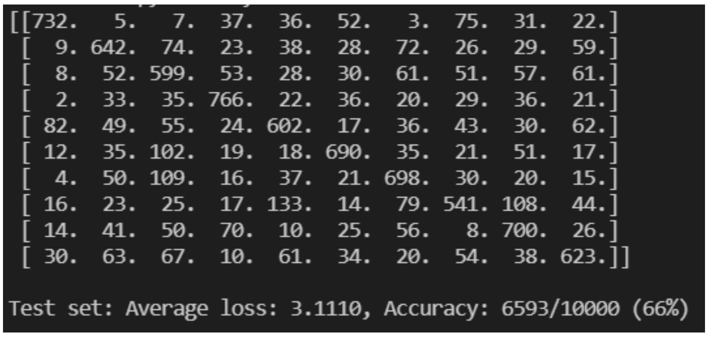
Project 1 – Character, Fractals and Hidden Unit Dynamics

COMP9444 – Neural Networks & Deep Learning

Part 1: Japanese Character Recognition

NetLin results



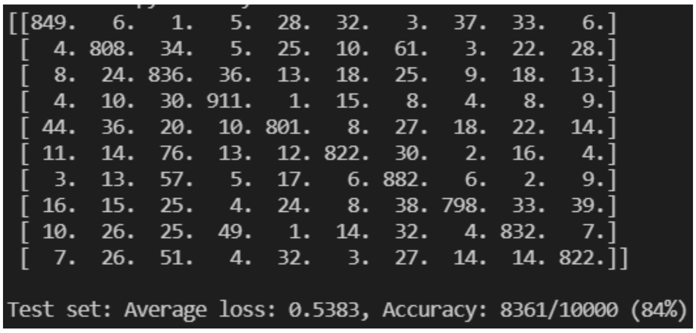
```
[[732.   5.   7.  37.  36.  52.   3.  75.  31.  22.]
 [  9. 642.  74.  23.  38.  28.  72.  26.  29.  59.]
 [  8.  52. 599.  53.  28.  30.  61.  51.  57.  61.]
 [  2.  33.  35. 766.  22.  36.  20.  29.  36.  21.]
 [ 82.  49.  55.  24. 602.  17.  36.  43.  30.  62.]
 [ 12.  35. 102.  19.  18. 690.  35.  21.  51.  17.]
 [  4.  50. 109.  16.  37.  21. 698.  30.  20.  15.]
 [ 16.  23.  25.  17. 133.  14.  79. 541. 108.  44.]
 [ 14.  41.  50.  70.  10.  25.  56.   8. 700.  26.]
 [ 30.  63.  67.  10.  61.  34.  20.  54.  38. 623.]]

Test set: Average loss: 3.1110, Accuracy: 6593/10000 (66%)
```

Test set: Average loss: 3.1108, Accuracy: 6593/10000 (66%)
Number of independent parameters: $10(28 \times 28 + 1) = 7850$

NetFull results

Hidden nodes: 100

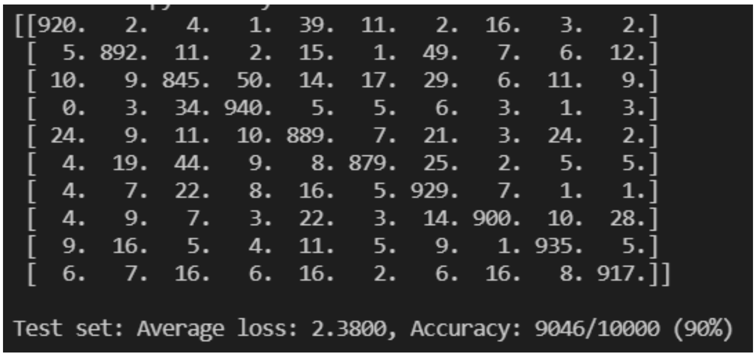


```
[[849.   6.   1.   5.  28.  32.   3.  37.  33.   6.]
 [  4. 808.  34.   5.  25.  10.  61.   3.  22.  28.]
 [  8.  24. 836.  36.  13.  18.  25.   9.  18.  13.]
 [  4.  10.  30. 911.   1.  15.   8.   4.   8.   9.]
 [ 44.  36.  20.  10. 801.   8.  27.  18.  22.  14.]
 [ 11.  14.  76.  13.  12. 822.  30.   2.  16.   4.]
 [  3.  13.  57.   5.  17.   6. 882.   6.   2.   9.]
 [ 16.  15.  25.   4.  24.   8.  38. 798.  33.  39.]
 [ 10.  26.  25.  49.   1.  14.  32.   4. 832.   7.]
 [  7.  26.  51.   4.  32.   3.  27.  14.  14. 822.]]

Test set: Average loss: 0.5383, Accuracy: 8361/10000 (84%)
```

Test set: Average loss: 0.5383, Accuracy: 8361/10000 (84%)
Total number of independent parameters: $(28 \times 28 + 1) \times 100 + (100 + 1) \times 10 = 79510$

NetConv results



```
[[920.   2.   4.   1.  39.  11.   2.  16.   3.   2.]
 [  5. 892.  11.   2.  15.   1.  49.   7.   6.  12.]
 [ 10.   9. 845.  50.  14.  17.  29.   6.  11.   9.]
 [  0.   3.  34. 940.   5.   5.   6.   3.   1.   3.]
 [ 24.   9.  11.  10. 889.   7.  21.   3.  24.   2.]
 [  4.  19.  44.   9.   8. 879.  25.   2.   5.   5.]
 [  4.   7.  22.   8.  16.   5. 929.   7.   1.   1.]
 [  4.   9.   7.   3.  22.   3.  14. 900.  10.  28.]
 [  9.  16.   5.   4.  11.   5.   9.   1. 935.   5.]
 [  6.   7.  16.   6.  16.   2.   6.  16.   8. 917.]]

Test set: Average loss: 2.3800, Accuracy: 9046/10000 (90%)
```

Layer 1: n_1 filters of size k_1 – n_1=30, k_1=5

Layer 2: n_2 filters of size k_2 – n_2=20,k_2=5

Total independent parameters: $n_1(k_1 * k_1 + 1) + n_2(k_2 * k_2 * n_1 + 1) + 10\left(\left(\frac{32-k_1+1}{2} - k_2 + 1\right)^2 * n_2 + 1\right) = 35810$

Discussion

Relative accuracy: Of the three models, the NetLin model was the least accurate, and the convolutional network NetConv was the most accurate. The accuracy the NetLin model was likely limited by the smaller number of parameters for learning. Compared to the other two models, the NetConv likely benefitted from using convolution because the nature of recognising written characters is such that nearby pixels in an image are more relevant to each other than pixels in other parts of the image.

Numbers of independent parameters: The NetLin model has significantly fewer independent parameters than the NetFull model, but while this makes for faster computation, it appears to have limited the learning capacity, as NetLin model was noticeably less accurate after training. However, while NetConv had fewer independent parameters, it was still able to achieve better accuracy, due to the suitability of a convolutional network to this application.

Confusion matrices: In the NetLin model, the 'ha'(5) and 'ma'(6) characters were often mistaken for the 'su'(2) character (likely due to having a similar horizontal line on top and a loop lower down). 'Ya' (7) was often mistaken for 'ha' or 're'(8) (likely due to the tall line(s) on the left). In the NetFull model, while mistakes were less common that for NetLin, the most common were: 'ha' and 'ma' mistaken for 'su', and 'ki'(1) mistaken for 'ma' (both the top and bottom halves of the characters are very similar in shape). In the case of NetConv, there was less of a visible trend in the confusion matrix. However, the same mistakes as with the previous models still had higher rates of misclassification than the rest of the matrix.

Part 2: Fractal Classification Task

Full2Net

Num hidden nodes: 20

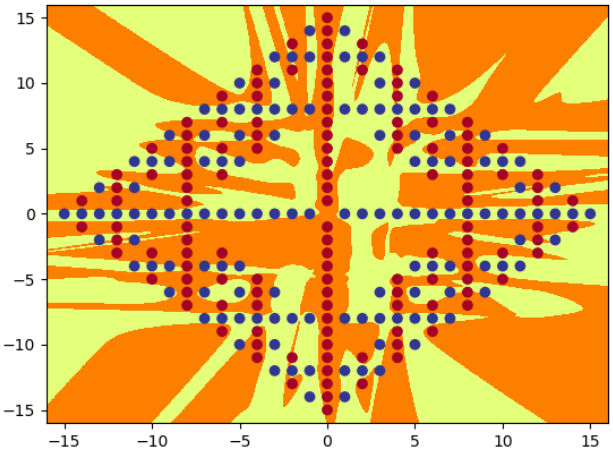
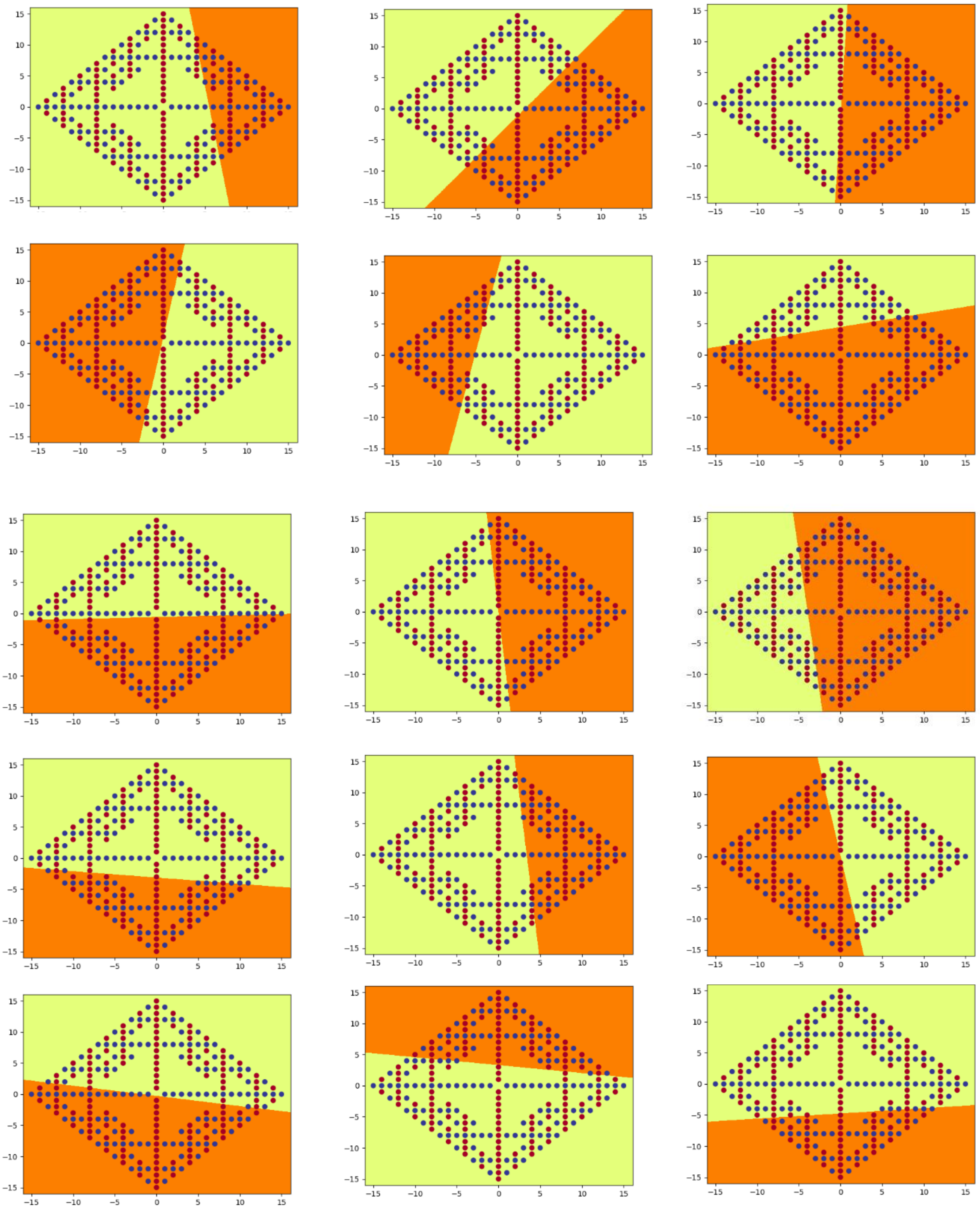


Figure 1 - out_full2_20.png

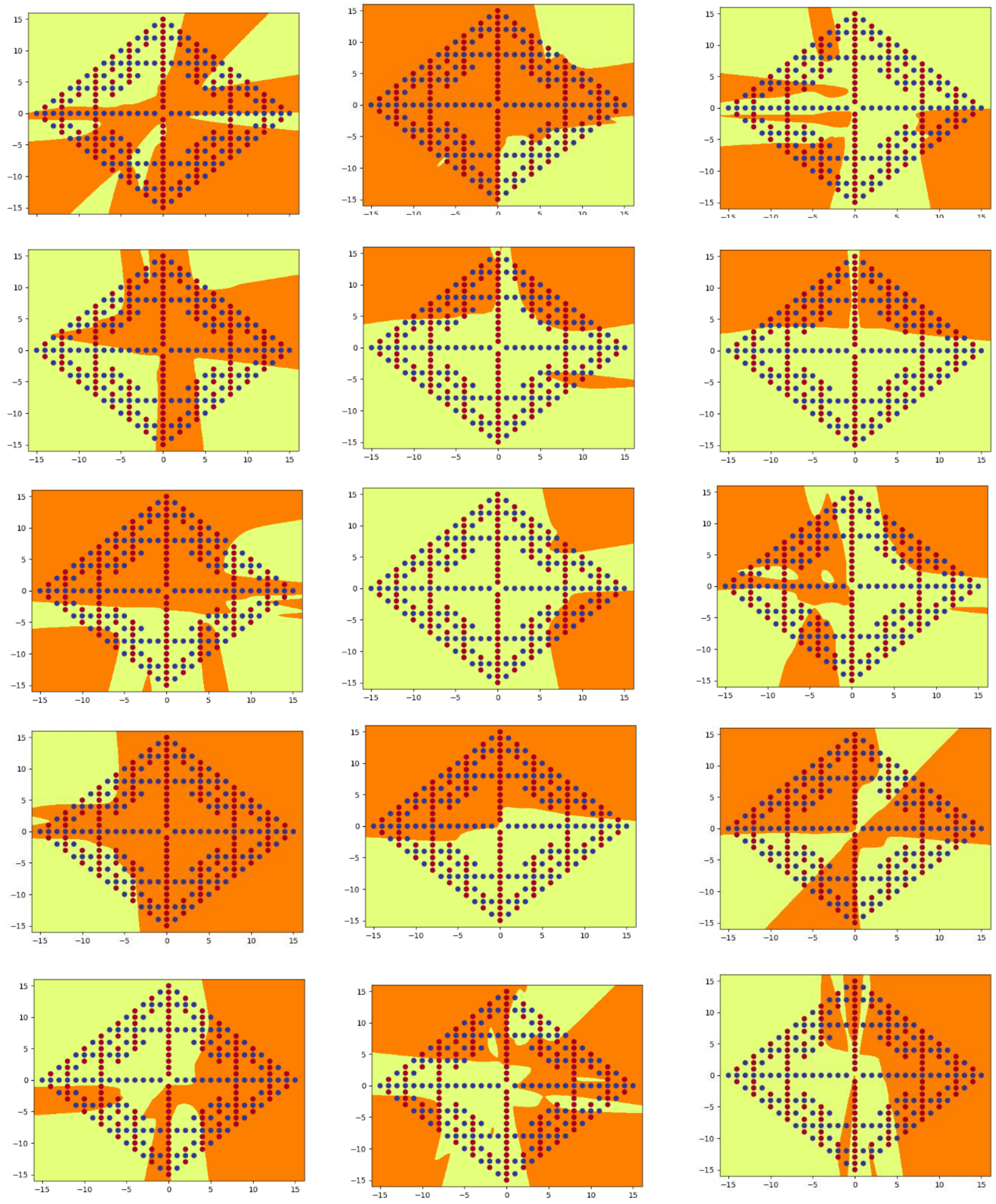
Total independent parameters: $20(2 + 1) + 20(20 + 1) + 1(20 + 1) = 501$

Full3Net

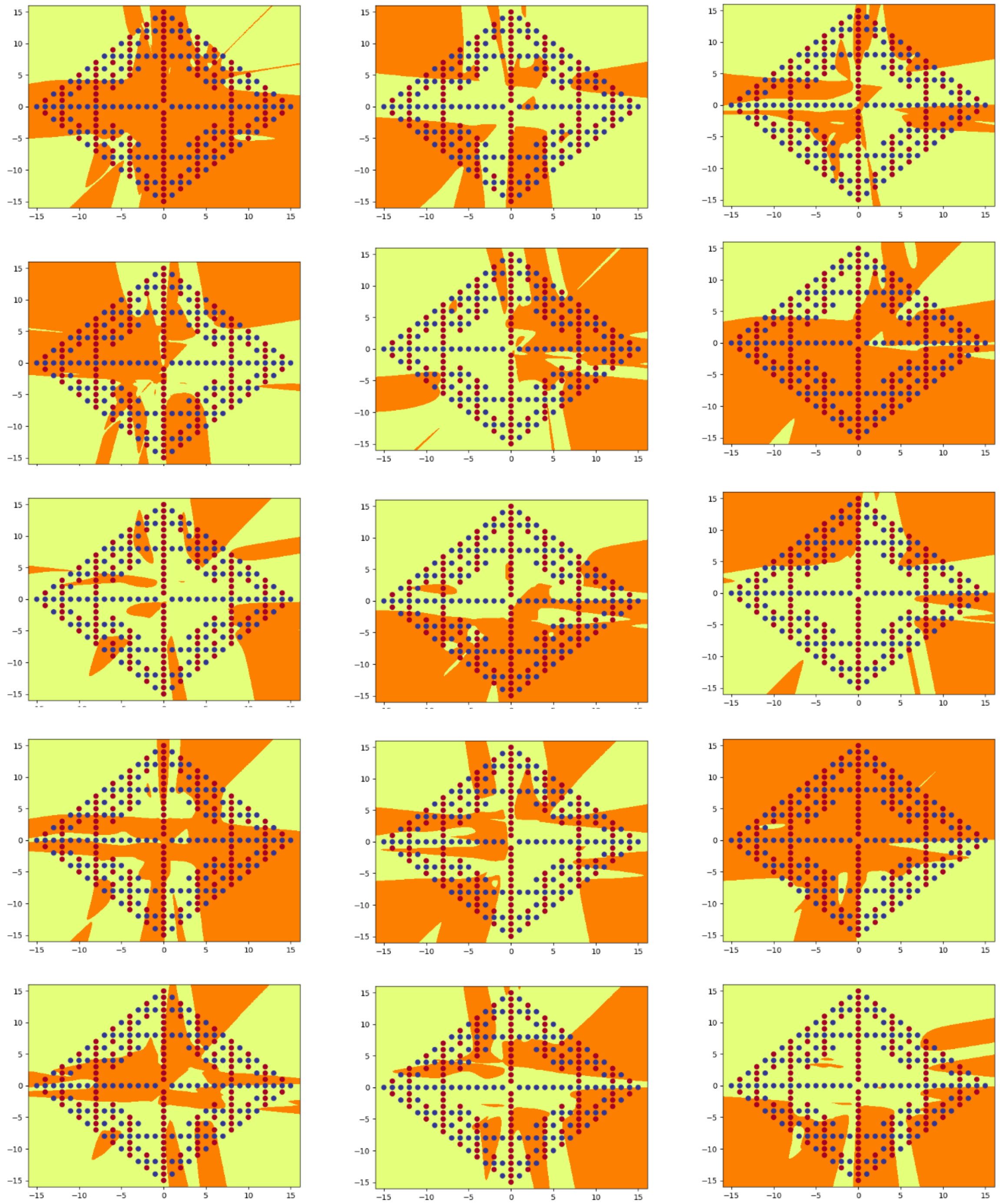
Layer 1 Functions



Layer 2 Functions



Layer 3 Functions



Output Function

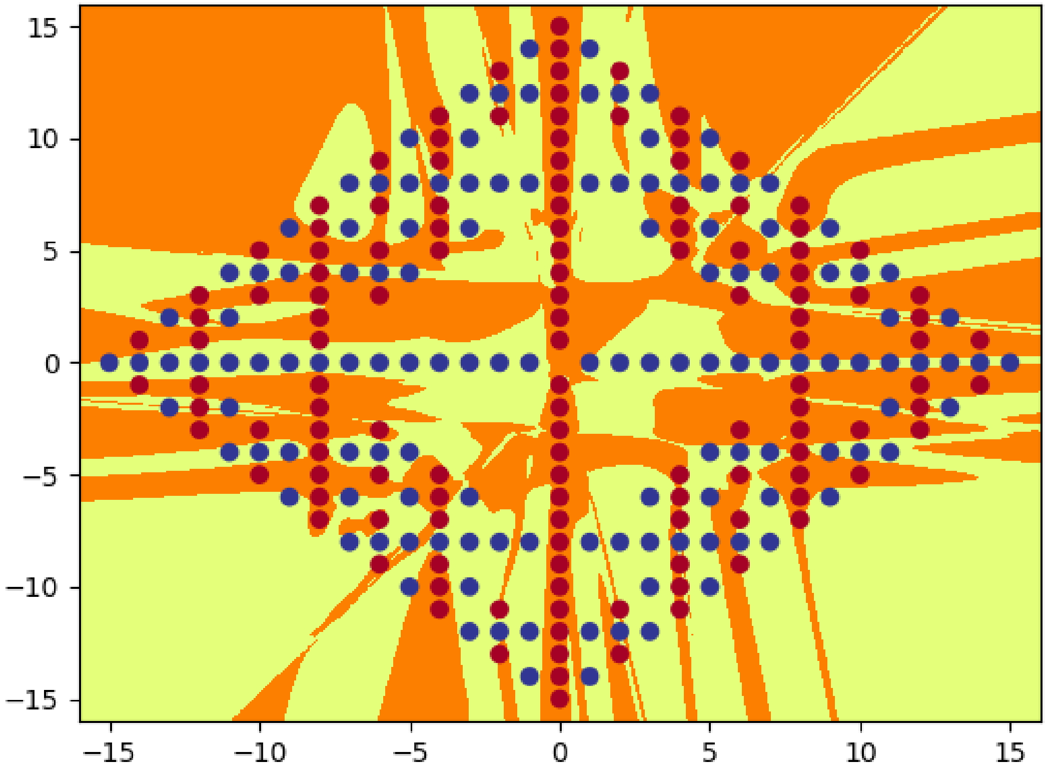


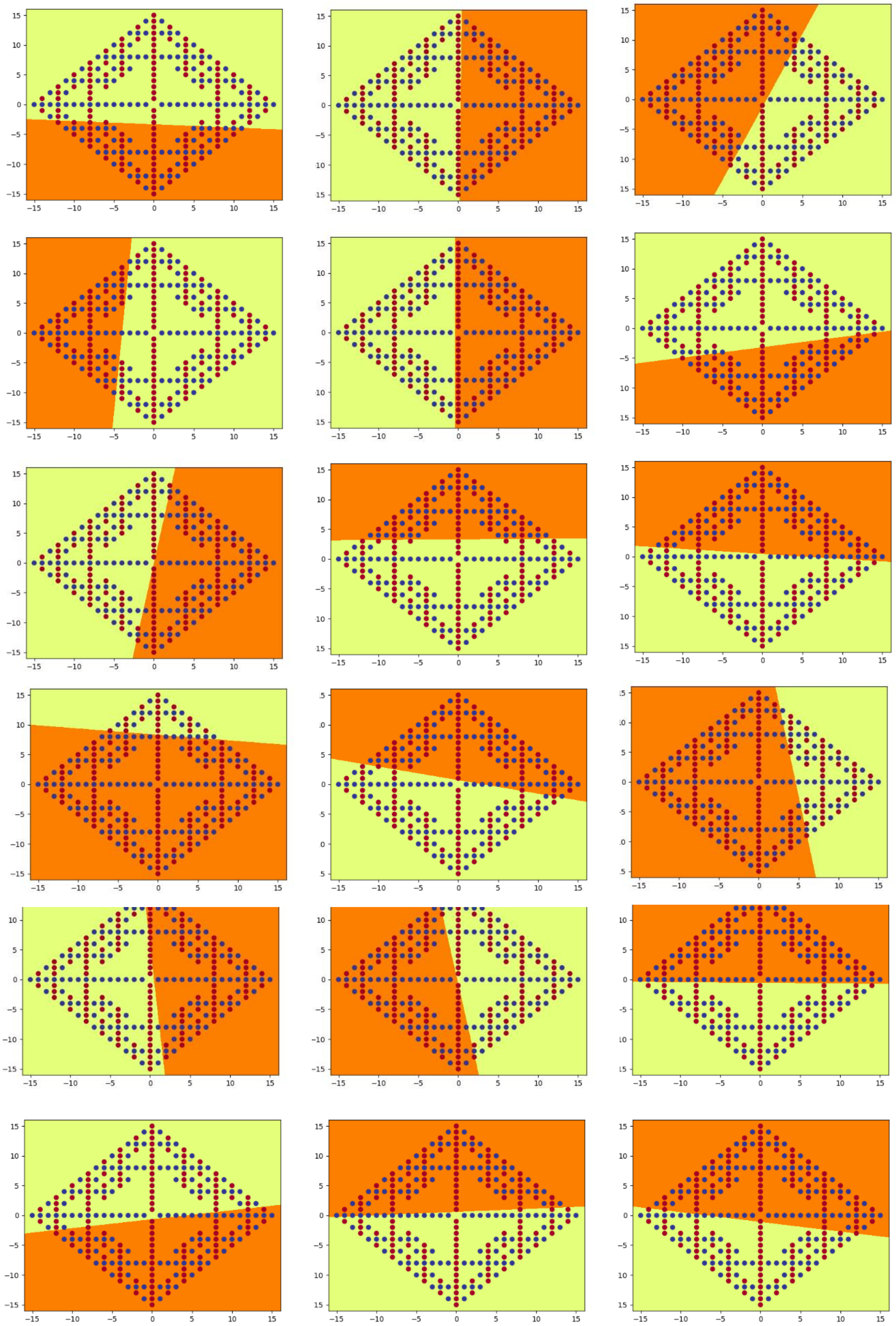
Figure 2 - out_full3_15.png

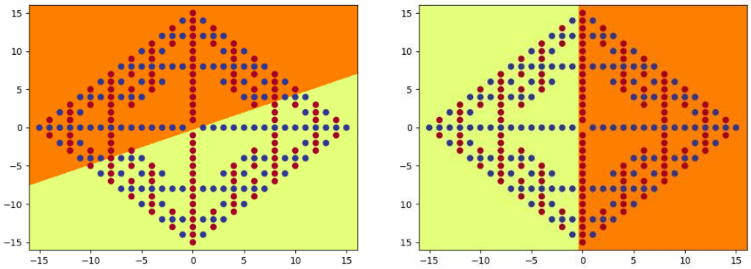
Num hidden nodes: 15

Total independent parameters: $h(2+1)+h(h+1)+h(h+1)+1(h+1)=541$

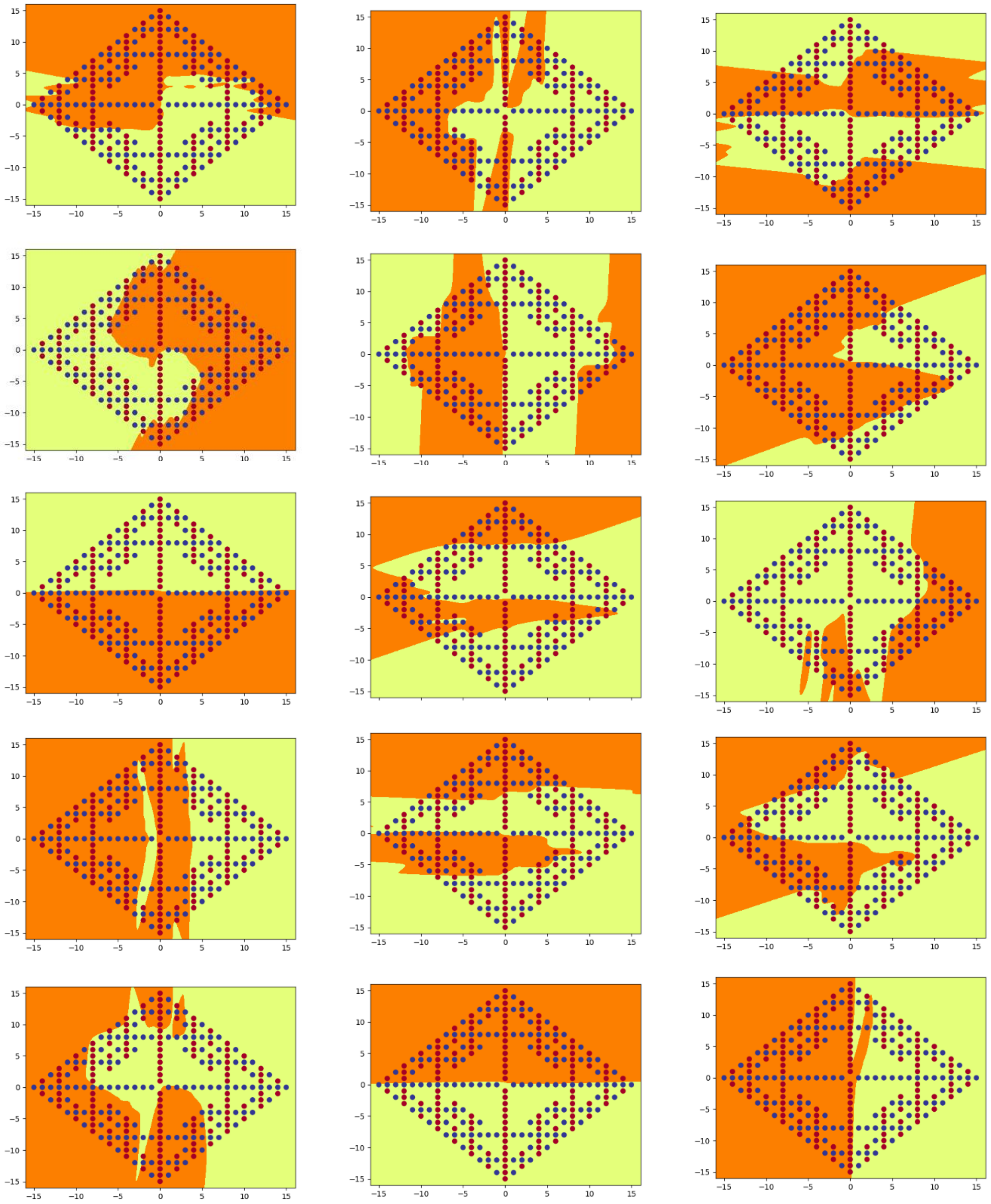
DenseNet

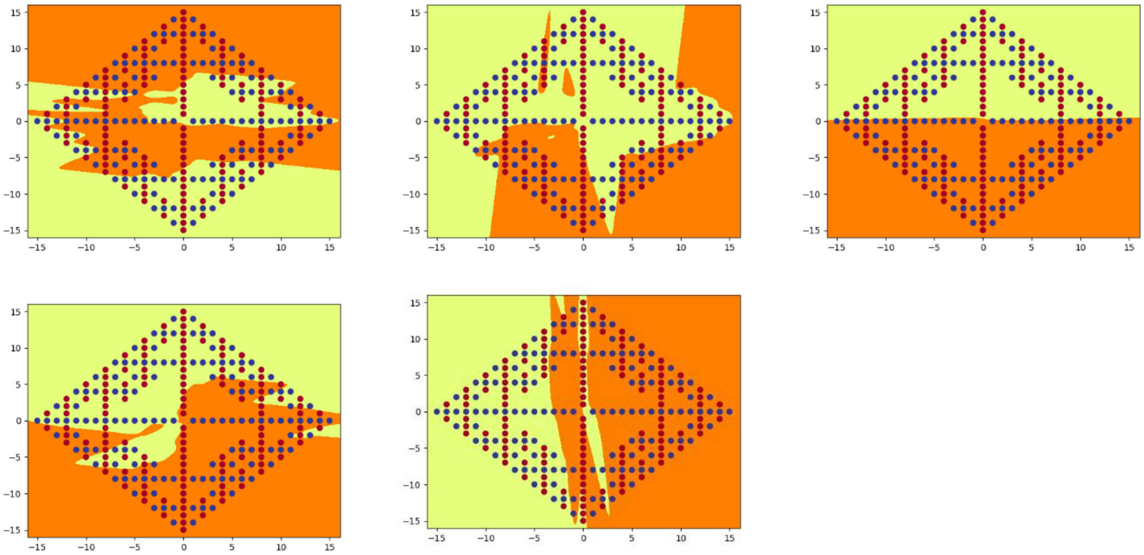
Layer 1 Functions





Layer 2 Functions





Output Function

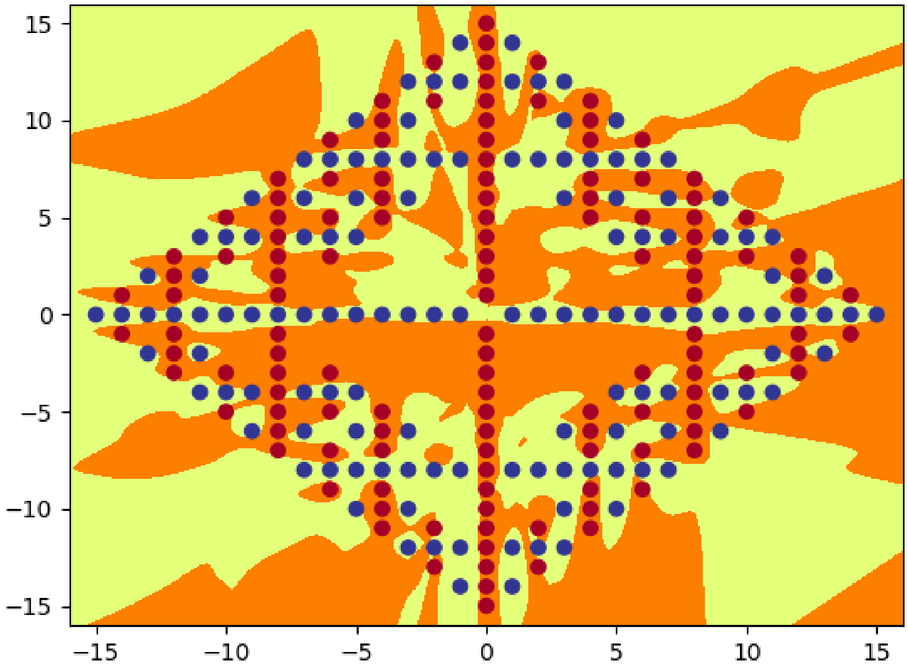


Figure 3 - out_dense_20.png

Num hidden nodes: 20 Num epochs: Around 38000

Total independent parameters: $h(2+1)+h(2+h+1)+1(2+h+h+1)=563$

Discussion

Independent parameters: While DenseNet had the largest number of independent parameters, and Full2Net the fewest, the number was fairly similar across the networks: around 500-600.

Qualitative description of layers of Full3Net & DenseNet: In Full3Net, the layers computed functions of varying complexity. The first layer computed very simple linear functions which separated the space into two halves. The second layer computed more complicated functions with less regular shapes, however the overall result usually split the space into 2-3 large regions and a number of smaller ones. The longest straight sections of the fractal were often separated from the surrounding areas. The functions computed by the third layer were more complicated again, with the space being split into more, smaller regions, and smaller parts of the fractal being segmentend.

In DenseNet, similarly to the other networks, the functions computed by the first layer are simple linear functions that separate the space into two regions. However, the functions computed by the second layer of DenseNet were qualitatively different to those of the second and third layers of Full3Net, in that the boundaries tended to be straighter, and most of the second-layer functions still only divided the region into two or three large regions.

Qualitative difference (if any) between the overall function of each network:

The output function primarily different in that the DenseNet produces boundaries that were generally more rounded, and fewer very thin sections of one colour. Furthermore, the overall shape of the DenseNet function appeared to, in most of the image, more closely resemble the fractal than the Full3Net. More common in the Full2Net and Full3Net was for points in image to be in a small area of the correct region, but mostly surrounded by the other region, suggesting that the fractal information had not been generalised well.

Part 3: Encoder Networks

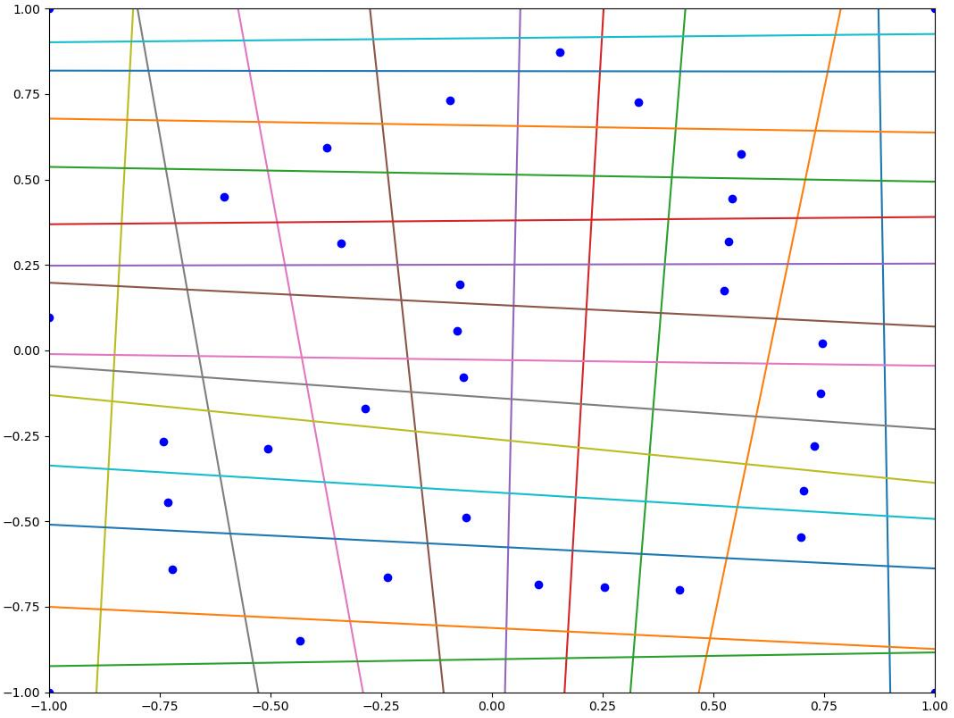
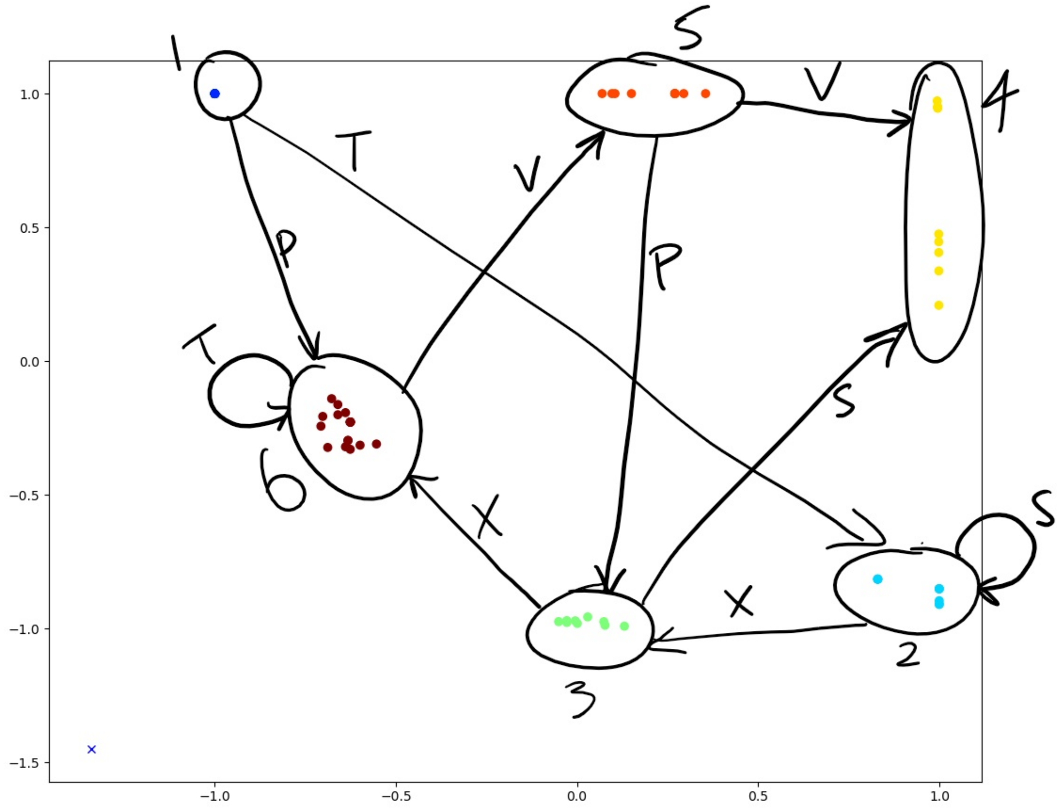


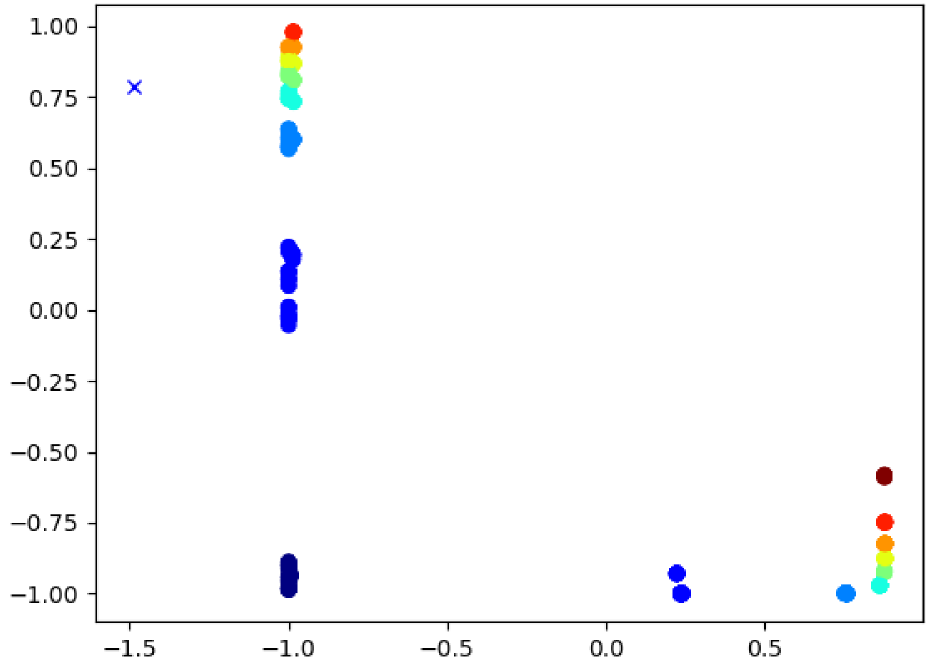
Figure 4 - Resulting hidden node activation graph

Part 4: Hidden Unit Dynamics for Recurrent Networks

SRN on Reber Grammar

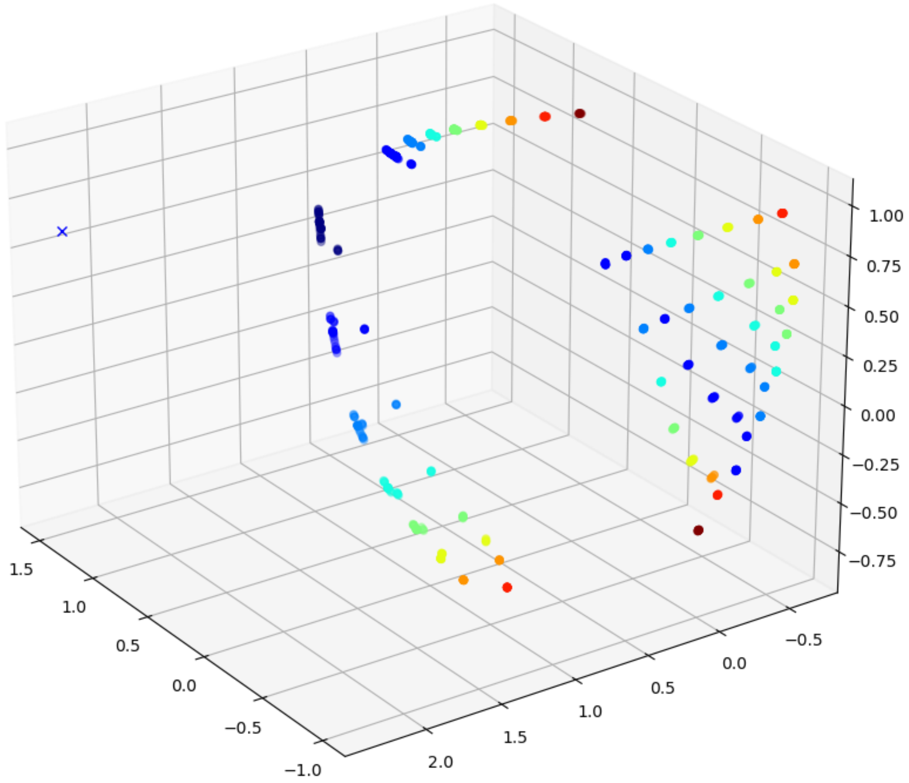


SRN on $a^n b^n$



When the first ‘a’ in a sequence in processed, the hidden unit activation from the bottom left of the graph (around (-1,-1)) to the bottom right corner. As consecutive ‘a’s are processed the activation shifts more to the right and upward. Once a ‘b’ is processed, indicating the end of that string of ‘a’s, the hidden unit activation jumps to the top left corner, where it works it’s way down as consecutive ‘b’s are processed. The network is therefore able to predict the last B in the sequence when the hidden unit activation has moved down on the graph sufficiently far from the top left corner. At this point, the activation jumps down further from around 0.0 to around -1.0 on the vertical axis, indication prediction of the following ‘a’. From here, the hidden unit activation again shifts to the right as ‘a’s are processed, continuing the cycle.

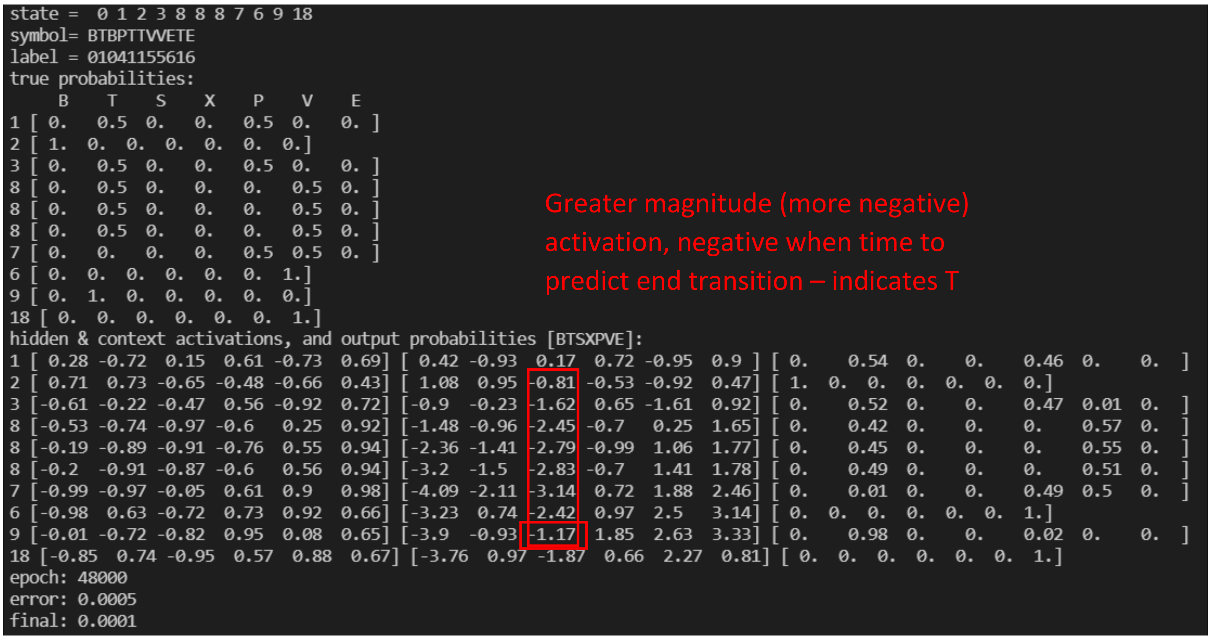
SRN on $a^nb^nc^n$



In this case, $a^nb^nc^n$ language prediction task is achieved with the hidden unit space split into three distinct regions, one for each letter in the task. Based on the hidden activation values printed by the program for each letter, it appears that the lower left region in the image is for the letter ‘a’, the upper middle region for ‘c’, and the wider region on the right is for ‘b’. As such, the hidden unit activation starts around the top left of the ‘a’ region when an ‘a’ is processed, and moves downwards as more consecutive ‘a’ characters are read. Once a ‘b’ is processed, the hidden unit activation jumps to the ‘b’ region, where is moves upwards/towards the middle blue area is counts down the consecutive ‘b’s. The network predicts the last ‘b’ in a sequence when the hidden unit activation reaches this area of the hidden unit space. Then, the hidden unit activation jumps to the region associated with the letter ‘c’, where all ‘c’s in the sequence are predicted. As the number of ‘c’s are counted down, the hidden unit activation moves to the left, towards to top of the ‘a’ region. Once the dark blue end of the ‘c’ region is reached, the hidden unit activation jumps to the top of the ‘a’ region and predicts the first ‘a’ in the next sequence.

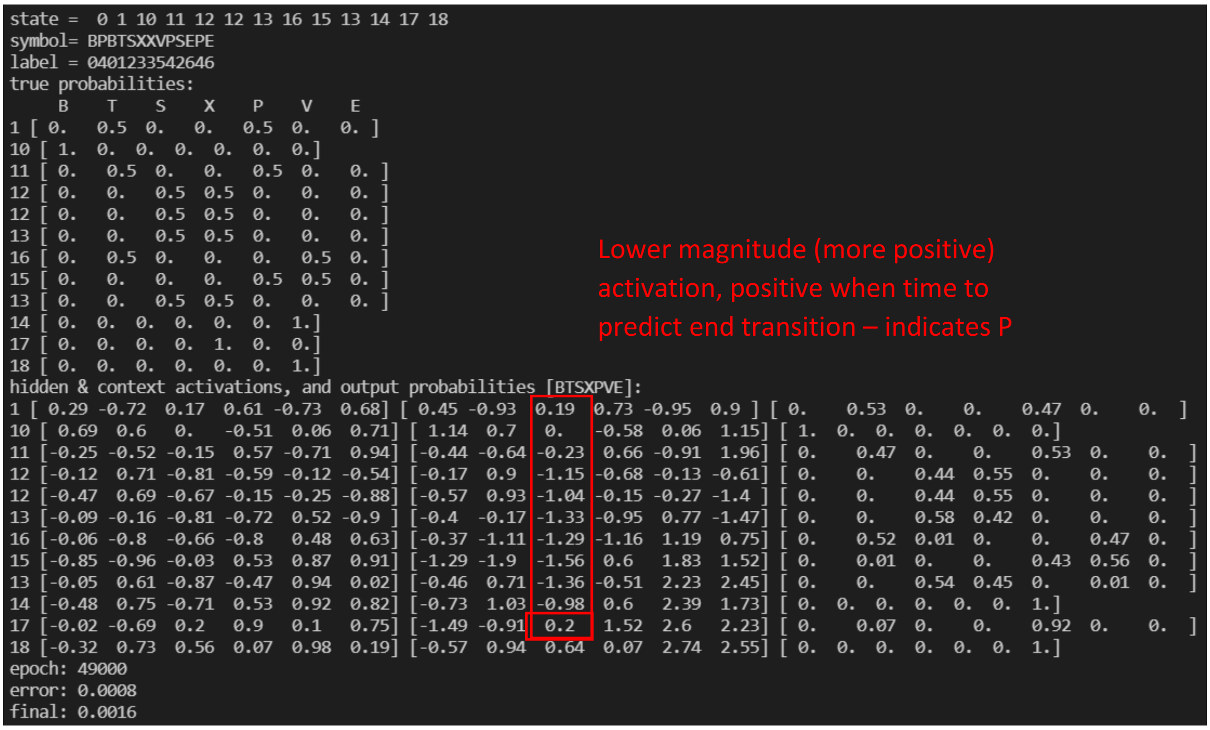
LSTM on Embedded Reber Grammar (6 hidden units)

When analysing the context unit activation in the LSTM it can be seen that, after processing an E character, one of the context units has a very different value (in particular, different sign) depending on whether it predicts a 'P' or a 'T' next. The only discernible pattern in the context unit activation between the initial and final transition, is that this same context unit has a consistently greater magnitide of negative activation, suggesting that this is where the information about which end transition to predict is stored.



```
state =  0 1 2 3 8 8 8 7 6 9 18
symbol= BTBPTTVVETE
label = 01041155616
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. ]
8 [ 0.   0.5  0.   0.   0.   0.5  0. ]
8 [ 0.   0.5  0.   0.   0.   0.5  0. ]
8 [ 0.   0.5  0.   0.   0.   0.5  0. ]
7 [ 0.   0.   0.   0.   0.5  0.5  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 & context activations, and output probabilities [BTSXPVE]:
1 [ 0.28 -0.72  0.15  0.61 -0.73  0.69] [ 0.42 -0.93  0.17  0.72 -0.95  0.9 ] [ 0.    0.54  0.    0.    0.46  0.    0.  ]
2 [ 0.71  0.73 -0.65 -0.48 -0.66  0.43] [ 1.08  0.95 -0.81 -0.53 -0.92  0.47] [ 1.  0.  0.  0.  0.  0.  0.]
3 [-0.61 -0.22 -0.47  0.56 -0.92  0.72] [-0.9  -0.23 -1.62  0.65 -1.61  0.92] [ 0.    0.52  0.    0.    0.47  0.01  0.  ]
8 [-0.53 -0.74 -0.97 -0.6   0.25  0.92] [-1.48 -0.96 -2.45 -0.7   0.25  1.65] [ 0.    0.42  0.    0.    0.    0.57  0.  ]
8 [-0.19 -0.89 -0.91 -0.76  0.55  0.94] [-2.36 -1.41 -2.79 -0.99  1.06  1.77] [ 0.    0.45  0.    0.    0.    0.55  0.  ]
8 [-0.2  -0.91 -0.87 -0.6   0.56  0.94] [-3.2  -1.5  -2.83 -0.7   1.41  1.78] [ 0.    0.49  0.    0.    0.    0.51  0.  ]
7 [-0.99 -0.97 -0.05  0.61  0.9   0.98] [-4.09 -2.11 -3.14  0.72  1.88  2.46] [ 0.    0.01  0.    0.    0.49  0.5   0.  ]
6 [-0.98  0.63 -0.72  0.73  0.92  0.66] [-3.23  0.74 -2.42  0.97  2.5   3.14] [ 0.  0.  0.  0.  0.  0.  1.]
9 [-0.01 -0.72 -0.82  0.95  0.08  0.65] [-3.9  -0.93 -1.17  1.85  2.63  3.33] [ 0.    0.98  0.    0.    0.02  0.    0.  ]
18 [-0.85  0.74 -0.95  0.57  0.88  0.67] [-3.76  0.97 -1.87  0.66  2.27  0.81] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 48000
error: 0.0005
final: 0.0001
```

Greater magnitude (more negative) activation, negative when time to predict end transition – indicates T



```
state =  0 1 10 11 12 12 13 16 15 13 14 17 18
symbol= BPBTSXXVPSEPE
label = 0401233542646
true probabilities:
     B    T    S    X    P    V    E
1 [ 0.   0.5  0.   0.   0.5  0.   0. ]
10 [ 1.  0.  0.  0.  0.  0.  0.]
11 [ 0.   0.5  0.   0.   0.5  0.   0. ]
12 [ 0.   0.   0.5  0.5  0.   0.   0. ]
12 [ 0.   0.   0.5  0.5  0.   0.   0. ]
13 [ 0.   0.   0.5  0.5  0.   0.   0. ]
16 [ 0.   0.5  0.   0.   0.   0.5  0. ]
15 [ 0.   0.   0.   0.   0.5  0.5  0. ]
13 [ 0.   0.   0.5  0.5  0.   0.   0. ]
14 [ 0.  0.  0.  0.  0.  0.  1.]
17 [ 0.  0.  0.  0.  1.  0.  0.]
18 [ 0.  0.  0.  0.  0.  0.  1.]
hidden & context activations, and output probabilities [BTSXPVE]:
1 [ 0.29 -0.72  0.17  0.61 -0.73  0.68] [ 0.45 -0.93  0.19  0.73 -0.95  0.9 ] [ 0.    0.53  0.    0.    0.47  0.    0.  ]
10 [ 0.69  0.6   0.   -0.51  0.06  0.71] [ 1.14  0.7   0.   -0.58  0.06  1.15] [ 1.  0.  0.  0.  0.  0.  0.]
11 [-0.25 -0.52 -0.15  0.57 -0.71  0.94] [-0.44 -0.64 -0.23  0.66 -0.91  1.96] [ 0.    0.47  0.    0.    0.53  0.    0.  ]
12 [-0.12  0.71 -0.81 -0.59 -0.12 -0.54] [-0.17  0.9  -1.15 -0.68 -0.13 -0.61] [ 0.    0.    0.44  0.55  0.    0.    0.  ]
12 [-0.47  0.69 -0.67 -0.15 -0.25 -0.88] [-0.57  0.93 -1.04 -0.15 -0.27 -1.4 ] [ 0.    0.    0.44  0.55  0.    0.    0.  ]
13 [-0.09 -0.16 -0.81 -0.72  0.52 -0.9 ] [-0.4  -0.17 -1.33 -0.95  0.77 -1.47] [ 0.    0.    0.58  0.42  0.    0.    0.  ]
16 [-0.06 -0.8  -0.66 -0.8   0.48  0.63] [-0.37 -1.11 -1.29 -1.16  1.19  0.75] [ 0.    0.52  0.01  0.    0.    0.47  0.  ]
15 [-0.85 -0.96 -0.03  0.53  0.87  0.91] [-1.29 -1.9  -1.56  0.6   1.83  1.52] [ 0.    0.01  0.    0.    0.43  0.56  0.  ]
13 [-0.05  0.61 -0.87 -0.47  0.94  0.02] [-0.46  0.71 -1.36 -0.51  2.23  2.45] [ 0.    0.    0.54  0.45  0.    0.01  0.  ]
14 [-0.48  0.75 -0.71  0.53  0.92  0.82] [-0.73  1.03 -0.98  0.6   2.39  1.73] [ 0.  0.  0.  0.  0.  0.  1.]
17 [-0.02 -0.69  0.2   0.9   0.1   0.75] [-1.49 -0.91  0.2   1.52  2.6   2.23] [ 0.    0.07  0.    0.    0.92  0.    0.  ]
18 [-0.32  0.73  0.56  0.07  0.98  0.19] [-0.57  0.94  0.64  0.07  2.74  2.55] [ 0.  0.  0.  0.  0.  0.  1.]
epoch: 49000
error: 0.0008
final: 0.0016
```

Lower magnitude (more positive) activation, positive when time to predict end transition – indicates P