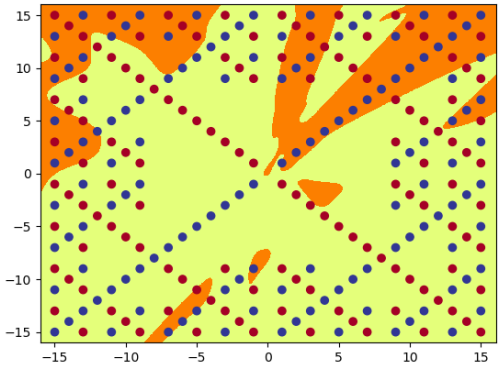
A collation of the independent parameters and required epochs for the three different networks is shown in Table 4. It can be seen that the number of independent parameters for DenseNet is the least which also requires the least epochs to converge. This may indicate a correlation between the number of Independent Parameters and epochs however it is difficult to validate this given that Full3Net and Full4Net violates the possible correlation.

Possible errors may come from the factors of small sampling size for epoch calculation, inclusion of outliers possibly due to stalling at a local minimum before converging again (case example test run 5 for Full4Net) as well as the stochastic nature of weight initialisation which will ensure different outcomes in every run. The first two factors can be attributed to lack of resources given the time constraints of this report and the last factor an expected source of error.

Another source may be due to the subjective definition of minimum number of hidden nodes which directly affects the number of independent parameters. Regardless, the lower number of epochs required for DenseNet showcases the effectiveness of shortcut connections in aiding faster training of networks.

b)

In Full4Net, all of the nodes in the first layer only separates the points into two sections by a straight line which to the human eye, is a seemingly random division. While the second layer exhibits more complex behaviour in the separation, majority of the points are still classified by separating the points into two distinct regions. The third layer quite complex behaviour, which is expected to be the most complex of the layers. The activations in these layers look to perform quite specific separations, for example,



This is the activation plot of Node 21 in layer 3. Its clear that majority of the points in the activation are blue points which indicates that this layer has a negative weight associated with it. Also opposed to layer 1 and layer 2, this layer performs quite specific classifications as opposed to the general classification of the other two layers.

In DenseNet, again the first layer is simply dividing the points into two regions in a straight line. This is expected behaviour as the input to the first hidden node is identical to a simple perceptron machine. That being said, the second hidden layer node is much more complex than the second hidden layer of Full4Net. Majority of the points are dividing and classifying points not just by dividing into two regions but by activating small pockets and groups throughout the plot. This shows the advantage of showing of shortcut connections in developing more complex activations in less amount of hidden layers.

c)

The outputs from the three functions, although classifying all points correctly, is classified through much different manners. The overall function of Full3Net is rather janky and spiky, in that there are several 'spikes' that are poking out at seemingly awkward directions as seen in figure 2. This classification oddities can be attributed to the network overfitting the points with its limited hidden layers, resulting in these distinctive marks.

The overall output from full4net, although still possessing some of the overfitting traits of spikes and sharp points as Full3Net, is definitely toned down and the division is much more rounded. Thanks to the extra layer of hidden nodes, the network is able to pick up on the intricacies of the division of points and is able to classify better.

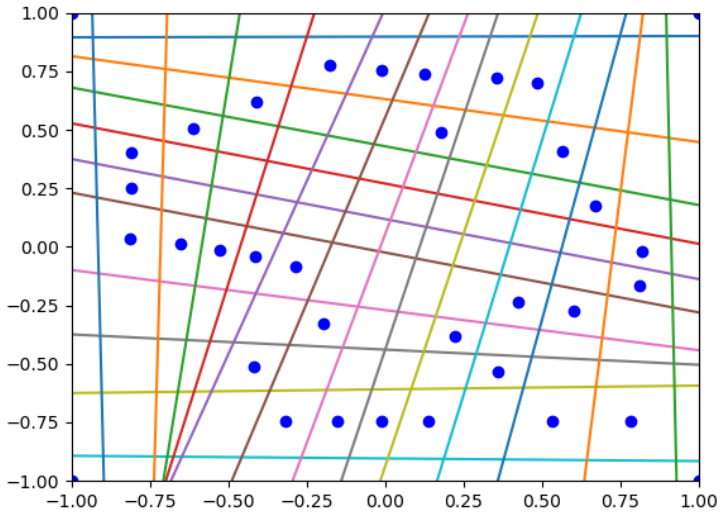
The output of DenseNet is the most rounded and exhibits almost none of the oddities and problems of both Full3Net and Full4Net, thanks to the shortcut connections. The division between the activated regions and non-activated regions are much smoother and not as jagged as the first two networks.

Encoder Networks

The ant35 matrix used:

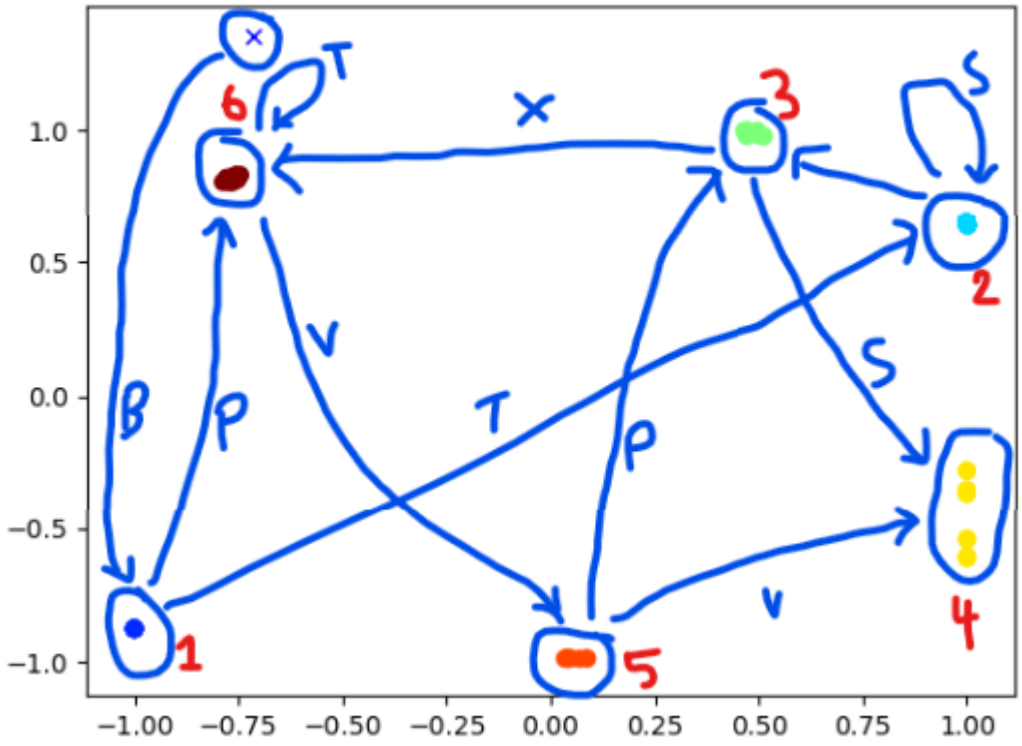
```
ant35 = torch.Tensor([
    [1,1,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1],
    [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,1],
    [1,0,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0],
    [1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0],
    [1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0],
    [1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0],
    [1,0,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,0,0,0,0,0],
    [1,1,0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0],
    [1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,0,0],
    [1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0],
    [1,1,1,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0],
    [1,1,1,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0],
    [1,1,1,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],
    [1,1,1,1,0,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,0,0],
    [1,1,1,1,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,0,0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1,0],
    [1,1,1,1,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0],
    [1,1,1,1,1,1,0,0,0,0,1,1,1,1,1,1,1,1,1,1,0,0,0],
    [1,1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,1,0,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0],
    [1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,1,1,0,0,1,1,1,1,1,1,1,1,1,0,0,0,0],
    [1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,0,0,0,0,0,0,0,0],
    [1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,0,0,0,0],
    [1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,0,0],
    [1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,0,0],
    [1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,0],
])
```

Where the first 4 arrays represent the corners. The resulting plot:

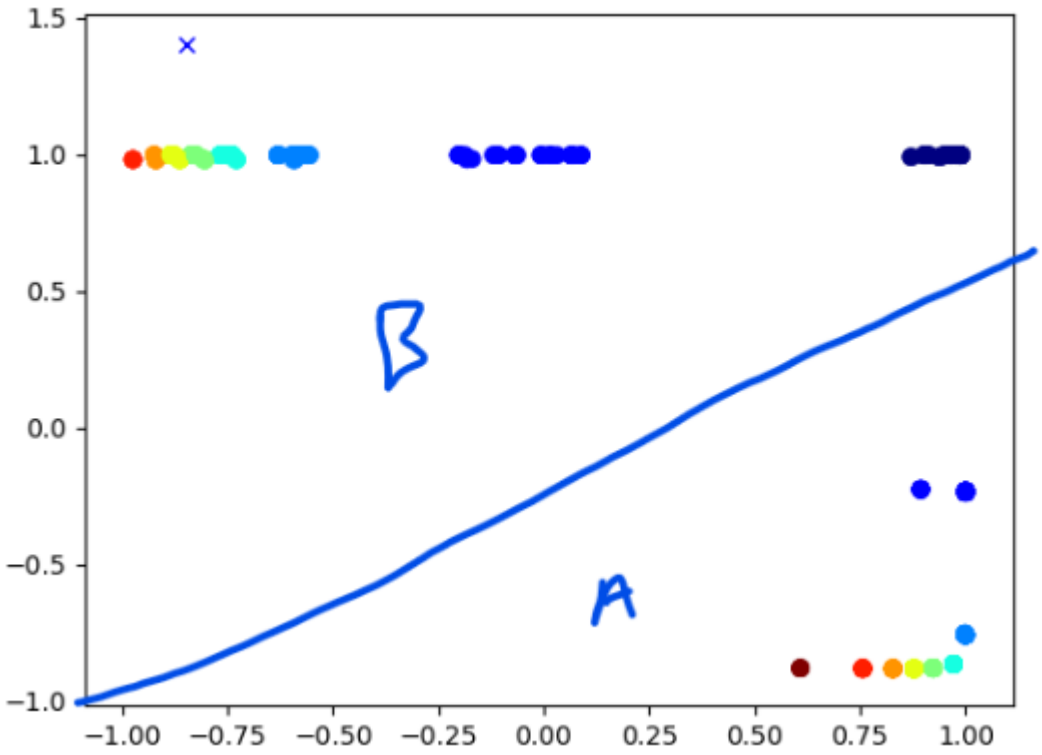


Hidden Unit Dynamics for Recurrent Networks

Annotated Reber plot



$a^n b^n$ plot



The $a^n b^n$ language requires that every sequence of consecutive a's must be followed by an equal number of consecutive b's. Looking at the terminal output,

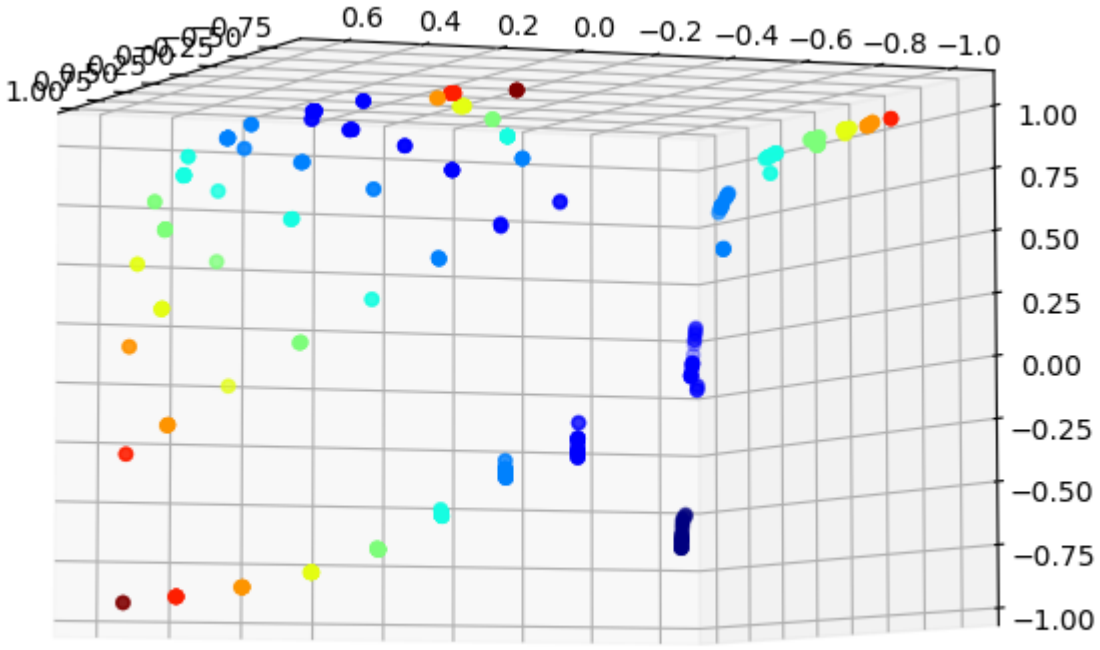
```
hidden activations and output probabi
A [ 0.89 -0.23] [0.84 0.16]
A [ 1.   -0.75] [0.86 0.14]
A [ 0.97 -0.86] [0.82 0.18]
B [ 0.92 -0.88] [0.75 0.25]
B [-0.73  0.98] [0. 1.]
B [-0.57  1.  ] [0. 1.]
B [0.03 1.  ] [0.02 0.98]
A [0.98 1.  ] [0.98 0.02]
A [ 1.   -0.23] [0.92 0.08]
A [ 1.   -0.76] [0.86 0.14]
A [ 0.97 -0.86] [0.82 0.18]
A [ 0.92 -0.88] [0.75 0.25]
B [ 0.88 -0.88] [0.67 0.33]
B [-0.8   0.98] [0. 1.]
B [-0.74  1.  ] [0. 1.]
B [-0.56  1.  ] [0. 1.]
B [0.07 1.  ] [0.03 0.97]
A [0.99 1.  ] [0.98 0.02]
A [ 1.   -0.24] [0.92 0.08]
A [ 1.   -0.76] [0.86 0.14]
A [ 0.97 -0.86] [0.82 0.18]
```

It classifies the probability of the points 'A' being in the lower right corner and the points 'B' being the long strand going across as seen in the annotated picture above.

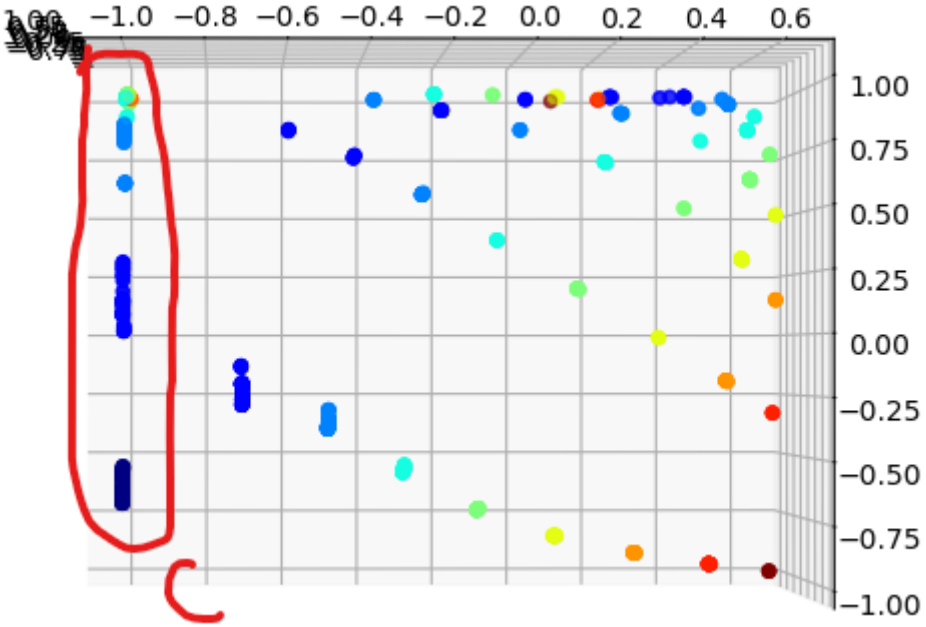
The network works by making use of two fixed points in the activation space, where one is attracting and one is repelling. The network counts up the a's as it oscillations, attracting towards the fixed point while the same would be done for b after all a is counted by counting down as it oscilates away from the repelling fixed point.

This can be seen in the diagram above as the colours in the A region go from blue to red, which indicates that the probability is increasing from blue to red. After the brown point, the network 'switches' to the B region and begins to count down from left to right of the plot before repeating the cycle again.

$a^n b^n c^n$ Plot

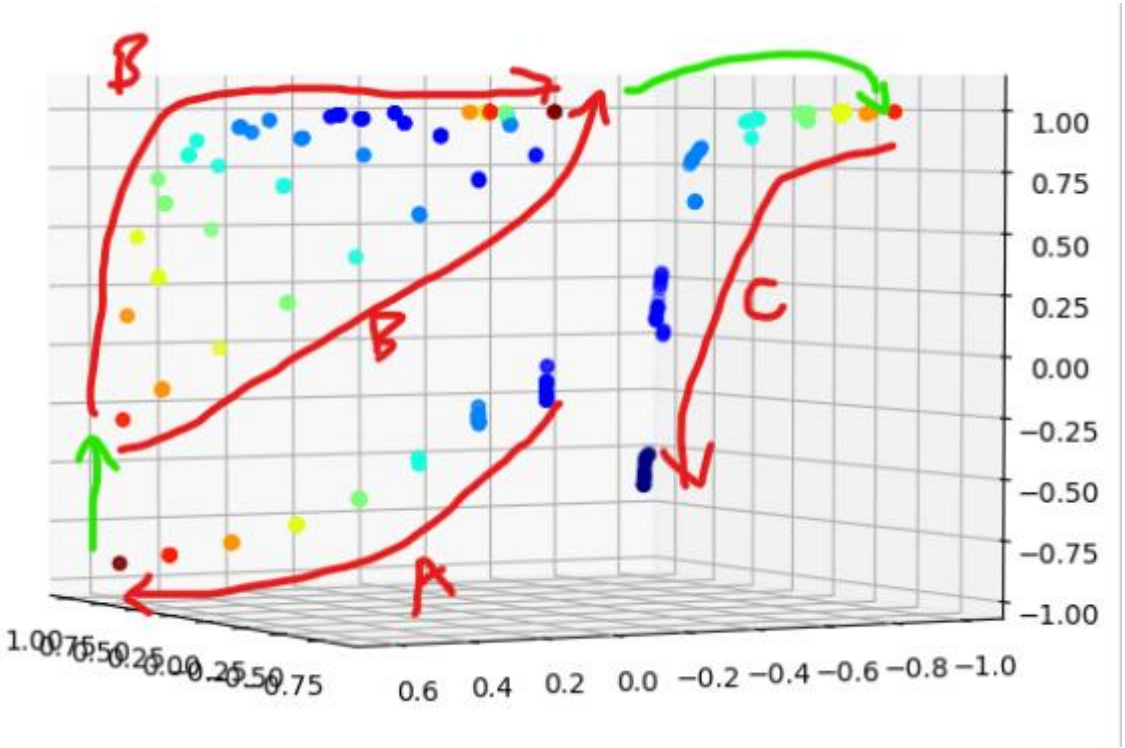


Extending upon the knowledge from the $a^n b^n$ case, the plot of $a^n b^n c^n$ looks to be separated into 3 distinct regions, two that look like a line and one that looks like a wing. As all three of these sections go from one part of the colour jet to another indication attraction and repelling from a fixed point, they can be separated into A, B and C regions.



```
B [-0.21  0.43  0.98] [0. 1. 0.]
C [-0.72  0.23  1.  ] [0.   0.01 0.99]
C [-0.43 -1.    0.98] [0. 0. 1.]
C [-0.26 -1.    0.99] [0. 0. 1.]
C [-0.02 -1.    0.98] [0. 0. 1.]
C [ 0.3  -1.    0.95] [0. 0. 1.]
C [ 0.65 -1.    0.77] [0. 0. 1.]
C [ 0.88 -1.    0.11] [0.   0.   0.99]
A [ 0.96 -1.   -0.64] [1. 0. 0.]
```

The two figures above show the plot in the y-z coordinate system and the output of C in the terminal output. The terminal shows that the y coordinate of the C points are all -1. This matches up with the line going down the value -1 and hence it has to be C. Similarly, the x-coordinates of A are all 1. Annotating the plot results in the figure below.

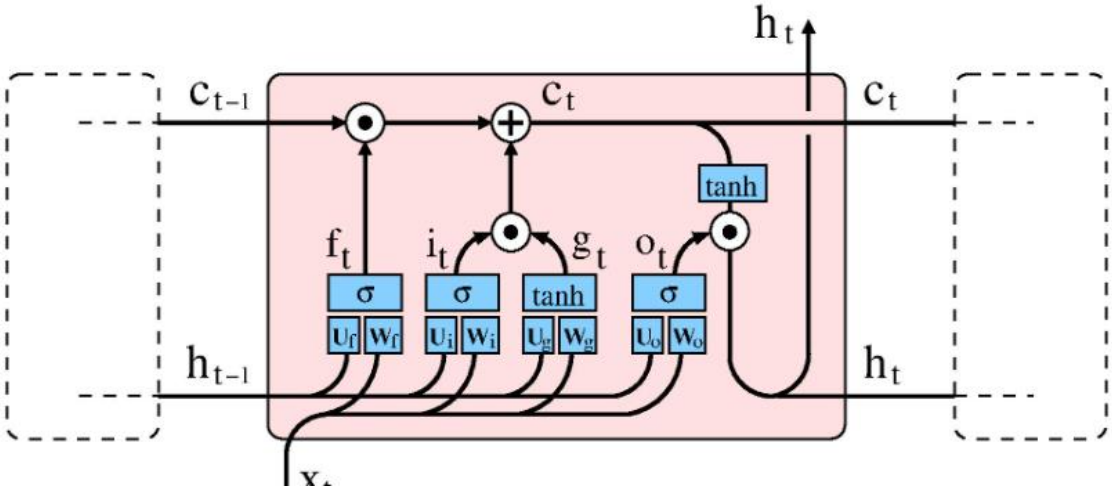


Again, similar to $a^n b^n$, the A region is attracted to the fixed point while B and C are repelled, resulting in the smooth cycle between the different regions.

LSTM

The advantage of using an LSTM network as opposed to an SRN network is the ability to retain information from a previous epoch and using that to help determine the output in the current epoch. This is helpful in situations such as a conversation where perhaps a pronoun used at the start impacts what pronoun needs to be used in the following sentences.

The basis of LSTM is to retain a 'context' layer on top of the hidden layer which will be sent to the next recursion in order to 'remember' information from the current layer. In order to determine how the system works, the context layer needs to be compared against the hidden layer.



The above figure shows a LSTM network works. Notice that the hidden node layer is simply the context layer with a tanh() activation and o_t added onto it. Printing out the context layer,

```
hidden activations and context and output probabilities [BTSXPVE]:
1 [ 0.69 -0.   -0.02 -0.44] [ 0.93 -0.   -0.03 -0.67] [ 0.1   0.29  0.    0.    0.57  0.03  0.  ]
2 [ 0.48  0.47  0.01  0.06] [ 0.55  0.65  0.02  0.07] [ 0.6   0.16  0.03  0.03  0.08  0.1   0.  ]
3 [ 0.84  0.65 -0.01 -0.62] [ 1.29  0.97 -0.01 -0.82] [ 0.02  0.56  0.    0.    0.35  0.06  0.  ]
4 [-0.07  0.84 -0.06 -0.11] [-0.07  1.79 -0.09 -0.12] [ 0.01  0.07  0.42  0.32  0.02  0.13  0.03]
4 [-0.67  0.71 -0.25 -0.43] [-0.84  1.82 -0.39 -0.51] [ 0.    0.    0.42  0.39  0.    0.02  0.17]
5 [-0.26  0.73 -0.11 -0.06] [-0.29  2.12 -0.76 -0.13] [ 0.    0.04  0.43  0.32  0.01  0.12  0.08]
8 [ 0.11  0.71 -0.2   0.04] [ 0.13  2.4  -1.07  0.06] [ 0.01  0.19  0.21  0.15  0.03  0.38  0.04]
7 [ 0.28  0.78 -0.69 -0.6 ] [ 0.3   1.96 -1.51 -0.75] [ 0.    0.43  0.02  0.02  0.24  0.22  0.07]
6 [-0.62  0.86 -0.77 -0.88] [-0.73  2.14 -2.09 -1.45] [ 0.    0.01  0.03  0.05  0.02  0.02  0.87]
9 [ 0.57  0.78 -0.9  -0.95] [ 0.65  1.62 -2.89 -2.21] [ 0.    0.31  0.    0.    0.64  0.03  0.03]
18 [-0.7   0.88 -0.7  -0.82] [-0.87  2.41 -3.18 -1.17] [ 0.    0.    0.01  0.02  0.01  0.01  0.95]
epoch: 3000
```

Where the second column is the context layer, it can be seen that the context layer is simply a linear transformation of the hidden layer.

```
18 [ 0.  0.  0.  0.  0.  0.  1.]
hidden activations and context and output probabilities [BTSXPVE]:
1 [-0.74 -0.61 -0.37 -0.45] [-0.99 -0.83 -0.4  -0.52] [ 0.    0.44  0.    0.    0.55  0.    0.  ]
10 [-0.88  0.58  0.66 -0.12] [-1.45  0.8   0.8  -0.29] [ 1.  0.  0.  0.  0.  0.  0.]
11 [-0.92 -0.35  0.    0.4 ] [-2.1  -0.65  0.    0.47] [ 0.    0.48  0.    0.    0.51  0.01  0.  ]
16 [-0.36 -0.17  0.64  0.27] [-0.38 -0.21  0.77  0.86] [ 0.    0.42  0.    0.    0.01  0.57  0.  ]
15 [-0.45  0.15 -0.4   0.84] [-0.54  0.61 -0.43  1.68] [ 0.    0.01  0.    0.    0.41  0.58  0.  ]
14 [ 0.72 -0.32 -0.79  0.68] [ 0.93 -0.59 -1.2   1.13] [ 0.  0.  0.  0.  0.  0.  1.]
17 [-0.7  -0.9  -0.09  0.89] [-0.88 -1.51 -0.95  1.46] [ 0.    0.39  0.    0.    0.6   0.    0.  ]
18 [ 0.57 -0.61 -0.61  0.38] [ 0.65 -0.86 -0.71  1.63] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 17000
18 [ 0.  0.  0.  0.  0.  0.  1.]
hidden activations and context and output probabilities [BTSXPVE]:
1 [-0.74 -0.62 -0.34 -0.45] [-0.99 -0.86 -0.37 -0.53] [ 0.    0.49  0.    0.    0.5   0.    0.  ]
2 [-0.48  0.36  0.58 -0.22] [-0.85  0.66  0.68 -1.23] [ 1.  0.  0.  0.  0.  0.  0.]
3 [-0.9  -0.42 -0.09  0.4 ] [-1.7  -0.65 -0.09  0.45] [ 0.    0.46  0.    0.    0.53  0.    0.  ]
4 [ 0.52 -0.27  0.03 -0.13] [ 0.67 -0.55  0.03 -0.58] [ 0.    0.    0.44  0.55  0.    0.    0.  ]
4 [ 0.83 -0.72 -0.04 -0.43] [ 1.42 -1.19 -0.04 -0.62] [ 0.    0.    0.43  0.56  0.    0.    0.  ]
4 [ 0.86 -0.8   0.1  -0.5 ] [ 1.82 -1.42  0.12 -0.72] [ 0.    0.    0.45  0.55  0.    0.    0.  ]
5 [ 0.65 -0.29  0.72 -0.03] [ 0.84 -0.41  0.94 -0.07] [ 0.    0.    0.56  0.43  0.    0.    0.  ]
6 [ 0.7  -0.56 -0.44  0.55] [ 1.03 -0.95 -0.52  0.8 ] [ 0.  0.  0.  0.  0.  0.  1.]
9 [-0.73 -0.93 -0.14  0.91] [-0.96 -1.78 -0.95  1.63] [ 0.    0.44  0.    0.    0.55  0.    0.  ]
18 [ 0.68 -0.48 -0.92  0.11] [ 0.89 -1.08 -1.61  0.42] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 26000
```

```
hidden activations and context and output probabilities [BTSXPVE]:
1 [-0.74 -0.64 -0.31 -0.48] [-0.99 -0.88 -0.33 -0.57] [ 0.    0.52  0.    0.    0.48  0.    0.  ]
10 [-0.87  0.56  0.62 -0.06] [-1.62  0.75  0.74 -0.27] [ 1.  0.  0.  0.  0.  0.  0.]
11 [-0.96 -0.42  0.03  0.27] [-2.32 -0.79  0.03  0.3 ] [ 0.    0.49  0.    0.    0.5   0.    0.  ]
16 [-0.82  0.04  0.69  0.19] [-1.2   0.05  0.86  0.74] [ 0.    0.47  0.    0.    0.02  0.51  0.  ]
15 [-0.65  0.26 -0.27  0.85] [-0.92  0.84 -0.27  1.48] [ 0.    0.    0.    0.    0.39  0.61  0.  ]
13 [ 0.66 -0.59  0.55  0.02] [ 0.79 -0.77  0.63  0.07] [ 0.    0.01  0.54  0.45  0.    0.    0.  ]
16 [-0.18 -0.21  0.68  0.27] [-0.2  -0.36  0.84  1.  ] [ 0.    0.36  0.    0.    0.    0.64  0.  ]
16 [-0.24 -0.17  0.63  0.29] [-0.44 -0.4   0.74  1.07] [ 0.    0.4   0.    0.    0.01  0.59  0.  ]
15 [-0.57  0.33 -0.43  0.9 ] [-0.72  0.73 -0.47  1.86] [ 0.    0.    0.    0.    0.44  0.56  0.  ]
14 [ 0.73 -0.49 -0.83  0.22] [ 0.94 -0.69 -1.28  0.25] [ 0.  0.  0.  0.  0.  1.]
17 [-0.6  -0.92 -0.13  0.77] [-0.76 -1.63 -0.45  1.06] [ 0.    0.6   0.    0.    0.39  0.    0.  ]
18 [ 0.43 -0.62 -0.38  0.27] [ 0.47 -0.83 -0.4   1.22] [ 0.    0.    0.    0.    0.    0.    0.99]
epoch: 36000
error: 0.0019
final: 0.1047
```

```
18 [ 0.  0.  0.  0.  0.  0.  1.]
hidden activations and context and output probabilities [BTSXPVE]:
1 [-0.74 -0.64 -0.27 -0.5 ] [-0.99 -0.87 -0.29 -0.59] [ 0.    0.44  0.    0.    0.56  0.    0.  ]
10 [-0.87  0.51  0.63 -0.05] [-1.57  0.68  0.76 -0.31] [ 1.  0.  0.  0.  0.  0.  0.]
11 [-0.97 -0.47  0.08  0.31] [-2.4  -0.77  0.08  0.35] [ 0.    0.43  0.    0.    0.57  0.    0.  ]
16 [-0.85  0.06  0.74  0.16] [-1.29  0.07  0.95  0.84] [ 0.    0.61  0.    0.    0.01  0.38  0.  ]
15 [-0.78  0.35 -0.19  0.87] [-1.28  0.89 -0.19  1.58] [ 0.    0.    0.    0.    0.45  0.55  0.  ]
13 [ 0.63 -0.59  0.62  0.02] [ 0.74 -0.76  0.74  0.1 ] [ 0.    0.03  0.58  0.39  0.    0.    0.  ]
16 [-0.3  -0.14  0.78  0.23] [-0.33 -0.25  1.05  1.05] [ 0.    0.41  0.    0.    0.    0.59  0.  ]
15 [-0.65  0.38 -0.27  0.91] [-0.92  0.81 -0.27  1.89] [ 0.    0.    0.    0.    0.28  0.72  0.  ]
14 [ 0.73 -0.48 -0.82  0.45] [ 0.94 -0.62 -1.17  0.52] [ 0.  0.  0.  0.  0.  0.  1.]
17 [-0.63 -0.92 -0.61  0.88] [-0.78 -1.59 -0.76  1.45] [ 0.    0.03  0.    0.    0.97  0.    0.  ]
18 [ 0.71 -0.87 -0.42  0.11] [ 0.9  -1.5  -0.45  0.47] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 46000
error: 0.0042
final: 0.0002
```

```
18 [ 0.  0.  0.  0.  0.  0.  1.]
hidden activations and context and output probabilities [BTSXPVE]:
1 [-0.74 -0.65 -0.25 -0.5 ] [-0.99 -0.87 -0.26 -0.59] [ 0.    0.48  0.    0.    0.52  0.    0.  ]
10 [-0.87  0.51  0.64 -0.04] [-1.55  0.68  0.77 -0.24] [ 1.  0.  0.  0.  0.  0.  0.]
11 [-0.97 -0.49  0.12  0.35] [-2.39 -0.77  0.13  0.39] [ 0.    0.46  0.    0.    0.54  0.    0.  ]
12 [ 0.37  0.03  0.09 -0.09] [ 0.66  0.04  0.09 -0.56] [ 0.    0.    0.47  0.52  0.    0.01  0.  ]
13 [ 0.65 -0.18  0.75 -0.01] [ 0.81 -0.28  0.98 -0.05] [ 0.    0.01  0.59  0.39  0.    0.01  0.  ]
14 [ 0.53 -0.34 -0.52  0.43] [ 0.63 -0.64 -0.6   0.6 ] [ 0.  0.  0.  0.  0.  0.  1.]
17 [-0.66 -0.91 -0.52  0.9 ] [-0.84 -1.59 -0.59  1.56] [ 0.    0.03  0.    0.    0.97  0.    0.  ]
18 [ 0.72 -0.84 -0.31  0.17] [ 0.91 -1.38 -0.32  0.73] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 50000
error: 0.0012
final: 0.0003
```

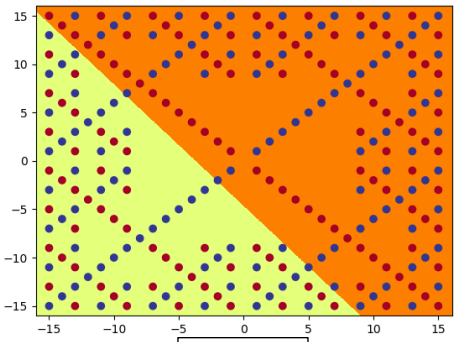
In order to test LSTM, a value preserved throughout many timesteps needs to be identified which will show that the T or P value is remembered from one timestep to the next. The screenshots above show the context every 10000 or so epochs. This value can be identified to be the first node in the first layer of the context layer as throughout the entirety of its runtime to 50000 epoch it has stayed consistent at -0.99. This shows that the value must have a low value for the forget gate and a high negative value for the output gate, which is fed back into the next timestep and causing a feedback loop of a negative number close to -1 for 50000 epochs.

This demonstrates long term memory capability of an LSTM network, a major advantage over a simple RSM network in many different tasks.

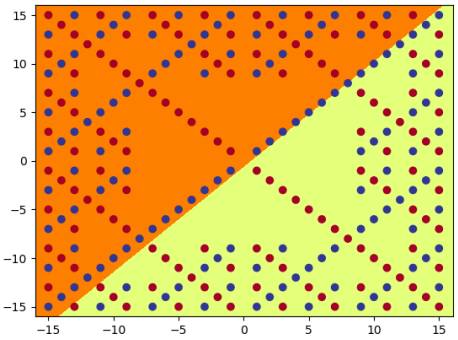
APPENDIX

Hidden Full4Net Plots

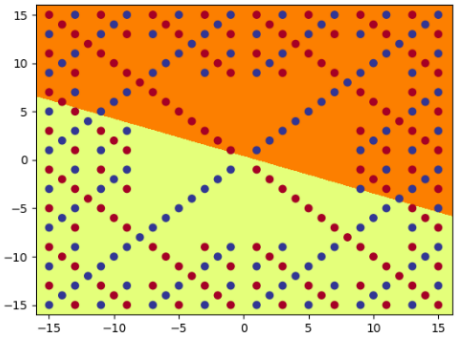
Layer 1



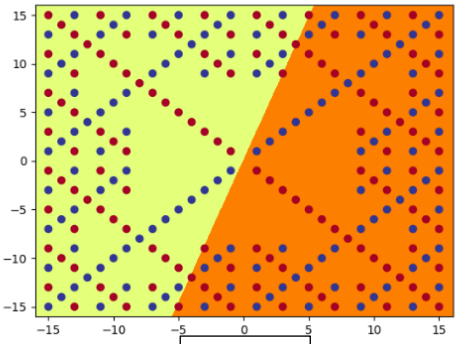
Node 0



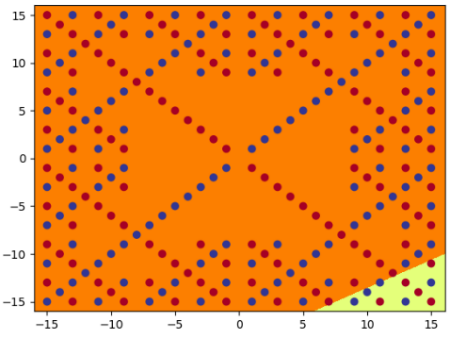
Node 1



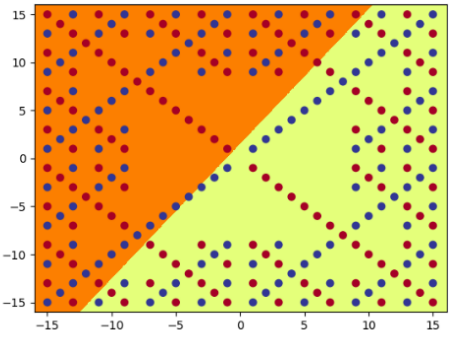
Node 2



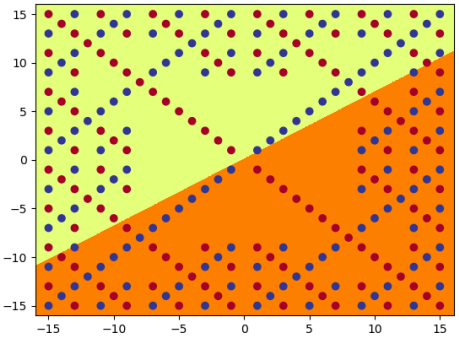
Node 3



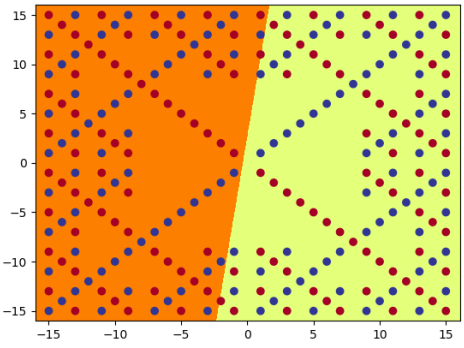
Node 4



Node 5

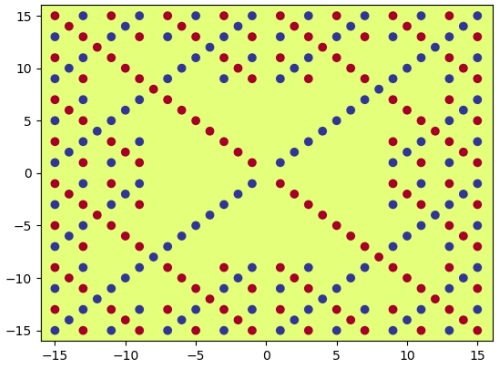


Node 6

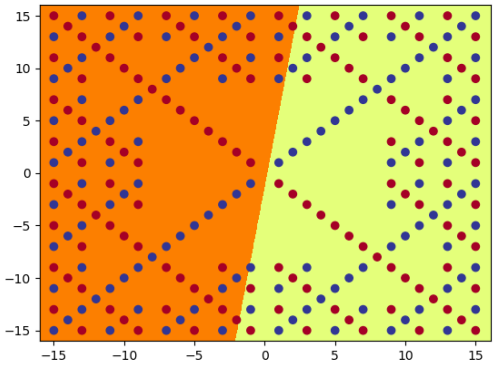


Node 7

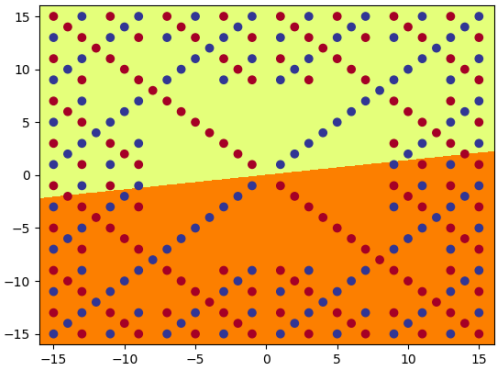
APPENDIX



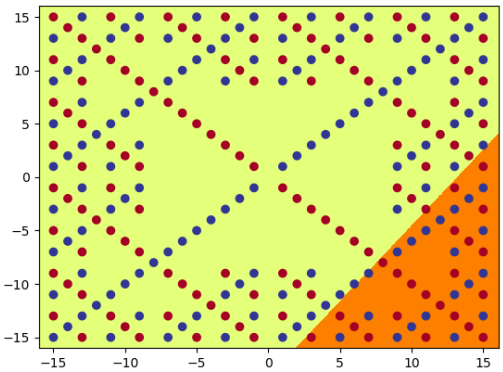
Node 8



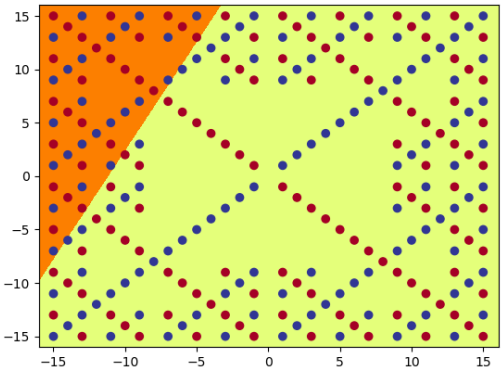
Node 9



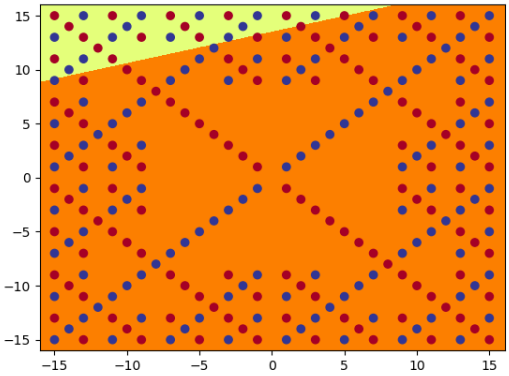
Node 10



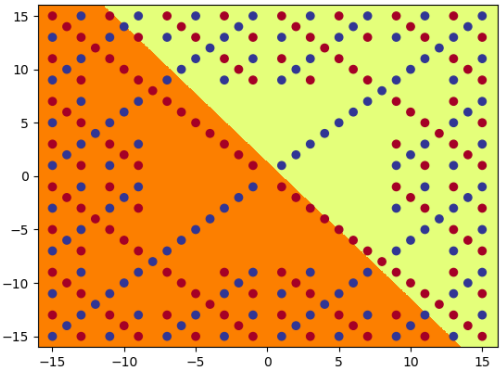
Node 11



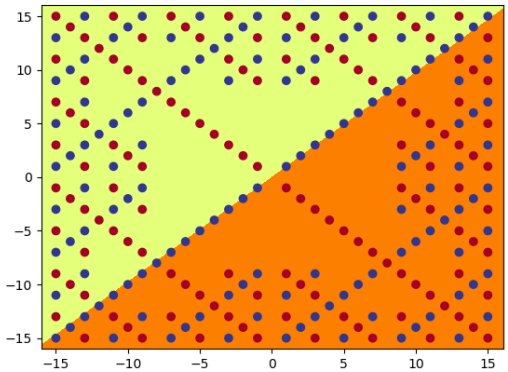
Node 12



Node 13

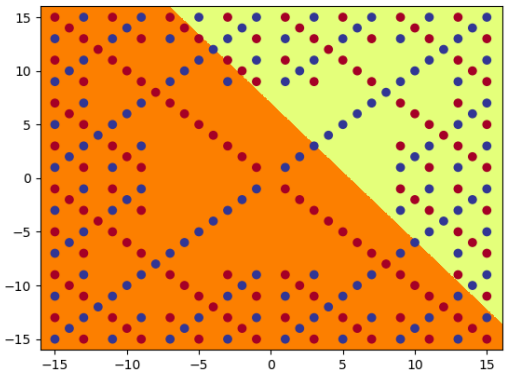


Node 14

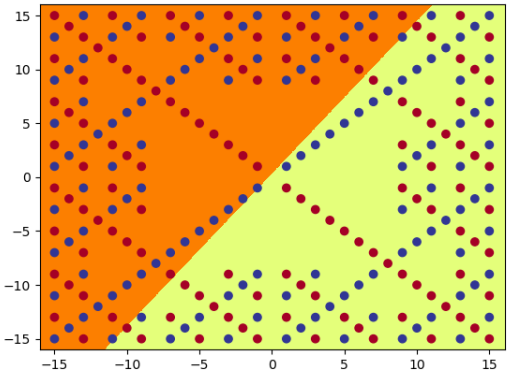


Node 15

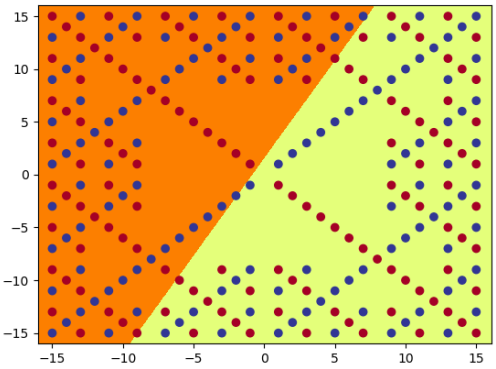
APPENDIX



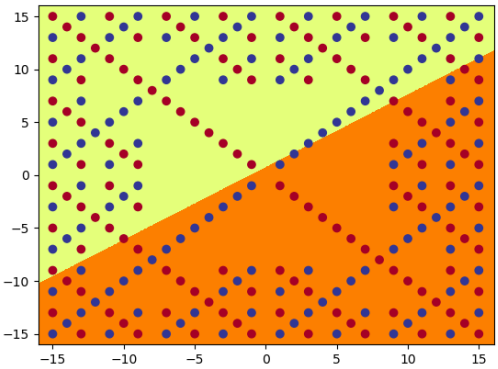
Node 16



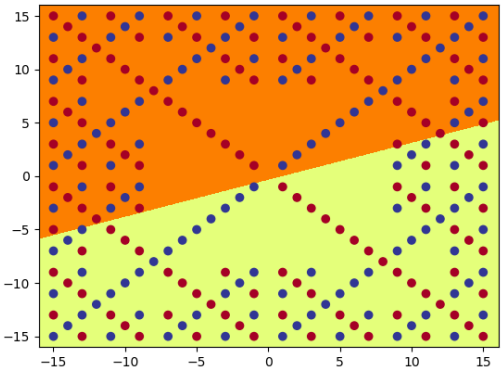
Node 17



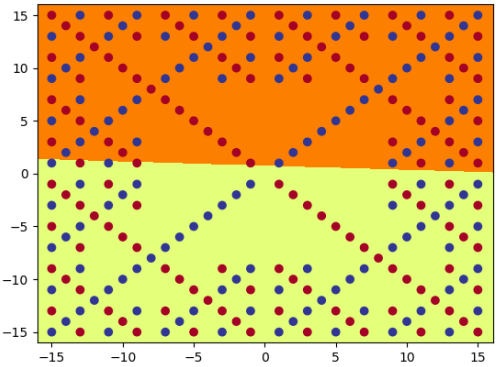
Node 18



Node 19

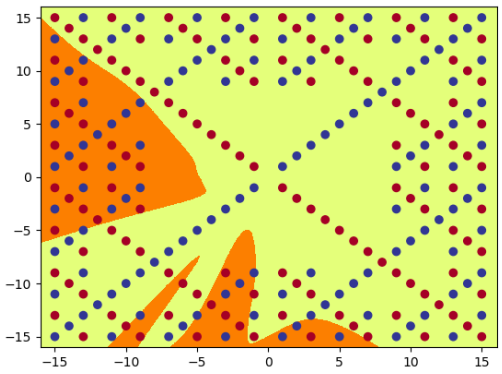


Node 20

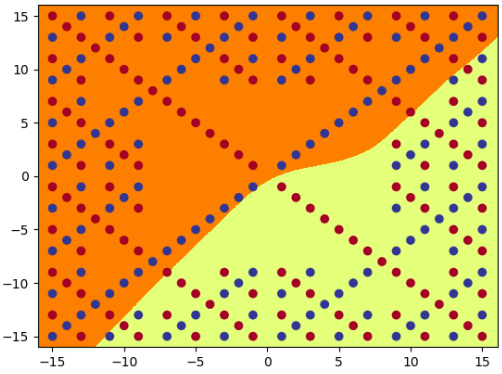


Node 21

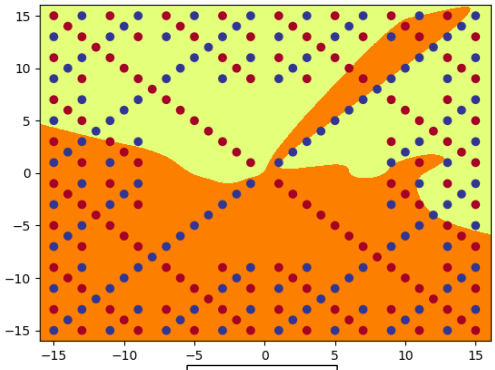
Layer 2



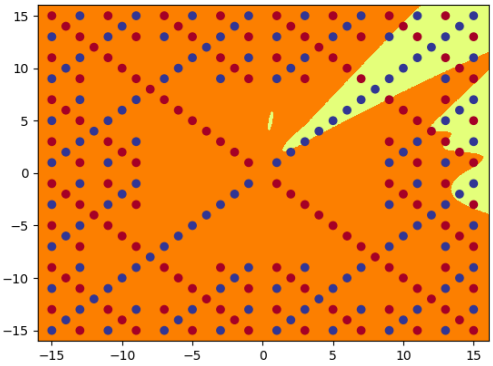
Node 0



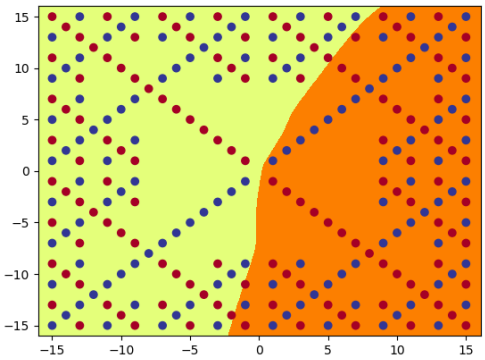
Node 1



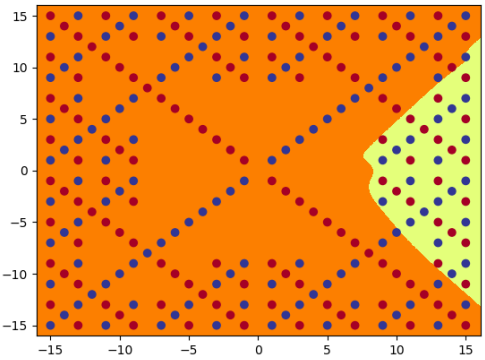
Node 2



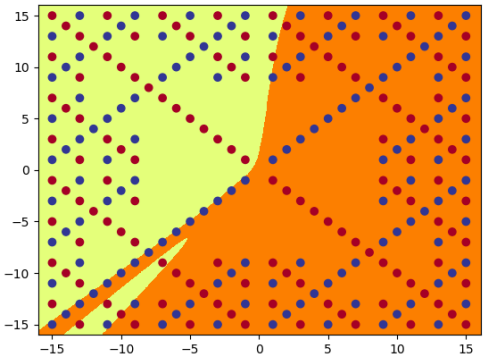
Node 3



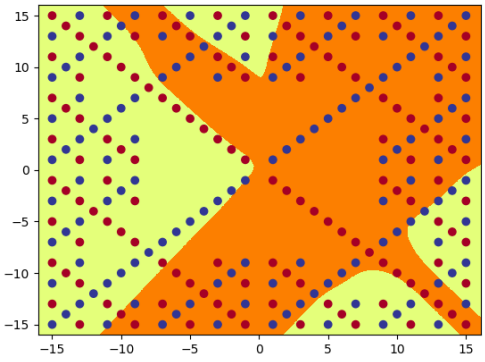
Node 4



Node 5

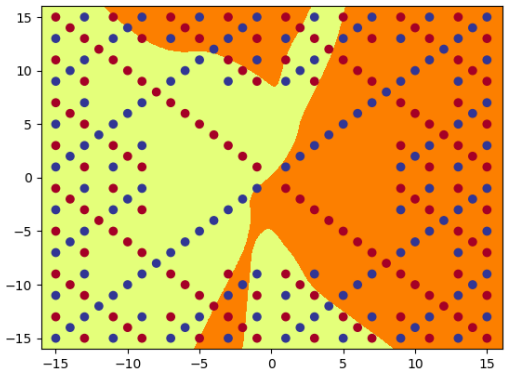


Node 6

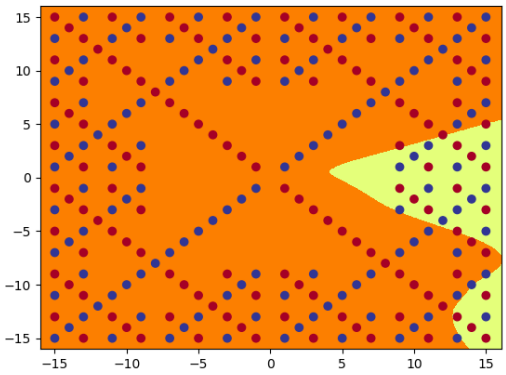


Node 7

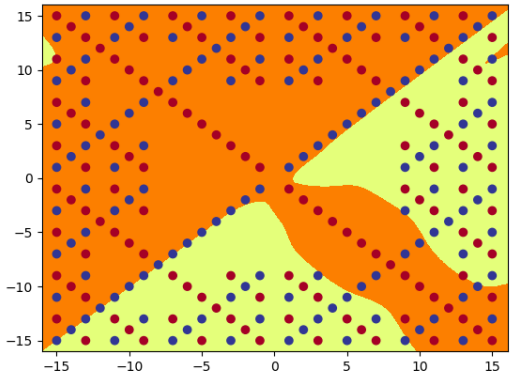
APPENDIX



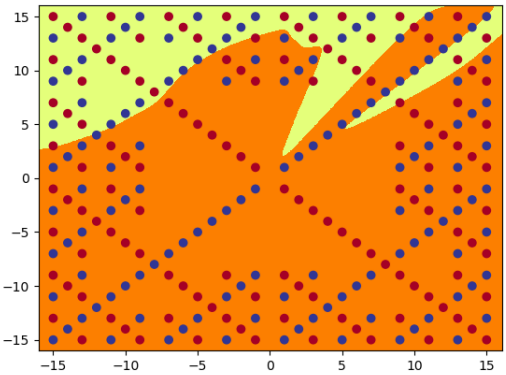
Node 8



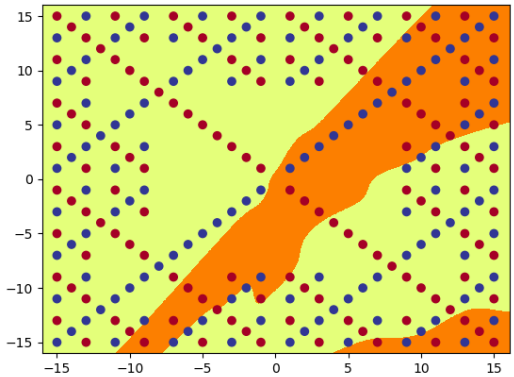
Node 9



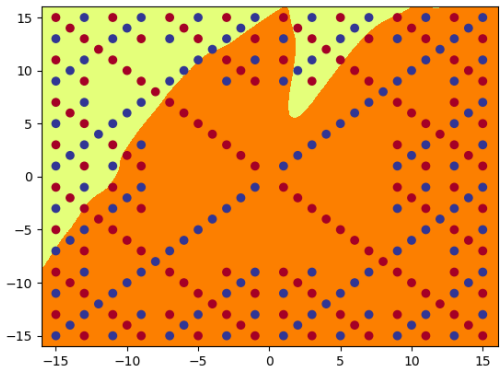
Node 10



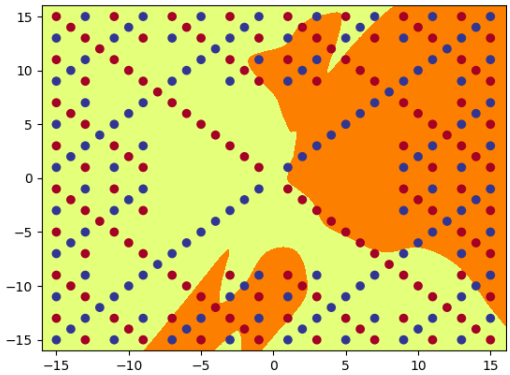
Node 11



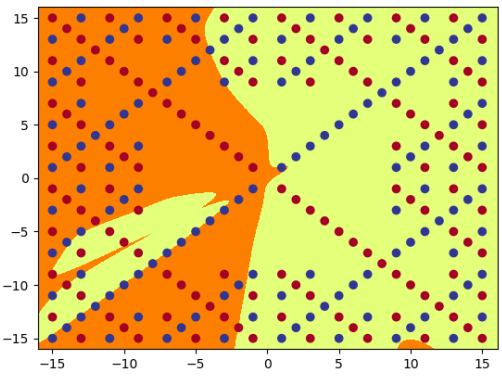
Node 12



Node 13

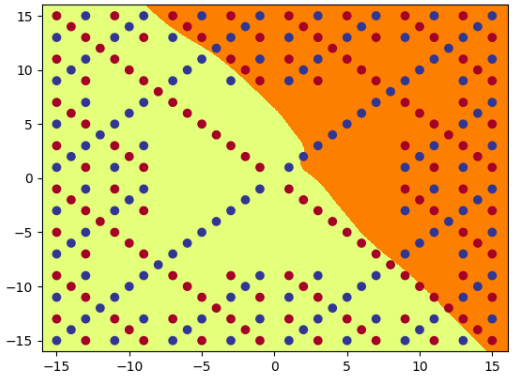


Node 14

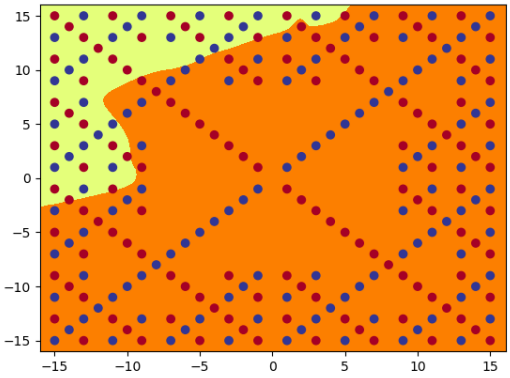


Node 15

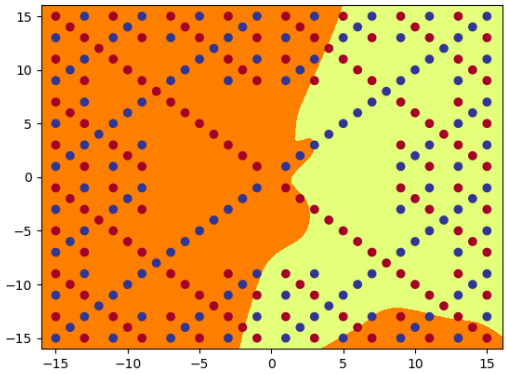
APPENDIX



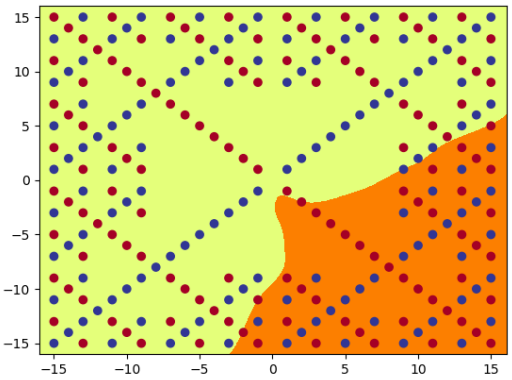
Node 16



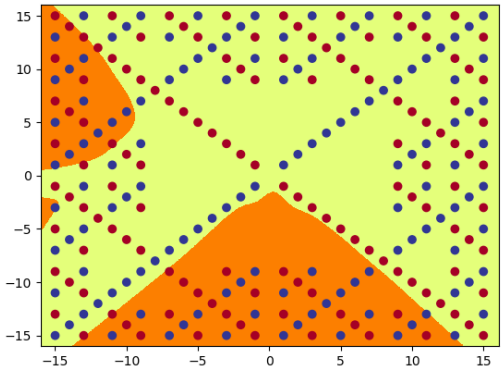
Node 17



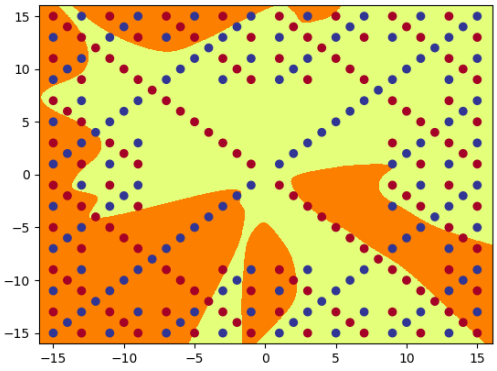
Node 18



Node 19

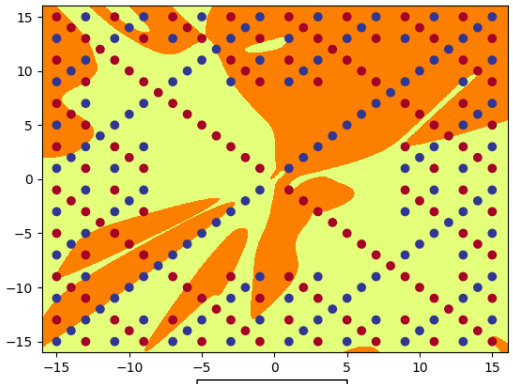


Node 20

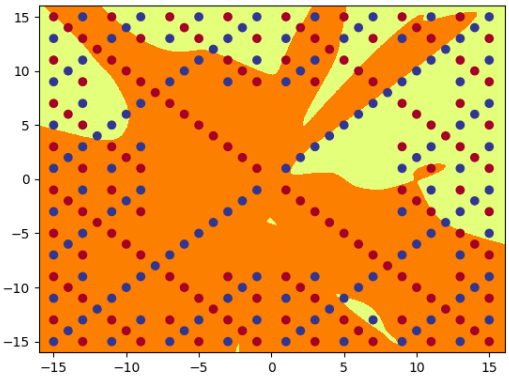


Node 21

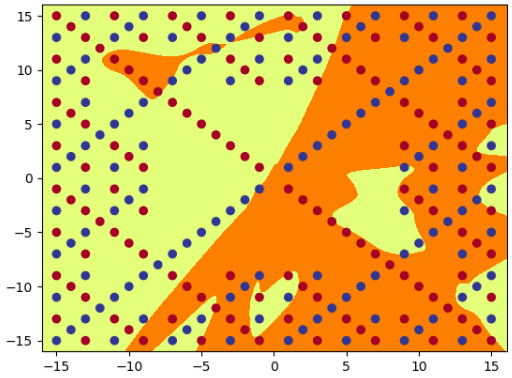
Layer 3



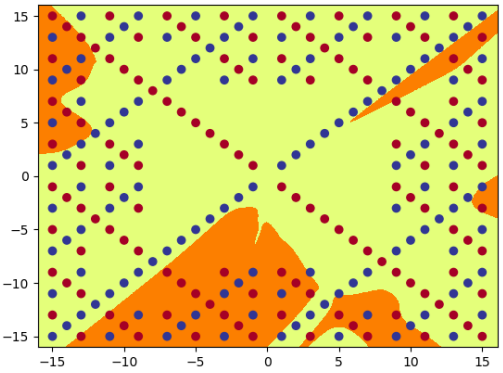
Node 0



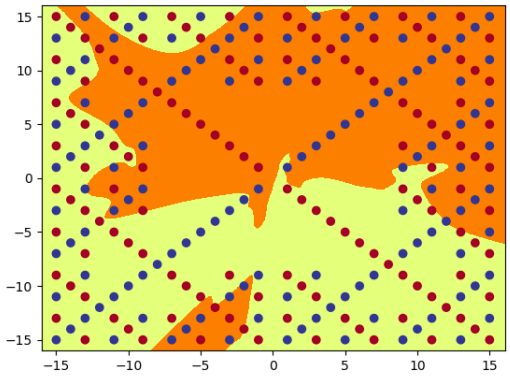
Node 1



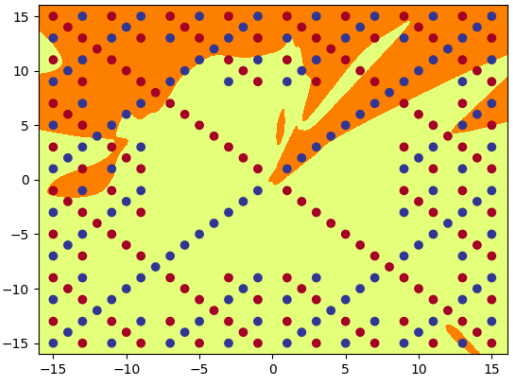
Node 2



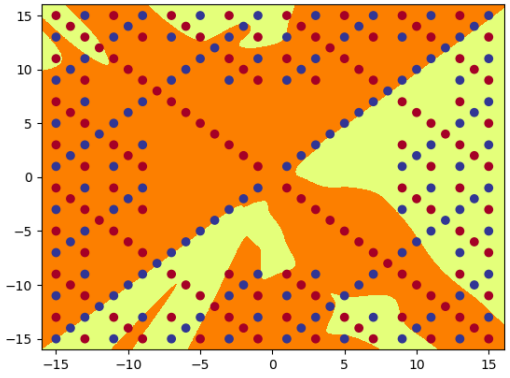
Node 3



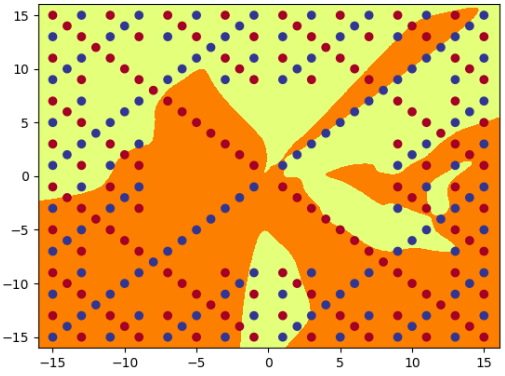
Node 4



Node 5

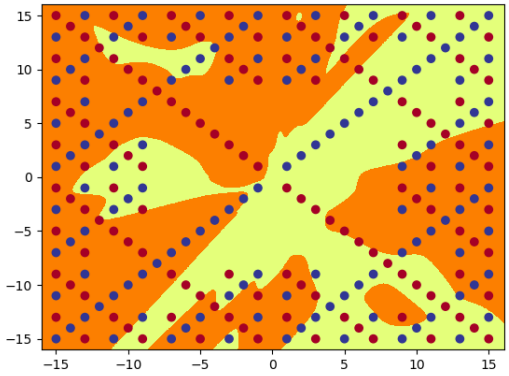


Node 6

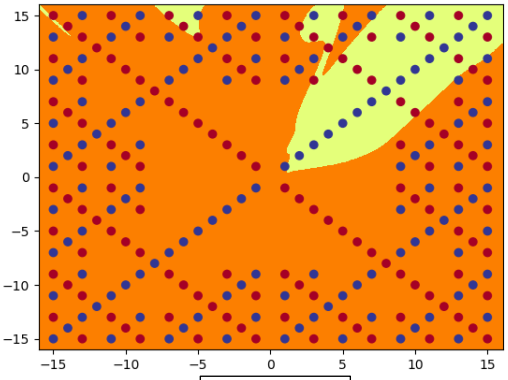


Node 7

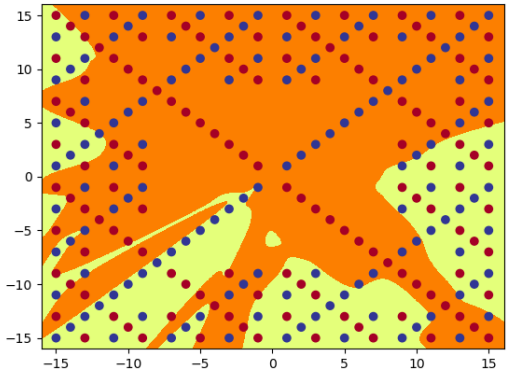
APPENDIX



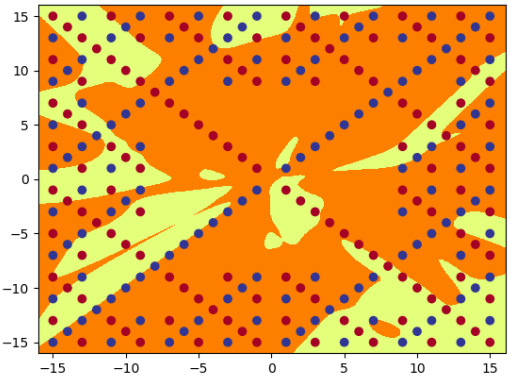
Node 8



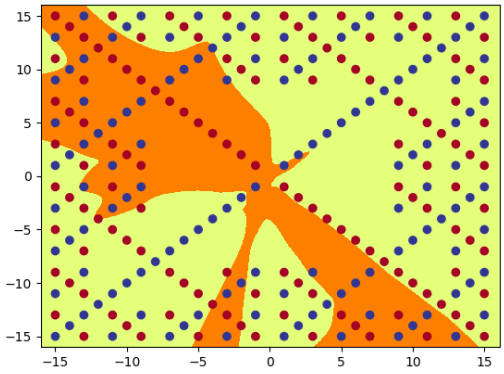
Node 9



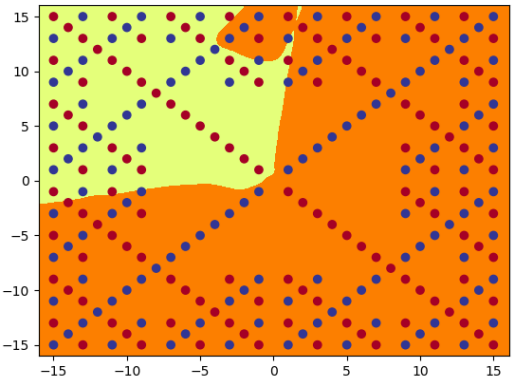
Node 10



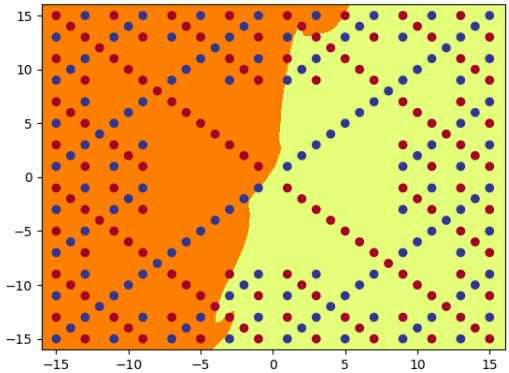
Node 11



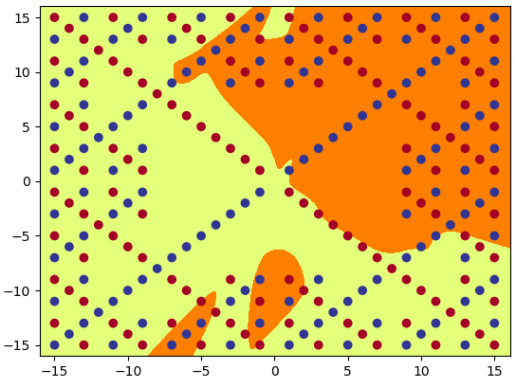
Node 12



Node 13

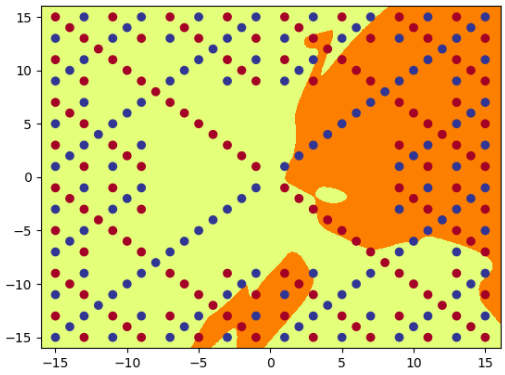


Node 14

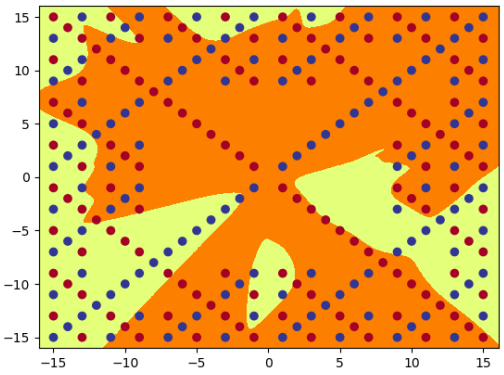


Node 15

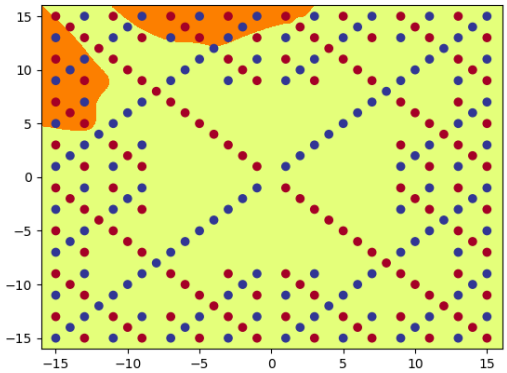
APPENDIX



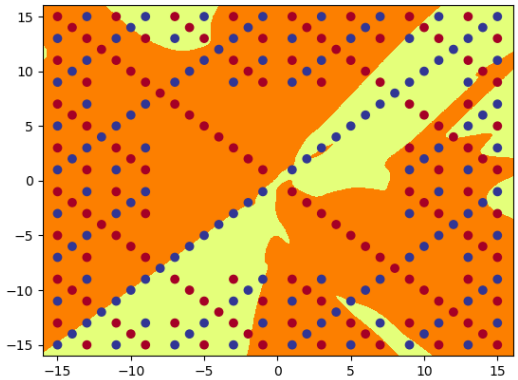
Node 16



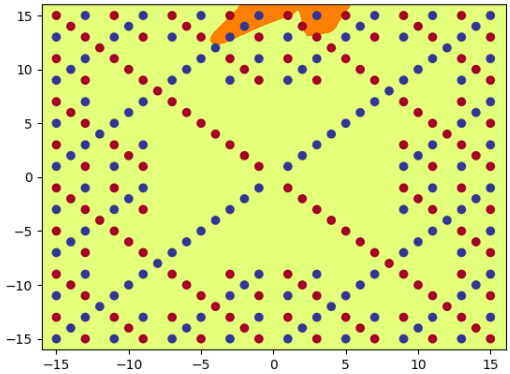
Node 17



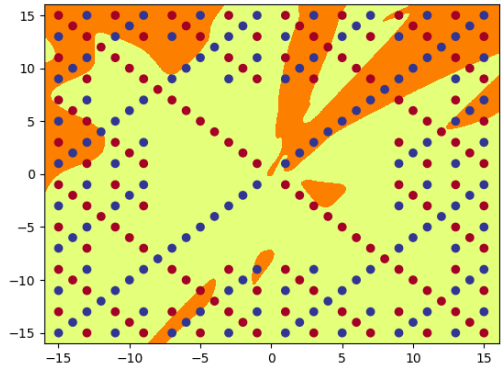
Node 18



Node 19



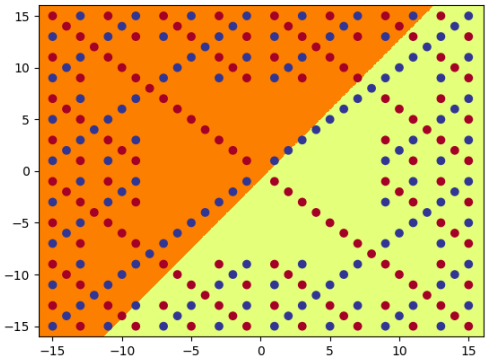
Node 20



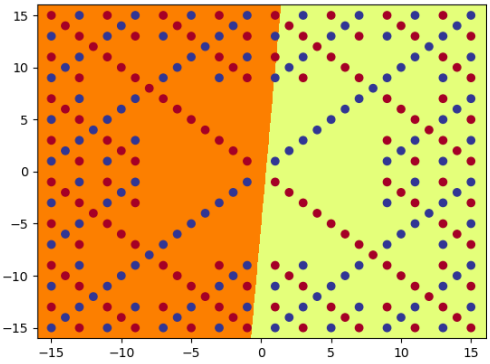
Node 21

Hidden DenseNet Plots

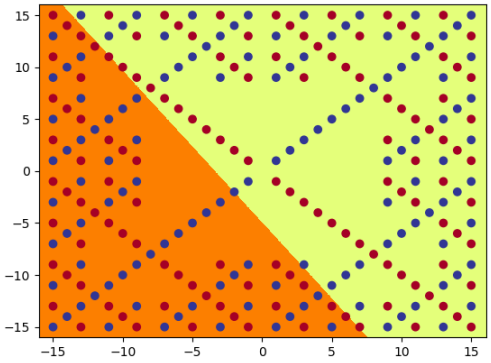
Layer 1



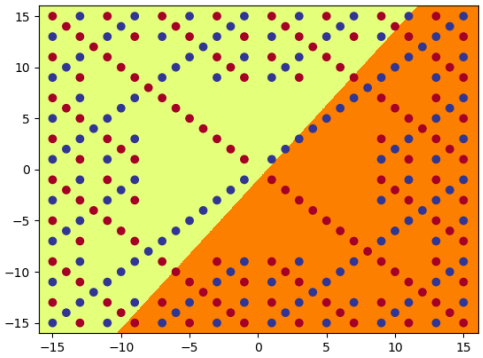
Node 0



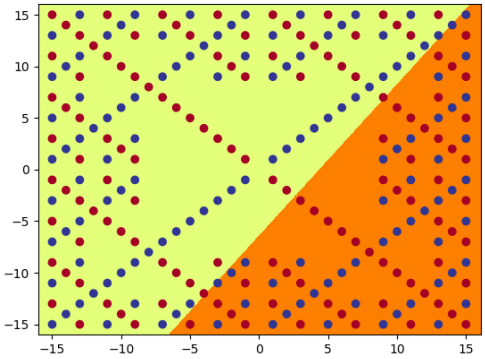
Node 1



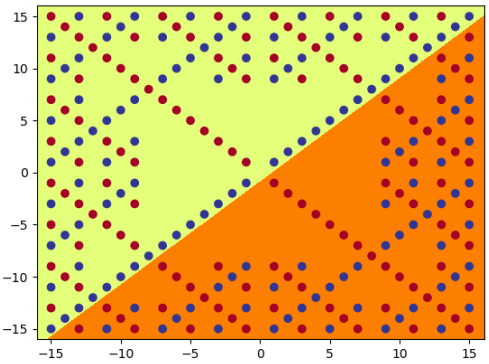
Node 2



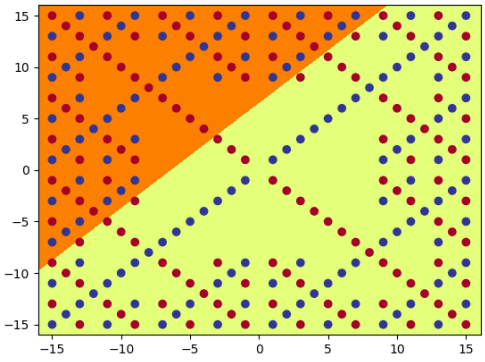
Node 3



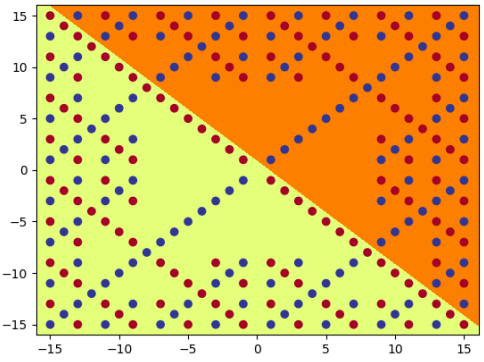
Node 4



Node 5

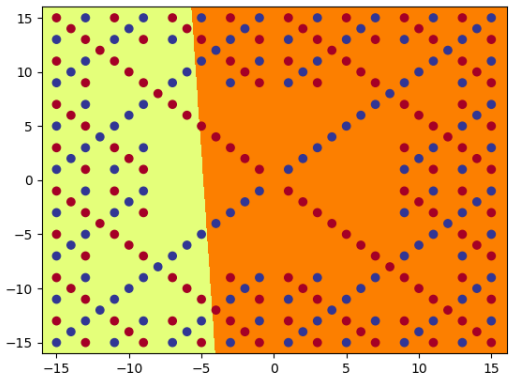


Node 6

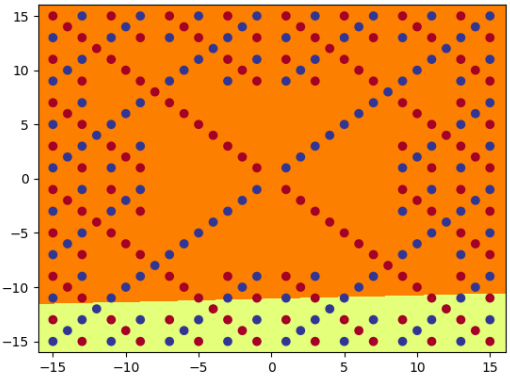


Node 7

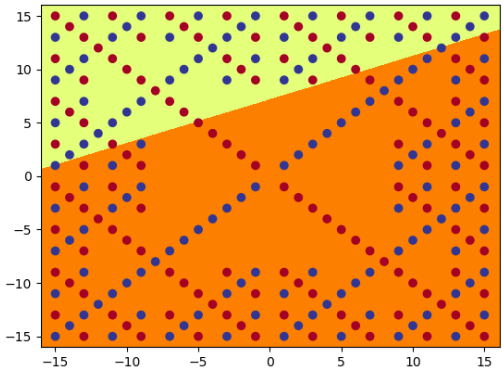
APPENDIX



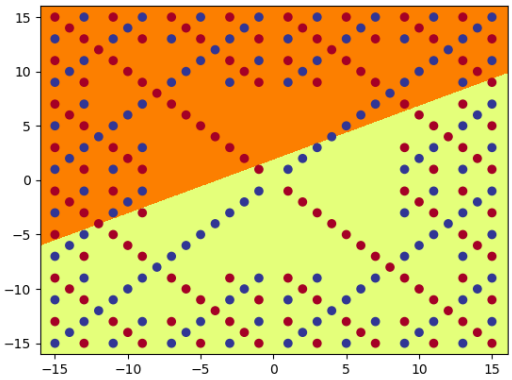
Node 8



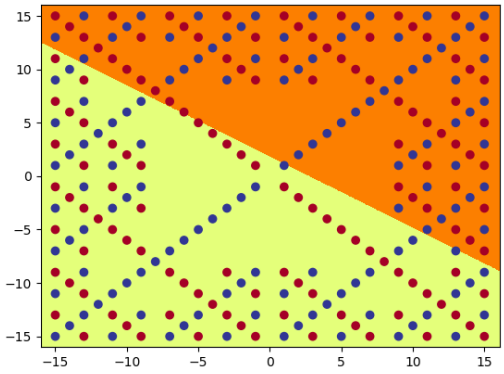
Node 9



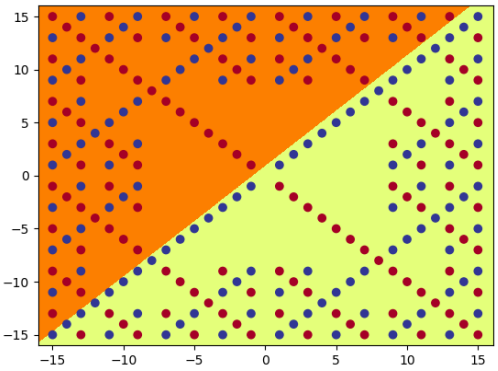
Node 10



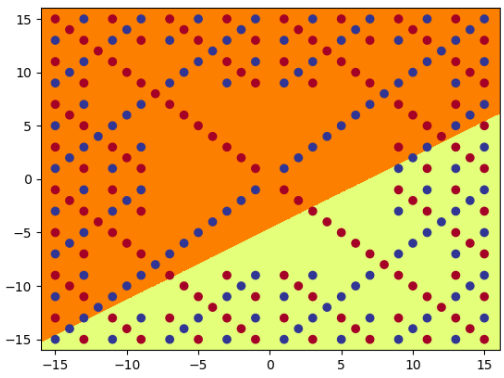
Node 11



Node 12

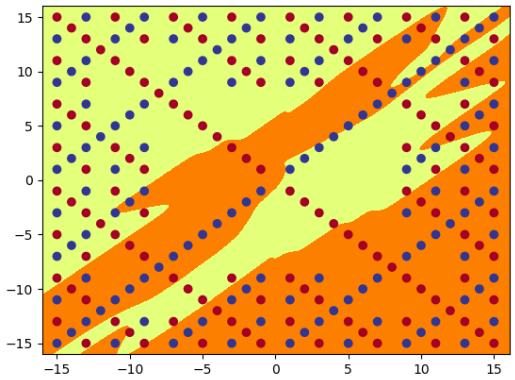


Node 13

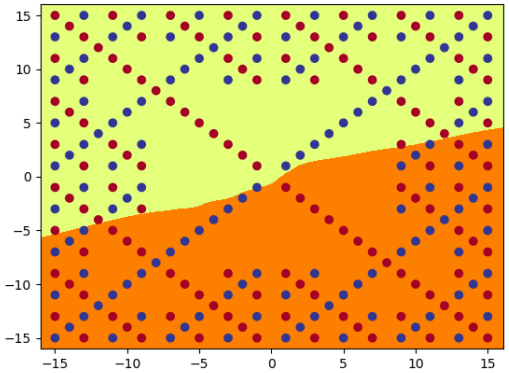


Node 14

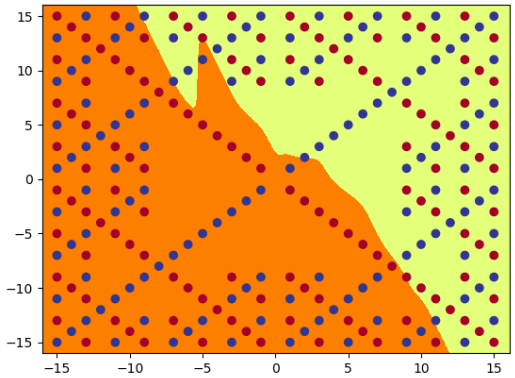
Layer 2



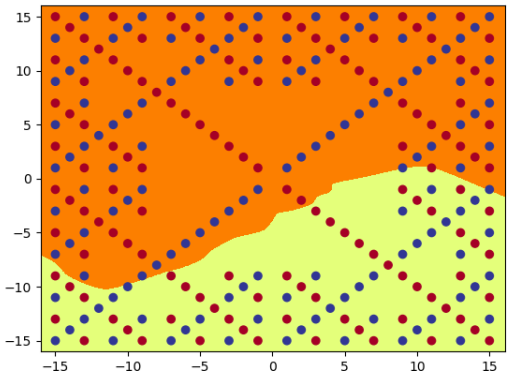
Node 0



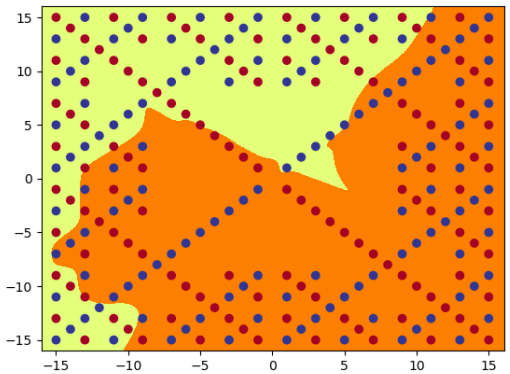
Node 1



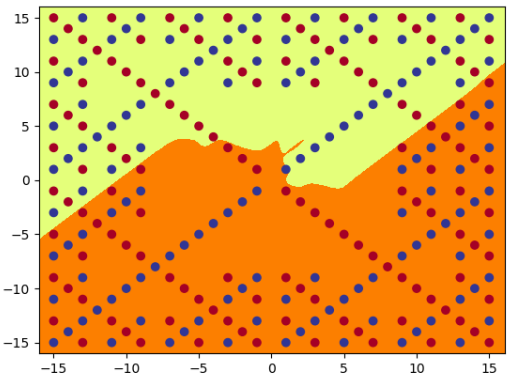
Node 2



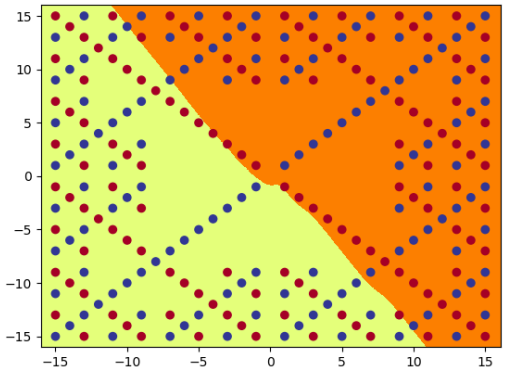
Node 3



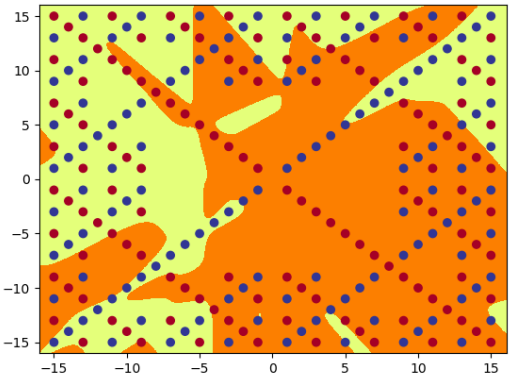
Node 4



Node 5

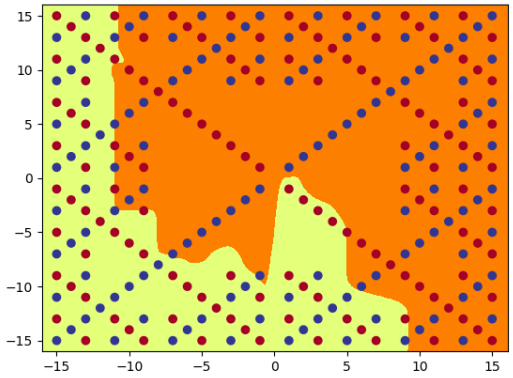


Node 6

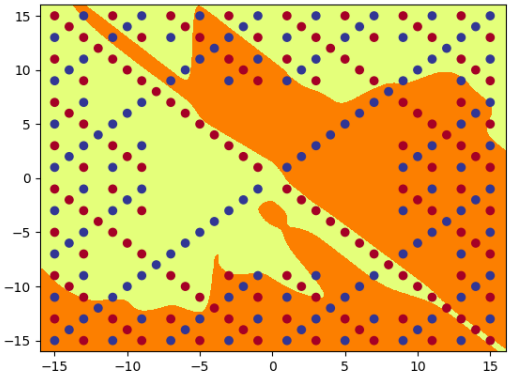


Node 7

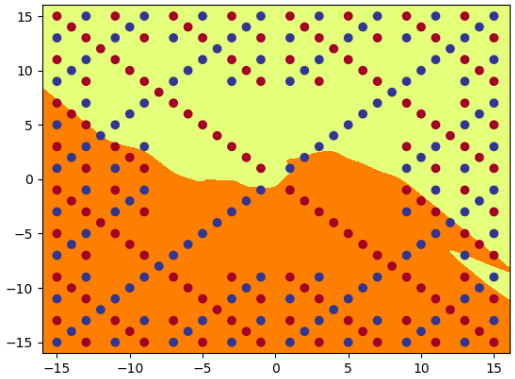
APPENDIX



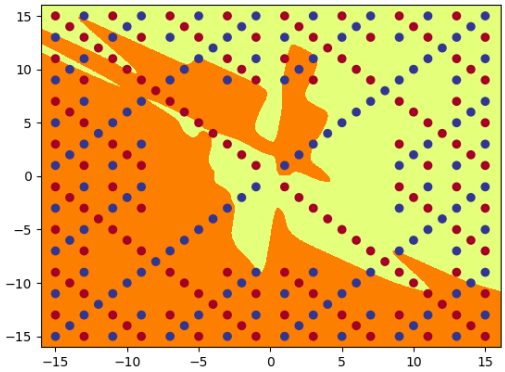
Node 8



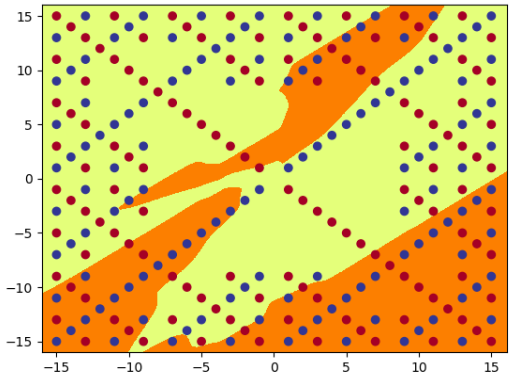
Node 9



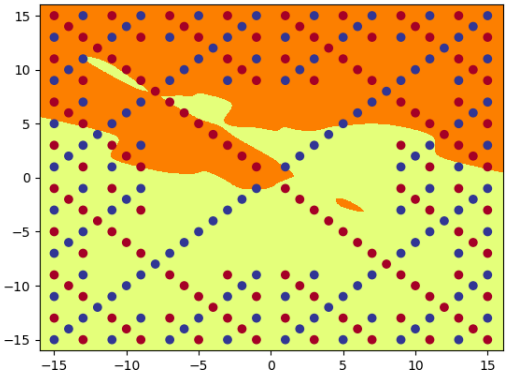
Node 10



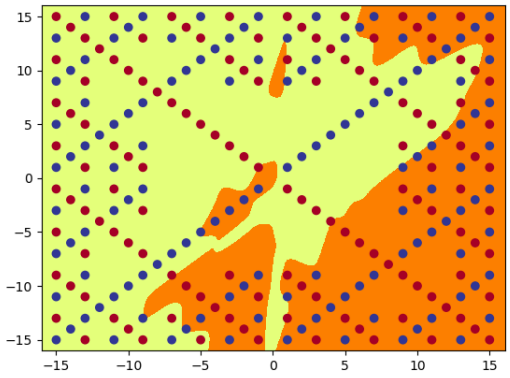
Node 11



Node 12



Node 13



Node 14