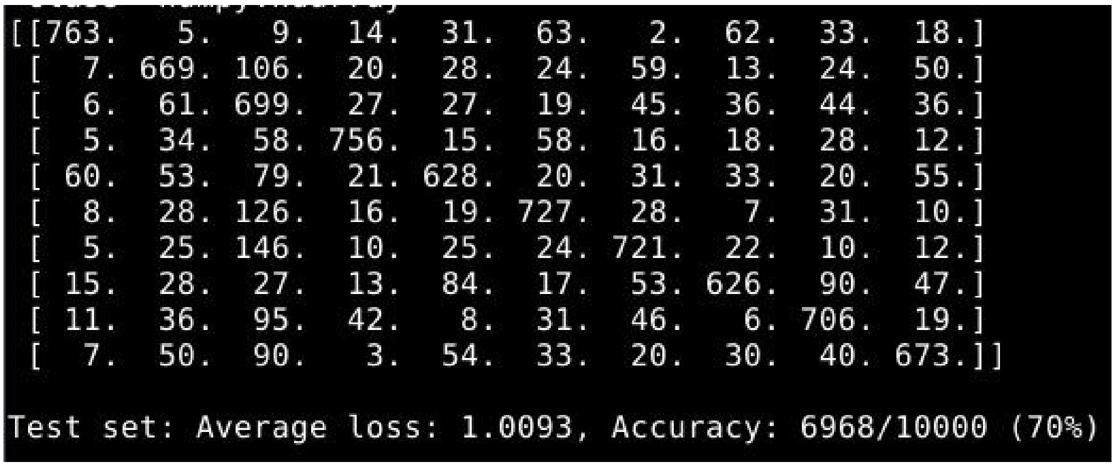
COMP9444 Assignment1

z5241178 Zihao Jiang

Part1

Q1.



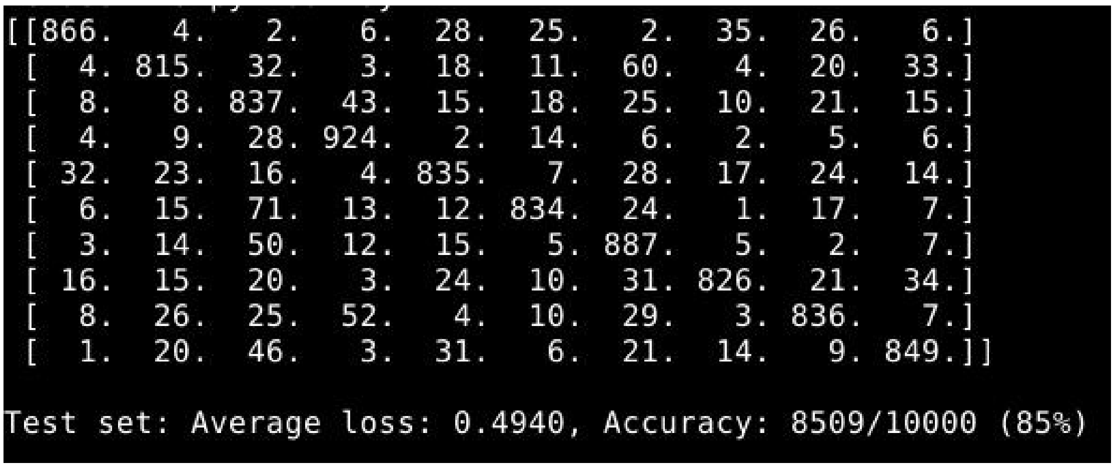
```
[[763.   5.   9.  14.  31.  63.   2.  62.  33.  18.]
 [  7. 669. 106.  20.  28.  24.  59.  13.  24.  50.]
 [  6.  61. 699.  27.  27.  19.  45.  36.  44.  36.]
 [  5.  34.  58. 756.  15.  58.  16.  18.  28.  12.]
 [ 60.  53.  79.  21. 628.  20.  31.  33.  20.  55.]
 [  8.  28. 126.  16.  19. 727.  28.   7.  31.  10.]
 [  5.  25. 146.  10.  25.  24. 721.  22.  10.  12.]
 [ 15.  28.  27.  13.  84.  17.  53. 626.  90.  47.]
 [ 11.  36.  95.  42.   8.  31.  46.   6. 706.  19.]
 [  7.  50.  90.   3.  54.  33.  20.  30.  40. 673.]]

Test set: Average loss: 1.0093, Accuracy: 6968/10000 (70%)
```

Q2.
300 hidden nodes.

Fully-connected layer:
1.weights per filter = 1 + 28 × 28 × 1 = 785
2.number of neurons = 300
3.Connections = 300 × 785 = 235500
4.independent parameters = 235500

Output layer:
1.weights per filter = 1 + 300 = 301
2.number of neurons = 10
3.Connections = 301 × 10 = 3010
4.independent parameters = 3010

So the total number of independent parameters in the network is 235500 + 3010 = 238510



```
[[866.   4.   2.   6.  28.  25.   2.  35.  26.   6.]
 [  4. 815.  32.   3.  18.  11.  60.   4.  20.  33.]
 [  8.   8. 837.  43.  15.  18.  25.  10.  21.  15.]
 [  4.   9.  28. 924.   2.  14.   6.   2.   5.   6.]
 [ 32.  23.  16.   4. 835.   7.  28.  17.  24.  14.]
 [  6.  15.  71.  13.  12. 834.  24.   1.  17.   7.]
 [  3.  14.  50.  12.  15.   5. 887.   5.   2.   7.]
 [ 16.  15.  20.   3.  24.  10.  31. 826.  21.  34.]
 [  8.  26.  25.  52.   4.  10.  29.   3. 836.   7.]
 [  1.  20.  46.   3.  31.   6.  21.  14.   9. 849.]]

Test set: Average loss: 0.4940, Accuracy: 8509/10000 (85%)
```

Q3.

The first convolutional layer has input_channel of 1 and output_channel of 64. The second layer has input_channel of 64 and output_channel of 256. Both of them have kernel_size of 5 and both of them are followed by 2 * 2 pooling layers. The model also has one fully connected layer which has input_size of 2048 and output_size of 256. The output layer has input_size of 256 and output_size of 10.

For first convolutional layer:
J = K = 28, L = 1, M = N = 5, P = 0, s = 1
1.weights per filter = 1 + 5 * 5 * 1 = 26
2.width and height = 1 + (28 - 5)/1 = 24
3.number of neurons = 24 * 24 * 64 = 36864
4.connections = 24 * 24 * 64 * 26 = 958464
5.independent parameters = 64 * 26 = 1664

For first pooling layer:
J = K = 24, L = 64, M = N = 2, P = 0, s = 2
1.weights per filter = 1 + 2 * 2 * 64 = 256
2.width and height = 24 / 2 = 12
3.number of neurons = 12 * 12 * 64 = 9216
4.connections = 12 * 12 * 64 * 256 = 2359296
5.independent parameters = 64 * 256 = 16384

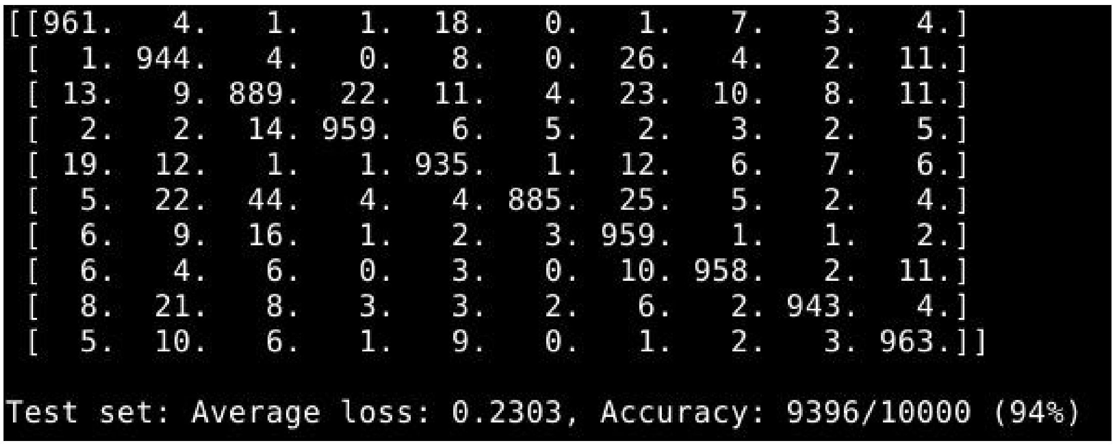
For second convolutional layer:
J = K = 12, L = 64, M = N = 5, P = 0, s = 1
1.weights per filter = 1 + 5 * 5 * 64 = 1601
2.width and height = 1 + (12 - 5)/1 = 8
3.number of neurons = 8 * 8 * 128 = 8192
4.connections = 8 * 8 * 128 * 1601 = 13115392
5.independent parameters = 128 * 1601 = 204928

For second pooling layer:
J = K = 8, L = 128, M = N = 2, P = 0, s = 2
1.weights per filter = 1 + 2 * 2 * 128 = 512
2.width and height = 8 / 2 = 4
3.number of neurons = 4 * 4 * 128 = 2048
4.connections = 4 * 4 * 128 * 512 = 1048576
5.independent parameters = 128 * 512 = 65536

For fully-connected layer:
1.weights per filter = 1 + 2048 = 2049
2.number of neurons = 256
3.Connections = 256 × 2049 = 524544
4.independent parameters = 524544

For output layer:
1.weights per filter = 1 + 256 = 257
2.number of neurons = 10
3.Connections = 10 × 257 = 2570
4.independent parameters = 2570

So the total number of independent parameters in the network is 1664 + 16384 + 204928 + 65536 + 524544 + 2570 = 815626


```
[[961.   4.   1.   1.  18.   0.   1.   7.   3.   4.]
 [  1. 944.   4.   0.   8.   0.  26.   4.   2.  11.]
 [ 13.   9. 889.  22.  11.   4.  23.  10.   8.  11.]
 [  2.   2.  14. 959.   6.   5.   2.   3.   2.   5.]
 [ 19.  12.   1.   1. 935.   1.  12.   6.   7.   6.]
 [  5.  22.  44.   4.   4. 885.  25.   5.   2.   4.]
 [  6.   9.  16.   1.   2.   3. 959.   1.   1.   2.]
 [  6.   4.   6.   0.   3.   0.  10. 958.   2.  11.]
 [  8.  21.   8.   3.   3.   2.   6.   2. 943.   4.]
 [  5.  10.   6.   1.   9.   0.   1.   2.   3. 963.]]

Test set: Average loss: 0.2303, Accuracy: 9396/10000 (94%)
```

Q4.
a. As shown above, we can conclude the relative accuracy of the three models as below.

Model	Accuracy
NetLin	70%
NetFull	85%
NetConv	94%

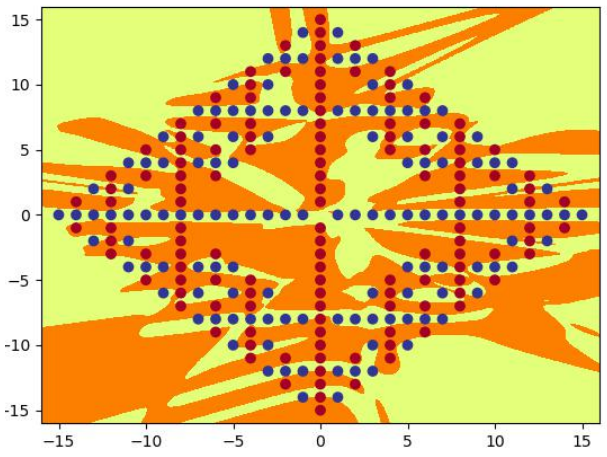
The result shows that NetConv has highest accuracy compared to others. NetLin has a very simple structure but its accuracy is also the smallest. For linear model, it takes a lot of time to learn features and also linear models are prone to overfitting, which leads to low accuracy. Also, linear models perform bad when the input data is non-linear. The fully connected network extracts the characteristics of the entire image, so it improves the accuracy, but at the same time it also generates a large number of parameters, which leads to the inaccuracy of the model calculation. At the same time, too many parameters may also lead to overfitting. Convolutional neural networks can effectively learn corresponding features from a large number of samples, avoiding complex feature extraction processes.

b. We can find that as the model becomes more and more complex, the number of independent parameters of the model become more and more. I think that more independent parameters may improve the accuracy of the model, but it also requires more time to train the model. The increase of independent parameters is helpful to the feature recognition of image details, so the convolutional neural network is the most accurate one of the three models.

c. I think there are two reasons. The first reason is because simple structured networks do not perform well in feature extraction, so it is prone to misclassification. The second reason is probably because the noise in the data. Those sample data are all handwritten so even though they all represent a same digit, they do not look same as each other.

Part2

Q2.



The number of hidden nodes that I choose is 24. Learning rate = 0.01, Initial weight size = 0.1

For the first fully-connected layer:

1.weights per filter = 1 + 2 = 3

2.number of neurons = 24

3.Connections = 24 × 3 = 72

4.independent parameters = 72

For the second fully-connected layer:

1.weights per filter = 1 + 24 = 25
2.number of neurons = 24
3.Connections = 24 × 25= 600
4.independent parameters = 600

For the output layer:
1.weights per filter = 1 + 24= 25
2.number of neurons = 1
3.Connections = 1 × 25 = 25
4.independent parameters = 25

So the total number of independent parameters in the network is 72 + 600 + 25 = 697.

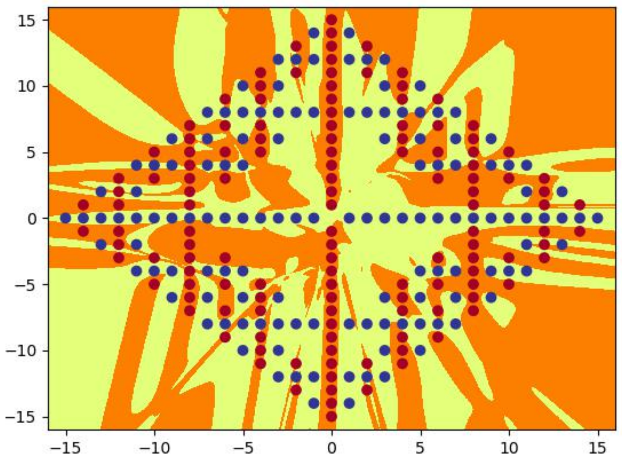
Q4.
The number of hidden nodes that I choose is 23. Learning rate = 0.01, Initial weight size = 0.1.
For the first fully-connected layer:
1.weights per filter = 1 + 2 = 3
2.number of neurons = 23
3.Connections = 23 × 3 = 69
4.independent parameters = 69

For the second fully-connected layer:
1.weights per filter = 1 + 23 = 24
2.number of neurons = 23
3.Connections = 23 × 24 = 552
4.independent parameters = 552

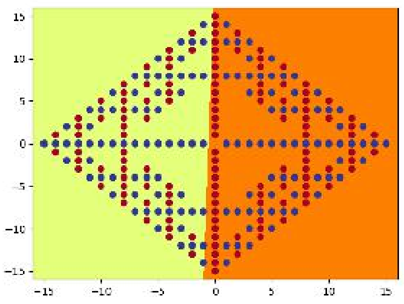
For the third fully-connected layer:
1.weights per filter = 1 + 23 = 24
2.number of neurons = 23
3.Connections = 23 × 24 = 552
4.independent parameters = 552

For the output layer:
1.weights per filter = 1 + 23 = 24
2.number of neurons = 1
3.Connections = 24 ×1 = 24
4.independent parameters = 24

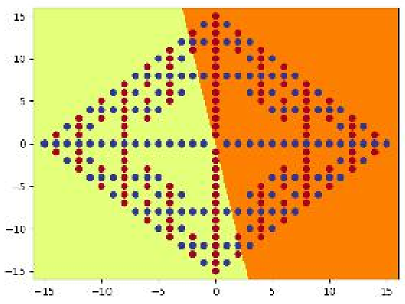
So the total number of independent parameters in the network is 69 + 552 + 552 + 24 = 1197 out_full3_23.png



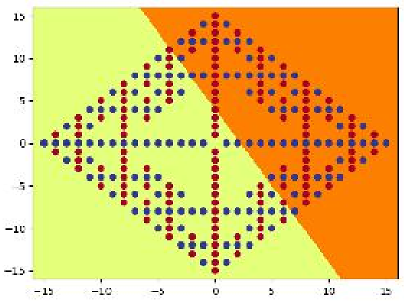
1_0



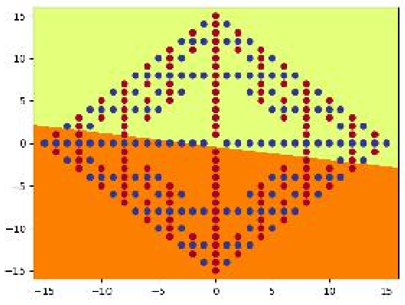
1_1



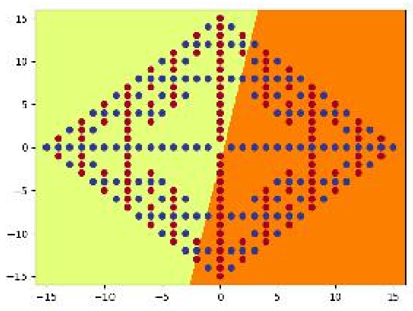
1_2



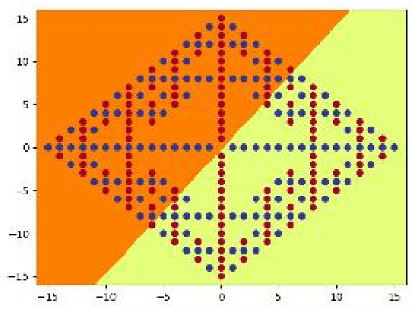
1_3



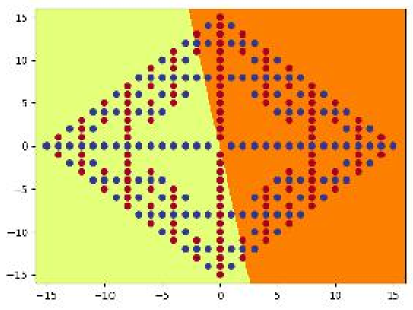
1_4



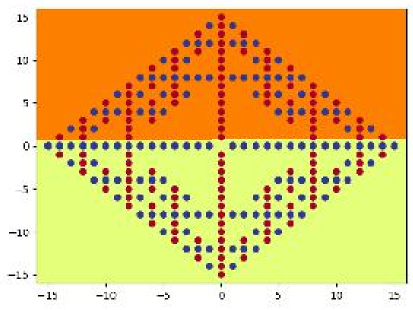
1_5



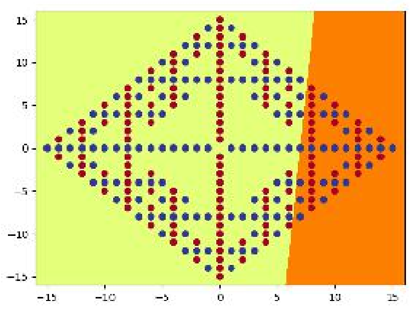
1_6



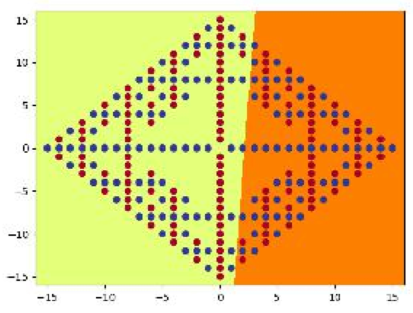
1_7



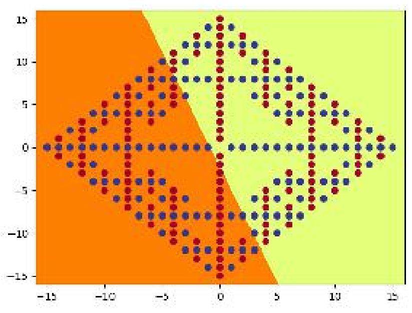
1_8



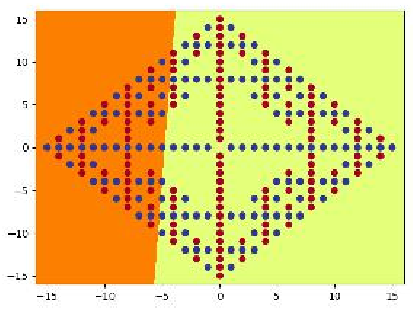
1_9



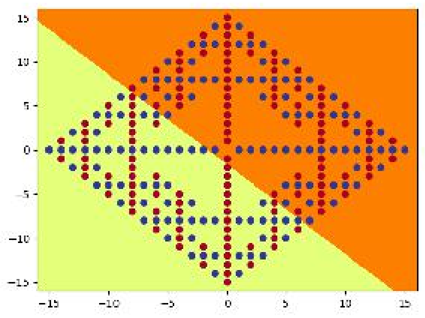
1_10



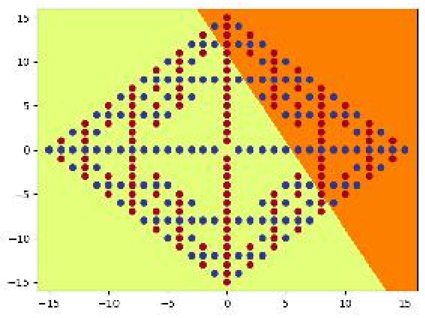
1_11



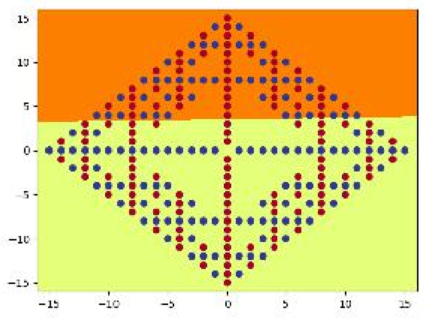
1_12



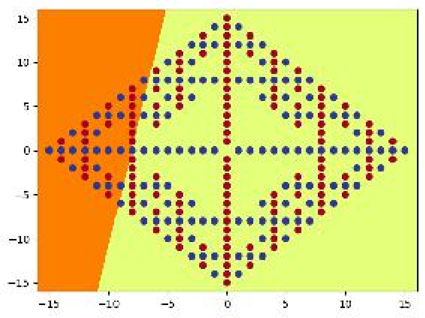
1_13



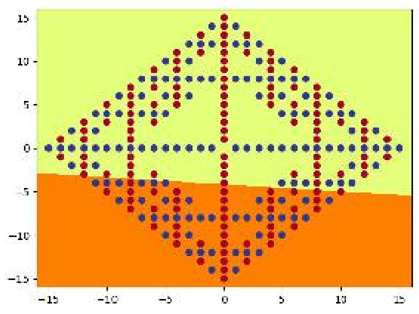
1_14



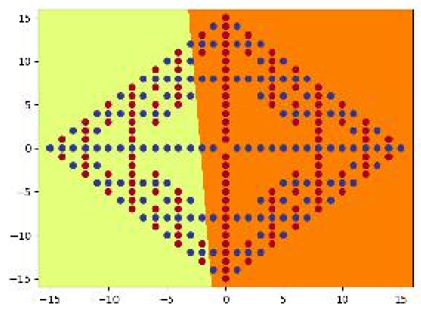
1_15



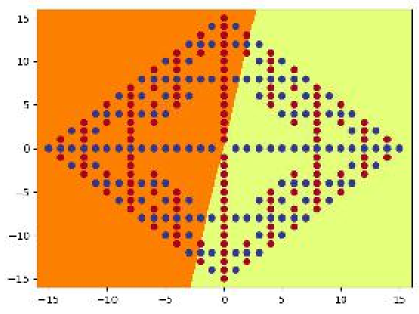
1_16



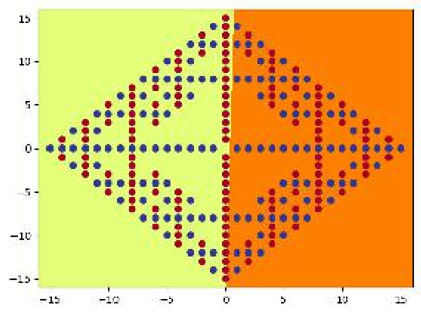
1_17



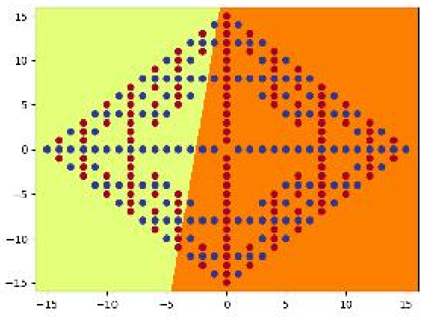
1_18



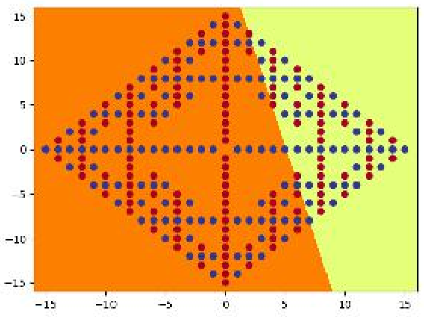
1_19



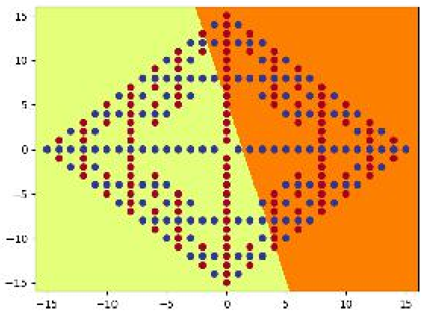
1_20



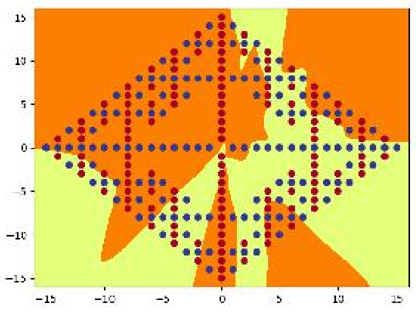
1_21



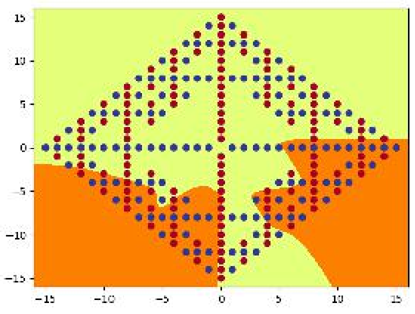
1_22



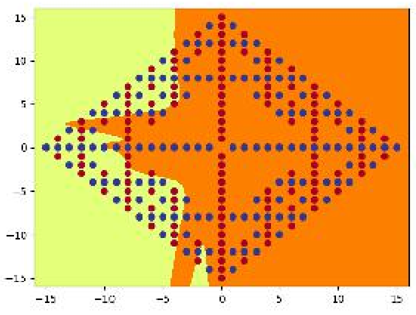
2_0



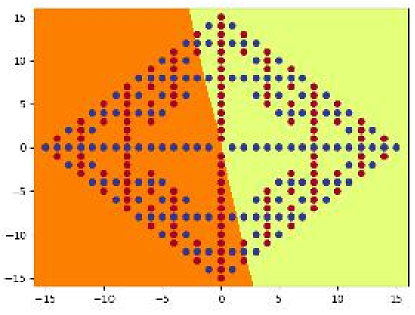
2_1



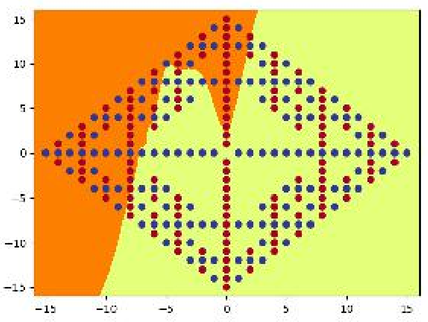
2_2



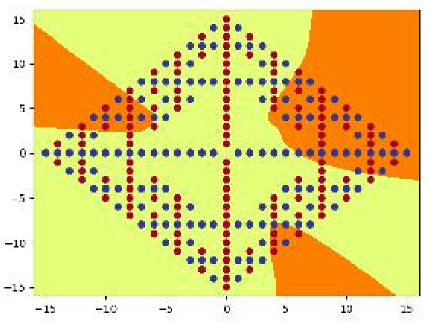
2_3



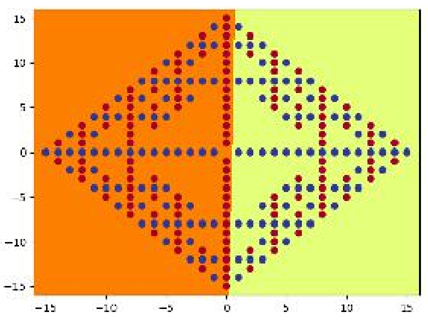
2_4



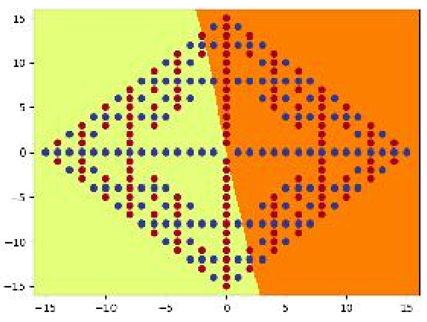
2_5



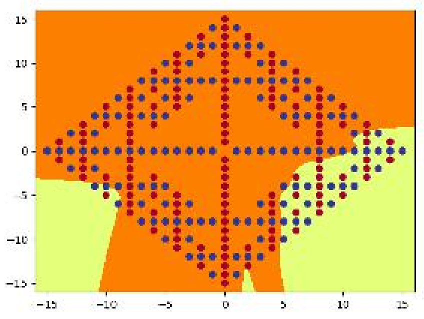
2_6



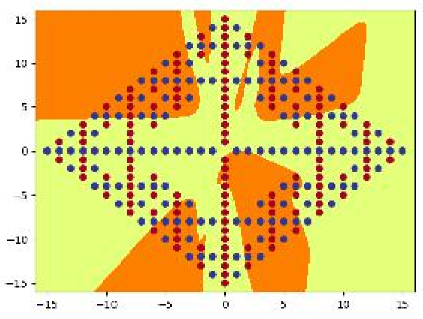
2_7



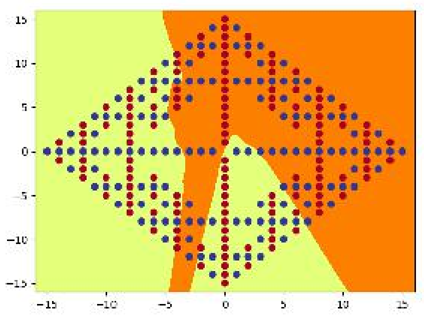
2_8



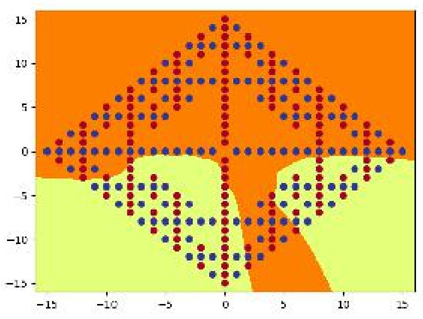
2_9



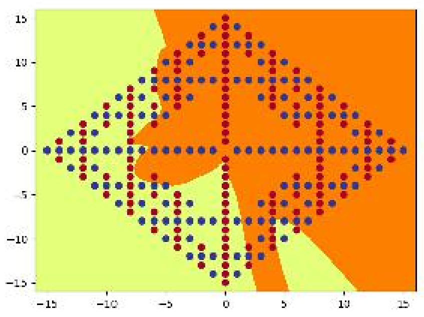
2_10



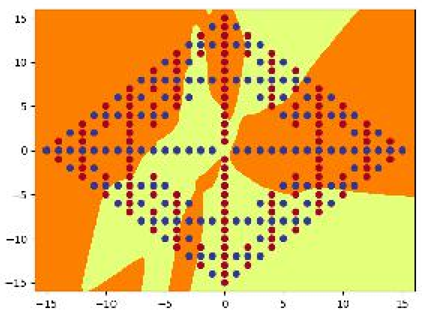
2_11



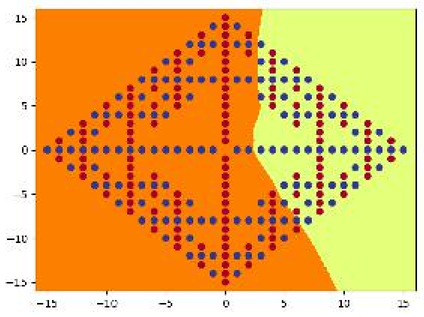
2_12



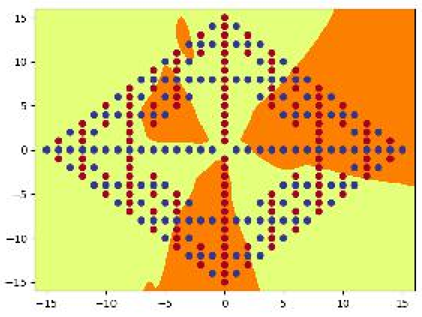
2_13



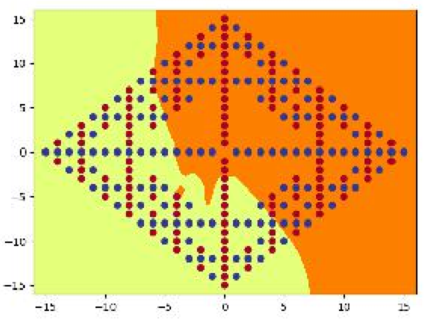
2_14



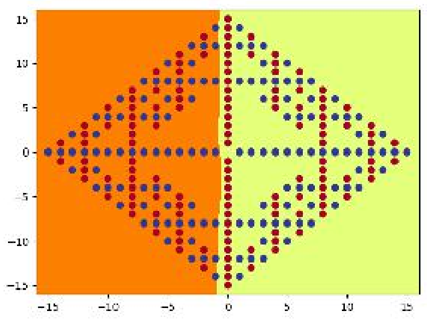
2_15



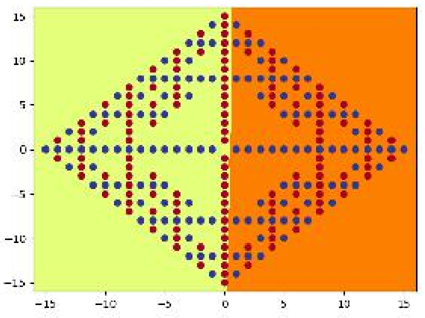
2_16



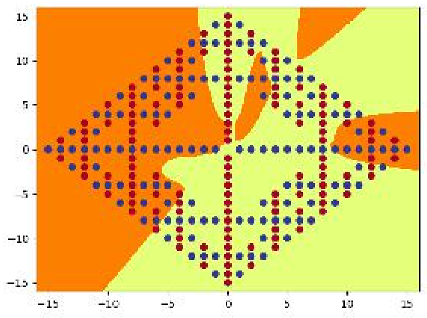
2_17



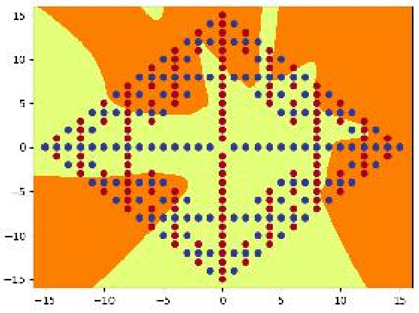
2_18



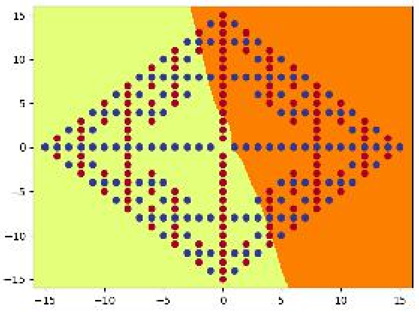
2_19



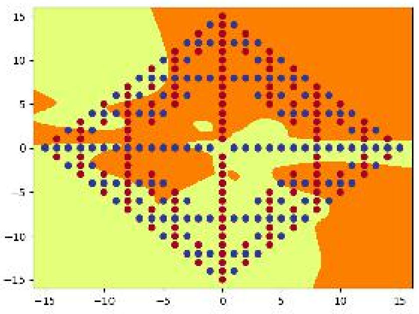
2_20



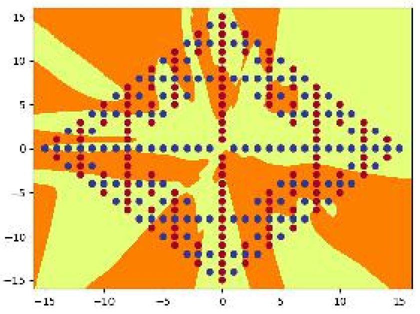
2_21



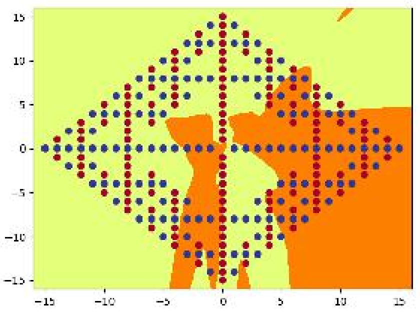
2_22



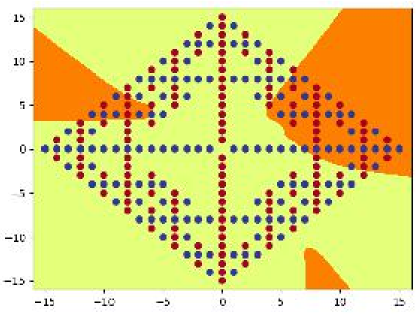
3_0



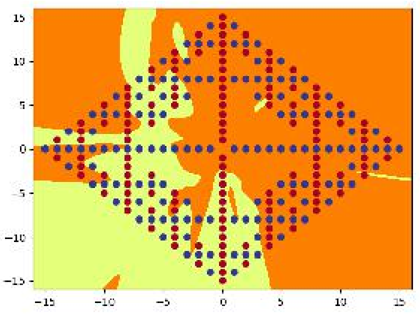
3_1



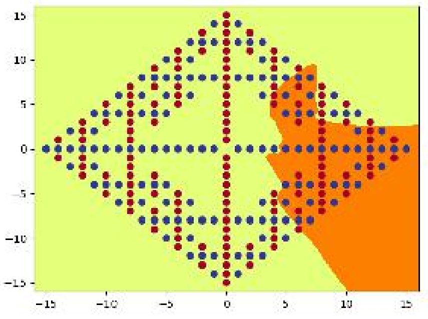
3_2



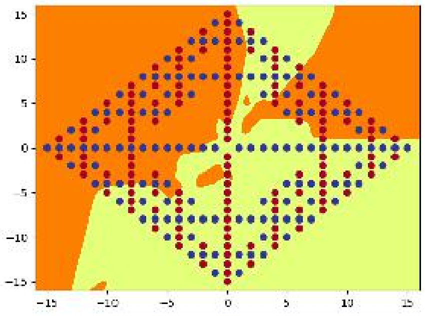
3_3



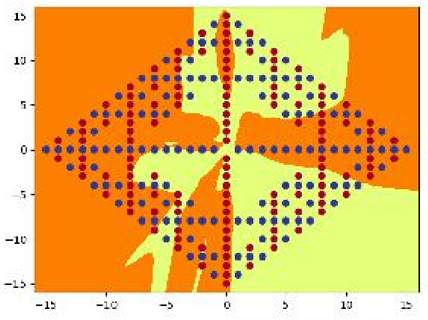
3_4



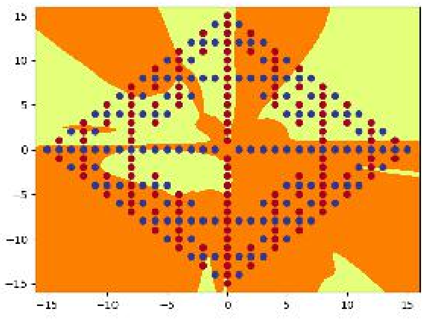
3_5



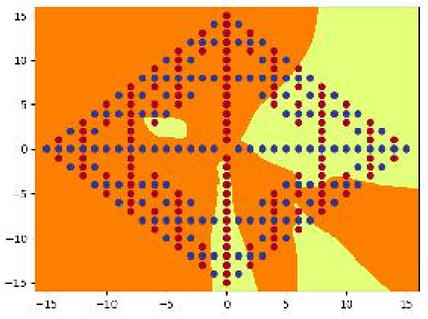
3_6



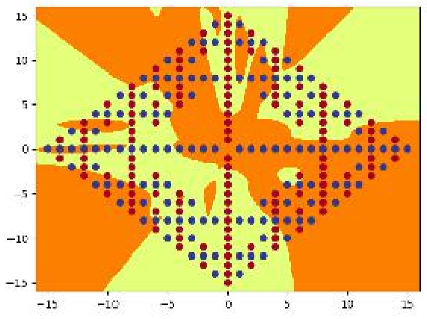
3_7



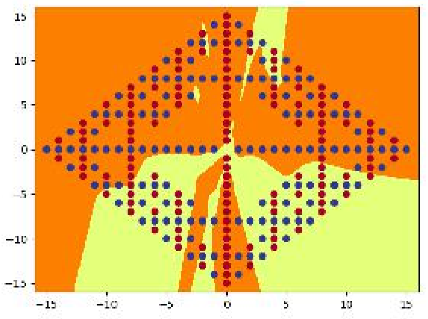
3_8



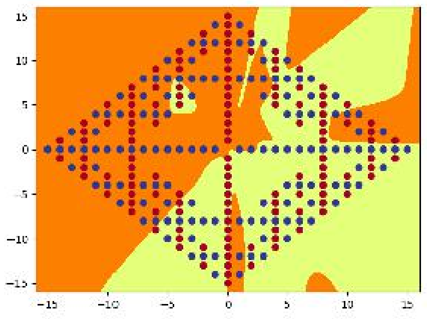
3_9



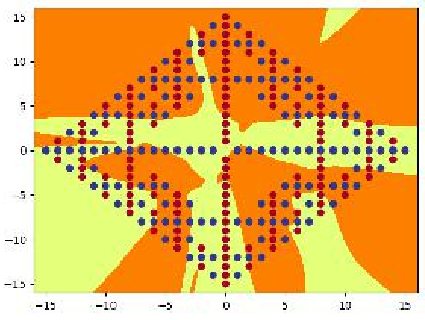
3_10



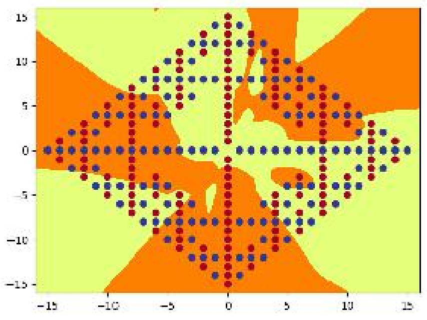
3_11



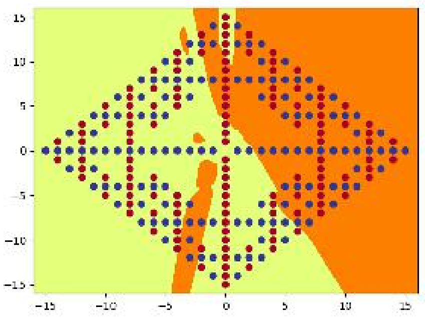
3_12



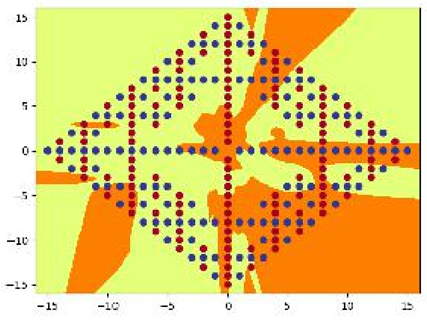
3_13



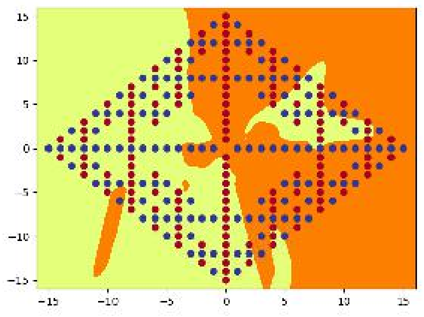
3_14



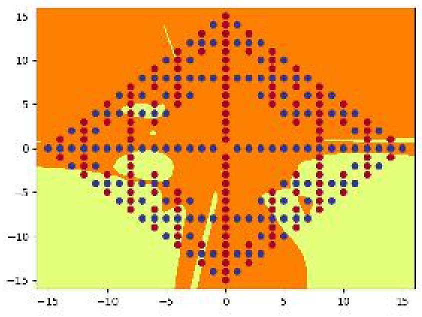
3_15



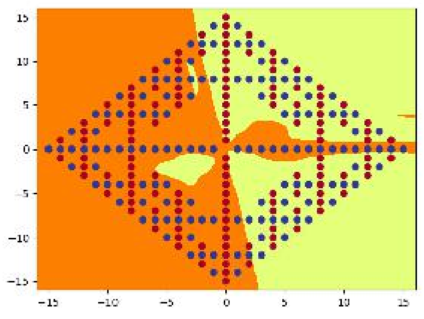
3_16



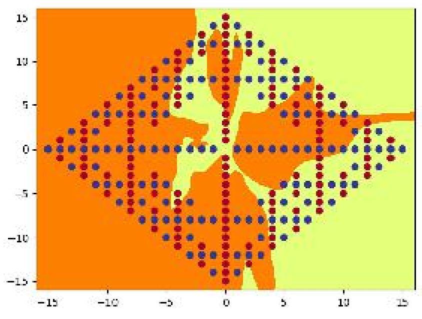
3_17



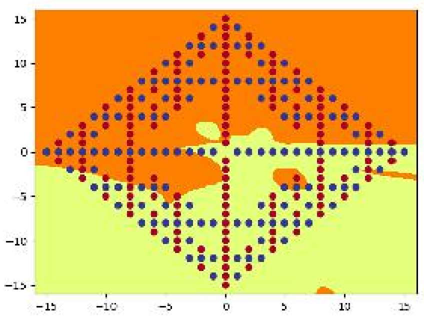
3_18



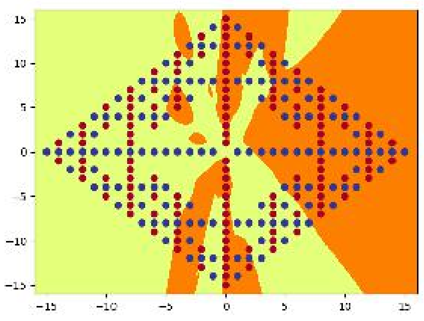
3_19



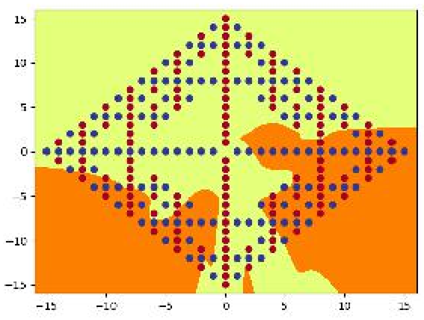
3_20



3_21



3_22



Q6.

The number of hidden nodes that I choose is 18. Learning rate = 0.01, Initial weight size = 0.1. Since there are shortcuts in the network, we still need to take those shortcuts into consideration when calculating independent parameters.

For the first fully-connected layer:
1.weights per filter = 1 + 2 = 3
2.number of neurons = 18
3.Connections = 18 × 3 = 54
4.independent parameters = 54

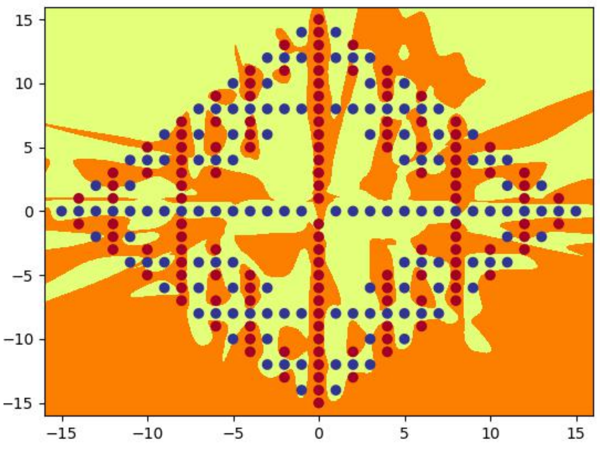
For the second fully-connected layer:
There is a shortcut between input layer and the second fully-connected layer. We first calculate independent parameters for this shortcut.
1.weights per filter = 1 + 2 = 3
2.number of neurons = 18
3.Connections = 18 × 3 = 54
4.independent parameters = 54
Then we calculate independent parameters of the second fully-connected layer without considering shortcut.
1.weights per filter = 1 + 18 = 19
2.number of neurons = 18

3.Connections = 18 × 19 = 342
4.independent parameters = 342
So total number of independent parameters of this layer is 342 + 54 = 396

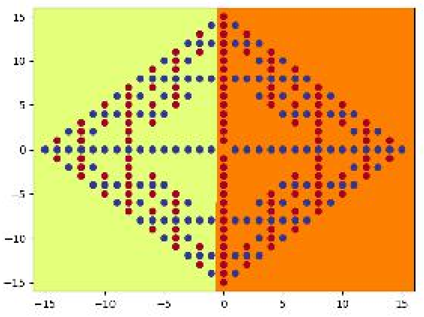
For the output layer:
There are two shortcuts on the output layer. We first calculate independent parameters for the first shortcut.
1.weights per filter = 1 + 2 = 3
2.number of neurons = 1
3.Connections = 3 × 1 = 3
4.independent parameters = 3
Then we calculate independent parameters for the second shortcut.
1.weights per filter = 1 + 18 = 19
2.number of neurons = 1
3.Connections = 1 × 19 = 19
4.independent parameters = 19
Then we calculate independent parameters for the output layer without considering shortcuts.
1.weights per filter = 1 + 18 = 19
2.number of neurons = 1
3.Connections = 1 × 19 = 19
4.independent parameters = 19
So total number of independent parameters of this layer is 3 + 19 + 19 = 41

So the total number of independent parameters in the network is 54 + 396 + 41 = 491

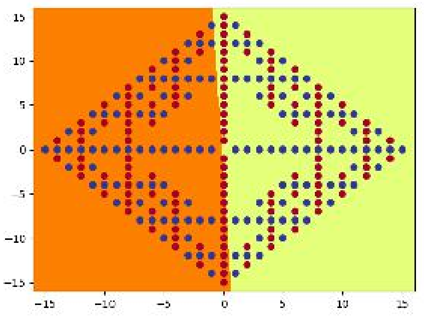
out_dense_23.png



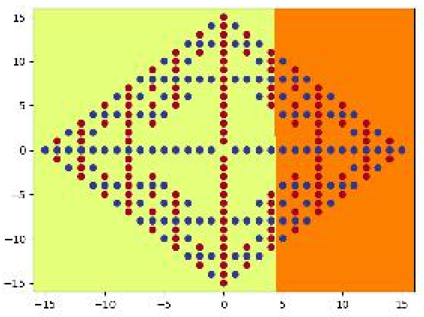
1_0



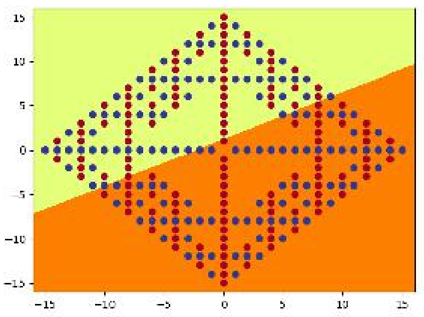
1_1



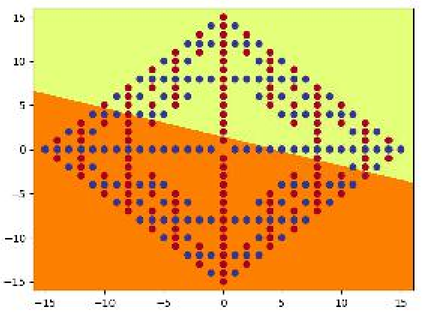
1_2



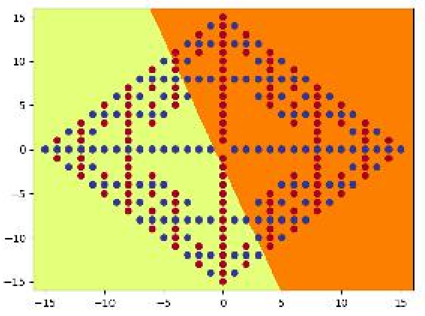
1_3



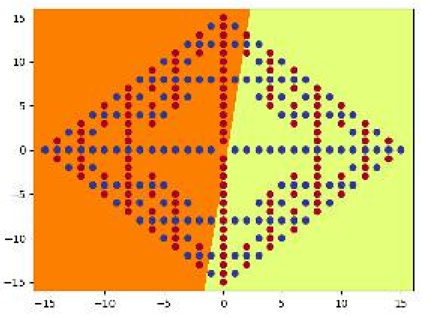
1_4



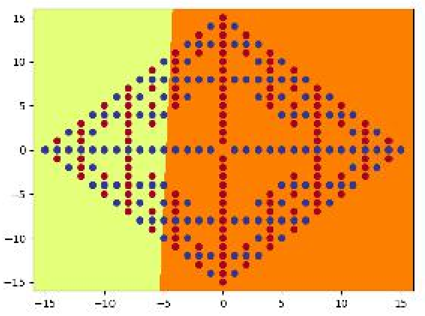
1_5



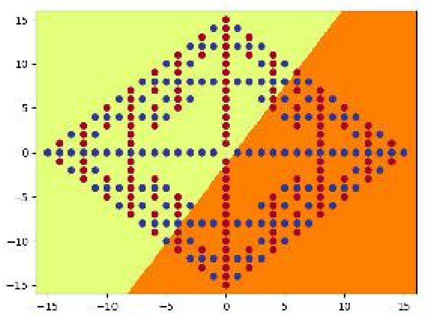
1_6



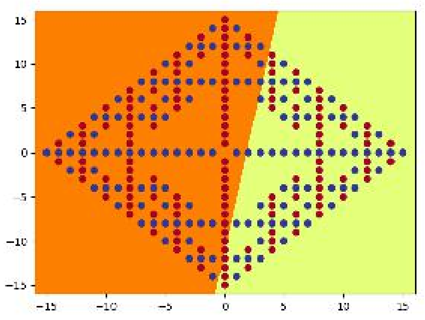
1_7



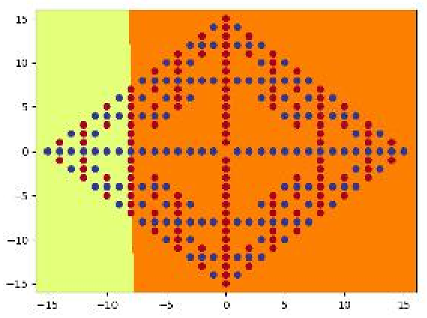
1_8



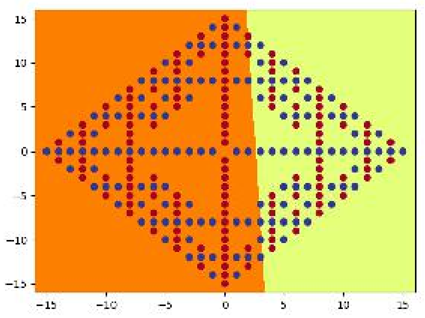
1_9



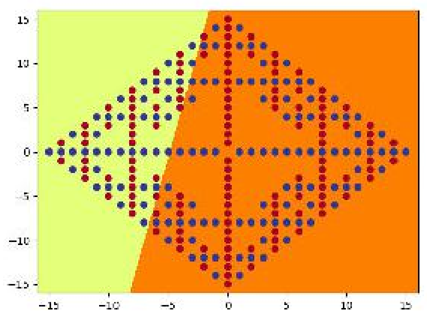
1_10



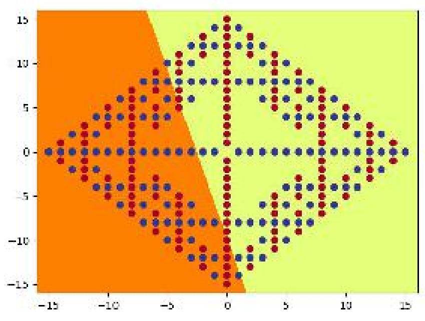
1_11



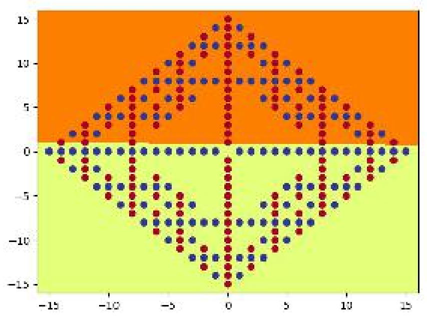
1_12



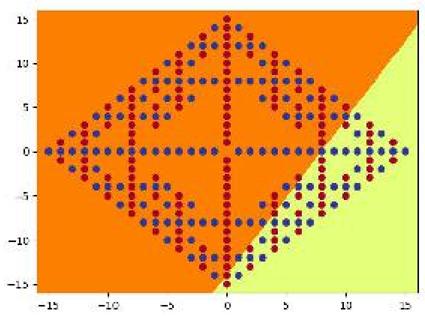
1_13



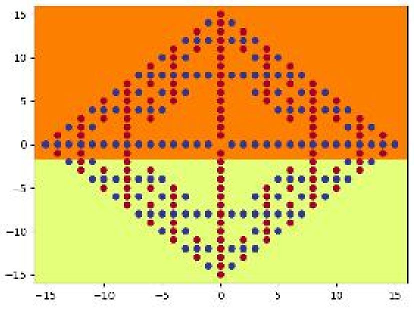
1_14



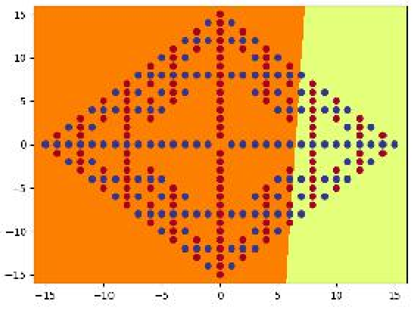
1_15



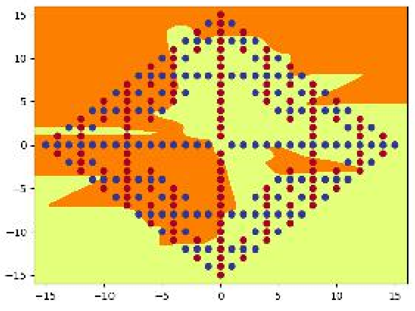
1_16



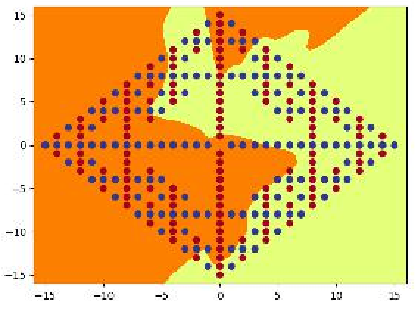
1_17



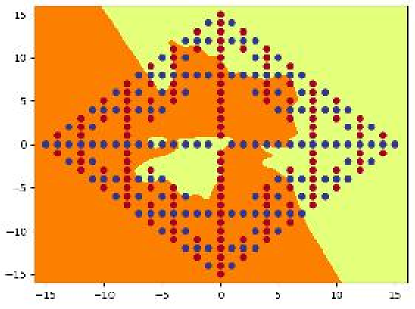
2_0



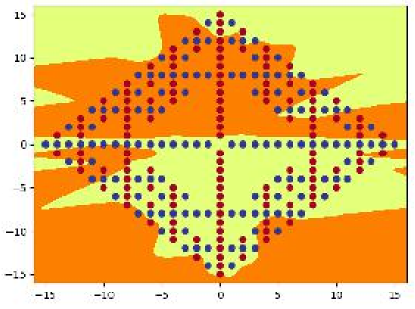
2_1



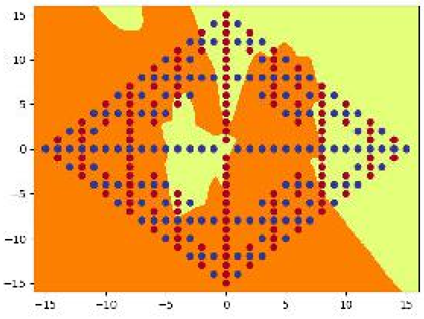
2_2



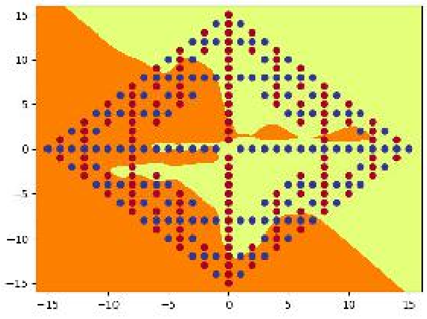
2_3



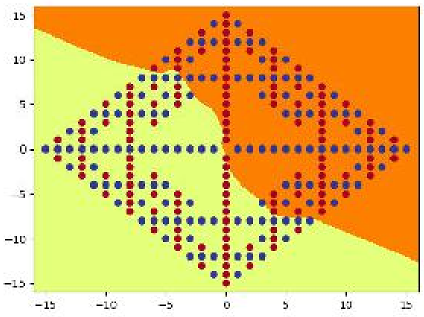
2_4



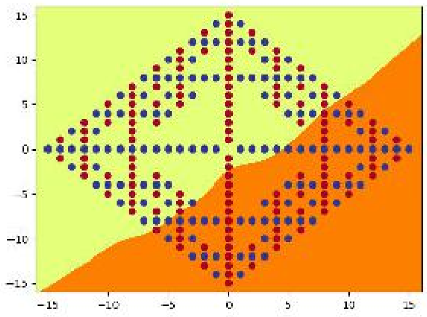
2_5



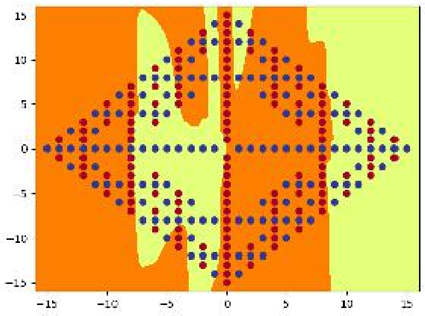
2_6



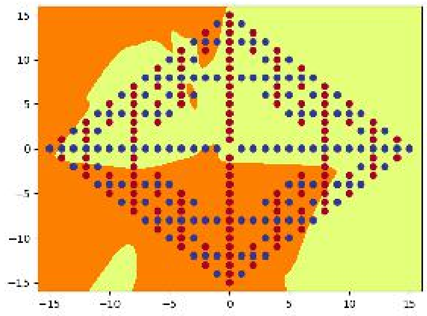
2_7



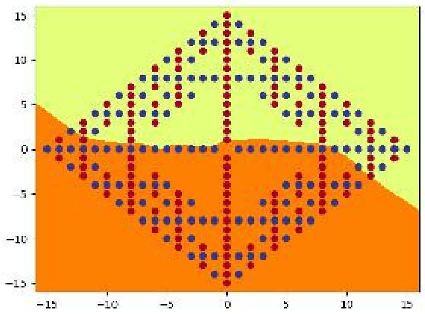
2_8



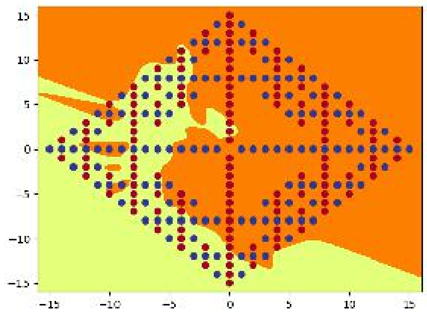
2_9



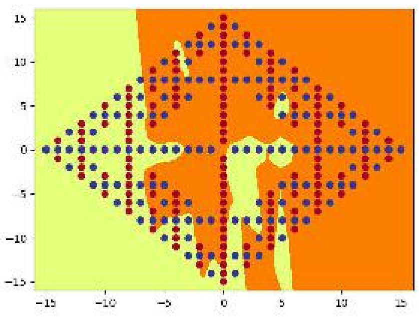
2_10



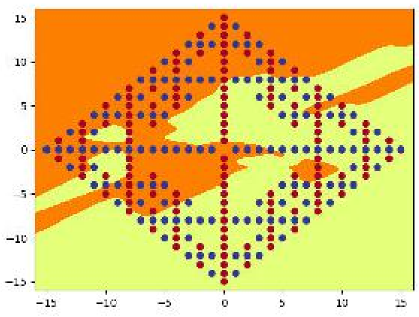
2_11



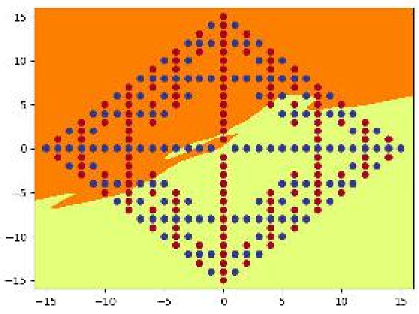
2_12



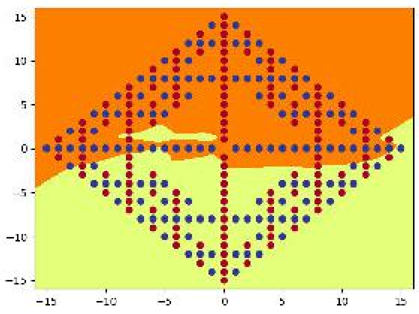
2_13



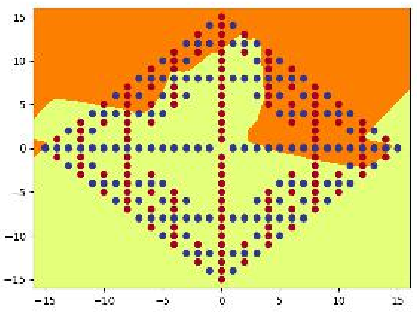
2_14



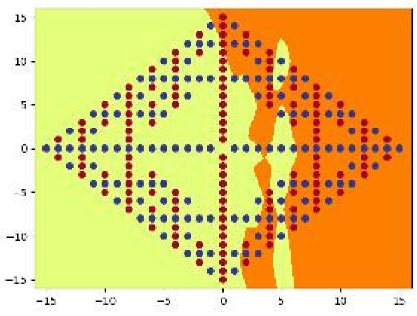
2_15



2_16



2_17



Q7

a.

Network structure	Independent parameters	Epochs
Full2Net	697	90000
Full3Net	1197	120000
DenseNet	491	57000

We can see that Densenet has the least number of independent parameters, and Full3Net has the most number of independent parameters. As for epochs, since Densenet has the least number of independent parameters, it has the least number of training epochs. Full3Net has the most number of independent parameters so it has the most number o training epochs.

b.

For Full3Net, the first fully-connected layer computes a function which is able to make linear classification and we can see that data points are divided by a line. The second fully-connected layer computes a function which is able to make non-linear classification and make linear classification at the same time. Compared with the first layer of the network, the second layer of the network extracts more detailed features about the input data. We can see from the generated plots that these hidden nodes have made some irregular classifications of the data. The third fully-connected layer computes a function which is able to make non-linear classification. Compared with the second layer, this layer extracts more detailed features to make the classification results more reliable. Output layer computes a function that generate the final correct classification result.

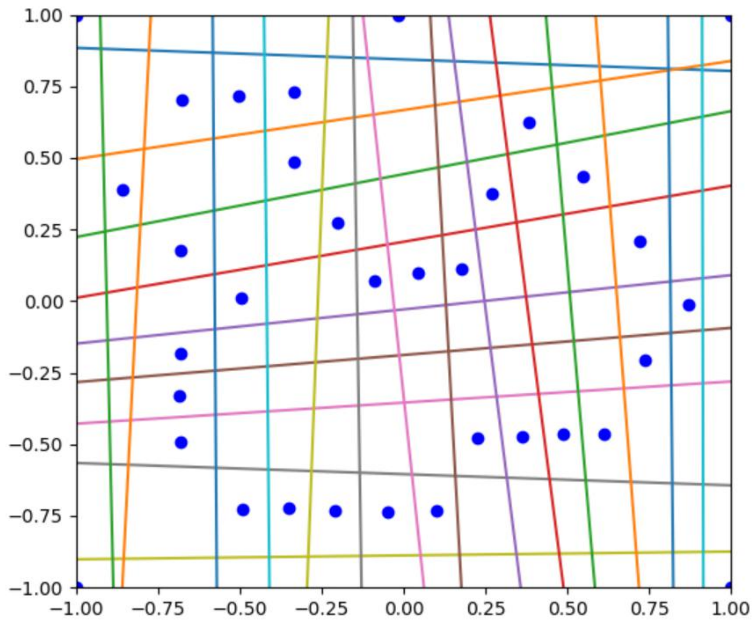
For DenseNet, the first fully-connected layer computes a function which is able to make linear classification and we can see that data points are divided by a line. The second fully-connected layer computes a function which is able to make non-linear classification. Due to the shortcut, the input of the second layer contains the output of the first layer and the training data, so the function calculated by the second layer is the result of combining these two different inputs. Output layer computes a function that generate the final correct classification result.

c.

From my observation, I think the output result of the output function calculated by DenseNet is cleaner and neater than the output result of the output function calculated by Full2Net and Full3Net. This may be because the existence of shortcut allows DenseNet to extract more detailed features. Compared with Full2Net, Full3Net has one more fully connected layer, which can extract more features, so the output result of Full3Net is less messy than Full2Net, even though it takes longer to train.

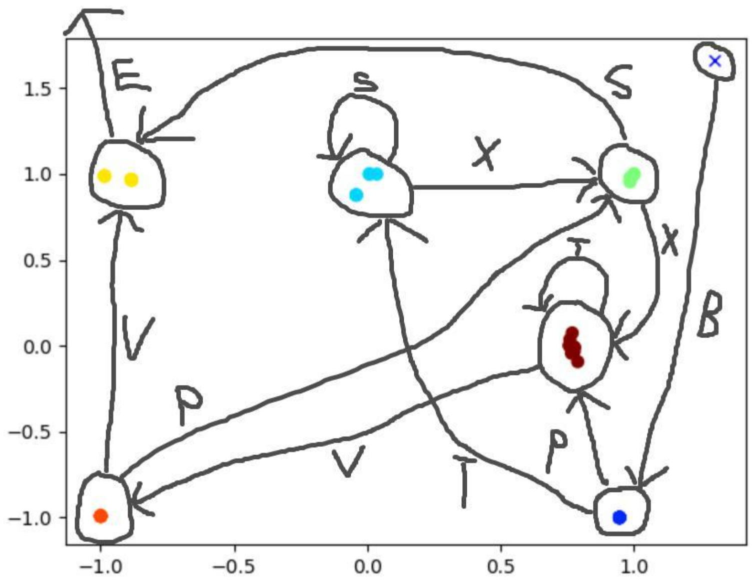
Part 3

Q1.

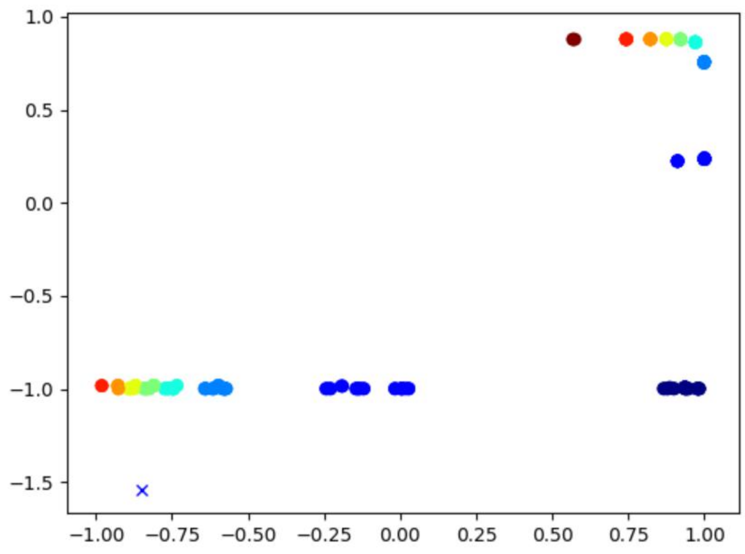


Part4

Q1.



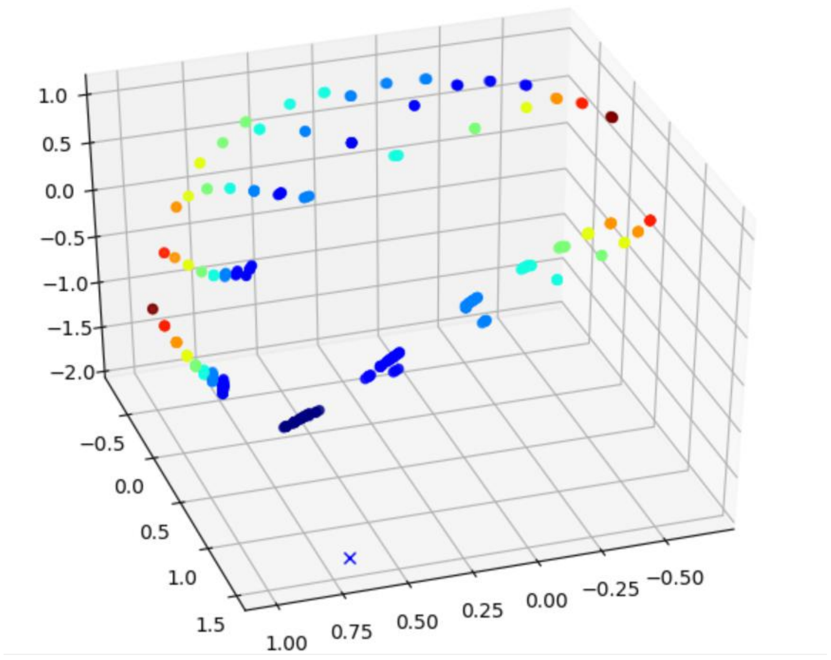
Q2.



Q3.

For the $a^n b^n$ task, the first b is not predictable, but subsequent b's and the initial a in the next subsequence are predictable. The network effectively counts up the a's as it oscillates towards the attractive fixed point and then counts down the same number of b's as it oscillates away from the repelling fixed point. When we meet the first a, we counts up the a's in the direction towards the attractive fixed point. When we meet the first b, we go to the repelling fixed point(b region). Then we predict b's and count down the number of b's and move away from the repelling fixed point until the count of b's is 0. Then we need to go back to the attractive fixed point(a region) and we can repeat above procedure until the string is fully predicted.

Q4.



Q5.

Similar to Q3, we can think that there are three fixed points in the network. The first b is unpredictable but subsequent b's and c's and the initial a in the next subsequence are predictable. When we meet the first a, the network effectively counts up the a's as it oscillates towards the attractive fixed point. When we meet the first b, we go to the repelling fixed point(b region). Then we count down the number of b's and move away from the repelling fixed point until the count of b's is 0. When we meet the first c, we go to the repelling fixed point(c region). Then we count down the number of c's and move away from the repelling fixed point until the count of c's is 0. When the count of c's is 0, we would jump back to a point in the "a region" and restart above procedure for next subsequence. This whole procedure would end when the entire string is predicted.