Project 1 – Characters, spirals, and hidden unit dynamics

z5205338 – Hugh Ebeling

Part 1: Japanese character recognition

1. Confusion matrix for linear network:

[[771. 5. 8. 13. 29. 63. 2. 62. 30. 17.]
[5. 672. 104. 18. 30. 23. 58. 13. 25. 52.]
[9. 61. 690. 26. 24. 21. 46. 38. 46. 39.]
[5. 36. 62. 755. 16. 58. 13. 18. 27. 10.]
[58. 54. 79. 20. 623. 19. 32. 37. 19. 59.]
[8. 28. 130. 17. 19. 722. 26. 8. 32. 10.]
[5. 25. 149. 9. 24. 25. 721. 20. 9. 13.]
[17. 32. 27. 13. 82. 16. 53. 624. 88. 48.]
[10. 40. 92. 41. 6. 29. 46. 6. 708. 22.]
[8. 50. 88. 3. 52. 32. 19. 30. 39. 679.]]

Accuracy: 6993/10000 ~= 70%

Adjusting the parameters for this did not yield better results

2. Confusion matrix:

[[886. 3. 1. 1. 29. 21. 4. 28. 25. 2.]
[6. 822. 28. 4. 20. 7. 68. 6. 20. 19.]
[8. 13. 853. 30. 14. 20. 22. 10. 22. 8.]
[3. 8. 24. 924. 3. 17. 3. 3. 8. 7.]
[35. 16. 14. 5. 861. 9. 25. 14. 17. 4.]
[6. 7. 53. 8. 11. 874. 21. 3. 8. 9.]
[2. 8. 27. 6. 10. 7. 930. 7. 1. 2.]
[20. 10. 19. 2. 26. 7. 27. 850. 14. 25.]
[9. 28. 21. 34. 3. 8. 20. 5. 868. 4.]
[5. 14. 43. 4. 24. 4. 19. 18. 13. 856.]]

Accuracy: 8720/10000 ~= 87%

Details: I used 150 layers and a learning rate of 0.02 to achieve this result

3. Confusion matrix:

[[946. 3. 1. 2. 25. 5. 2. 12. 1. 3.]
[2. 930. 6. 1. 3. 0. 31. 4. 3. 20.]
[12. 13. 877. 30. 7. 6. 36. 10. 4. 5.]
[0. 2. 23. 956. 6. 1. 2. 3. 2. 5.]
[22. 7. 6. 8. 908. 4. 20. 6. 9. 10.]
[3. 13. 31. 8. 2. 886. 33. 11. 4. 9.]
[4. 8. 16. 1. 2. 0. 966. 2. 0. 1.]
[6. 8. 0. 2. 13. 2. 11. 928. 4. 26.]
[6. 11. 4. 2. 8. 5. 6. 2. 953. 3.]
[5. 12. 6. 2. 6. 3. 0. 3. 1. 962.]]

Accuracy: 9312/10000 = 93%

Details: First convolutional layer is 64 with a kernel size of 4 and second layer has an input of 64 and output of 128 with a kernel size of 4

4. Discuss what you have learned

a. The relative accuracy of the 3 models

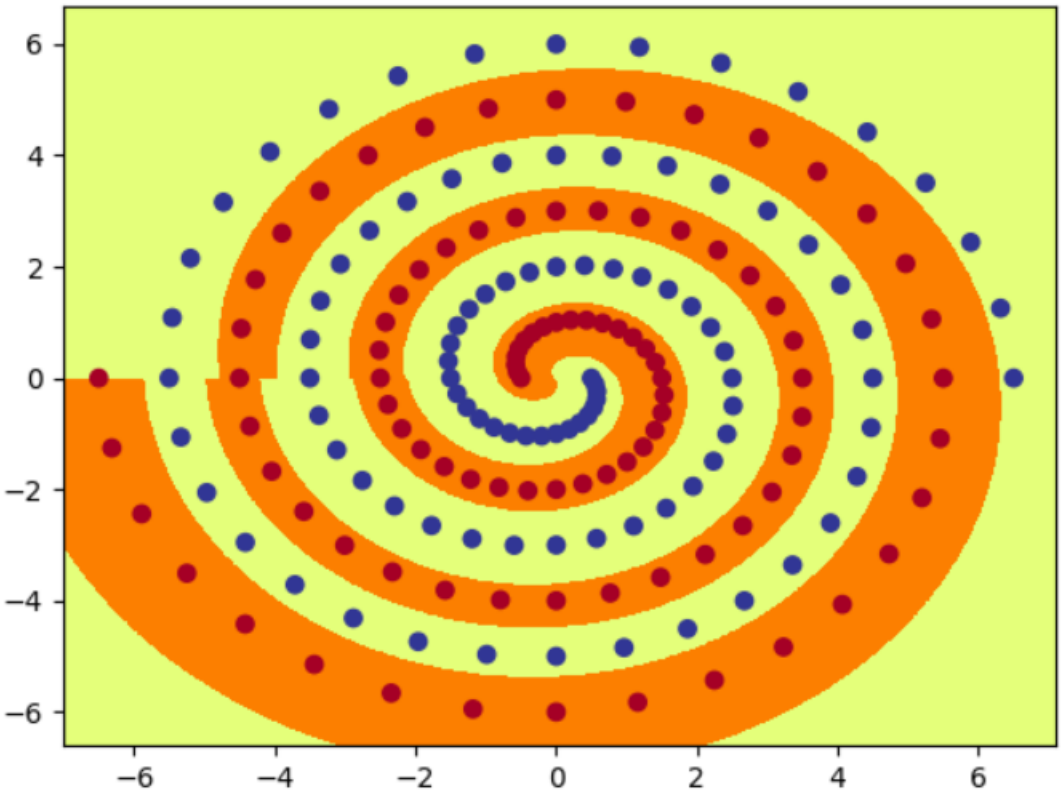
The linear method of determining which letter is which is not very accurate, as there is no way to fully adjust to the different ways of writing each letter. By adding another linear layer to the first method the accuracy improves however is still below 90%. The convolutional network is more accurate than both of these methods as it is a method that can recognise patterns in an images rather than exact placement of pixels.

b. The confusion matrix of each model: which characters are most likely to be mistaken for which other characters and why
   The linear method due to being very inaccurate makes it difficult to determine which letters are mistaken for eachother. Letters that show an accuracy below 70% (below 700 on the diagonal) appear to be more likely to be mistaken for eachother. While letters 1 and 4 appear to be more unique to the system and thus the system is more successful at guessing them.
   The full method can be analysed in a similar way letters 2 and 7 are the least successfully guessed and are more likely mistaken for eachother. While letters 4 and 6 are more successful and thus aren't as likely to be mistaken for a different letter.
   The convolutional confusion matrix shows where the bigger mistakes are made in particular any mistake over 30 for example letter 7 is mistaken for letter 3 but not the other way around.
c. I explored the use of different convolution and kernel sizes without much change in the final result adjusting the linear final output also did not change much after the 2048 as too many connections causes the convolutions to memorise the training set instead of generalising the problem. I also changed the learning rate of all of these with little change other than in the full net with significantly increased the percentage compare to the default 0.01 learning rate.
d. Additionally I learnt that different types of data require different methods for classifying and the different methods can benefit one dataset and disadvantage another.

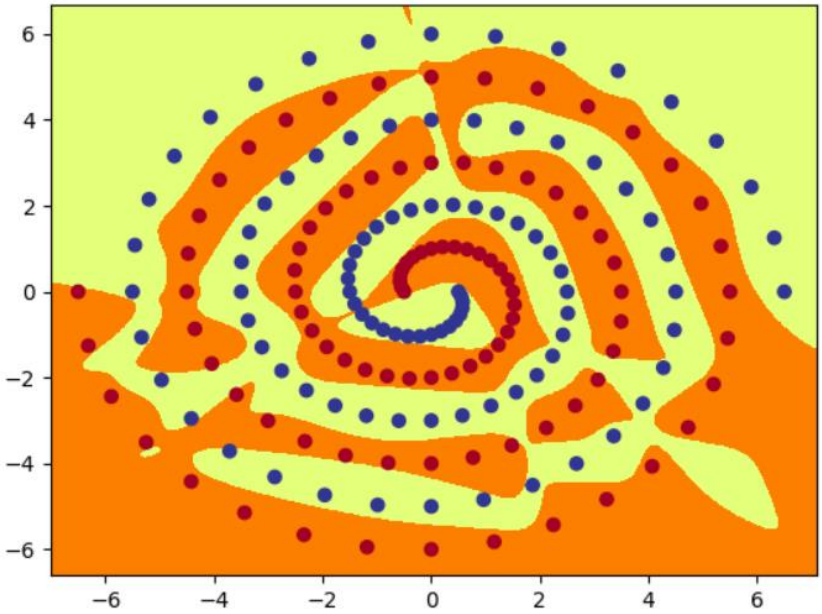
Part 2: Twin spirals task

1. See code in spiral.py

2. I found that a minimum number of 7 hidden nodes seems to consistently classify the training data with a learning rate of 0.005 the output is



3. See code in spiral.py
4. 10 hidden seems to be consistent with a learning rate of 0.03 and initial value set to 0.04 the output is



5. See the output code in spiral.py
   See the appendix for all the output images of the hidden nodes
6. Discuss

a. The qualitative difference between the functions computed by the hidden layer nodes PolarNet and RawNet and a brief description of how the network uses these functions to achieve the classification

Since polarnet is fed data that has been converted from cartesian to polar form the network finds it a lot easier to determine where the spiral is as it just has to determine the distance from the centre (which is zero) which is given in the data the hidden nodes reveal a pattern of having the same gradient and just being a different distance from the left I can assume the output node has a gradient perpendicular to these and can thus determine the correct placement of the dots in the spiral.

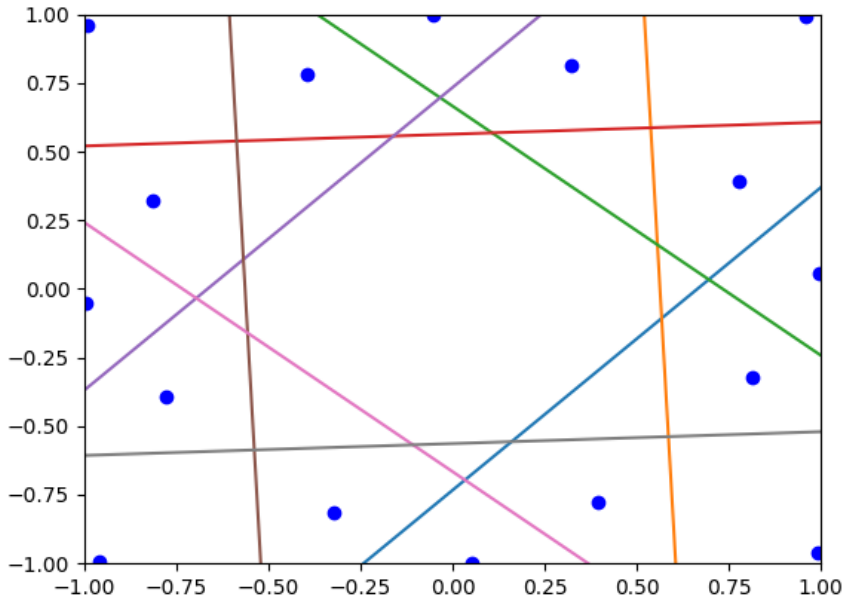
Whereas the raw input which just receives the cartesian coordinates for the spiral struggles to fit the activations within the 2 hidden layers as it needs to get every angle of the spiral by increasing the hidden layers the network is able to produce a more refined image with less activated space that isn't the red dots. The polar network will always be more accurate.

b. The effect of different values for initial weight size on the speed and success of learning for RawNet

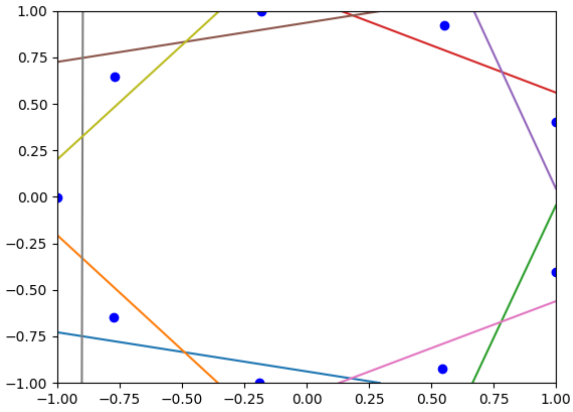
By decreasing the initial weight from 0.1 on Raw net the network was able to reach a 100% accurate image faster and with less hidden nodes. By increasing this number it made the network slower and less able to reach the accurate result by the 20000 epoch limit.

Part 3: Hidden Unit dynamics

1. The star image

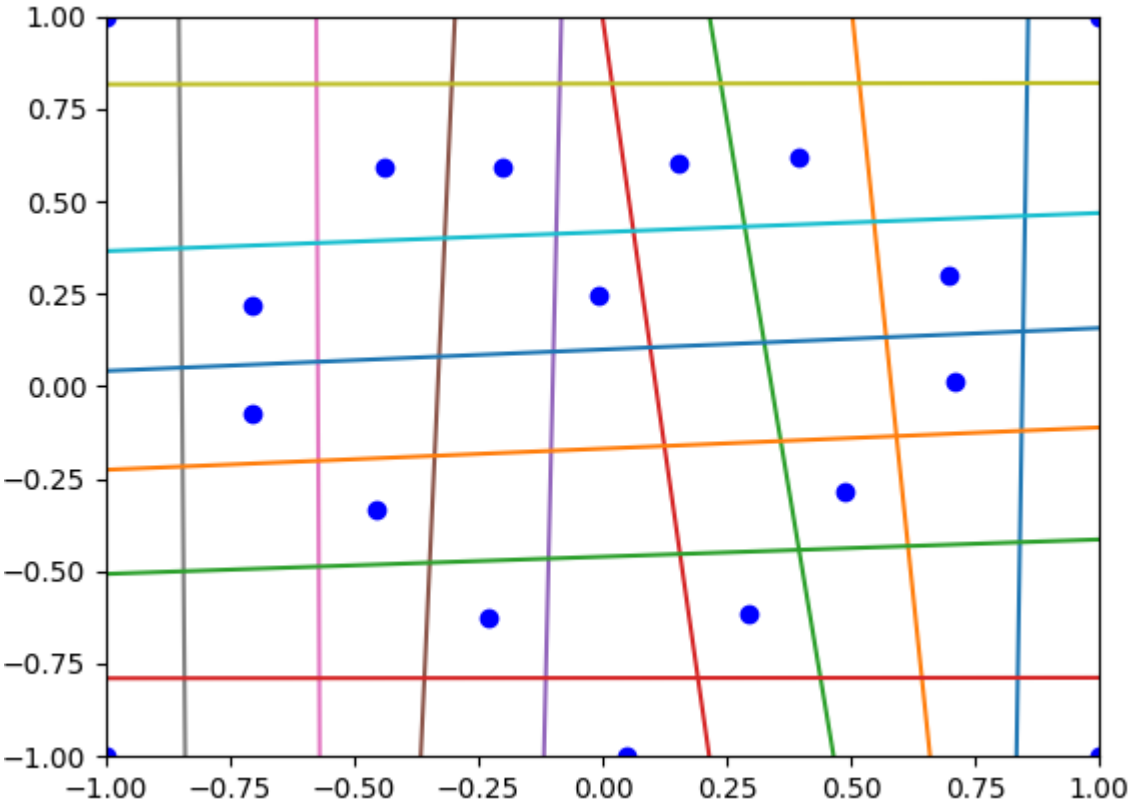


2. The 11 increasing images are in the appendix the final image is here



As the training progresses the activations start in the centre and progressively move outwards finally forming a somewhat regular polygon where each dot is just about equidistance from 2 other dots. The output boundaries do not appear at the start as they are outside of the plot boundary. It isn't until image 5 that they come into frame and start forming the basis of a polygon with no concave edges. There are 8 activations visible and 8 boundary lines visible.

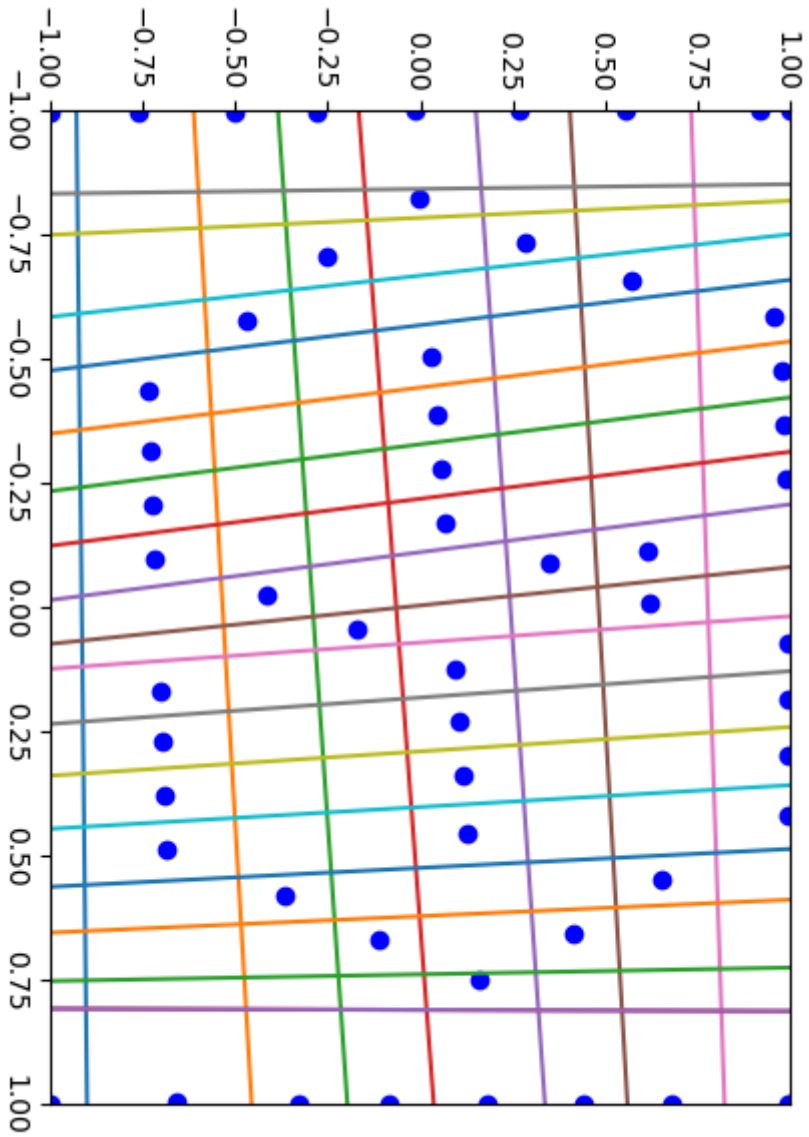
3.



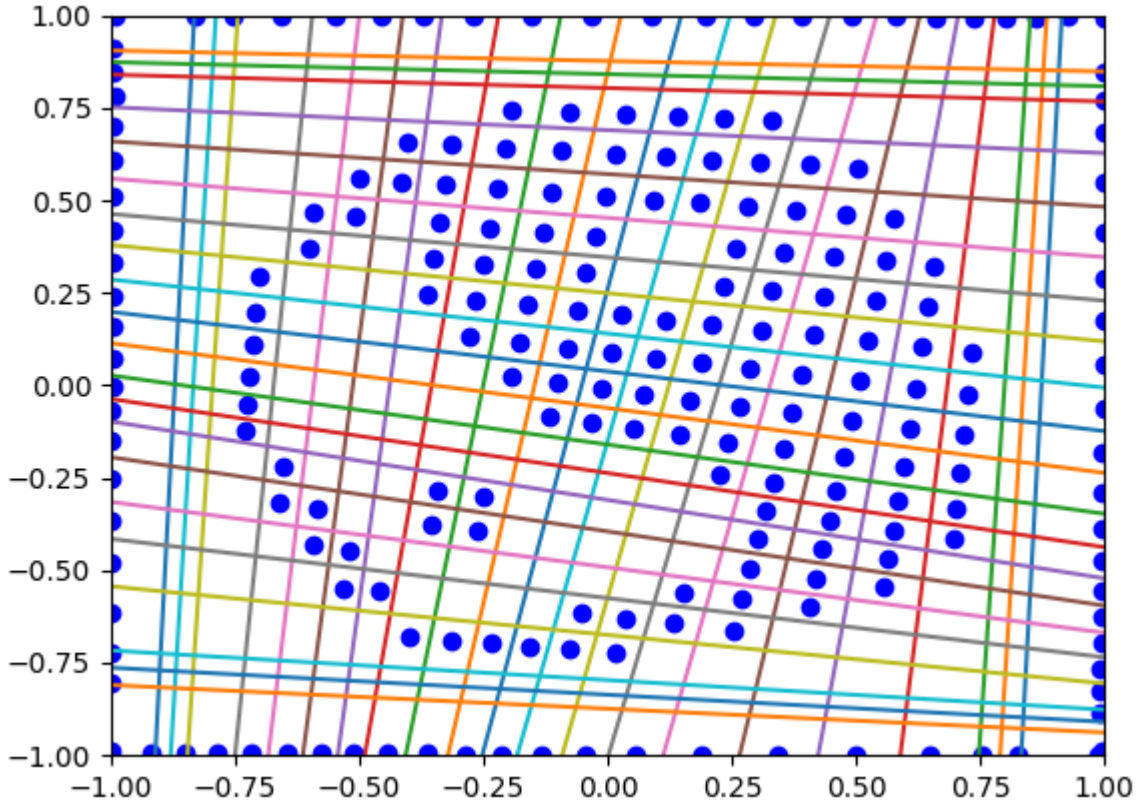
I didn't originally notice the 4 corner dots but by placing the dots in the 4 corners of the plot the heart is usually centred and all the spacing is equidistance.

4. My target images were the space invaders alien and a ying and yang symbol

Target 1 is what is known as the universal S a worldwide phenomena that has appeared in almost all school kids books at some point. I have rotated it 90 degrees as the encoder did not seem to display it vertically as planned.



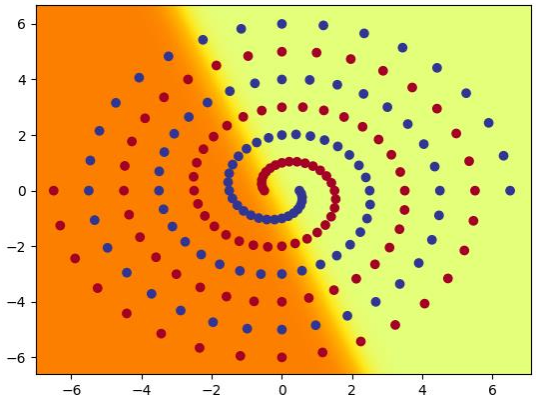
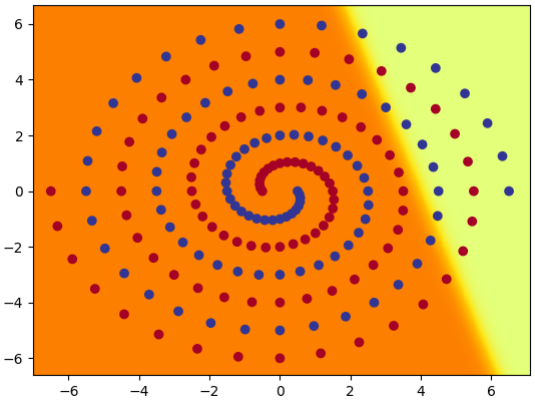
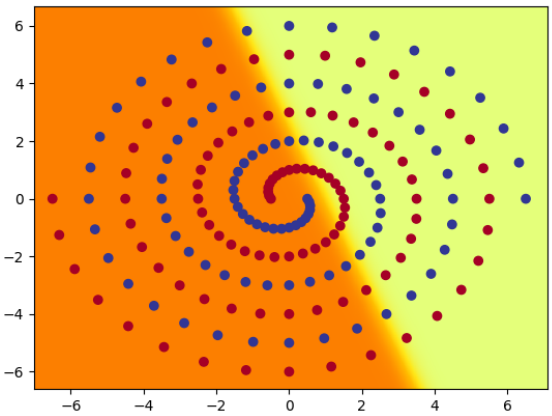
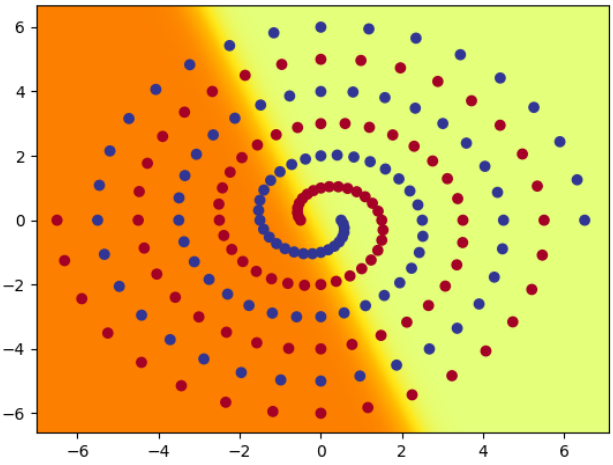
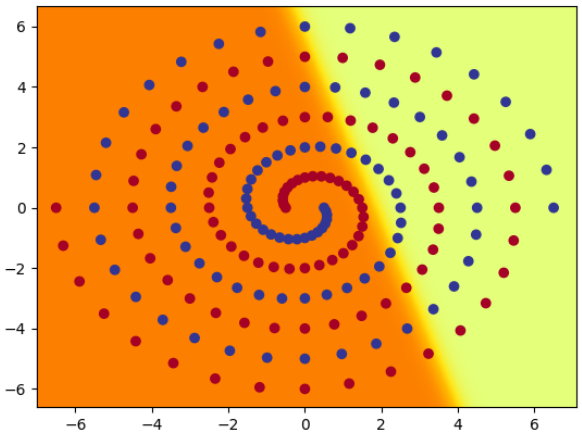
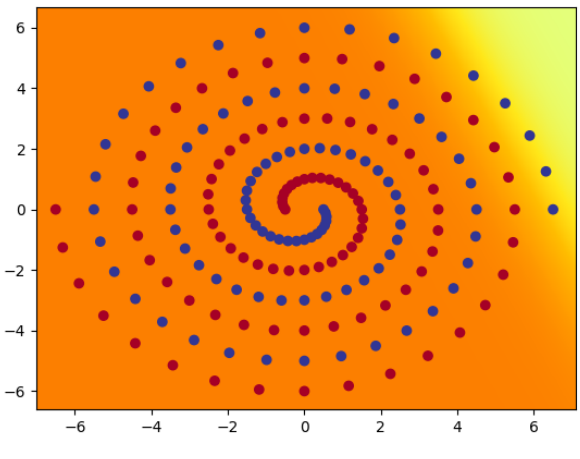
Target 2 is a ying and yang symbol

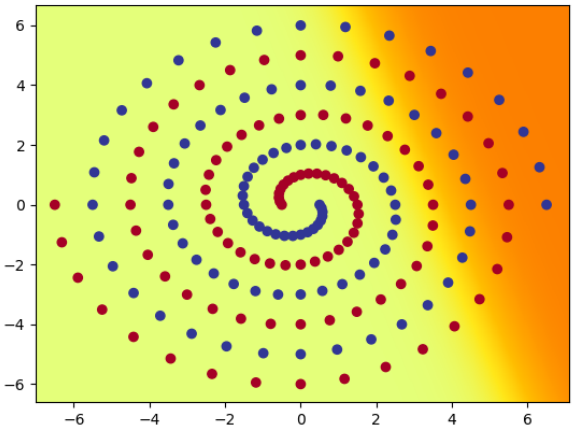
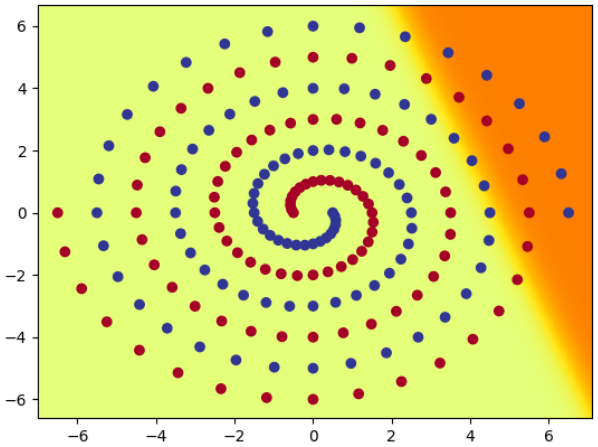


I have created an algorithm for creating the tensors that make these pixels art images it is commented at the bottom of the encoder.py file.

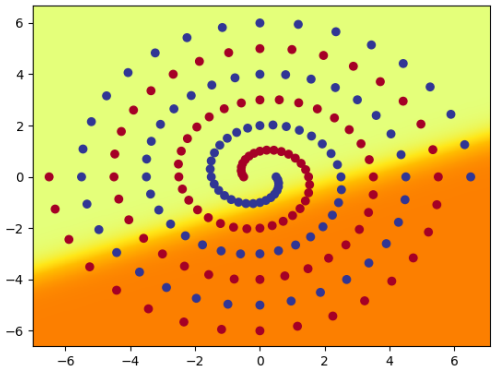
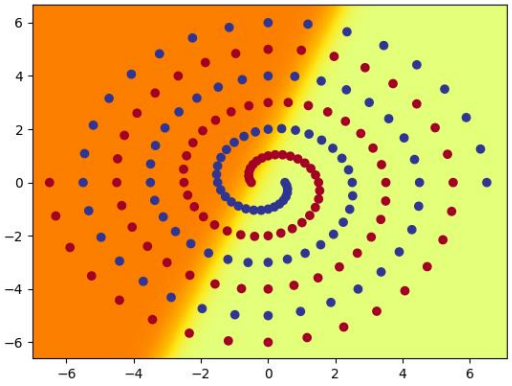
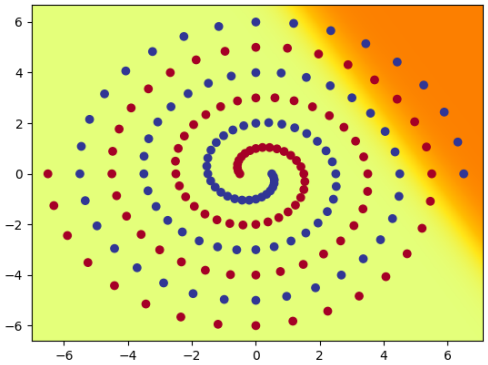
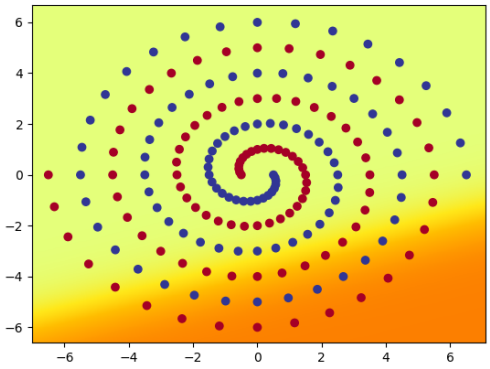
Appendix

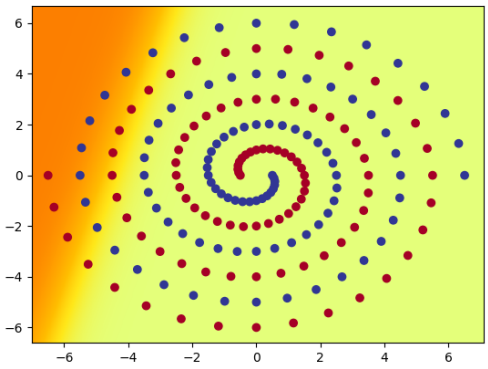
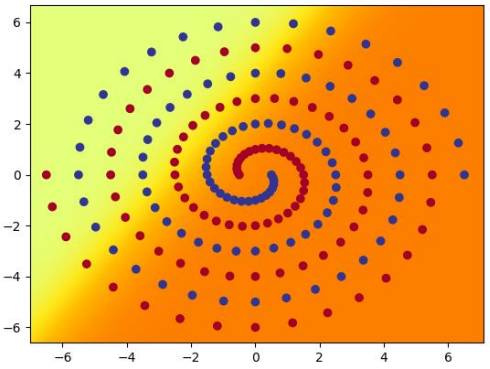
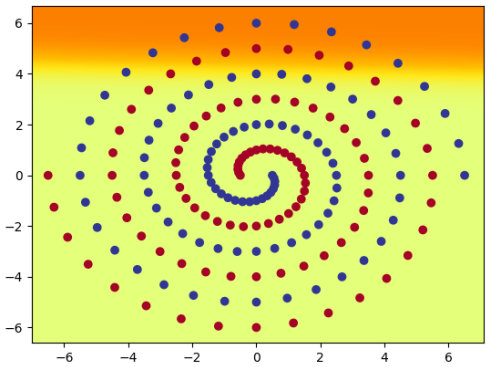
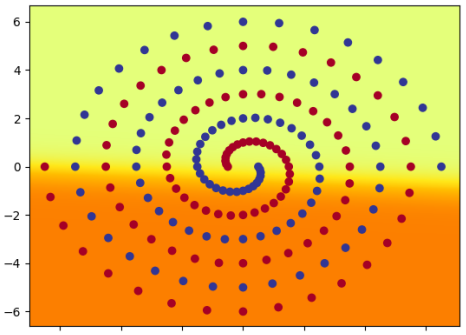
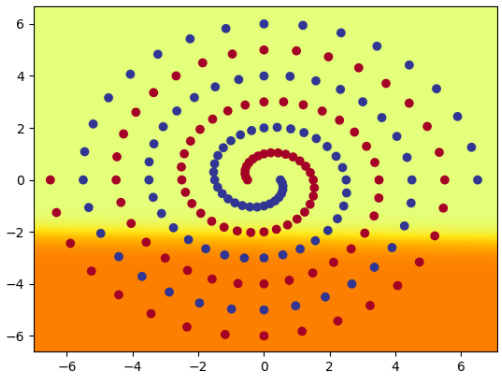
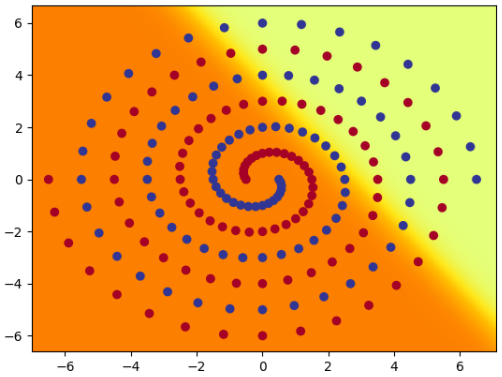
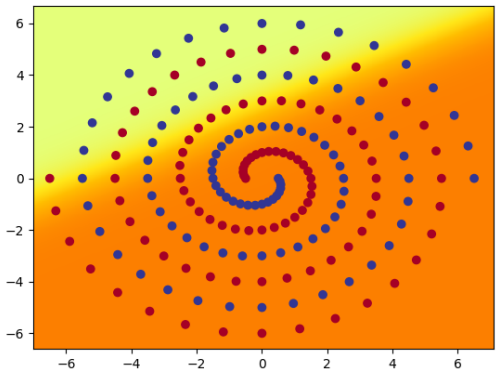
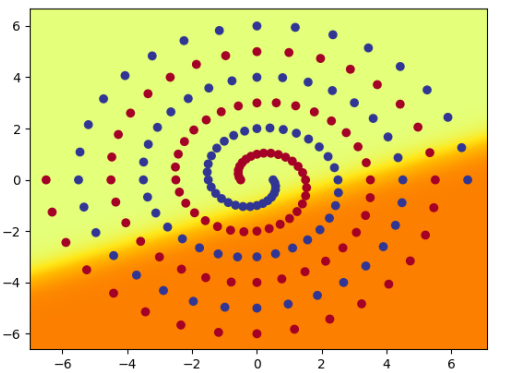
Below is the polar hidden nodes

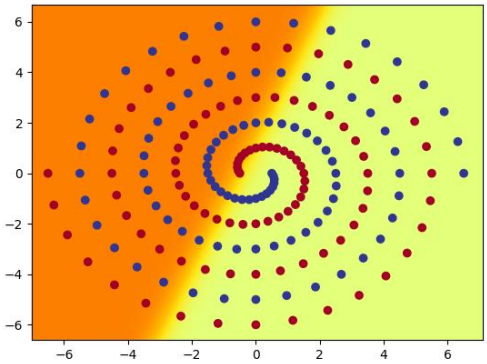
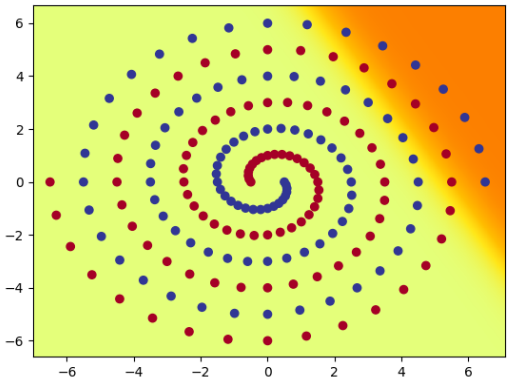
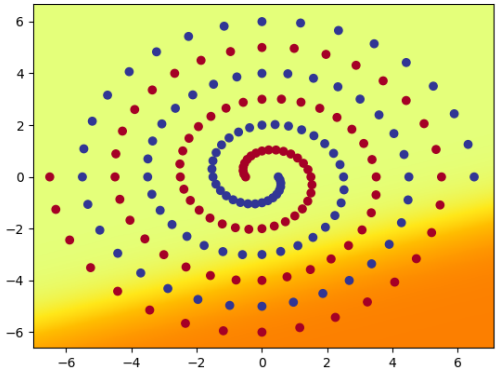
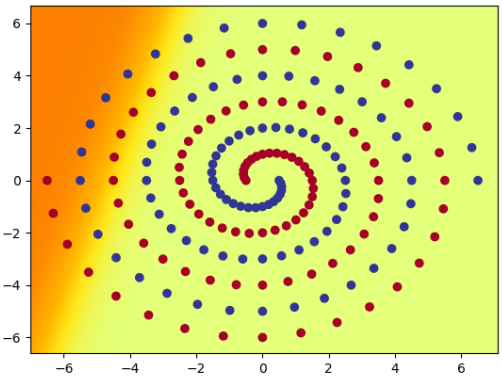
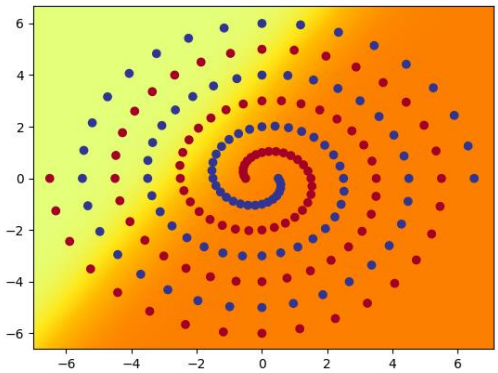
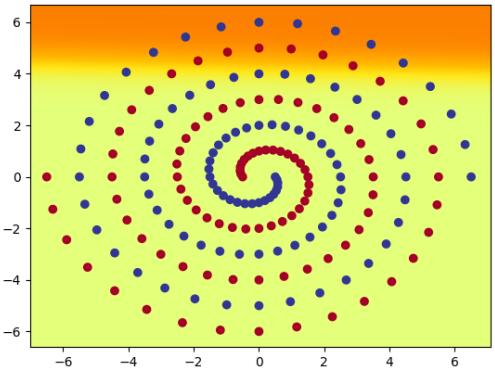







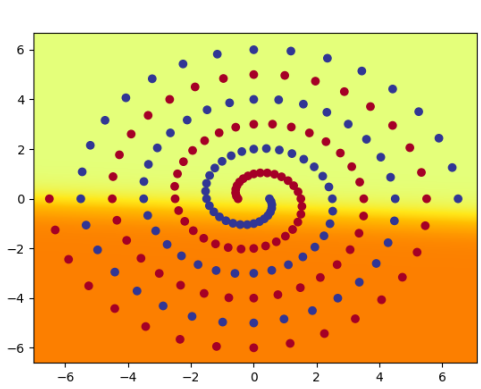
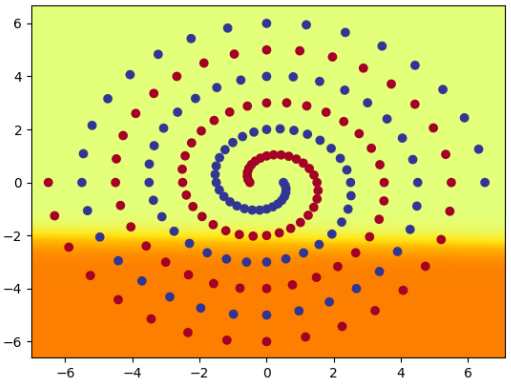
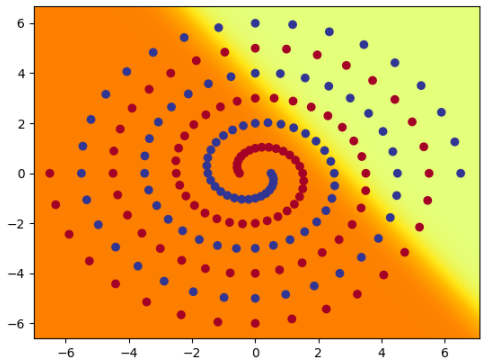
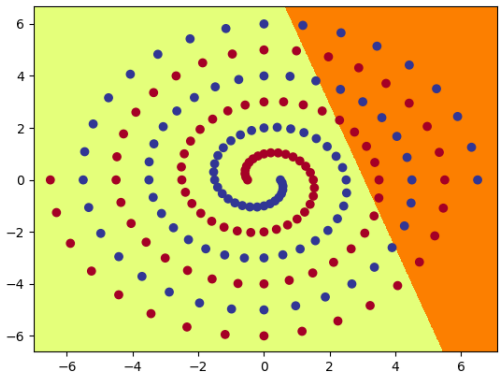



Below are the activations for the raw nodes

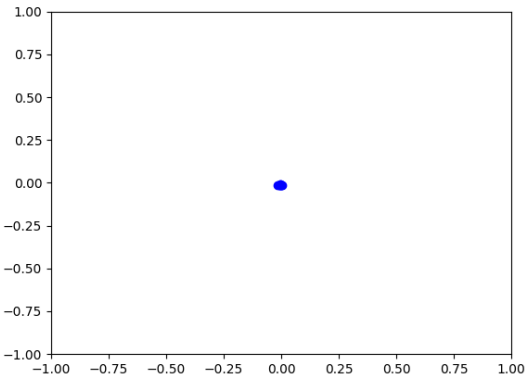
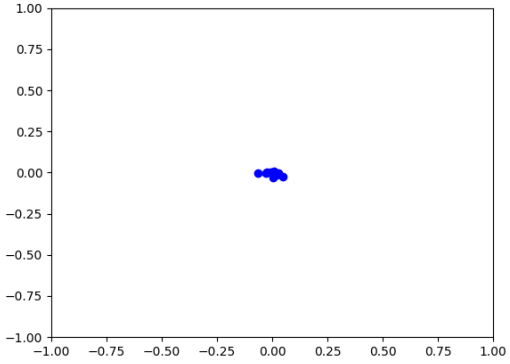





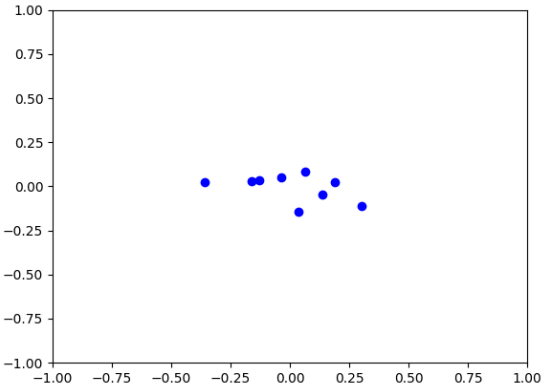
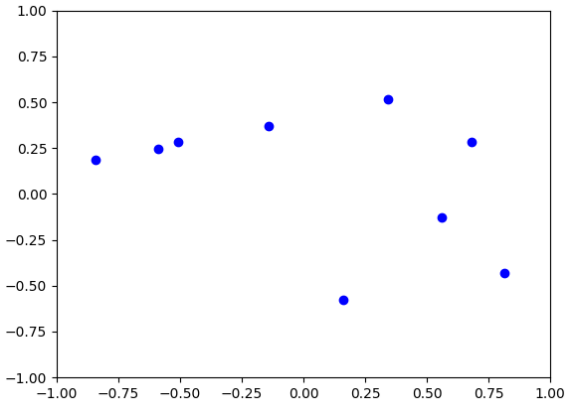
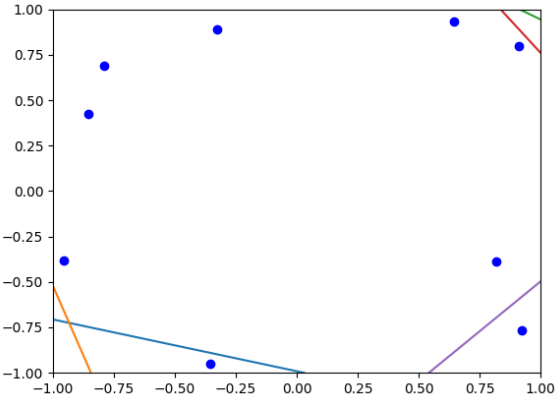
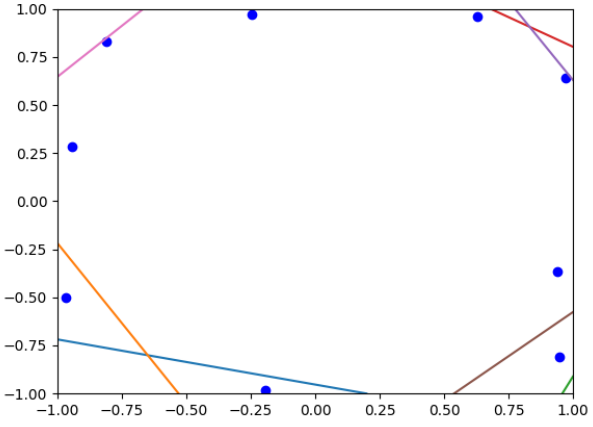
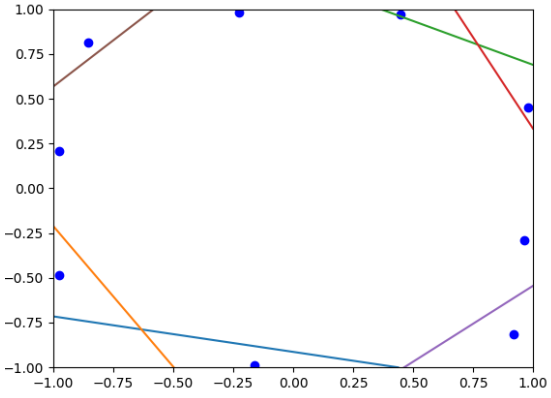
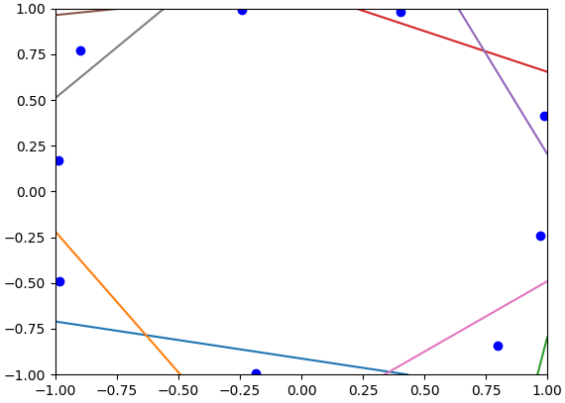
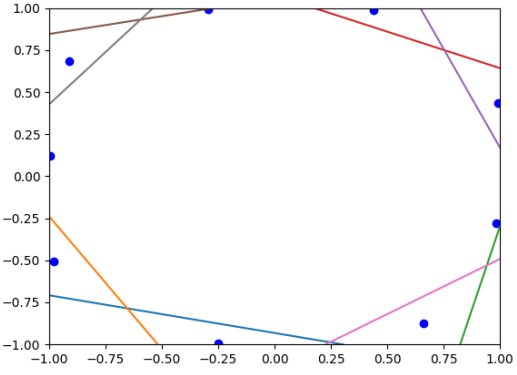
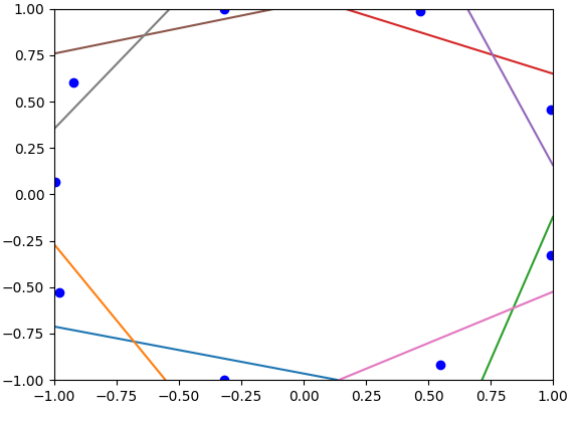









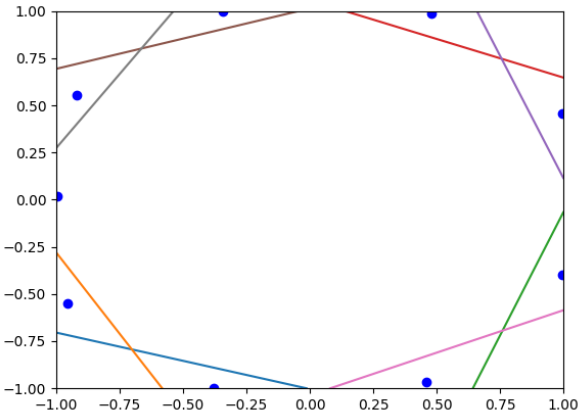







The first 11 images from the encoder input question


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