HOMEWORK 1

COMP 9444 Neural Network

UTTKARSH SHARMA
Z5269665

COMP9444 Neural Networks and Deep Learning

Question 1 :

Part 1

The final accuracy and confusion matrix are as follows :

Confusion Matrix :

[[768. 5. 7. 12. 31. 64. 2. 62. 30. 19.]

[6. 670. 106. 18. 28. 23. 60. 12. 25. 52.]

[8. 64. 692. 26. 25. 20. 46. 36. 46. 37.]

[5. 36. 63. 757. 15. 54. 13. 18. 28. 11.]

[62. 53. 80. 19. 619. 19. 34. 37. 20. 57.]

[7. 26. 127. 17. 20. 725. 26. 9. 33. 10.]

[5. 24. 147. 9. 24. 24. 726. 19. 8. 14.]

[18. 28. 25. 12. 82. 16. 55. 625. 90. 49.]

[11. 40. 96. 44. 6. 29. 47. 6. 702. 19.]

[9. 52. 92. 3. 52. 30. 19. 29. 41. 673.]]

Accuracy :

Test set: Average loss: 1.0101, Accuracy: 6957/10000 (70%)

Part 2:

The final accuracy and confusion matrix are as follows :

Confusion Matrix :

[[851. 4. 1. 8. 30. 34. 1. 34. 28. 9.]

[5. 808. 34. 6. 22. 12. 57. 4. 23. 29.]

[9. 10. 826. 37. 14. 18. 27. 15. 25. 19.]

[3. 6. 38. 905. 3. 18. 5. 4. 7. 11.]

[44. 29. 20. 7. 800. 11. 30. 18. 20. 21.]

[8. 11. 69. 11. 11. 842. 25. 2. 13. 8.]

[3. 12. 50. 8. 14. 6. 893. 3. 2. 9.]

[13. 14. 28. 5. 28. 10. 33. 809. 30. 30.]

[9. 33. 27. 41. 7. 14. 27. 1. 837. 4.]

[7. 21. 52. 5. 34. 9. 17. 17. 9. 829.]]

Accuracy :

Test set: Average loss: 0.5181, Accuracy: 8400/10000 (84%)

Part 3 :

Confusion Matrix :

The final output is as follows :

[[963. 2. 2. 1. 10. 1. 1. 12. 6. 2.]

[4. 925. 15. 1. 11. 1. 30. 1. 4. 8.]

[9. 4. 902. 28. 8. 12. 12. 4. 13. 8.]

[3. 0. 13. 965. 2. 6. 6. 1. 2. 2.]

[18. 7. 6. 6. 920. 11. 12. 5. 12. 3.]

[2. 8. 29. 6. 4. 931. 11. 1. 2. 6.]

[4. 3. 21. 4. 8. 0. 956. 2. 0. 2.]

[15. 2. 9. 5. 1. 2. 10. 937. 5. 14.]

[8. 5. 7. 12. 1. 5. 1. 1. 960. 0.]

[11. 4. 14. 4. 4. 2. 6. 2. 5. 948.]]

Accuracy :

Test set: Average loss: 0.3510, Accuracy: 9407/10000 (94%)

The output of 10 test cases:

Test Case Number	Accuracy
1	93
2	94
3	94
4	94
5	94
6	94
7	94
8	93
9	94
10	94

Part 4 :

The following are the major learnings from this project :

1. The approach for learning the models :
    a. The takeaway was to first try to find a base model, which works, then in incremental steps try to improve upon the accuracy of the model.
2. The structure of the Neural Networks :
    a. The effects that are observed amongst different Neural Networks when hidden units are employed and how they fare relative to each others
3. The effect of meta-parameters:
    a. The meta parameters are an important tool that need to be used wisely, they have the power to both improve and effect the values, of the accuracy.

In essence we follow the principle of Ockam's Razor, and first find the greedy solution and then improve upon it.

Part a)

The accuracy was improved significantly across the three models, the following table shows the accuracy of the models.

Model	Accuracy
Linear	70
Fully Connected	84
Convolutional	93

As can be observed the effect of adding the hidden nodes in the models had a significant increase in the accuracy of the models.

Also, this assignment showed how the architecture of the Neural networks can significantly improve upon the accuracy of the network.

Part b)

Comparing the three confusion matrices, we can observe along the diagonals that the relative accuracy is increasing when we transition towards the convolutional neural network.

The reason for that being that in CNN layer ten to learn more complex features leveraging the hidden nodes of the CNN layer.

Upon checking the confusion matrix, it can be observed that there are cases where the predicted values are relatively close to each other, hence there is a possibility o the model predicting wrong values.

For eg :

[[768. 5. 7. 12. 31. 64. 2. 62. 30. 19.]

[6. 670. 106. 18. 28. 23. 60. 12. 25. 52.]

[8. 64. 692. 26. 25. 20. 46. 36. 46. 37.]

[5. 36. 63. 757. 15. 54. 13. 18. 28. 11.]

[62. 53. 80. 19. 619. 19. 34. 37. 20. 57.]

[7. 26. 127. 17. 20. 725. 26. 9. 33. 10.]

[5. 24. 147. 9. 24. 24. 726. 19. 8. 14.]

[18. 28. 25. 12. 82. 16. 55. 625. 90. 49.]

[11. 40. 96. 44. 6. 29. 47. 6. 702. 19.]

[9. 52. 92. 3. 52. 30. 19. 29. 41. 673.]]

The values marked in yellow are fairly close to each other due to which there is a possibility for them to being predicted incorrectly.

Part c)

I did the changes of checking relative nodes and learning rates along with checking different functions the observations are as follows:

Linear :

Meta-Parameters/Hidden Nodes	***Accuracy/Loss***	***Observation***
Hidden Nodes =10, learning rate = 0.1	66/1.53	Here the accuracy is low and not withing the permissible limits also note the high loss values
Hidden Nodes = 10, learning rate = 0.001	66/1.2	The lowering learning rate proved useful in reducing the loss but did not improve upon the accuracy of the model
Hidden Nodes = 100, learning rate = 0.1	68/1.1	The accuracy increased with the increase in the hidden nodes but the loss function was not effected.

Hidden Nodes =500	70/1.0101	I tried to minimize the loss,and increased the value of the hidden nodes. Which led to the increase in the accuracy
Hidden Nodes = 1000	70/1.0202	Even when we increase the hidden nodes then the accuracy of the model was not effected.

Upon changing the parameters of the function, I observed that the while an increase in learning rate may prove to be useful but to was prone to get stuck in the local minima, that could be attributed due to the fact that it may miss out on important points of the minima for the plane of the neural network.

An increase in hidden nodes proved useful, but then the accuracy got saturated, which suggest the phenomenon of an increase in error rate upon having too many hidden nodes.

Fully Connected Model:

Meta-Parameters	Accuracy/Loss	Observation
Hidden Nodes =10	67/1.0657	10 Hidden Nodes proved to be quite less for this model and hence resulted in a lower accuracy score
Hidden Nodes = 50	81/0.6031	An increase in the hidden nodes, resulted in learning more features and hence accuracy significantly increased
Hidden Nodes = 75	83/0.5597	The hidden nodes proved useful which resulted in an increase of the accuracy
Hidden Nodes =100	84/0.5181	The vale of 100 was the most significant one where the model gave repeatedly successful results.
Hidden Nodes =140	84/0.50	The value of 140 while resulted in the increase of the accuracy but when ran multiple times resulted did not gave consistent results.

Increasing the hidden nodes proved quite useful for getting the most accurate results, while accuracy could be achieved at 90 hidden nodes, however when ran multiple times,

the accuracy was not consistent and hence the final model has 100 hidden nodes.

Convolution Network :

The test for this model, took a longer time as compared to the rest, below is the subset of the table comprising of the tests that were executed.

All the tests below have the kernel size of 5

In/out Channel	Hidden Nodes	Accuracy/Loss	Observation
1/5 Conv-1 5/12 Conv-2	100	91/0.4113	This was the base model which was incrementally improved upon
1/10 Conv-1 10/24 Conv-2	100	92/0.4005	Incremental increase for the RGB layers led to a 1% increase in the accuracy.
1/5 Conv-1 5/12 Conv-2 Lr = 0.05	100	93/0.4602	While increasing the learning rate resulted in the higher accuracy but the results were not consistent.
1/10 Conv-1 10/24 Conv-2	50	0.3510/94	The accuracy was consistently observed with this layer
1/64 Conv-1 64/24 Conv-2	100	93/0.337	

Thus we observed the phenomenon as to how the learning rate both positively and adversely effects the model and how best to both tackle it and be cognizant of it.

Question 2 :

Part 2 :

I ran the code for multiple values of hidden nodes,how-ever two stood out the most and their tables are as follows :

Testing when Hidden Node are 6

Case Number	***Epochs***	***Loss/Accuracy***
1.	8700	0.1103/100
2.	20000	0.0235/77.84
3.	6000	0.1004/100
4.	6500	0.0780/100
5.	20000	0.0110/91.75
6.	20000	0.0234/77.84
7.	20000	0.0482/99.48
8.	5700	0.0856/100
9.	20000	0.0234/77.84
10.	9000	0.0216/100

As can be observed the accuracy is fluctuating a lot even changing the learning rate did not prove useful hence I tested for the accuracy when the hidden nodes were 7.

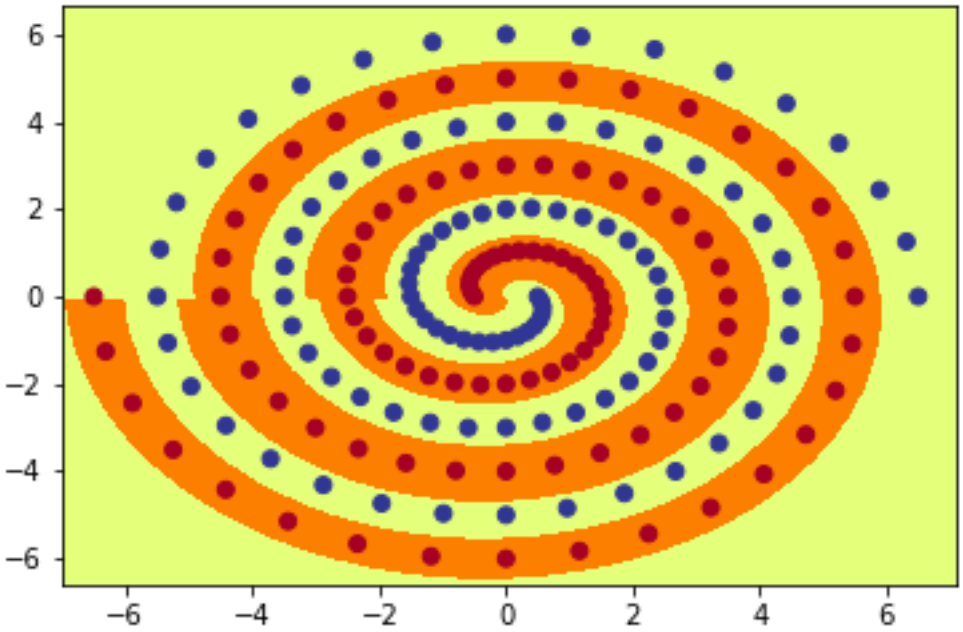
Testing when Hidden Node are 7.

Case Number	***Epochs***	***Loss/Accuracy***
1.	6300	0.1398/100
2.	5600	0.0361/100
3.	6200	0.1249/100
4.	3300	0.0241/100
5.	2500	0.0175/100
6.	11900	0.1351/100
7.	4800	0.1351/100
8.	5100	0.0101/100
9.	2900	0.0206/100
10.	9000	0.0216/100

As can be osbserved having hidden nodes of 7 gave consistent accuracy for the 10 times it ran.

Each time care was taken that the kernels of the IDE were restarted so that the initial weights are accuracte.

The polar net output is as follows :



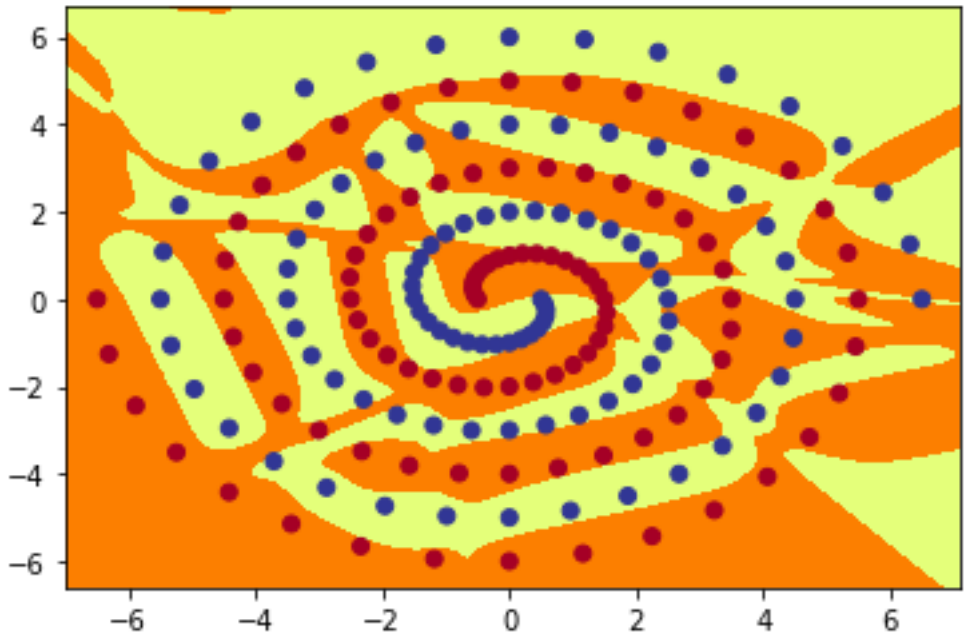
Part 4 :

I ran multiple tests for the layers and following are the subset of the results

Testing when Hidden Node are 10 init weights =0.22 and learning rate =0.01.

Case Number	***Epochs***	***Loss/Accuracy***
1.	18500	0.0149/100
2.	6100	0.0127/100
3.	9200	0.0454/100
4.	7300	0.0127/100
5.	12700	0.0140/100
6.	20100	0.0164/97.94
7.	6500	0.0218/100
8.	6500	0.0218/100
9.	18450	0.0140/100
10.	15235	0.0123/100

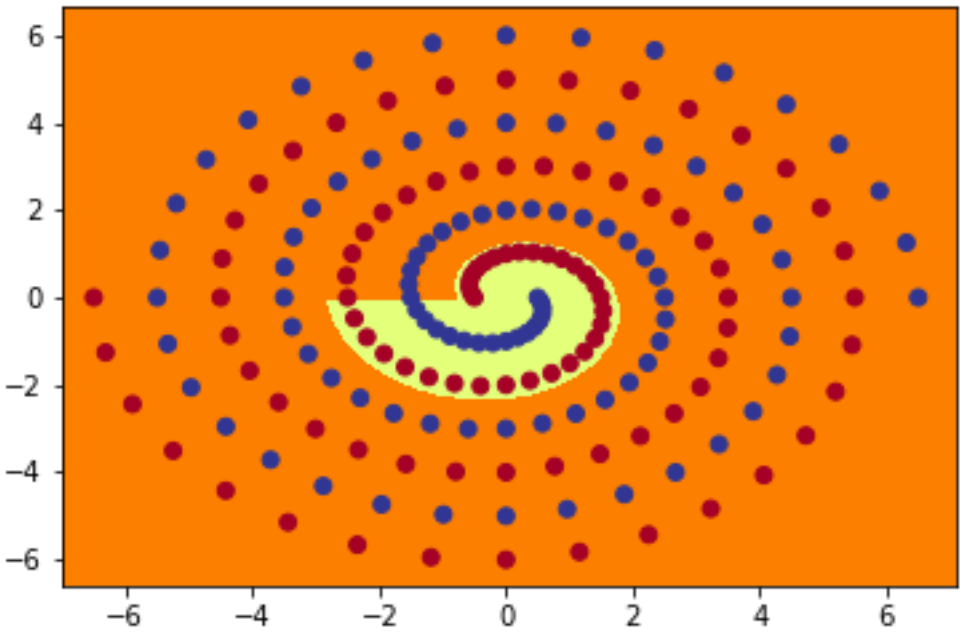
For this model, I ran the above mentioned configuration 10 times and observed the accuracy of 100 for 9/10 models and the following is the image :

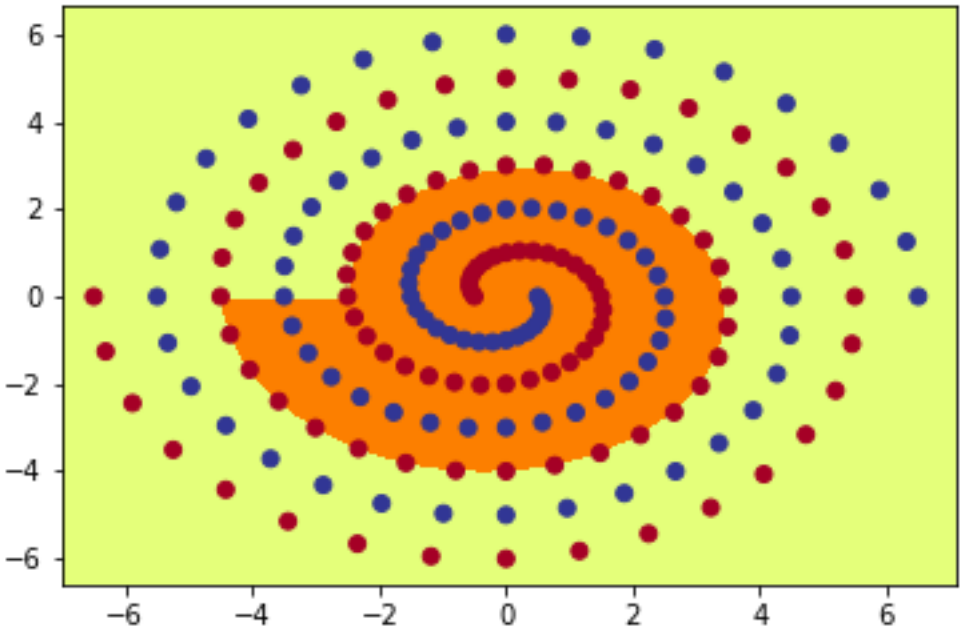


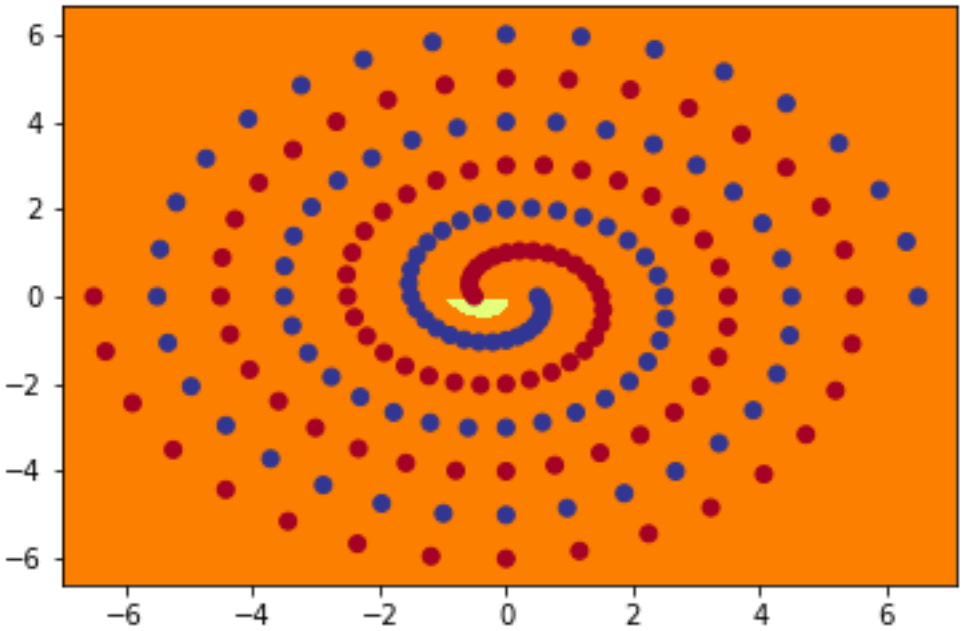
Part 5 :

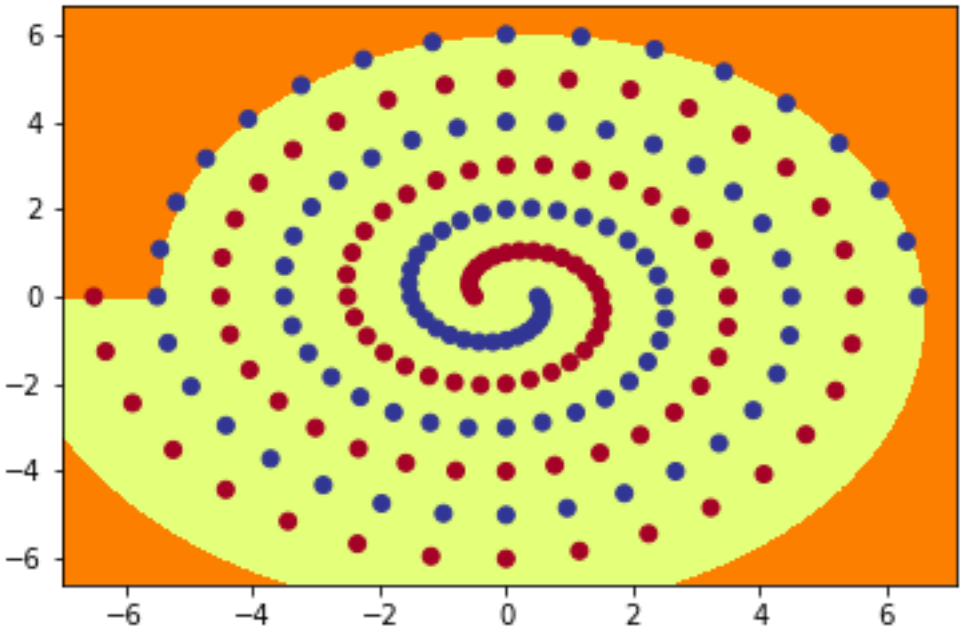
The Layer Nodes images are as follows :

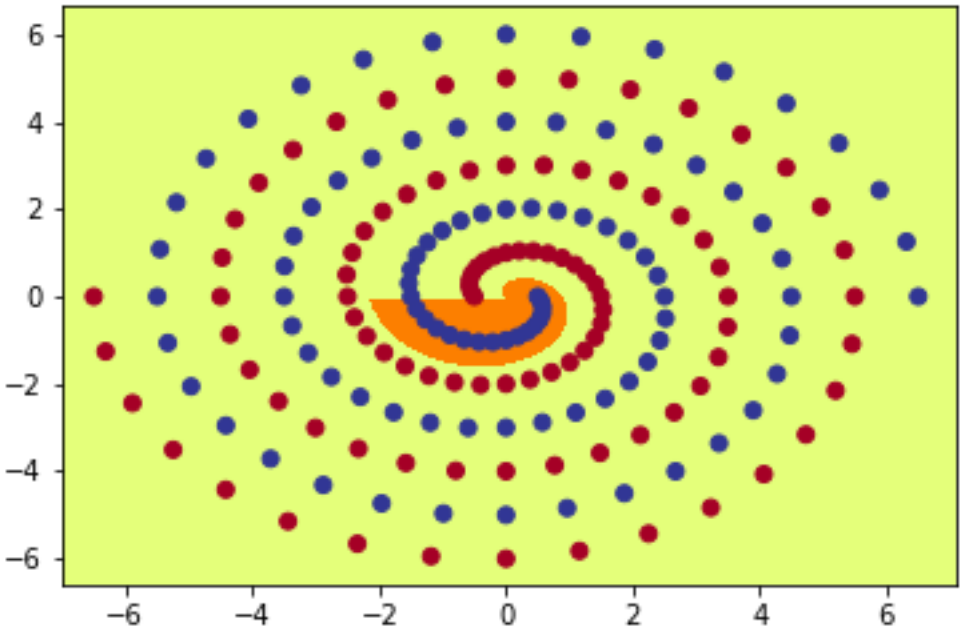
For Polar Net :

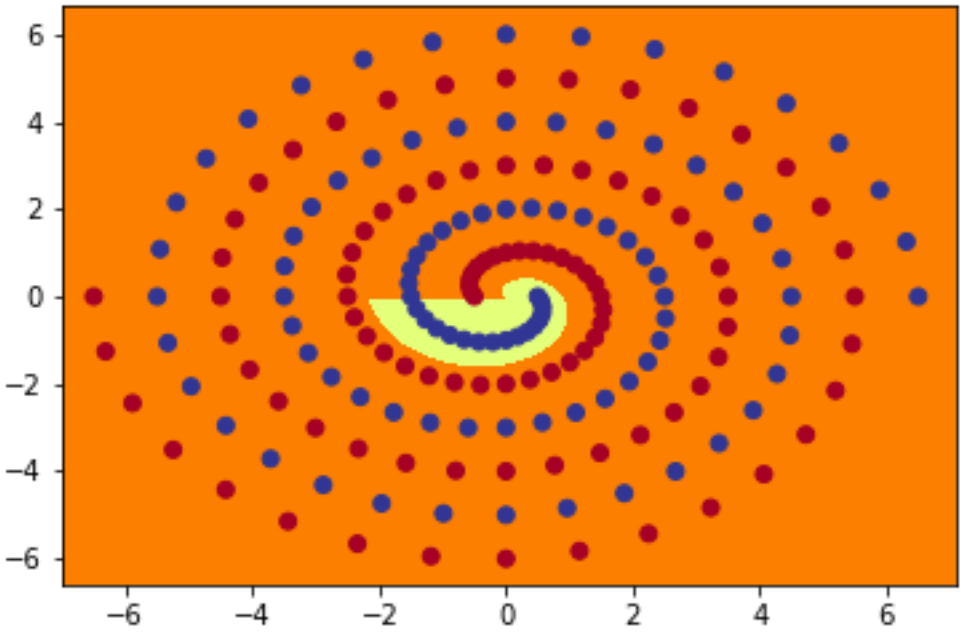


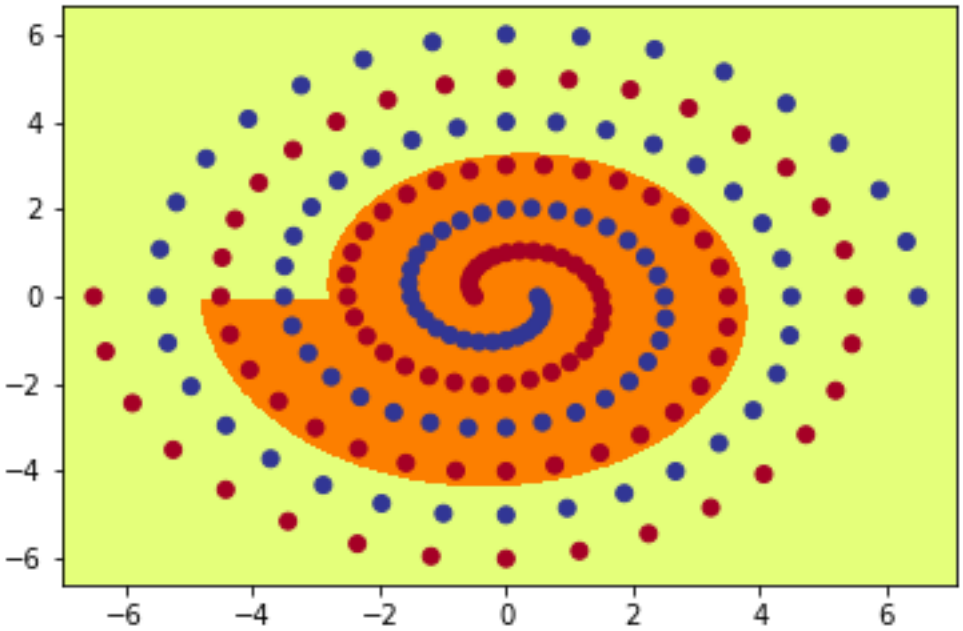

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

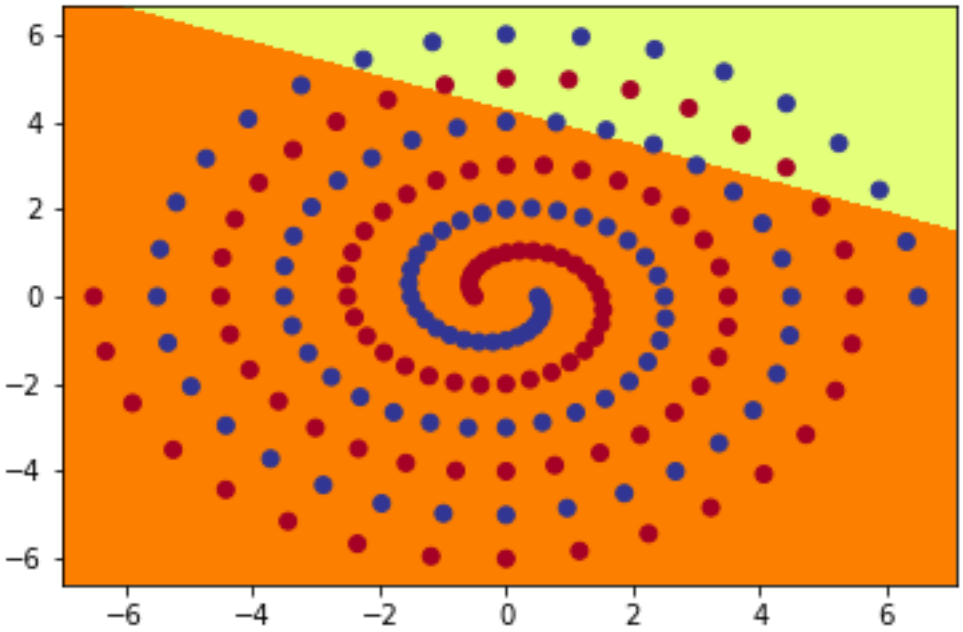

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

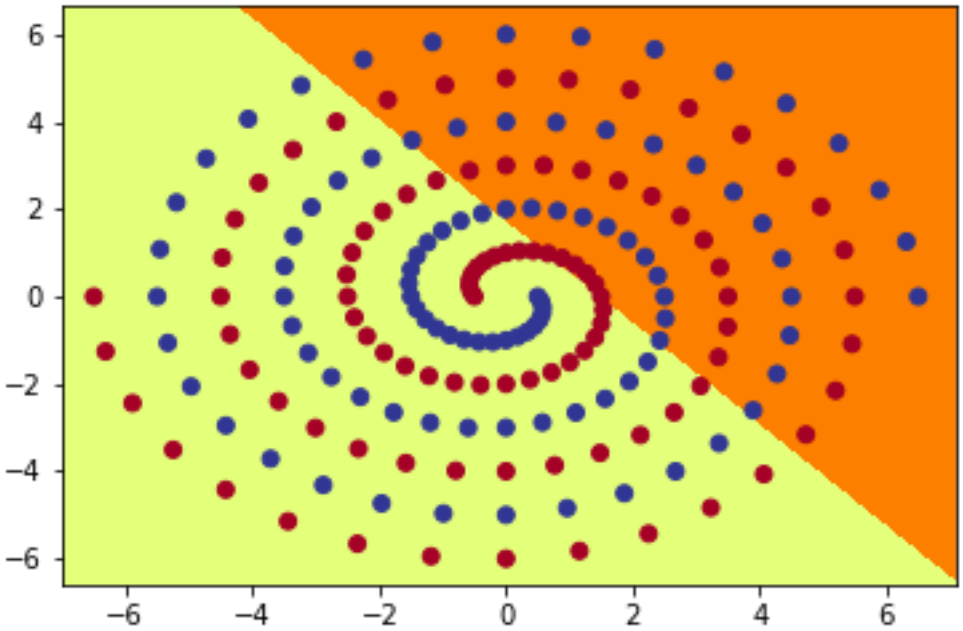


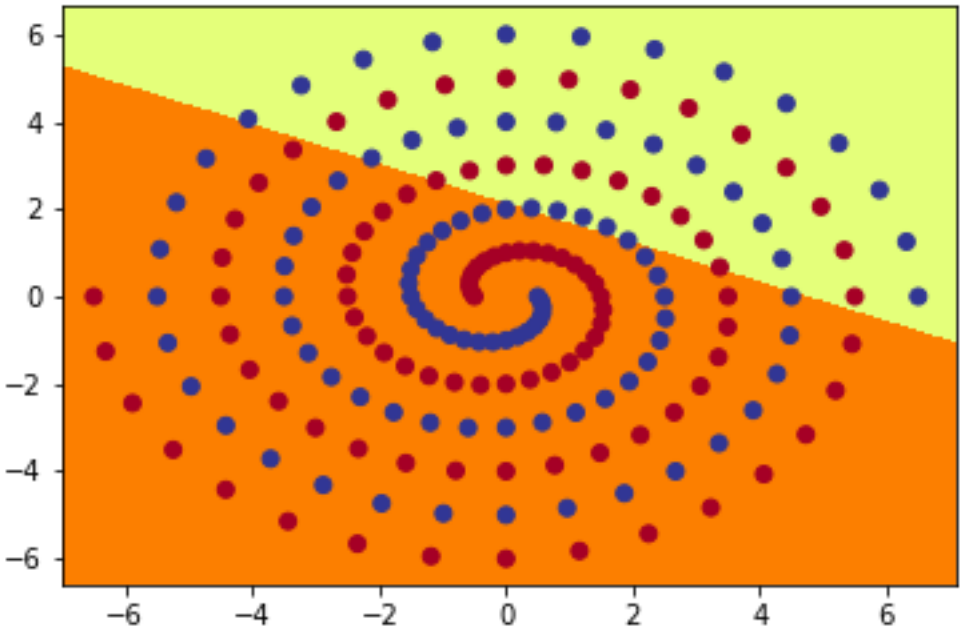


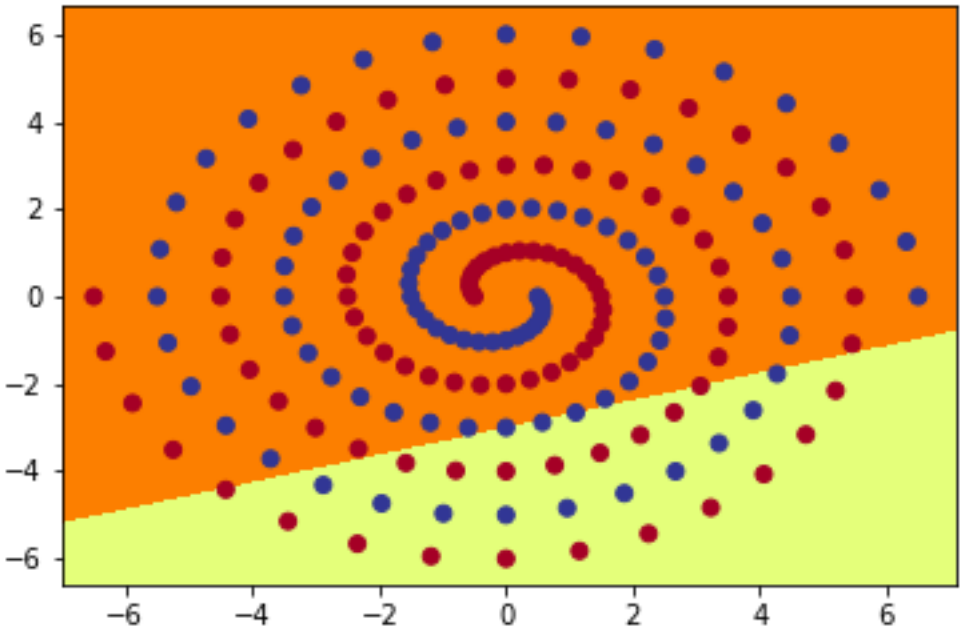
For Raw Net:

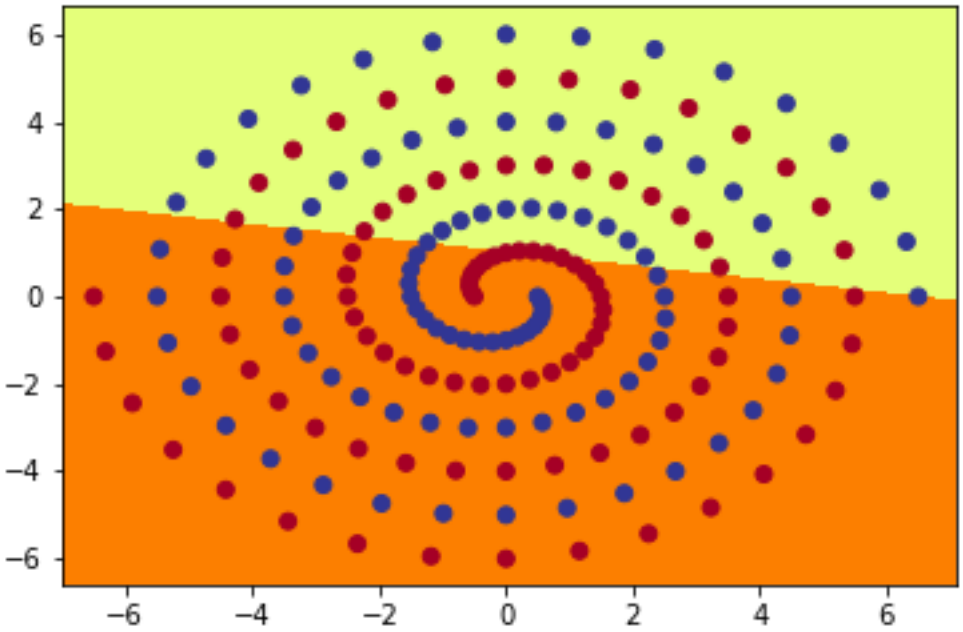
Layer Node : 1 Images

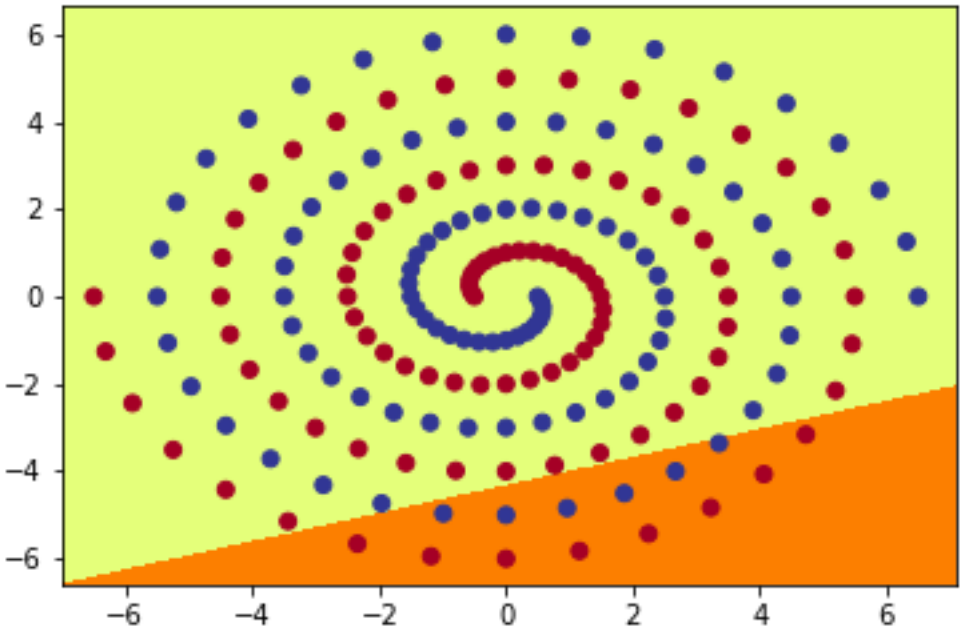

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

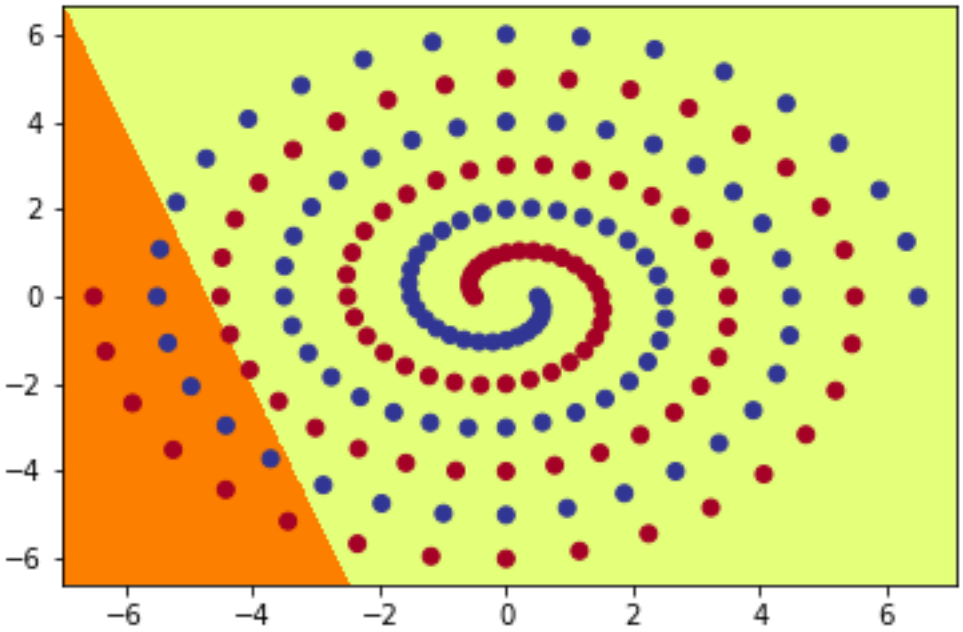

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

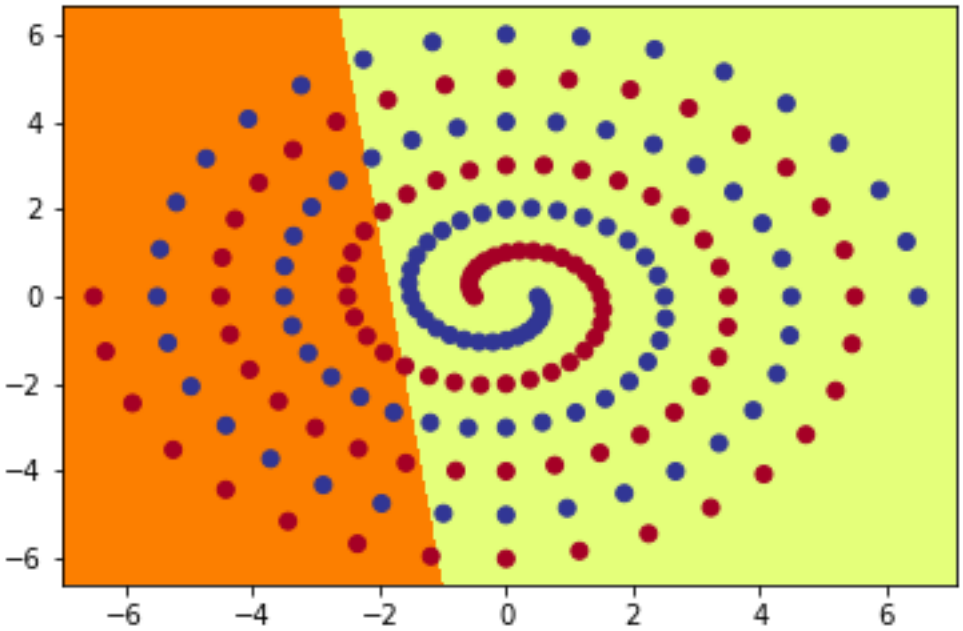

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

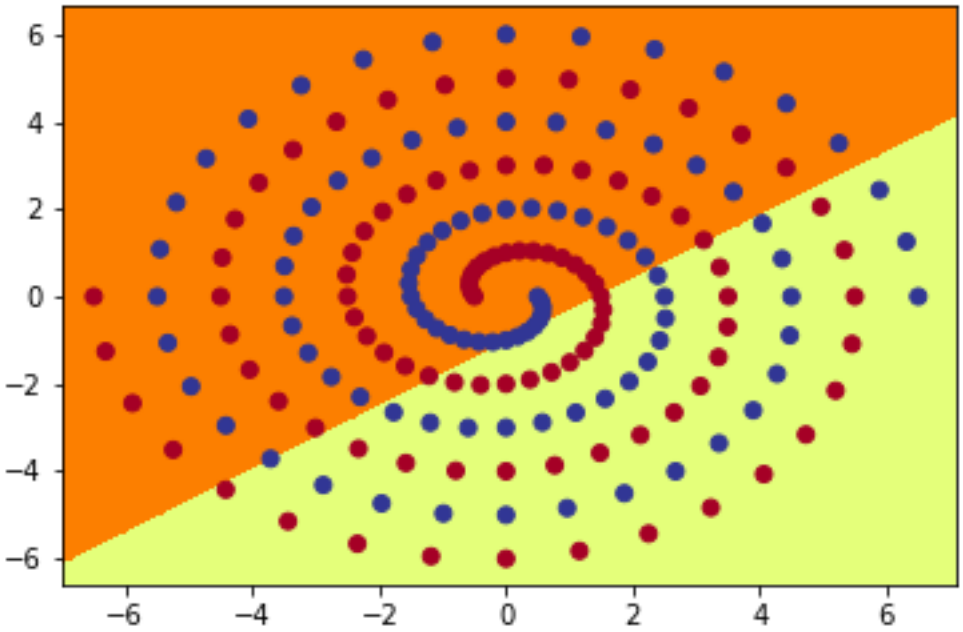

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

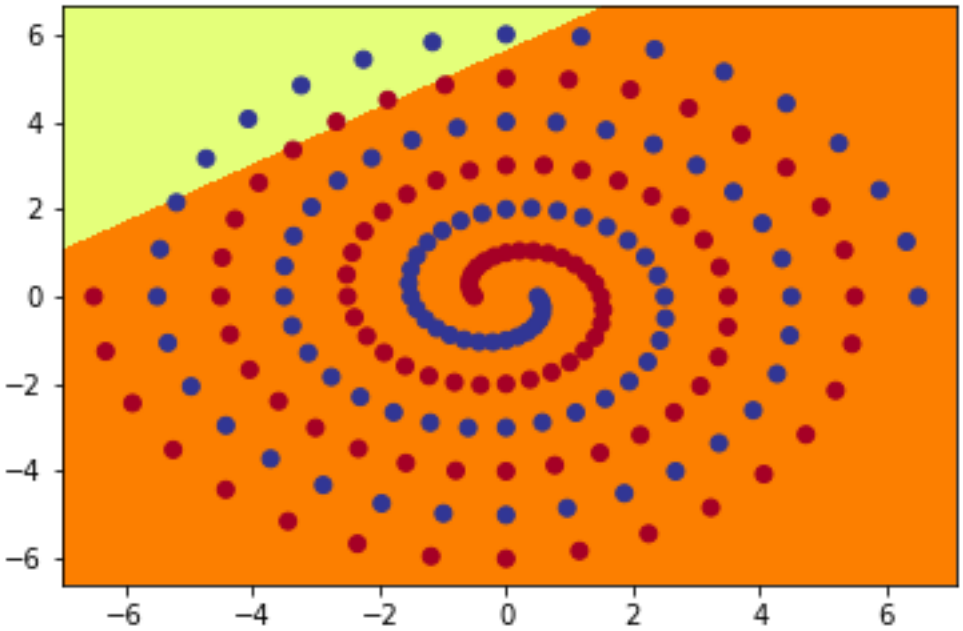

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

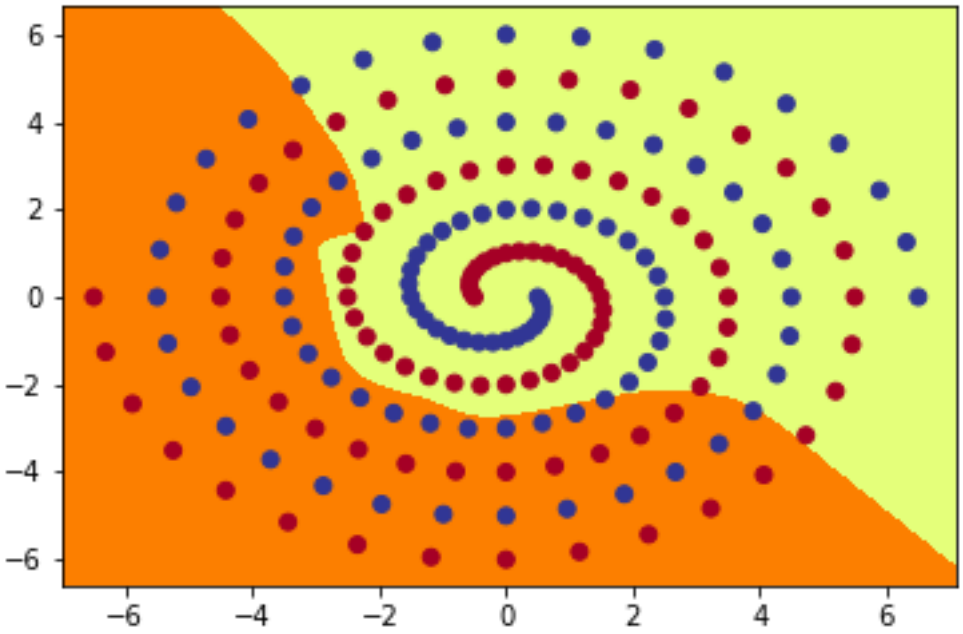

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

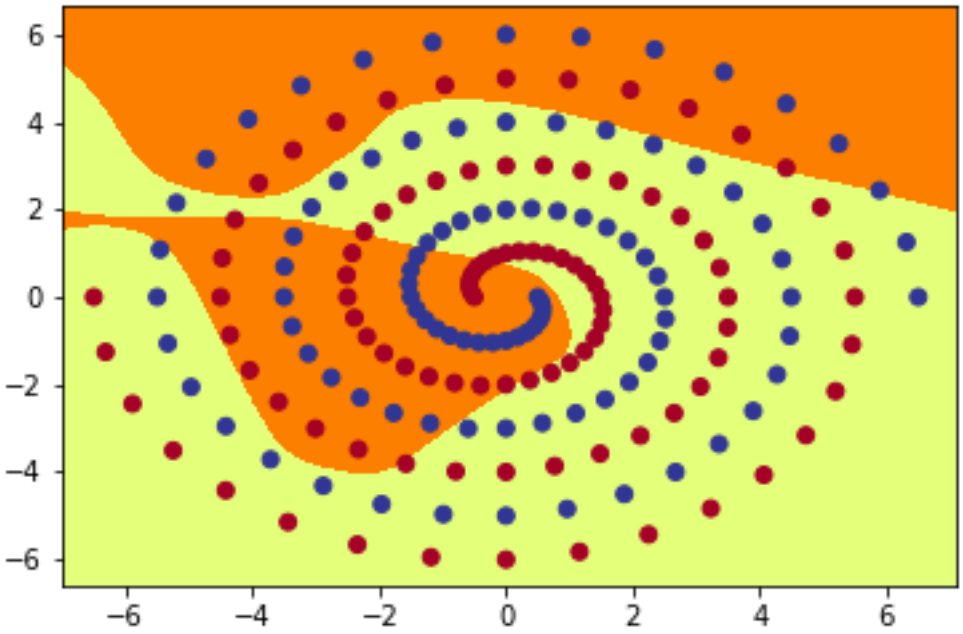

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

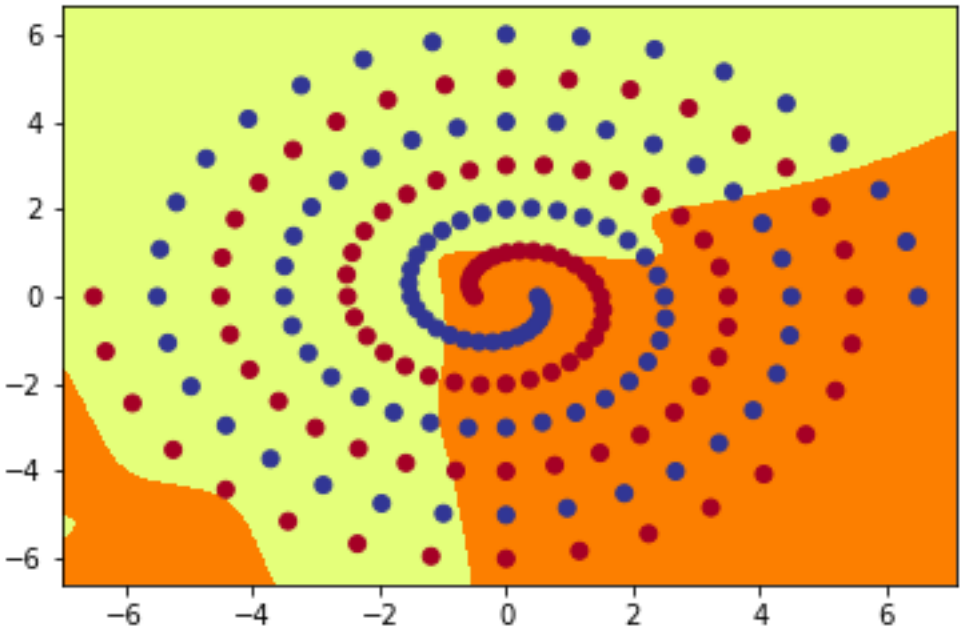

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

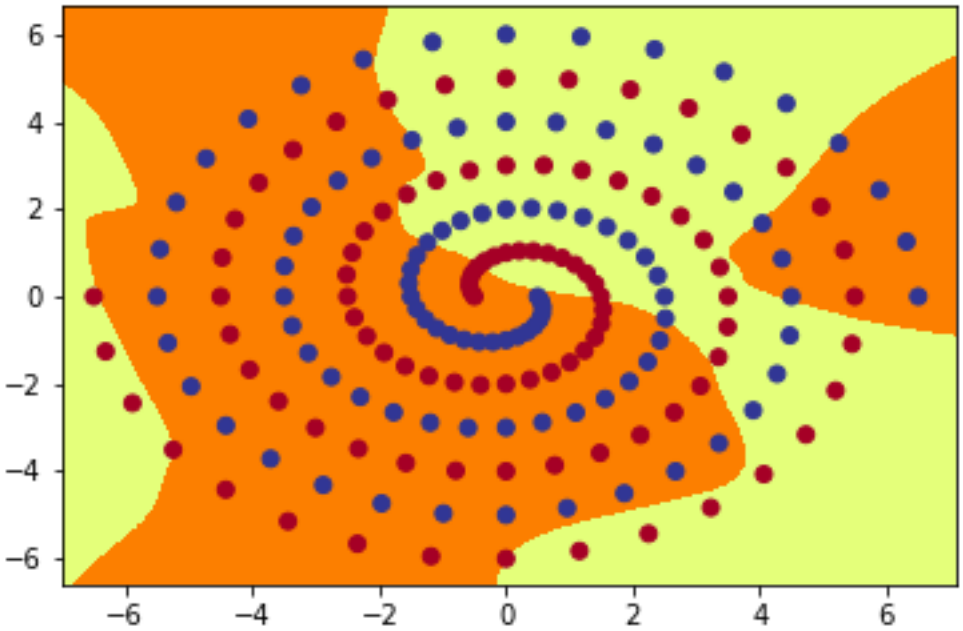

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

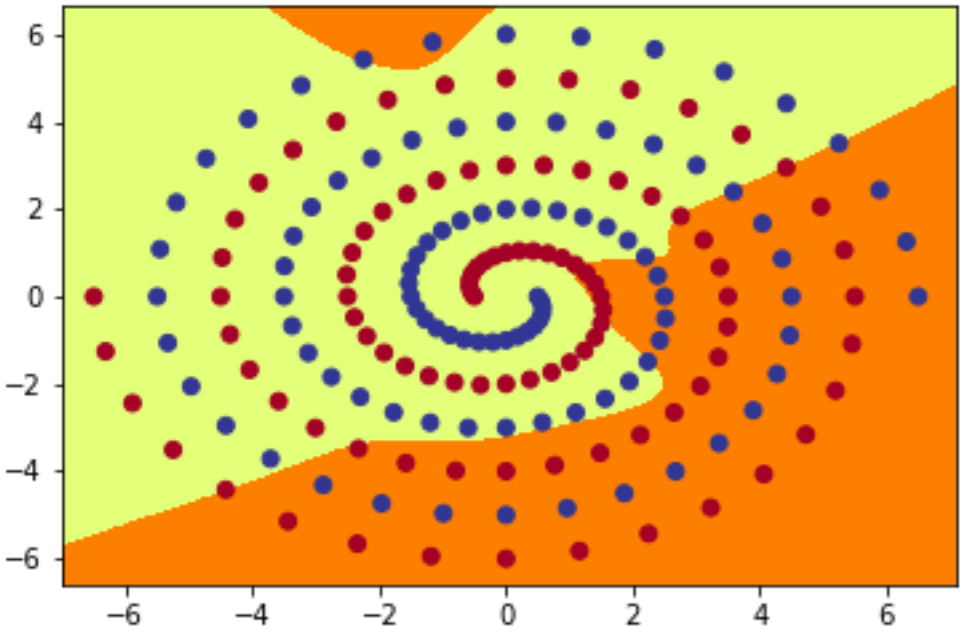
Layer 2 Images are as follows :

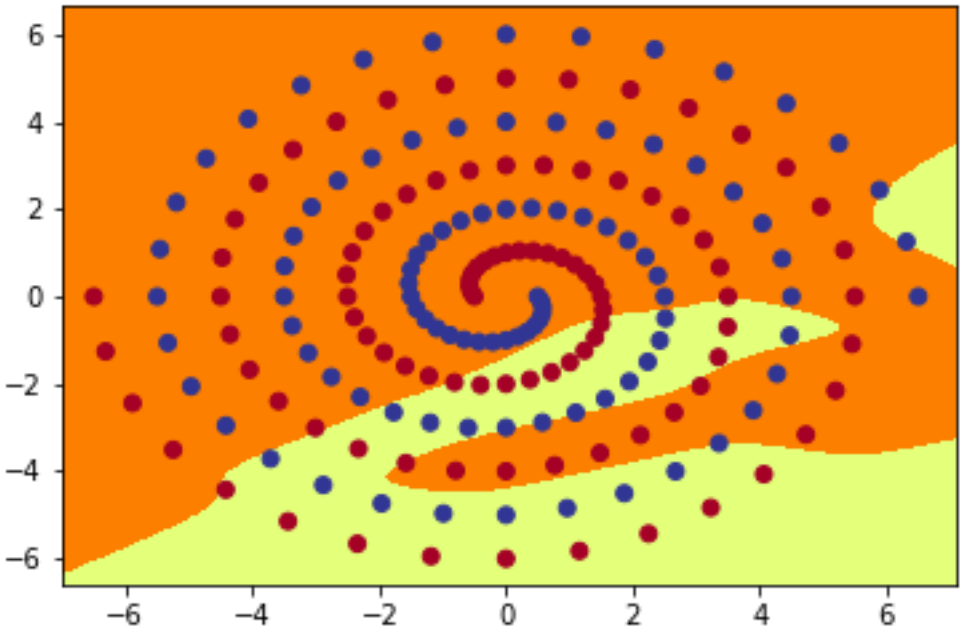


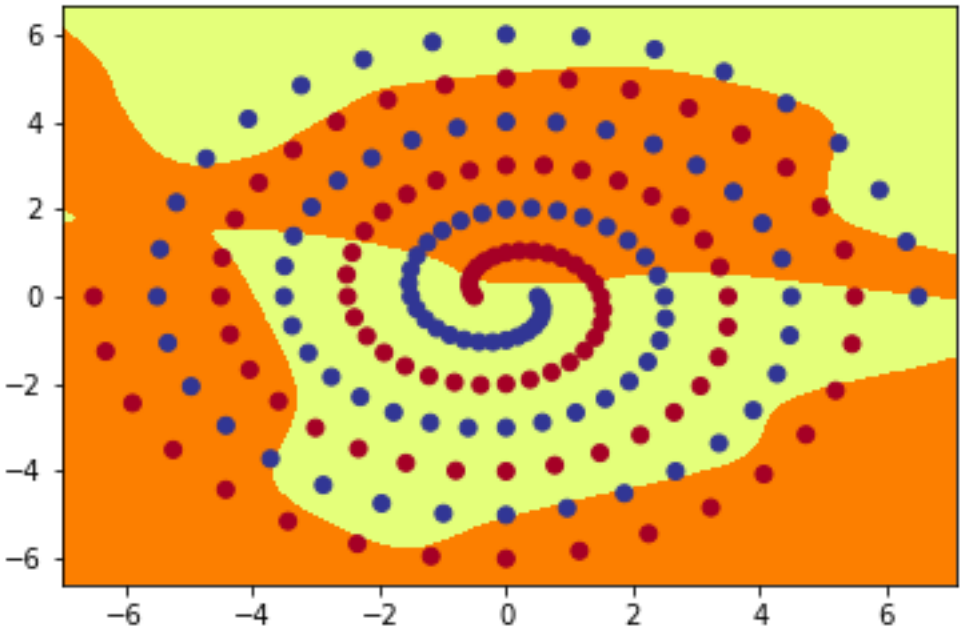


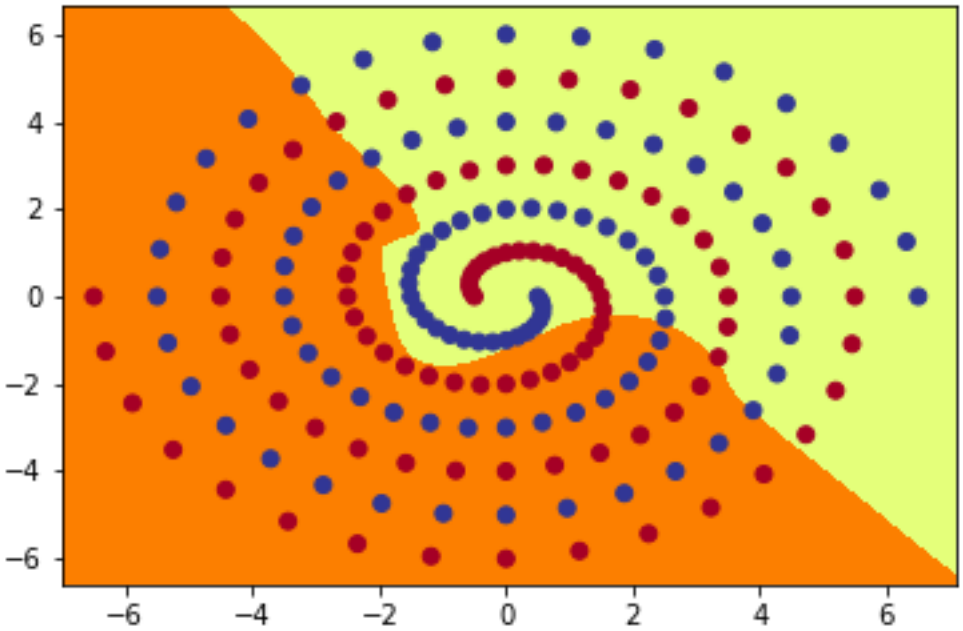

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

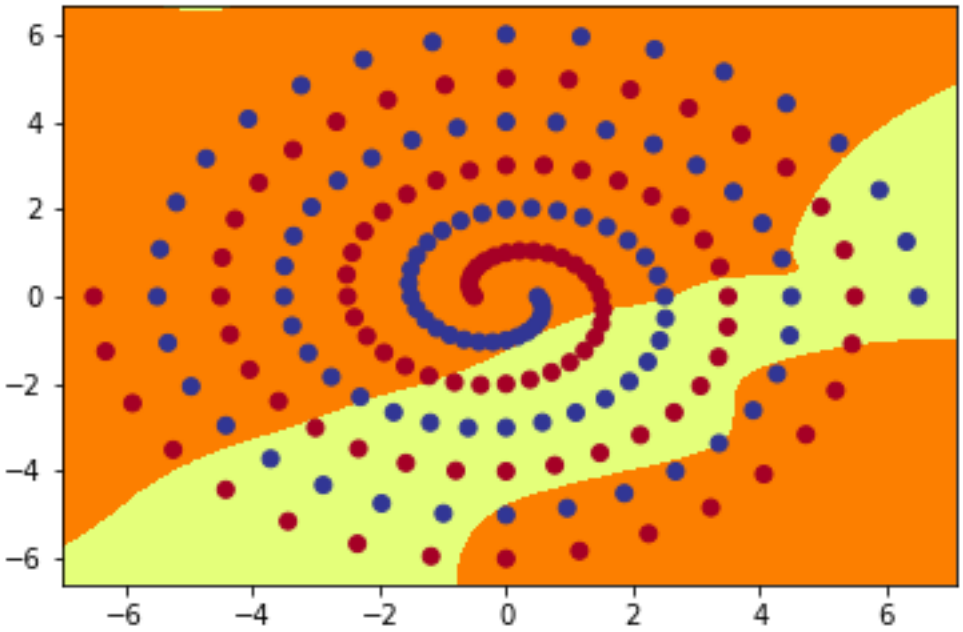

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6

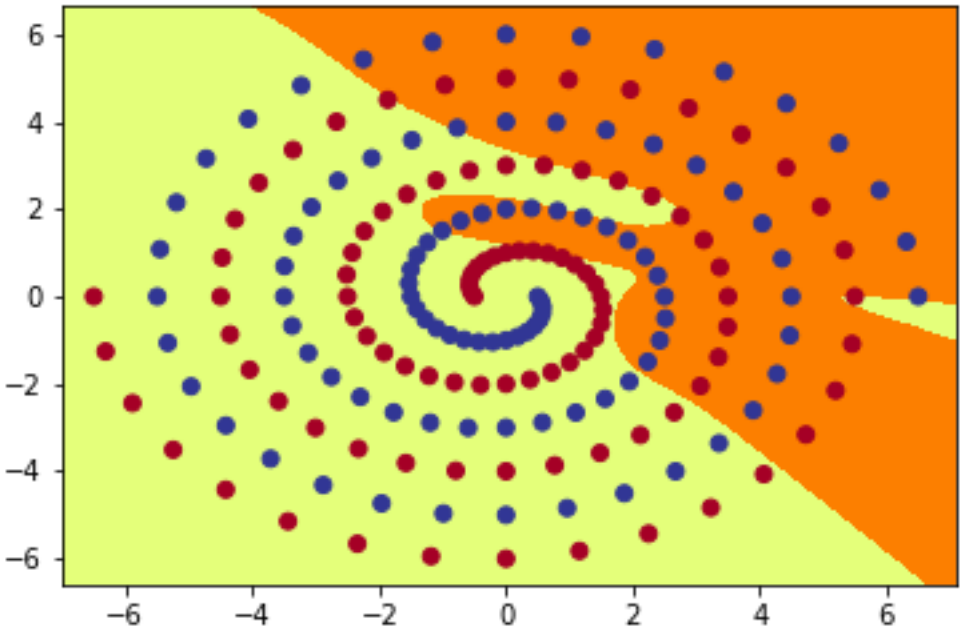

6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6


6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6





Part 6 :

A.

The PolarNet and RawNet have the different mode of working, following is the explanation for both.

Model : PolarNet:

This model, relies on the polar co-ordinates using which can effectively learn different parts of the twin spiral problem.

Being a deep learning model, it first learns the separate layers which can be observed by crisp straight line separations in the images created.

Then it combines the different learnings to get the final output.

Model : Raw Net

Unlike Polar Net raw net does not have the liberty of being given accurate pin point values, rather the input is given in a way that the model works in the vicinity.

Due to this approach, first the model learns the outer spiral as can be observed by the output of the layer 1 images.

Once the model has learnt the task it moves to learning more complex internal images.

B.

The effect of changing the weights and size are as follows :

Meta parameters	***Epochs***	***Accuracy/Loss***	***Observation***
Hidden Node = 10 Lr = 0.01 Init=0.01	*20000*	53.61/0.6929	The model got stuck in a point of minima and did not move from there resulting into being stopped midway.
Hidden Node = 10 Lr = 0.01 Init=0.2	20000	99.48/0.0292	While the accuracy was increased but on successive tests it was observed that the accuracy was very fluctative,
Hidden Node = 10 Lr = 0.01 Init=0.25	20000	97.94/0.0130	This did not prove much useful and only decreased the accuracy
Hidden Node = 10 Lr = 0.01 Init=0.23	4300	100/0.0162	This proved to be the best case scenario, as the model both learned early and also was within the permissible limits
Hidden Nodes = 11	19000	99.95/0.0240	I tried increasing the hidden nodes but the

			improvement was not very significant.

Part C:

The following are the observations by changing the batch sizes

For ***PolarNet*** with batch size of 194 and weight of 0.23

Case No	***Epoch***	***Accuracy***
1.	2000	100
2.	20000	66
3.	20000	78
4.	1700	100
5.	4000	100
6.	18000	100
7.	10000	100
8.	1800	100
9.	20000	60
10.	3400	100

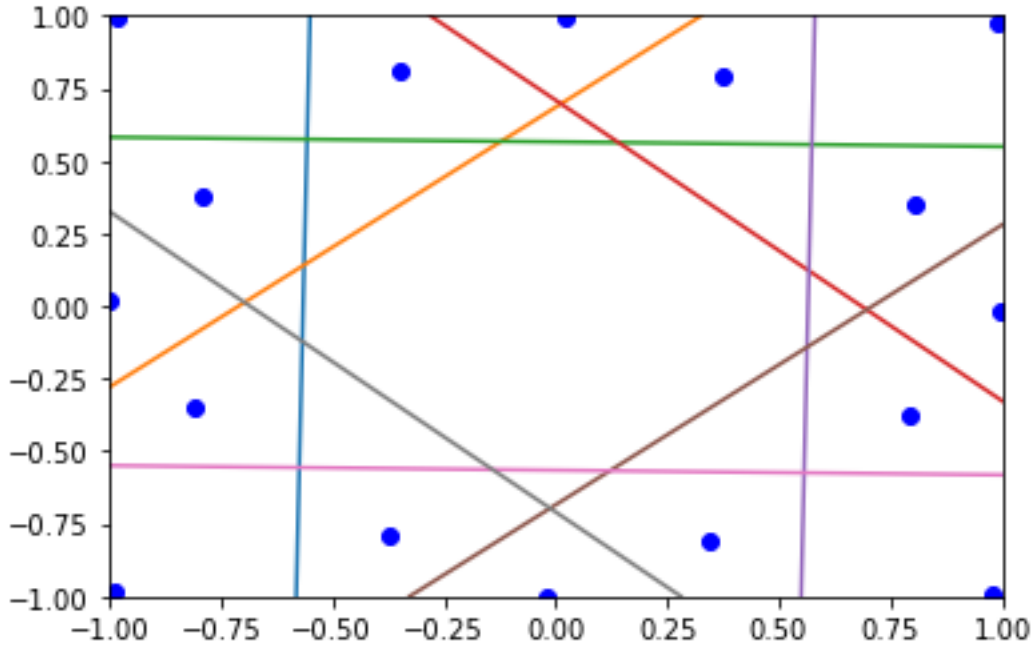
For RawNet with batchsize of 194 and weight of 0.23

Case No	***Epoch***	***Accuracy***
1.	2000	100
2.	17000	100
3.	20000	99.98
4.	15300	100
5.	4500	100
6.	20000	98.14
7.	20000	95
8.	1800	100
9.	20000	96
10.	4400	100

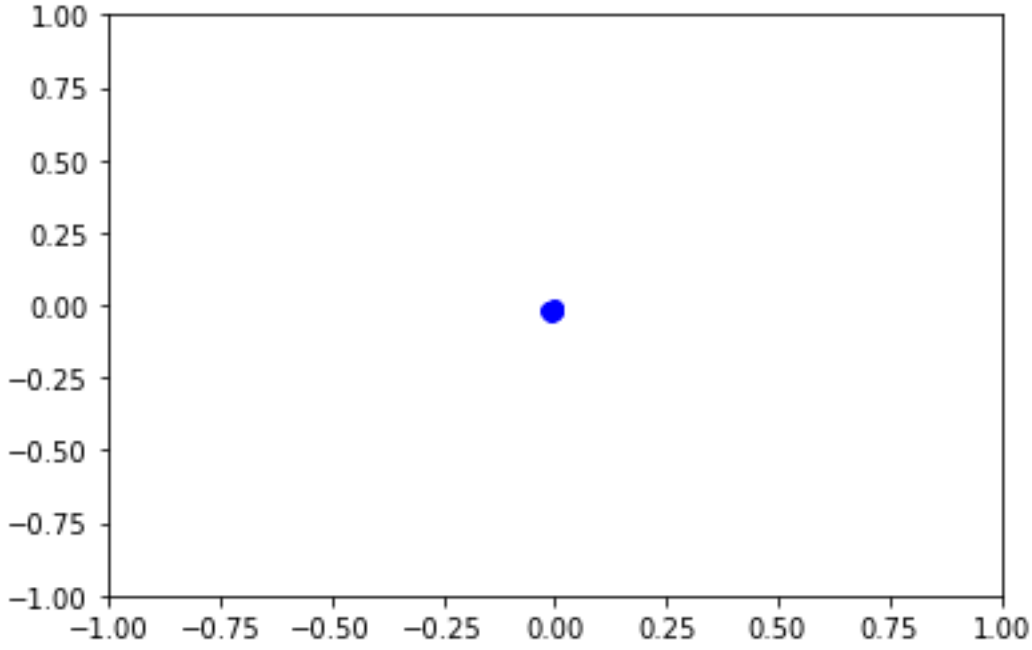
Increasing the batch size resulted in the model learning the data in lesser epochs as compared to when the model was learning in smaller batch sizes.

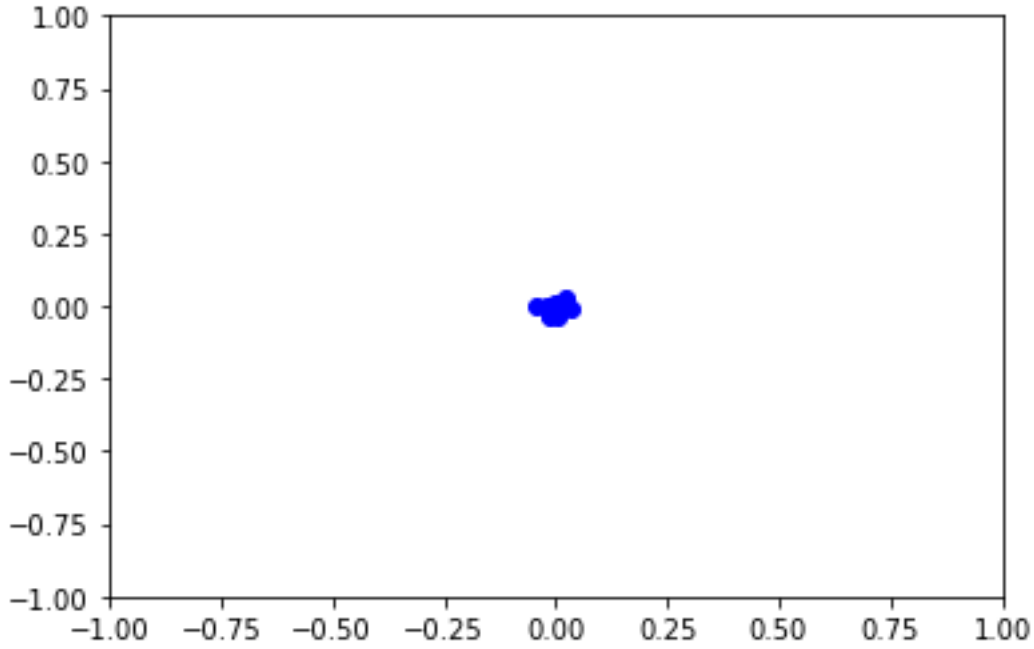
Question 3

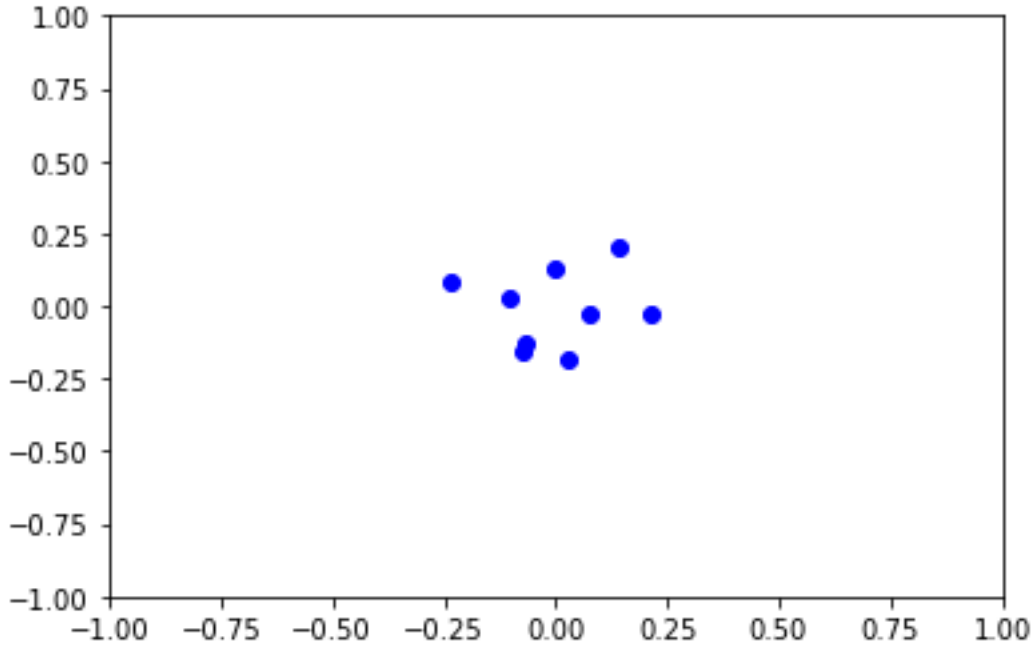
Part 1 : The output is as follows :

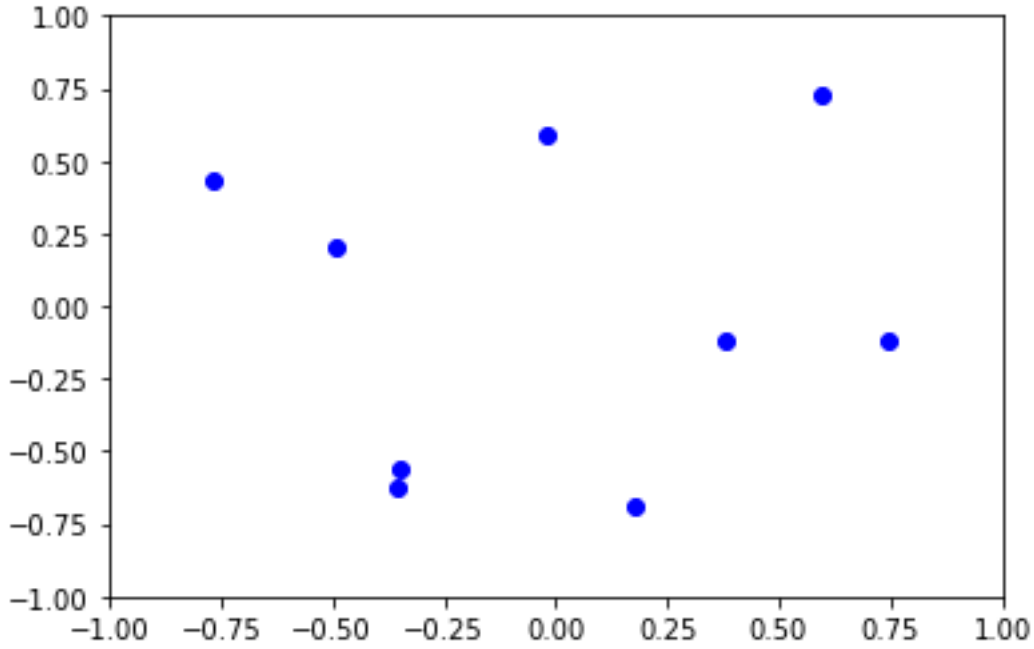


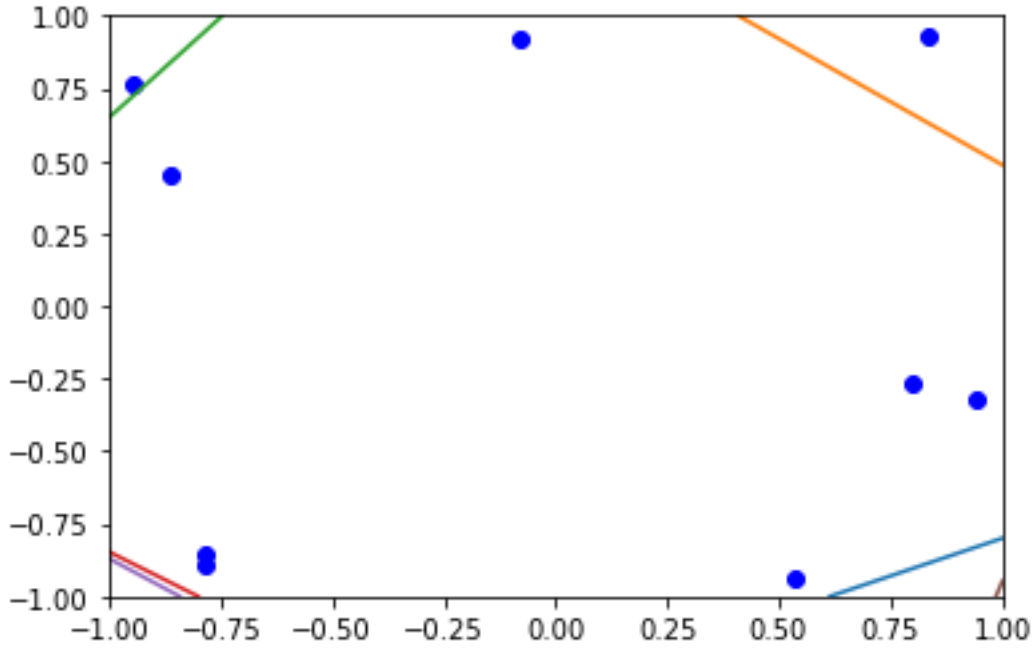
Part 2 :

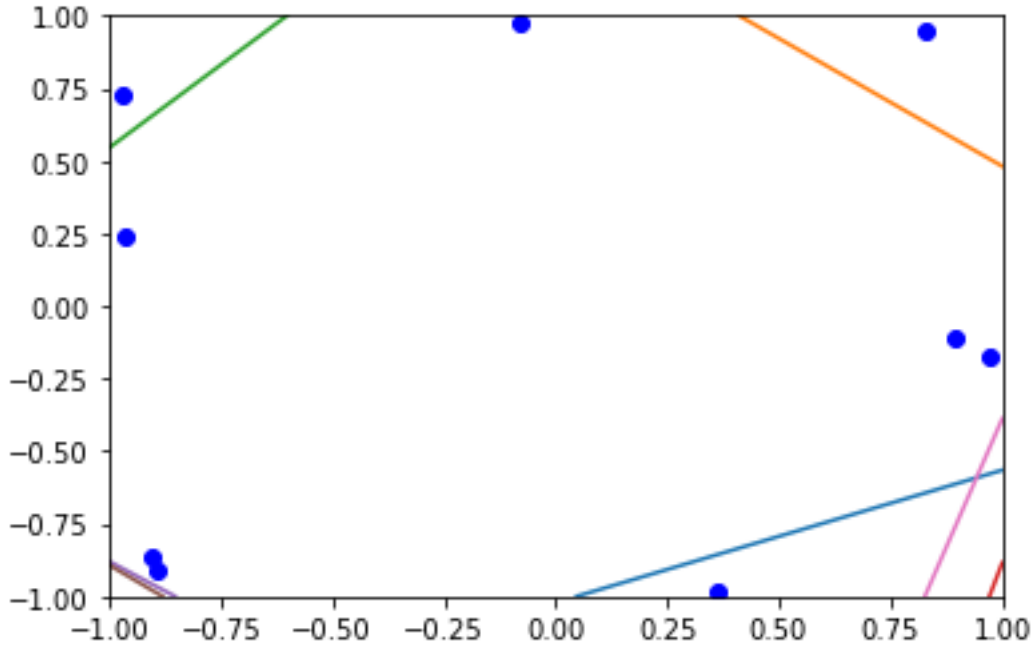


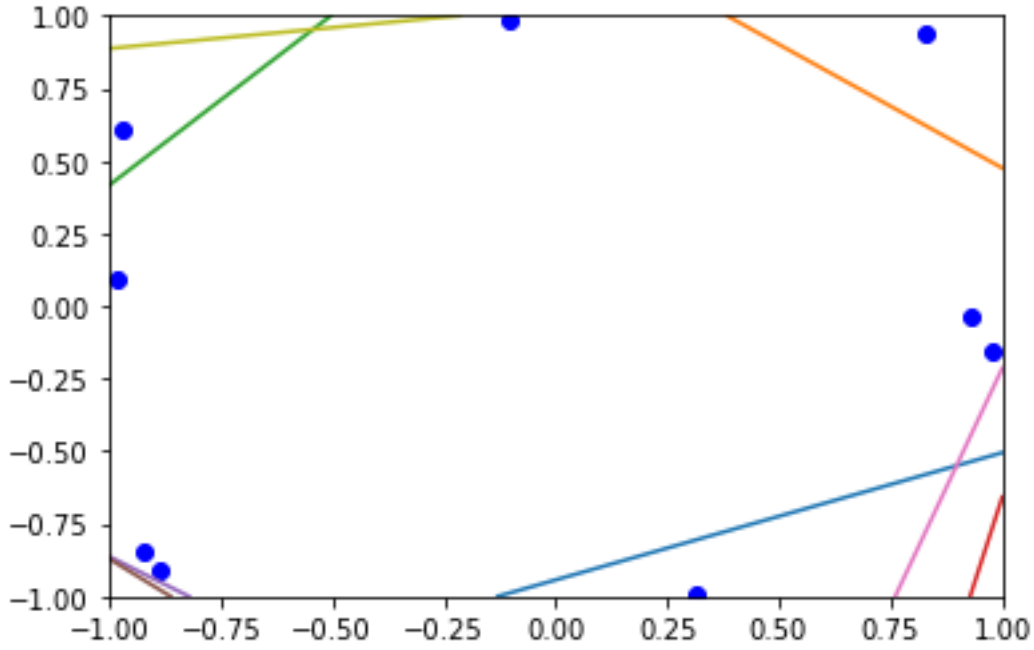

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

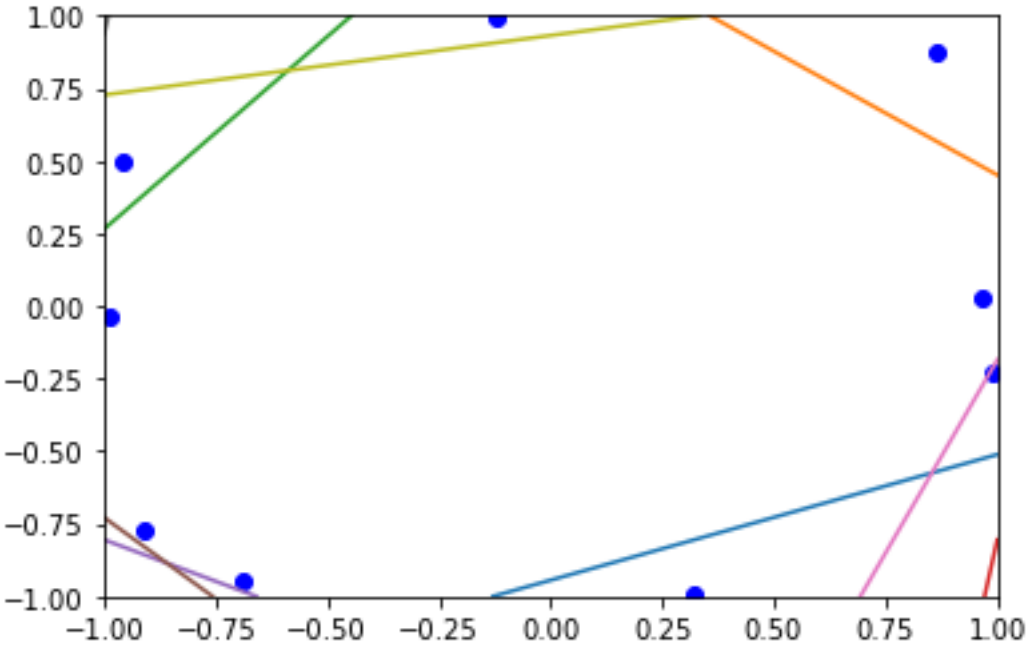

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

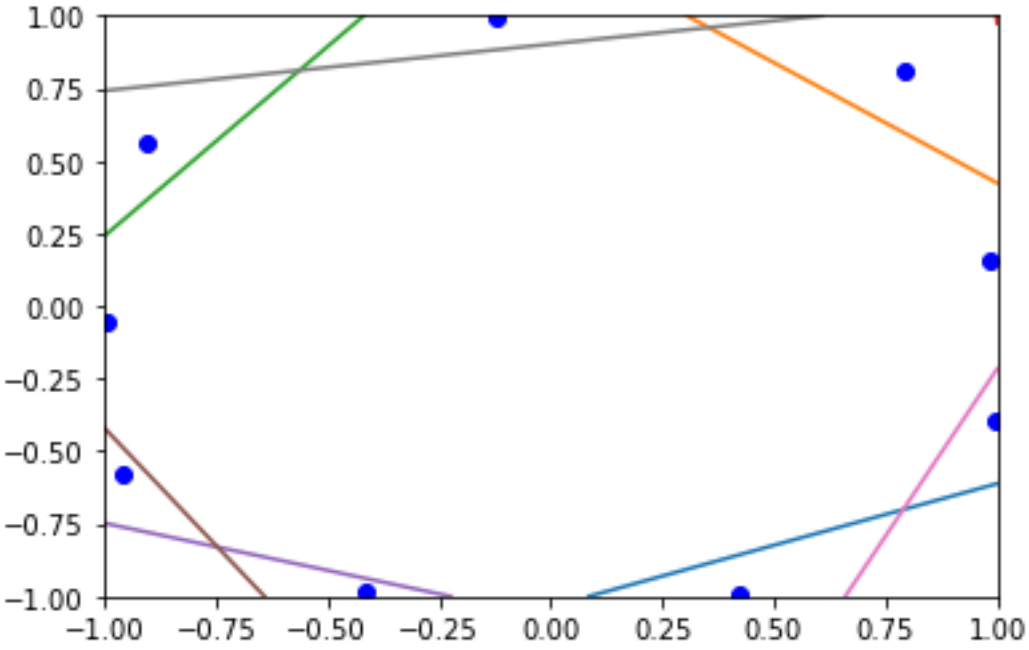

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

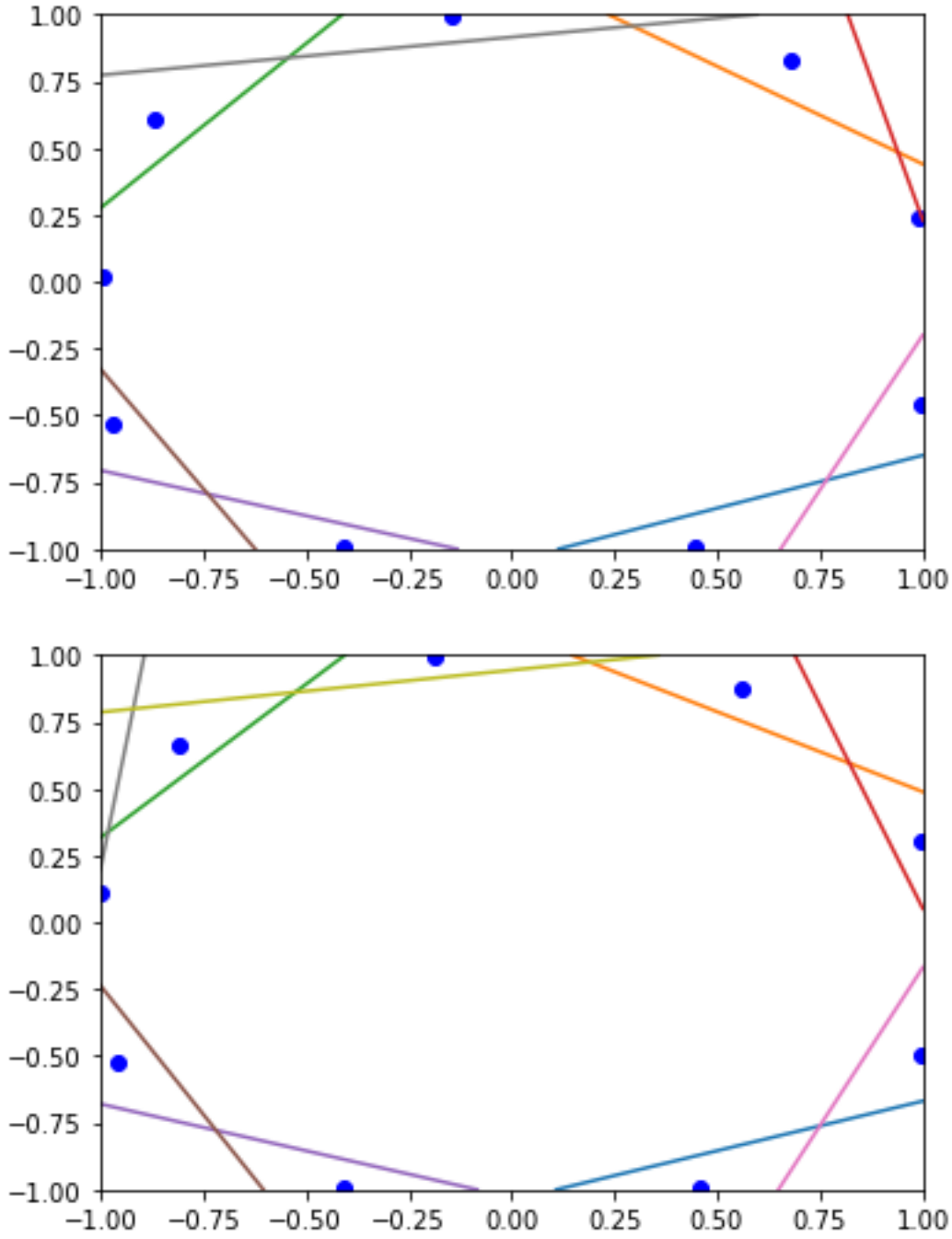

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00



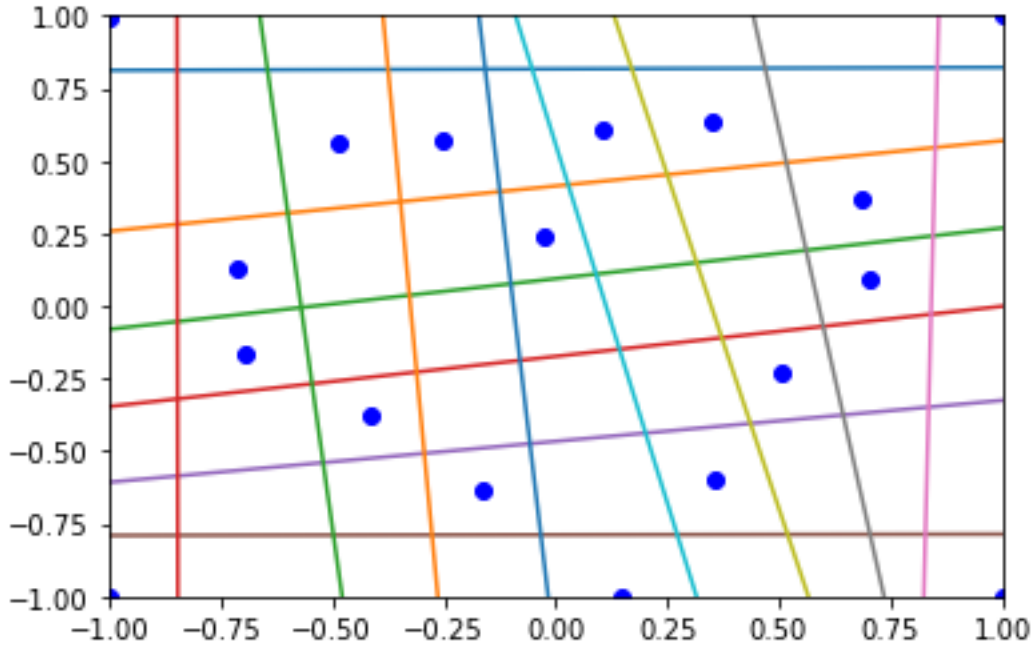
First it tries to plot the simple values of the hidden units in an incremental value, then the lines are an attempt to divide the plotted points into different planes.

In essence the assumption is that this is an infinite plane onto which the values of the input to hidden are being plotted for N2N encoders.

As can be observed the initial weights could be zero due to which the points are clustered, as we increase the combination of the weights to the hidden layers then the points get dispersed and we try to divide them in different planes.

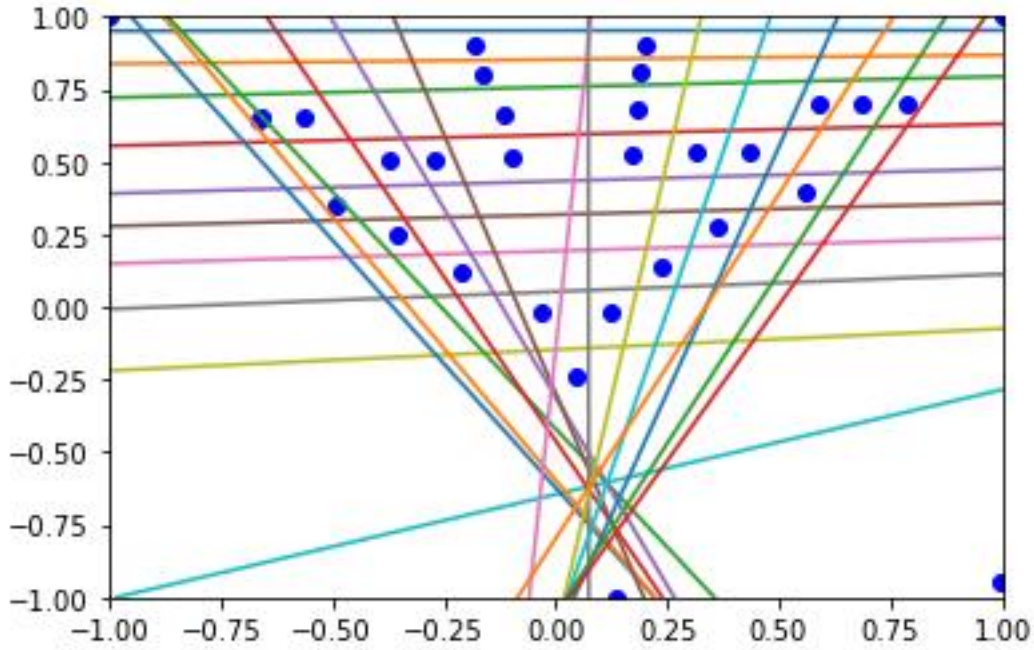
The end result is the last figure where it has computed the hidden layer inputs and results are observed.

Part 3 : The output of the image is as follows :



Part 4 : The patterns are as follows :

The first pattern is that of the batman logo :



The second pattern is that of an infinity symbol.