Project 1 Report

Part 1

1. The final confusion matrix and accuracy of NetLin.

```
<class 'numpy.ndarray'>
[[766.   5.   9.  14.  31.  61.   2.  63.  30.  19.]
 [  7. 664. 108.  18.  29.  22.  57.  16.  27.  52.]
 [  7.  58. 695.  26.  27.  20.  46.  38.  45.  38.]
 [  4.  37.  60. 761.  14.  54.  14.  19.  25.  12.]
 [ 61.  53.  79.  19. 623.  20.  33.  37.  20.  55.]
 [  8.  26. 125.  17.  19. 724.  27.   8.  34.  12.]
 [  5.  21. 148.  10.  27.  25. 723.  20.   8.  13.]
 [ 16.  28.  27.  13.  87.  16.  53. 620.  92.  48.]
 [ 11.  36.  96.  41.   8.  30.  43.   6. 707.  22.]
 [  8.  51.  90.   3.  54.  32.  19.  31.  40. 672.]]

Test set: Average loss: 1.0098, Accuracy: 6955/10000 (70%)
```

2. The final confusion matrix and accuracy of NetFull when the number of hidden nodes is 500. I tried 30, 50, 80, 100, 200, 350, 500, 800 and 1000 for hidden nodes. When the number of hidden nodes is more than 200, the accuracy is around 85%.

```
<class 'numpy.ndarray'>
[[852.   4.   2.   4.  27.  32.   3.  39.  30.   7.]
 [  6. 821.  33.   2.  15.  12.  61.   7.  18.  25.]
 [  7.  11. 830.  49.  14.  18.  25.  12.  18.  16.]
 [  3.  10.  26. 921.   3.  15.   5.   3.   7.   7.]
 [ 34.  26.  24.   6. 820.   8.  31.  19.  20.  12.]
 [  8.  16.  76.   8.  11. 833.  25.   1.  17.   5.]
 [  3.  12.  44.   8.  14.   5. 898.   9.   1.   6.]
 [ 19.  14.  23.   2.  20.   9.  34. 830.  20.  29.]
 [ 11.  27.  27.  46.   3.   8.  30.   3. 838.   7.]
 [  3.  18.  43.   3.  33.   5.  19.  18.  13. 845.]]

Test set: Average loss: 0.4934, Accuracy: 8488/10000 (85%)
```

3. The final confusion matrix and accuracy of NetConv.

For the first convolutional layer, the input is the image, the output is 20 feature maps, and the kernel is a 5*5 square.

For the second convolutional layer, the input is 20 feature maps, the output is 50 feature maps, and the kernel is a 5*5 square.

The window for maximum pooling is 2*2.

```
<class 'numpy.ndarray'>
[[954.   4.   0.   0.  18.   1.   1.  16.   3.   3.]
 [  2. 932.   7.   0.  11.   0.  30.   4.   4.  10.]
 [  9.   8. 894.  31.   9.   4.  18.  12.   4.  11.]
 [  2.   3.  20. 952.   4.   3.   6.   5.   1.   4.]
 [ 17.  13.   3.   6. 924.   0.  11.  11.   6.   9.]
 [  4.  15.  56.   4.   4. 886.  12.   9.   3.   7.]
 [  5.  11.  24.   2.   3.   1. 947.   6.   0.   1.]
 [  2.   3.   6.   0.   3.   0.   5. 966.   4.  11.]
 [  4.  13.  15.   6.  11.   2.   1.   2. 940.   6.]
 [  3.   1.  11.   2.   2.   0.   2.   5.   1. 973.]]

Test set: Average loss: 0.2371, Accuracy: 9368/10000 (94%)
```

4.

a. The accuracy of linear network is lowest among these three networks. The accuracy of the 2-layer fully connected network is higher than that of the linear network but is lower than that of the convolutional network. The accuracy of the convolutional network is highest. Therefore, for image recognition, using the convolutional network is the best. The linear network and the fully connected network is not good enough because the fitting is not adequate.

b. For confusion matrix, the position corresponding to the diagonal represents the serial number of the characters which are recognized correct. Thus, we only need to find the largest number except the diagonal. The row is the target character and the corresponding column is the one recognized by the network.

For linear network, the largest number except diagonal is 148. The row number for this figure is 6 and the column number is 2, which means that the 6th character “ma” is most likely to recognize as 2nd “su” by mistake.

For fully connected network, the largest number except diagonal is 76. The row number for this figure is 5 and the column number is 2, which means that the character 5th “ha” is most likely to recognize as 2nd “su”.

For convolutional network, the largest number except diagonal is 56. The row number for this figure is 5 and the column number is 2, which means that the character “ha” is most likely to recognize as “su”.

The reason why these characters are recognized as wrong characters is that they have similar structure and the handwriting is similar as well. Besides, the handwriting is not clear enough to recognize, so networks may make mistake.

c. (1) linear network.

First, I changed learning rate but did not change others. I tried the values 0.001, 0.002, 0.005, 0.02, 0.05, 0.2 and 0.5. The default value of learning rate is 0.01. If the value is between 0.005 and 0.05, the accuracy is around 70%(69% or 70%). When the value is larger, like 0.2 and 0.5, the accuracy is lower(less than 65%). When the value is smaller, like 0.001, the accuracy is around 67%.

Then, I changed momentum. I tried the values 0.1, 0.3, 0.7 and 0.9. The default value of momentum is 0.5. When the momentum changes, the accuracy is basically unchanged

(around 69% or 70%).

(2) Fully connected network

First, I did not change other data except learning rate. I tried the values 0.001, 0.002, 0.005, 0.02, 0.05, 0.2 and 0.5. When the value is between 0.005 and 0.05, the accuracy is not lower than 80%, and the accuracy is highest (90%) when the value is 0.05. When the value is larger than 0.05, the accuracy decreases. The accuracy will drop to 62% when the learning rate is 0.5. Similarly, when the learning rate is smaller than 0.005, the accuracy will decrease as well.

Second, I did not change other data except momentum. I tried the values 0.1, 0.3, 0.7 and 0.9. When the value becomes larger, the accuracy increases. The accuracy is highest when the momentum is 0.9, with 90%.

(3) Convolutional network

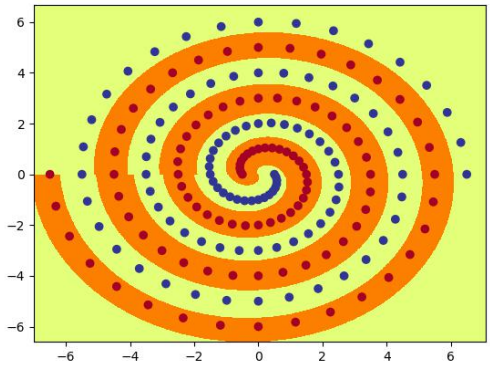
First, I changed the convolution layer. For the first convolution layer, the input is the image, the output is 16 feature maps. For the second convolution layer, the input is the 16 feature maps, the output is 32 feature maps. As for the fully connected layer, the parameter out_features is 200. The accuracy is stable, around 93%.

Then, I changed the convolution layers to (1, 8, 3, 1) and (8, 16, 3, 1) respectively, which means the kernel size is 3*3 and the parameter out_channels also changed. I also changed the parameter out_features of the fully connected layer to 50. The accuracy is lower, about 90%, which means that this network is not fitting enough.

Part 2

2. First, I used the default value 10 for hidden nodes and the final accuracy is 100%. Then I tried value 9, 8, 7 and 6 respectively. When the number of hidden nodes is 7, the final accuracy within 20000 epochs is 100%. However, when the number of hidden nodes is 6, the final accuracy is around 77% and the number of epochs is more than 20000 (other metaparameters are as default). Therefore, the minimum number of hidden nodes within 20000 epochs is 7.

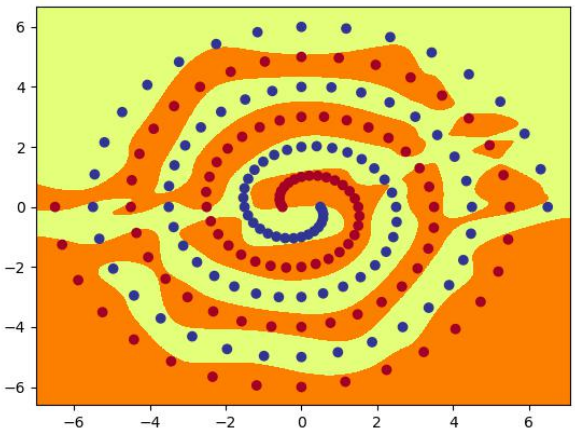
The graph polar_out.png when the number of hidden nodes is 7.



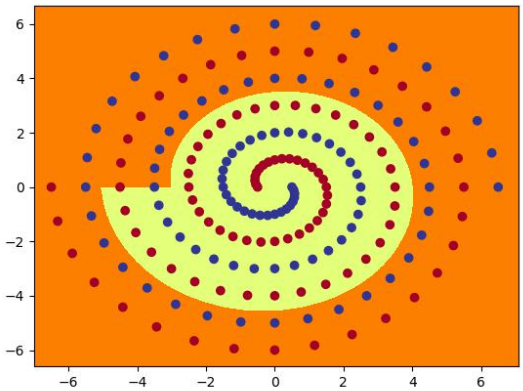
4. First, I used the default value for hidden nodes (10) and the size of the initial weights (0.1), and the final accuracy within 20000 epochs at all runs is 100%.

Then, I tried value 9 for hidden nodes without changing other parameters. The final

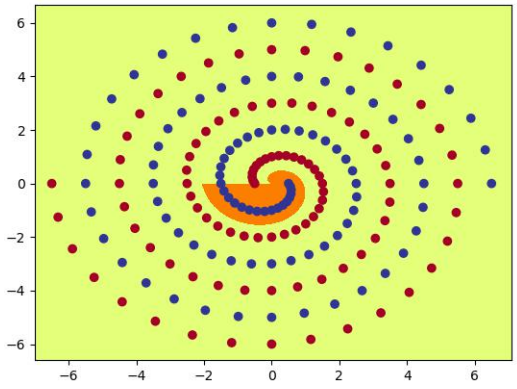
accuracy at all runs is around 55% and the number of epochs is larger than 20000.
As for the size of the initial weights, I tried the value 0.12, 0.15 and 0.05. The final accuracy within 20000 epochs in all runs is 100% for both 0.12 and 0.15, but the accuracy of the value 0.05 is around 50% within more than 20000 epochs.
The picture raw_out.png when the number of hidden nodes is 10 and the size of the initial weights is 0.12 (all other metaparameters are as default).



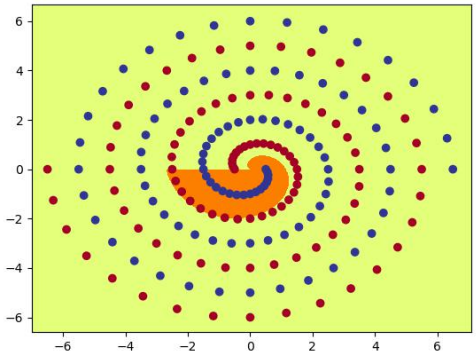
5.
(1) PolarNet
The number of hidden nodes is 7 and other metaparameters not changed.
· polar1_0



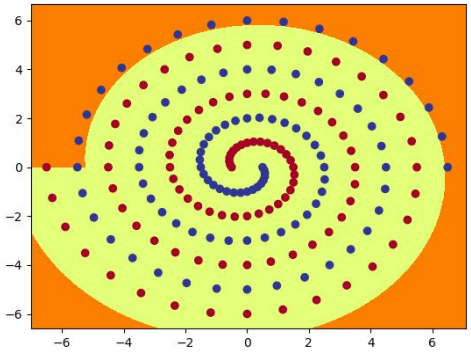
· polar1_1



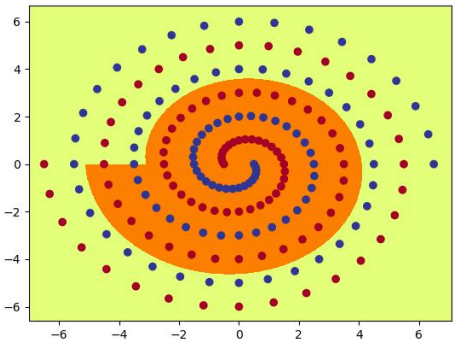
· polar1_2



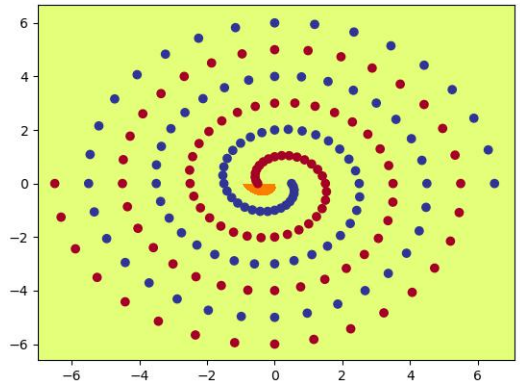
· polar1_3



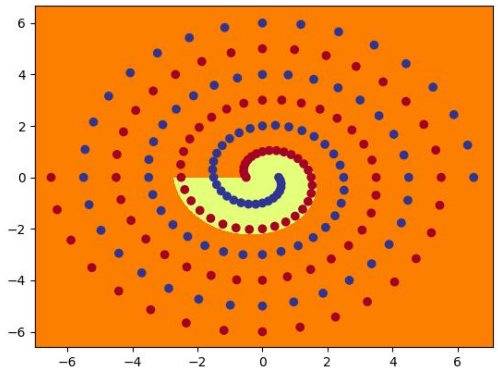
· polar1_4



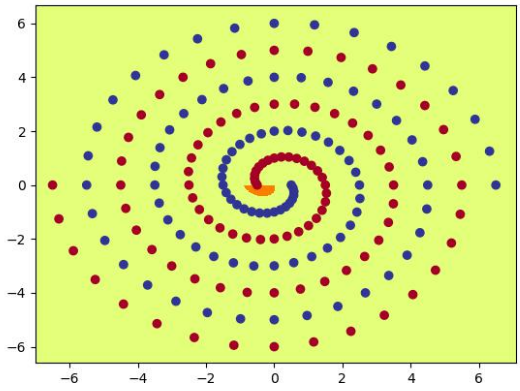
· polar1_5



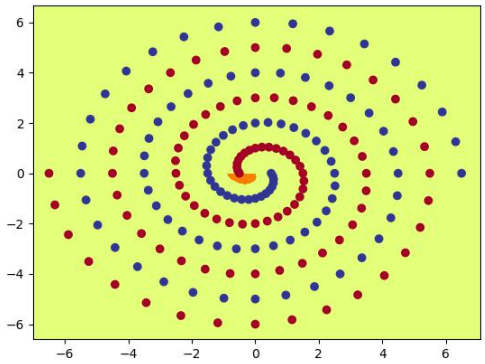
· polar1_6



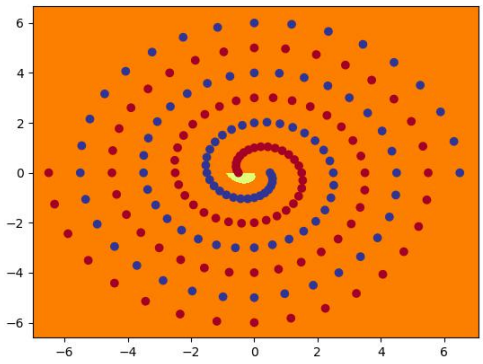
· polar1_7



· polar1_8



· polar1_9

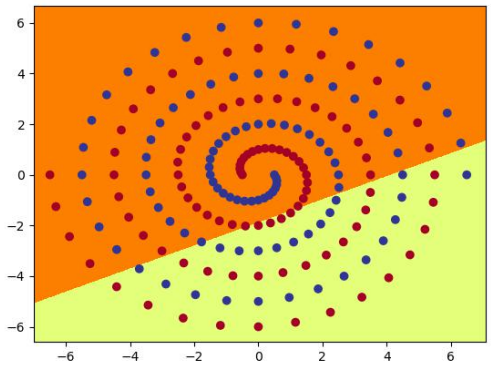


(2) RawNet

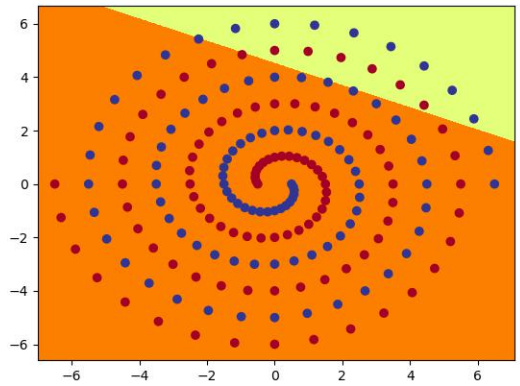
The number of hidden nodes is 10 and the size of the initial weights is 0.12.

① raw1

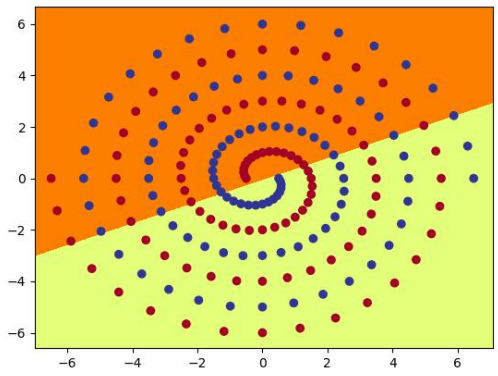
· raw1_0



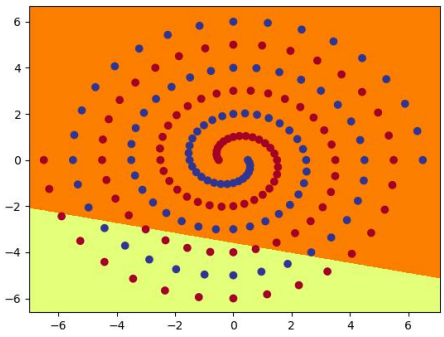
· raw1_1



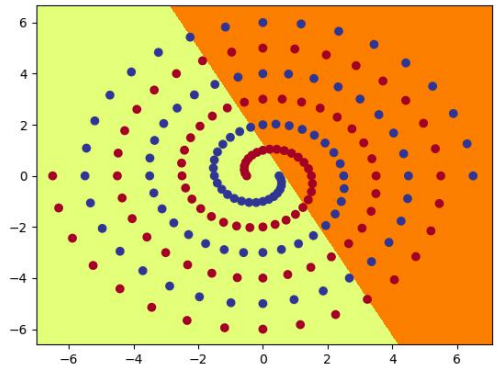
· raw1_2



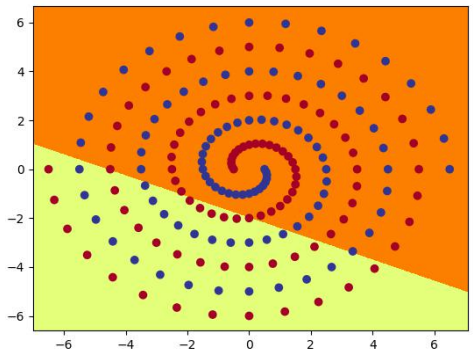
· raw1_3



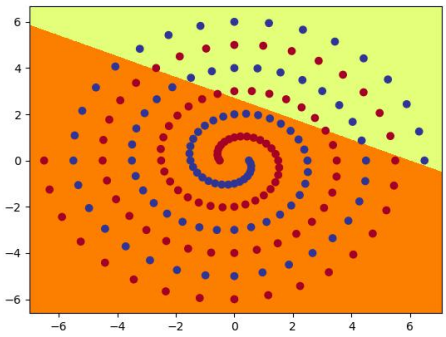
· raw1_4



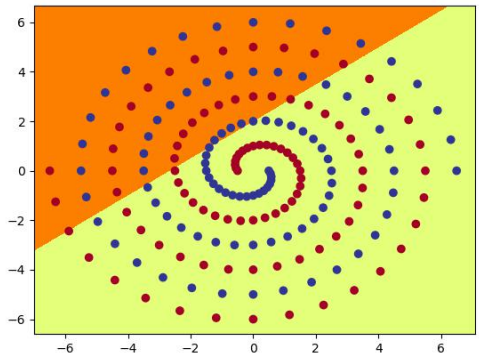
· raw1_5



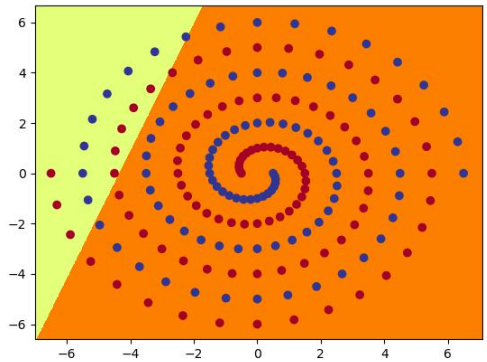
· raw1_6



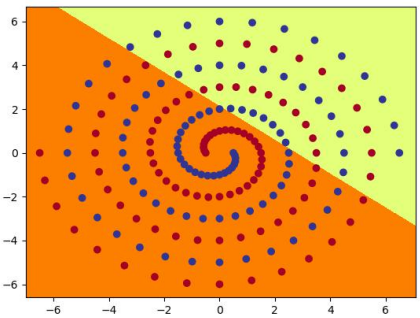
· raw1_7



· raw1_8

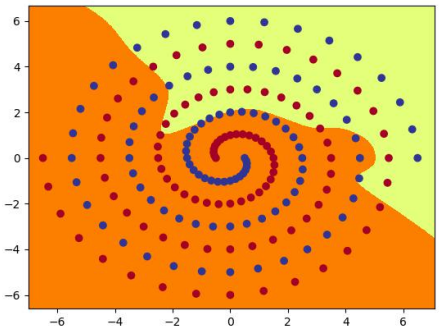


· raw1_9

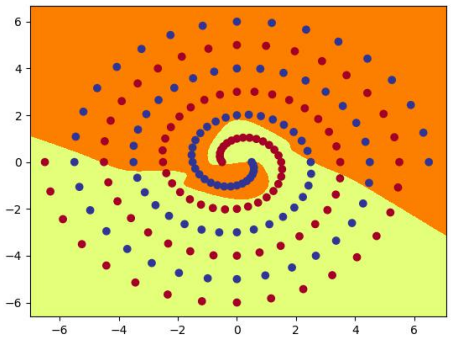


② raw2

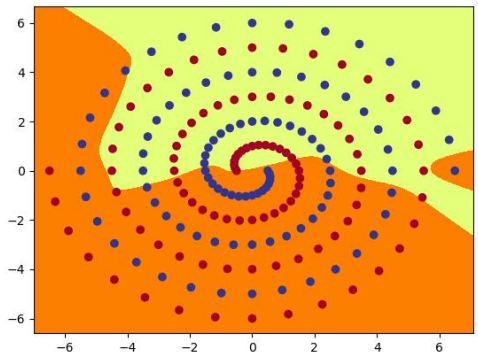
· raw2_0



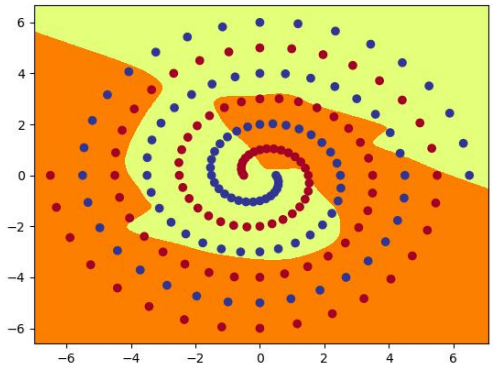
· raw2_1



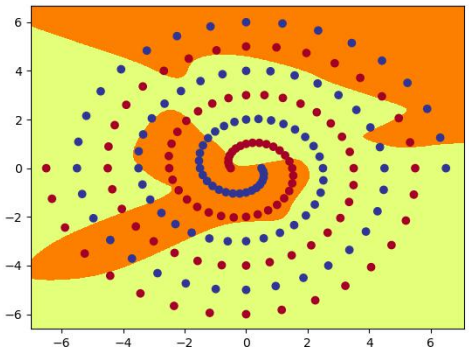
· raw2_2



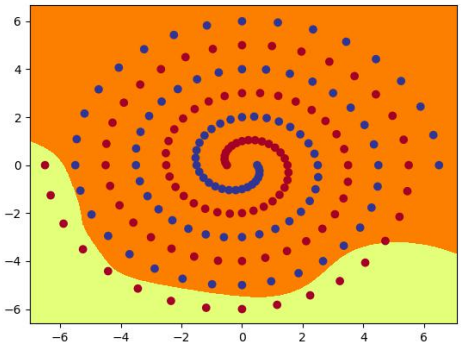
· raw2_3



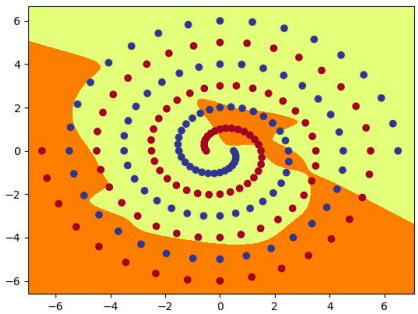
· raw2_4



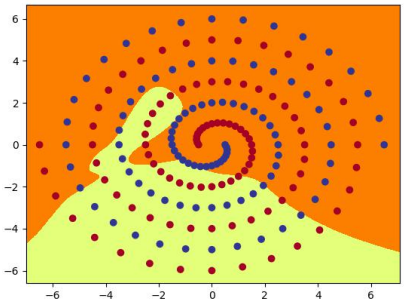
· raw2_5



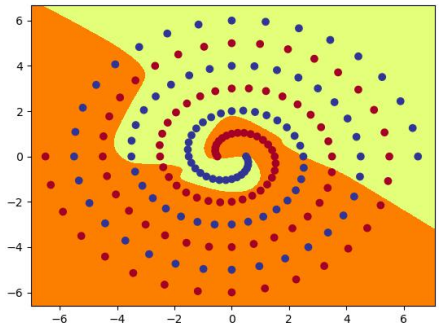
· raw2_6



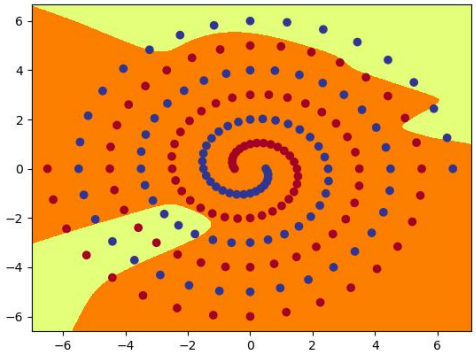
· raw2_7



· raw2_8



· raw2_9



6.

a. In terms of the PolarNet, the function for each hidden layer nodes is not linear. However, for RawNet, the first hidden layer achieves the linear decision boundary but the second hidden layer nodes still compute non-linear function.

For both PolarNet and RawNet, each hidden layer nodes will only classify parts of points, which means they just achieve a part of correct classification boundary. Then using weighted sum to make the network achieve the completed classification.

b. I tried values 0.2, 0.5, 0.05, 0.01, 0.005 and 0.001 for RawNet. When the value is 0.2, the number of epochs is larger than that of default value (0.1), which means that the learning speed is slower with initial weight size increasing. When the value is 0.5, the final accuracy is around 96%, so the effect is not very successful. With values 0.05, 0.01,

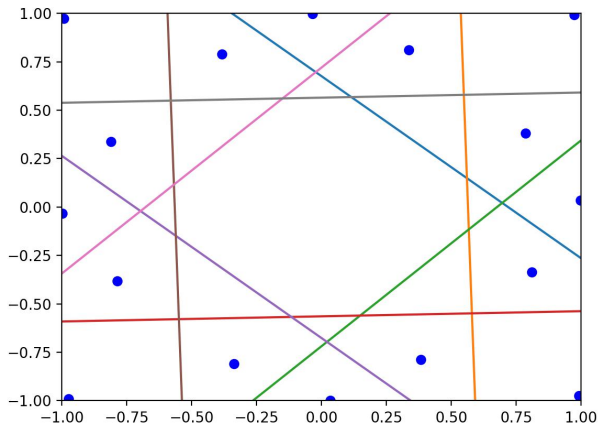
0.005 and 0.001, the final accuracy is around 50% in all runs, which means that these values cause failure. Thus, the value around 0.1 and 0.2 will result in success.

c. (1) I changed batch size to 194 for PolarNet without other parameters changed. It achieves 100% accuracy within 900 epochs. Then I increased the batch size to 388, and the number of epochs is 700. Therefore, the learning speed will be faster with batch size increasing.

(2) I used SGD instead of Adam for RawNet without other parameters changed. It can not achieve 100% accuracy within 20000 epochs and the final accuracy is around 60% in all runs. SGD is the most common optimizer and there is no acceleration effect. Momentum is an improved version of SGD, and RMSprop is an upgraded version of Momentum. Adam is an improved version of RMSprop, so the effect of Adam is better than SGD.

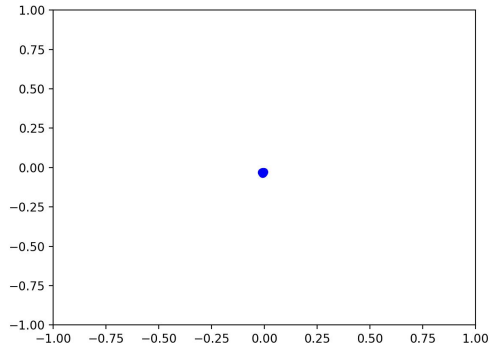
Part 3

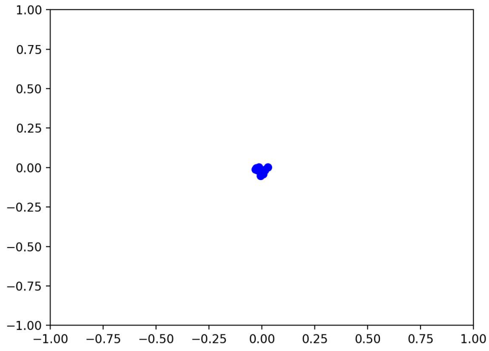
1.

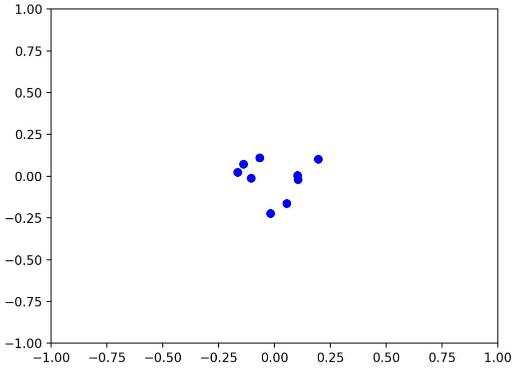


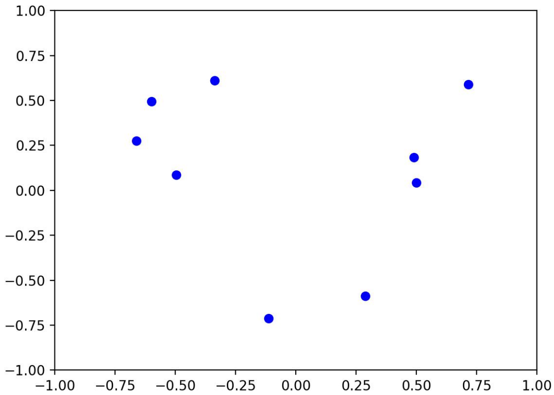
2.

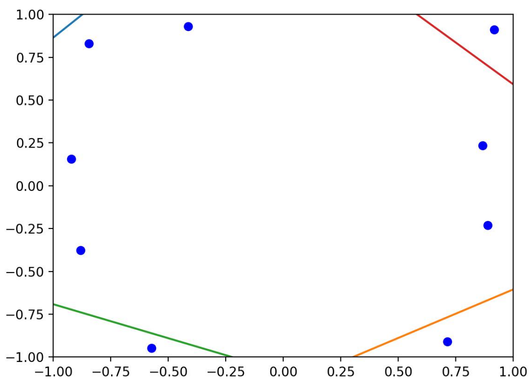
(1) The first 11 images

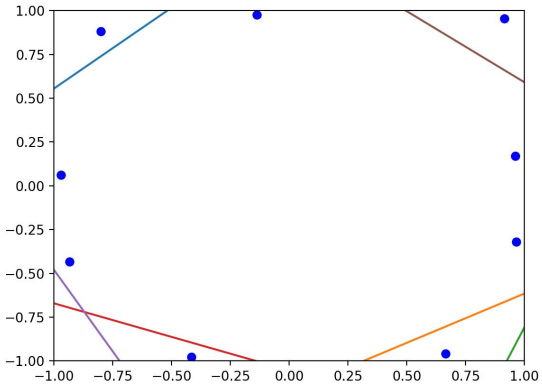


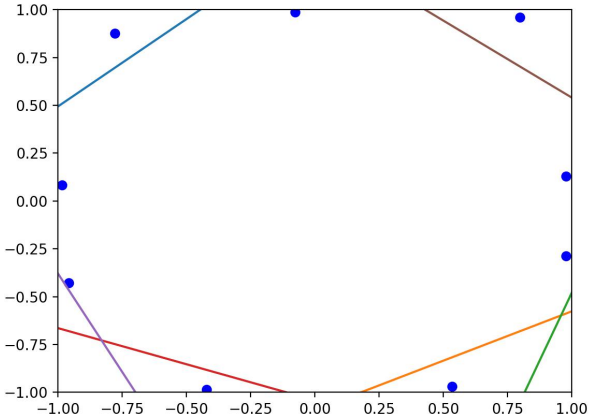

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

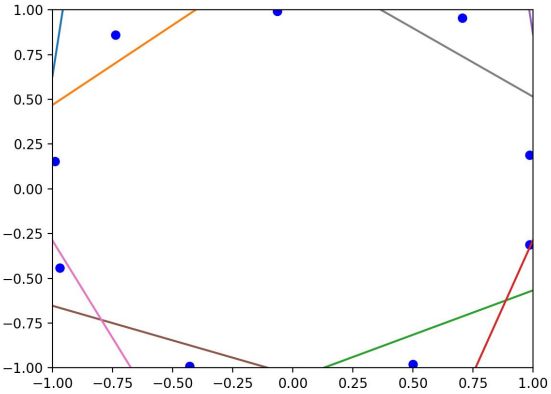

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

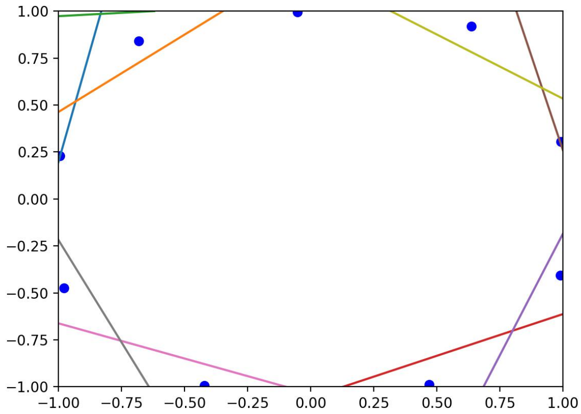

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

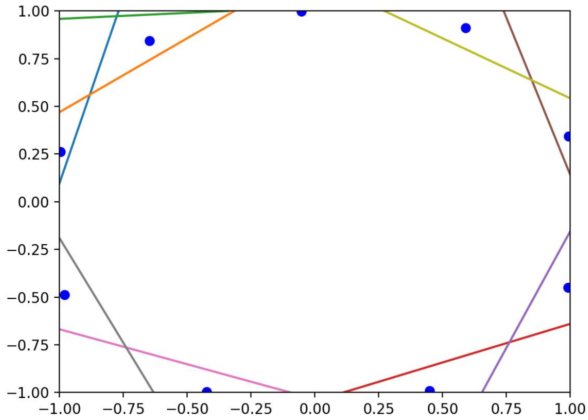

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00

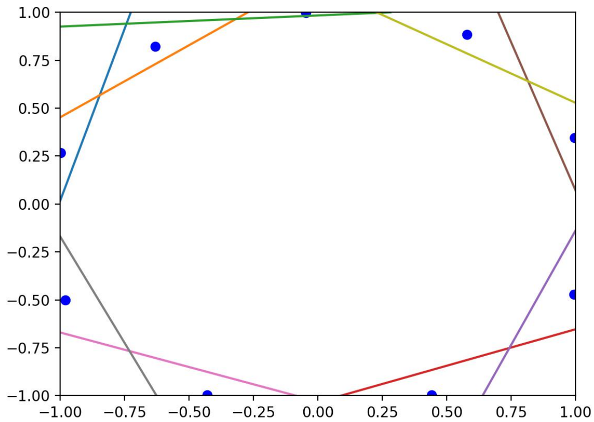

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00


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

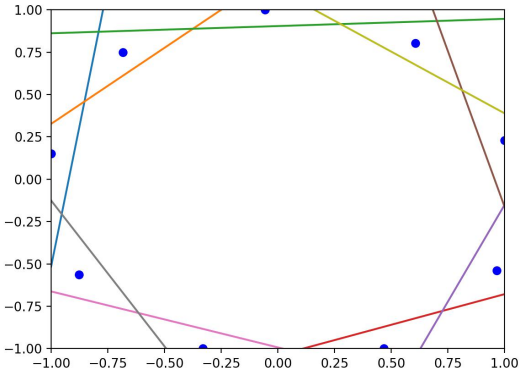

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.00 −0.75 −0.50 −0.25 0.00 0.25 0.50 0.75 1.00





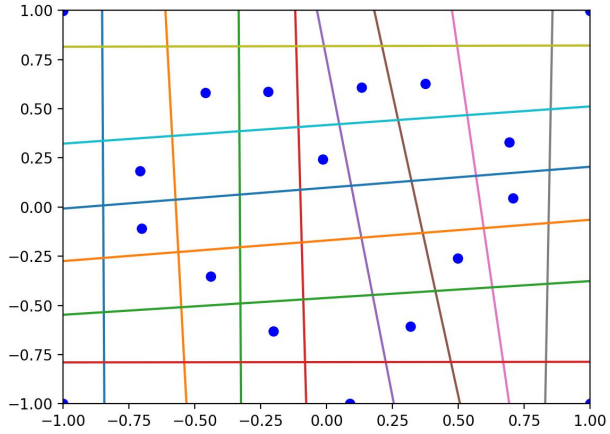


(2) The final image

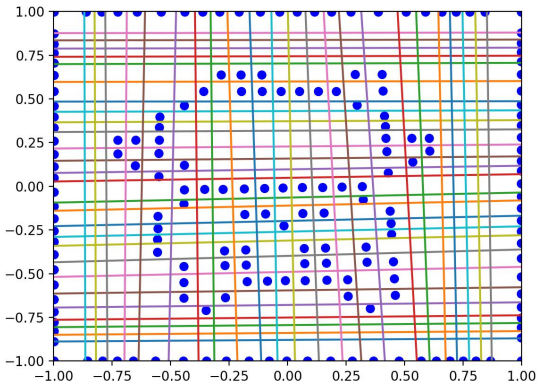


(3) Dots: Starting from the center point and gradually spreading to the boundary. The number gradually increases and finally stabilizes at 9. Dots finally move to the boundary and are divided by lines.
Lines: Lines shrink inward from the boundary. The number of lines is 0 at the beginning, then gradually increases and finally stabilizes at 9. All lines divide all dots in the last.

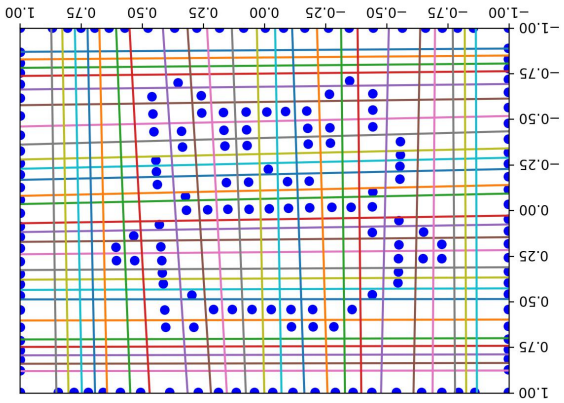
3.



4.(1) target1: Bear
· The beginning image

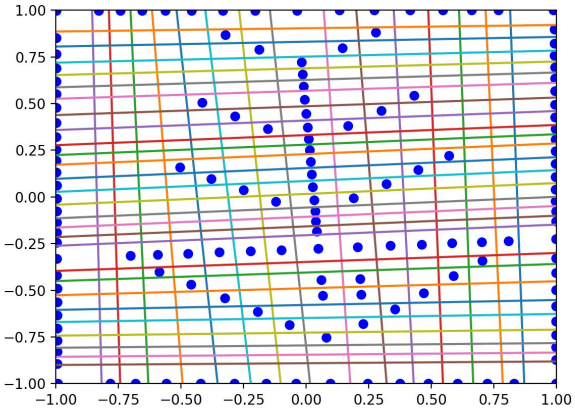


· The inverted image



(2) target2: Fish bones

· The beginning image



· The inverted image