Assignment 1

Chang Wang

Part 1

1.

```
[[767.   5.   7.   5.  60.   8.   5.  18.  10.   8.]
 [  5. 670.  63.  37.  53.  29.  20.  28.  40.  53.]
 [  9. 108. 691.  58.  76. 122. 150.  27.  97.  83.]
 [ 12.  18.  26. 758.  19.  17.  10.  12.  39.   3.]
 [ 30.  28.  26.  15. 627.  19.  27.  87.   6.  51.]
 [ 64.  22.  20.  58.  20. 727.  24.  16.  32.  31.]
 [  2.  58.  47.  14.  32.  26. 718.  52.  46.  20.]
 [ 62.  13.  35.  18.  36.   9.  21. 622.   7.  30.]
 [ 30.  27.  48.  26.  20.  32.  11.  88. 699.  41.]
 [ 19.  51.  37.  11.  57.  11.  14.  50.  24. 680.]]

Test set: Average loss: 1.0085, Accuracy: 6959/10000 (70%)
```

2.

```
[[846.   8.   6.   6.  47.   9.   3.  13.   8.   2.]
 [  5. 813.  23.  15.  32.  14.  12.  13.  39.  24.]
 [  2.  34. 817.  31.  22.  72.  67.  18.  27.  47.]
 [  4.   5.  40. 897.   8.  11.   7.   3.  48.   5.]
 [ 26.  21.  12.   5. 799.  12.  17.  24.   2.  28.]
 [ 30.   8.  20.  15.   7. 829.   6.   8.   9.  10.]
 [  4.  49.  29.   6.  28.  23. 864.  36.  30.  22.]
 [ 41.   6.  16.   6.  19.   2.  10. 821.   3.  13.]
 [ 36.  24.  19.  10.  20.  19.   3.  28. 829.  13.]
 [  6.  32.  18.   9.  18.   9.  11.  36.   5. 836.]]

Test set: Average loss: 0.5354, Accuracy: 8351/10000 (84%)
```

3.

```
[[965.   4.  11.   3.  20.   4.   6.  10.   3.   9.]
 [  4. 941.   5.   3.  12.   8.   4.   7.  17.   5.]
 [  3.   6. 915.  16.   3.  47.  14.  10.  11.  18.]
 [  0.   1.  24. 969.   4.   4.   1.   2.   7.   3.]
 [ 18.   4.   6.   0. 925.   3.   6.   2.   2.   6.]
 [  3.   3.  13.   2.   4. 909.   2.   4.   5.   0.]
 [  1.  29.   9.   2.  15.  14. 962.  13.   9.   4.]
 [  2.   2.   5.   1.   3.   2.   1. 940.   2.   4.]
 [  3.   4.   3.   1.  11.   4.   1.   3. 943.  10.]
 [  1.   6.   9.   3.   3.   5.   3.   9.   1. 941.]]

Test set: Average loss: 0.3056, Accuracy: 9410/10000 (94%)
```

4.

a) For the model Netlin, it is insufficient complexity and under fitting.

For the model Netfull, it is not suitable for image recognition tasks. This is because it has too many parameters, when the image becomes larger, the number of parameters will be much larger, thus its scalability is poor. Moreover, It is known that the more layers the network has, the stronger its expression ability will be, but we cannot have a deep fully connected neural network, and this will limit its capability.

For the model NetConv, it has many advantages. It is a local connection, each neuron is no longer connected to all the neurons in the upper layer, this could reduce a lot of parameters. Moreover, a group of connections can share the same weight, instead of each connection having a different weight, this could also reduce parameters.

b) For the model Netfull and NetConv, the character “ha” is most likely to be mistaken for character “su”, and for the model Netlin, its also likely to be mistaken for “su”, but “ma” is most likely to be mistaken for “su”. This might because their handwriting are similar.

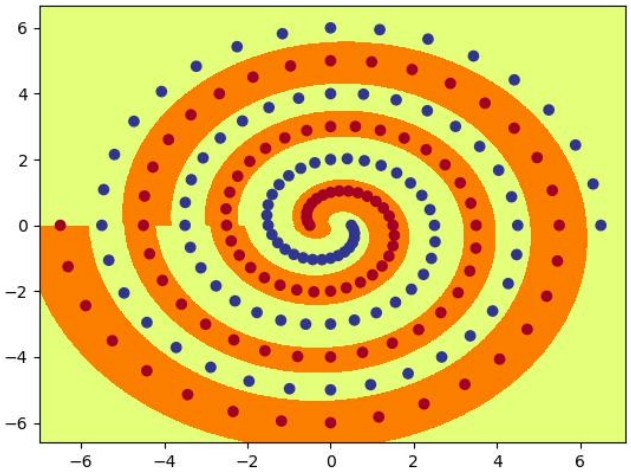
c) I changed the hidden nodes for the model NetConv, the value range should be (10, 39200), and I found when the value is small, code runs faster but the accuracy is lower. When I keep increasing this value, though the speed is slowing down, the accuracy are increasing and achieved 94% at 1000.

I also tried to removed relu, and found that the accuracy decreased.

Part 2

Q1. Code is provided in the file named spiral.py.

Q2. Here is the polar_out.png:



When I tried 5 for hidden nodes, the accuracy remained around 78 in 10000-20000 epochs

```
ep:20200 loss: 0.0237 acc: 77.84
ep:20300 loss: 0.0237 acc: 77.84
ep:20400 loss: 0.0237 acc: 77.84
ep:20500 loss: 0.0237 acc: 77.84
```

I got 100% accuracy within 20000 epoch when hidden nodes is 6.

```
ep:11800 loss: 0.0073 acc: 67.01
ep:11900 loss: 0.0074 acc: 67.01
ep:12000 loss: 0.0079 acc: 67.01
ep:12100 loss: 0.1295 acc: 100.00
```

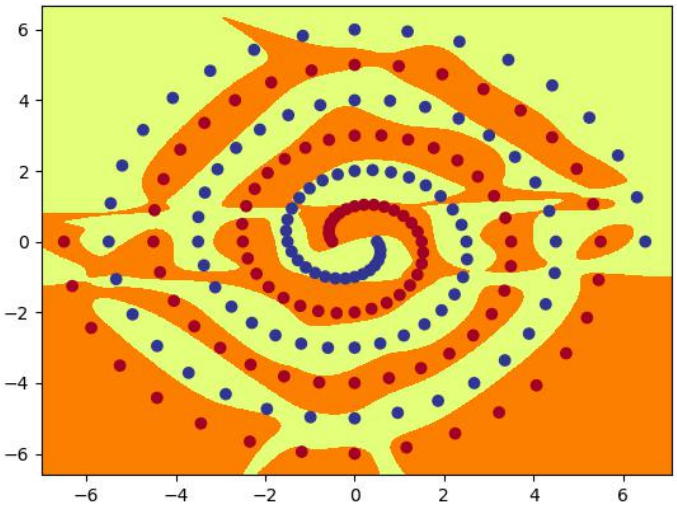
Therefore, the minimum number of hidden nodes is 6.

Q3. Code is provided in the file named spiral.py.

Q4. The value I found for the size of the initial weight is 0.2.

```
ep: 9300 loss: 0.0287 acc: 99.48
ep: 9400 loss: 0.0284 acc: 99.48
ep: 9500 loss: 0.0281 acc: 99.48
ep: 9600 loss: 0.0278 acc: 100.00
```

Here is the row_out.png:



Q5. Code is provided in the file named spiral.py.

Q6. When the number of hidden nodes is 10, I tried 0.001, 0.01, 0.1, 0.2 and 0.3 for the initial weight, and here is the output:

0.001:

```
ep:19800 loss: 0.7131 acc: 50.52
ep:19900 loss: 0.7131 acc: 50.52
ep:20000 loss: 0.7131 acc: 50.52
ep:20100 loss: 0.7131 acc: 50.52
```

0.01:

```
ep: 3600 loss: 0.0818 acc: 98.97
ep: 3700 loss: 0.0715 acc: 99.48
ep: 3800 loss: 0.0629 acc: 99.48
ep: 3900 loss: 0.0557 acc: 100.00
```

0.1:

```
ep: 3100 loss: 0.0620 acc: 97.94
ep: 3200 loss: 0.0551 acc: 98.45
ep: 3300 loss: 0.0492 acc: 98.97
ep: 3400 loss: 0.0438 acc: 100.00
```

0.2:

```
ep: 4200 loss: 0.0149 acc: 99.48
ep: 4300 loss: 0.0147 acc: 99.48
ep: 4400 loss: 0.0145 acc: 99.48
ep: 4500 loss: 0.0143 acc: 100.00
```

0.3:

```
ep: 6400 loss: 0.0204 acc: 99.48
ep: 6500 loss: 0.0198 acc: 99.48
ep: 6600 loss: 0.0193 acc: 99.48
ep: 6700 loss: 0.0188 acc: 100.00
```

It seems that 0.1 is better, however, when I tried 0.1 for the initial weight again, the output changed.

```
ep: 5000 loss: 0.0161 acc: 99.48
ep: 5100 loss: 0.0155 acc: 99.48
ep: 5200 loss: 0.0150 acc: 99.48
ep: 5300 loss: 0.0139 acc: 100.00
```

In this case I tried 0.01 and 0.2 again because they are similar with 0.1.
0.01:

```
ep: 2500 loss: 0.2035 acc: 88.66
ep: 2600 loss: 0.1638 acc: 94.85
ep: 2700 loss: 0.1267 acc: 97.94
ep: 2800 loss: 0.0998 acc: 100.00
```

0.2:

```
ep: 4400 loss: 0.0275 acc: 98.45
ep: 4500 loss: 0.0273 acc: 98.97
ep: 4600 loss: 0.0258 acc: 99.48
ep: 4700 loss: 0.0242 acc: 100.00
```

It is hard to judge which is the best, but I think initial weight = 0.01 is good. Then I need to find the minimal number of hidden nodes.
Hidden nodes = 5:

```
ep:19700 loss: 0.0835 acc: 93.81
ep:19800 loss: 0.0834 acc: 93.81
ep:19900 loss: 0.0834 acc: 93.81
ep:20000 loss: 0.0834 acc: 93.81
ep:20100 loss: 0.0833 acc: 93.81
```

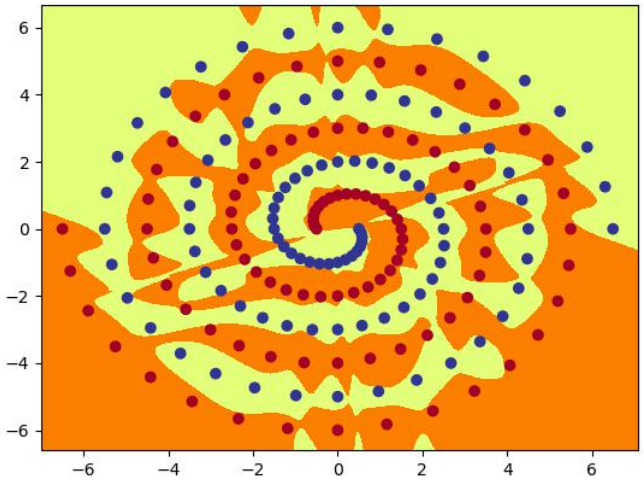
Hidden nodes = 6:

```
ep:19700 loss: 0.0764 acc: 98.45
ep:19800 loss: 0.0764 acc: 98.45
ep:19900 loss: 0.0764 acc: 98.45
ep:20000 loss: 0.0764 acc: 98.45
ep:20100 loss: 0.0763 acc: 98.45
```

Hidden nodes = 7:

```
ep:14800 loss: 0.0540 acc: 97.94
ep:14900 loss: 0.0543 acc: 97.94
ep:15000 loss: 0.0539 acc: 98.97
ep:15100 loss: 0.0536 acc: 100.00
```

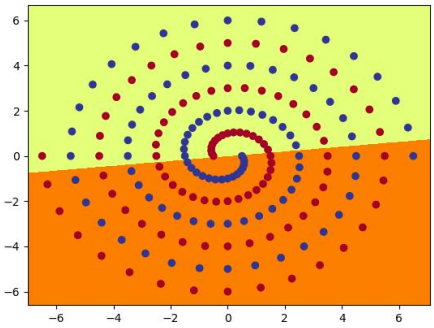
In conclusion, a good value for initial weight size is 0.01, and the minimum number of hidden nodes is 7. Here is the short_out.png:

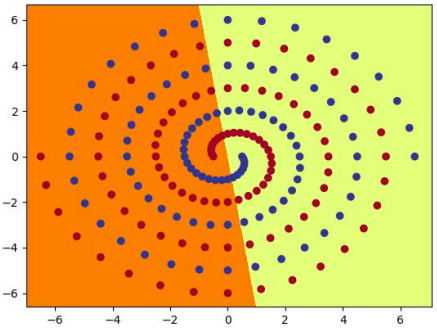


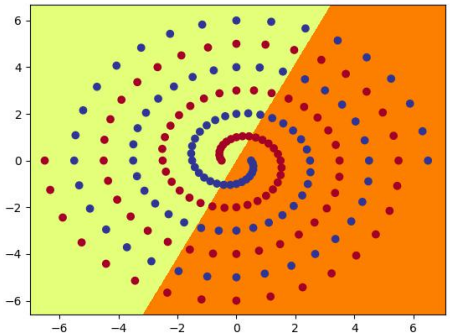
Q7.

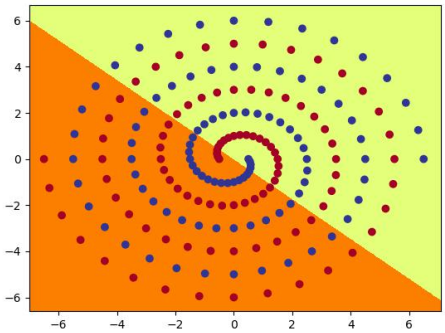
ShortNet(hid = 7)：

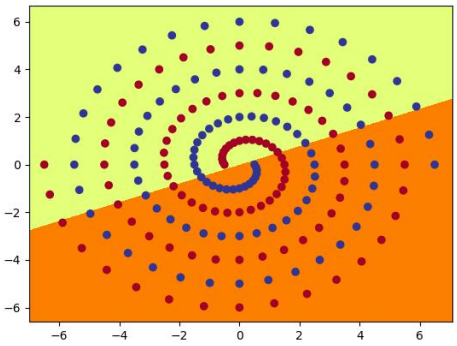
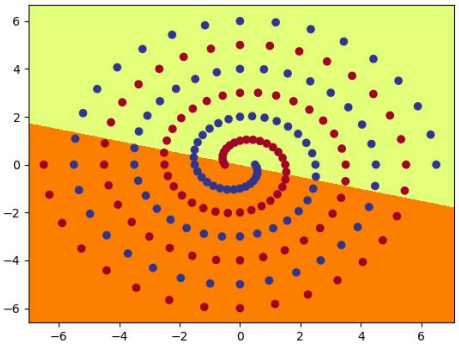
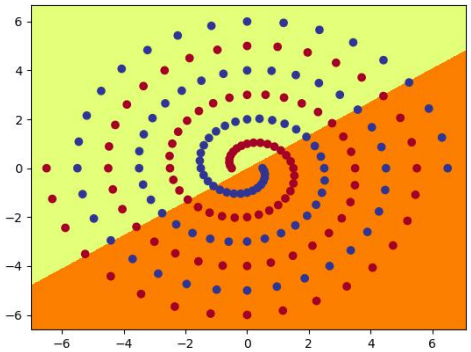
Hid 1:



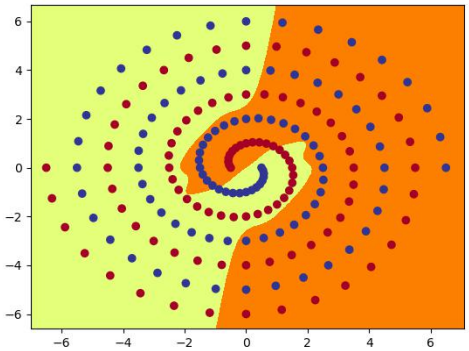
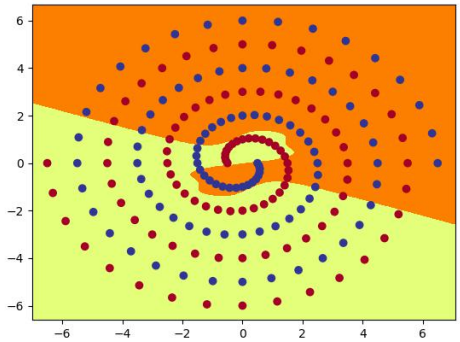
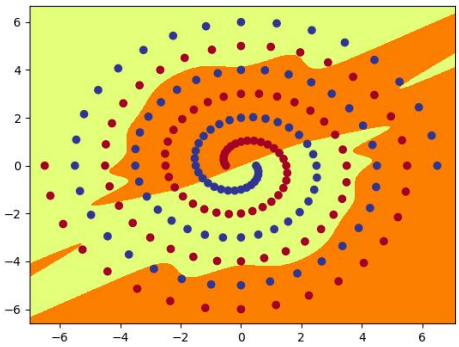
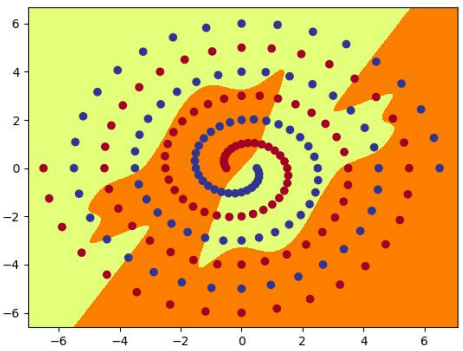


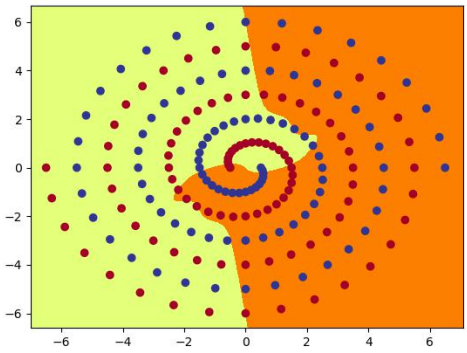
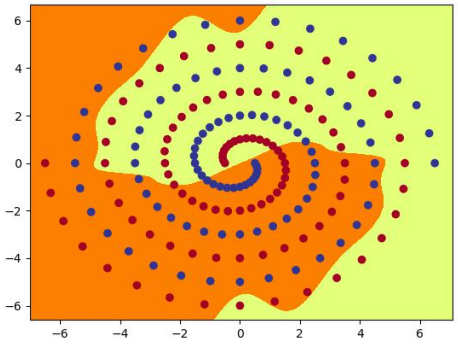
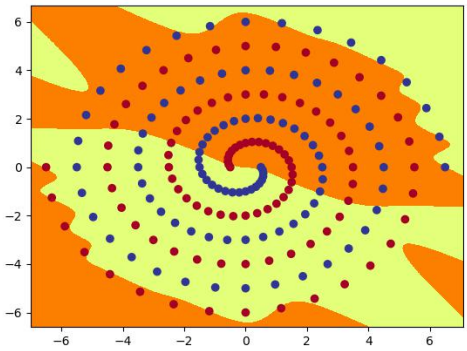




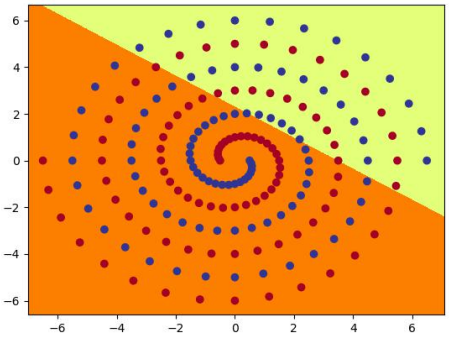
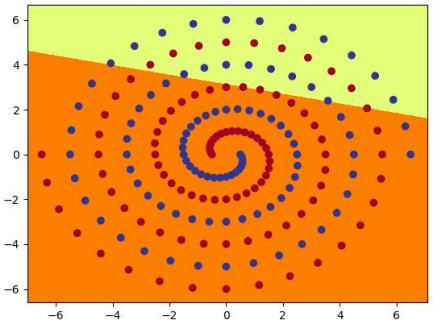
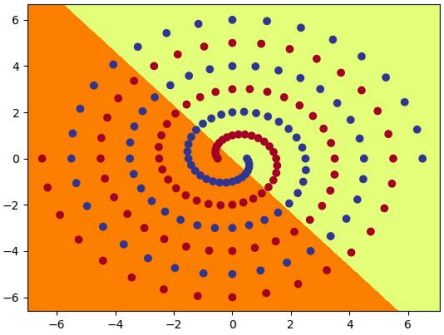
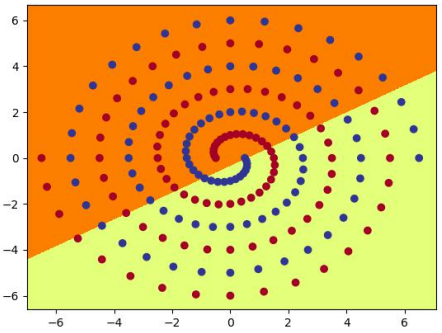
Hid 2:

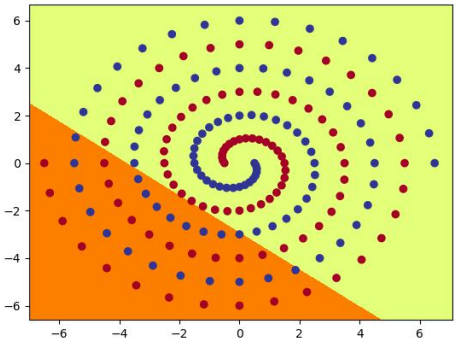
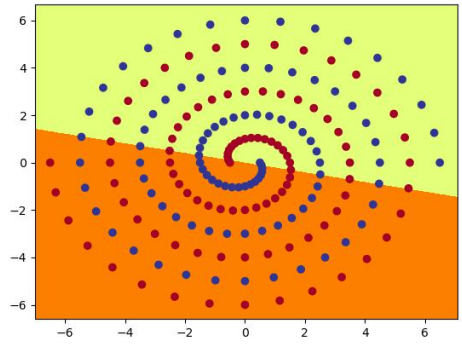
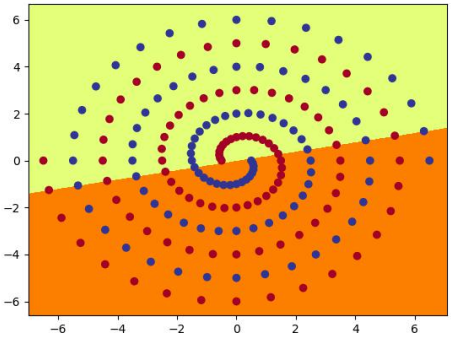
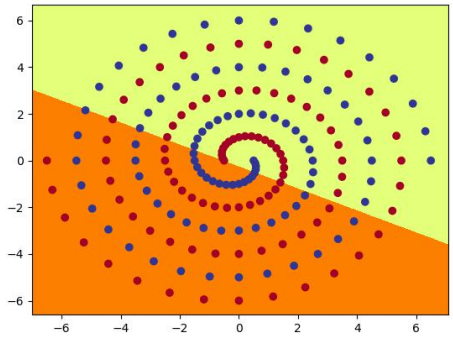
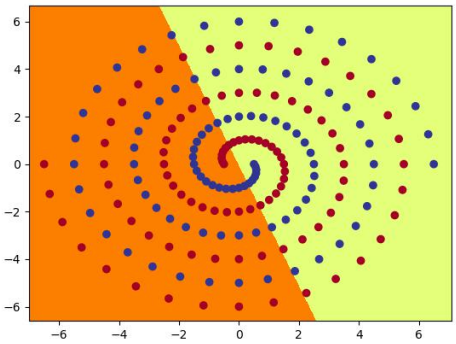
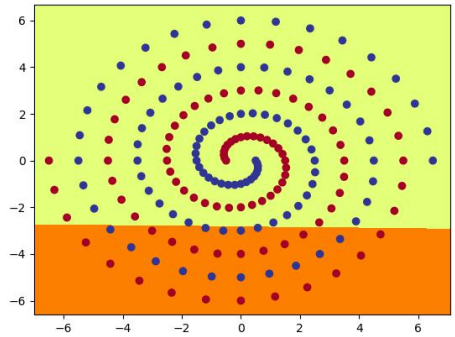





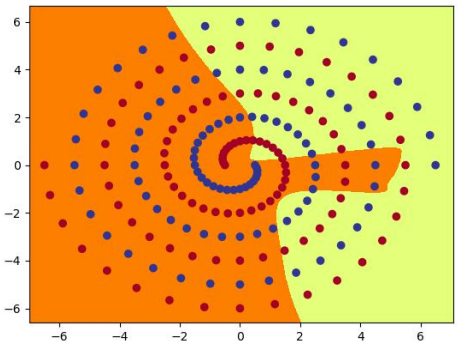
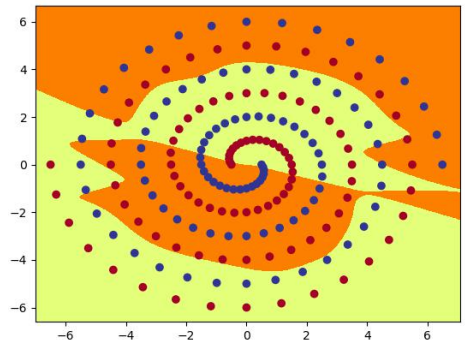
RawNet:

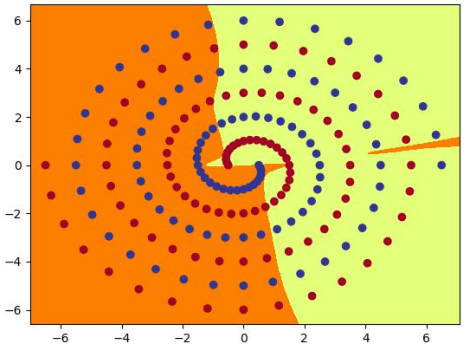
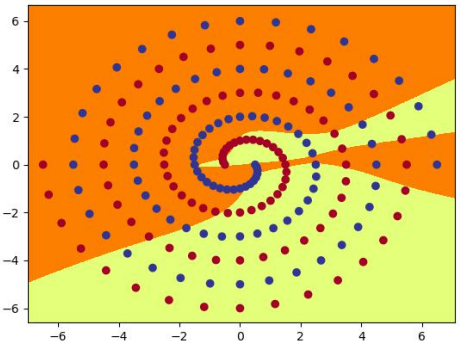
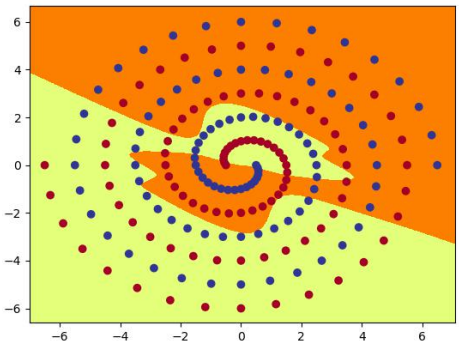
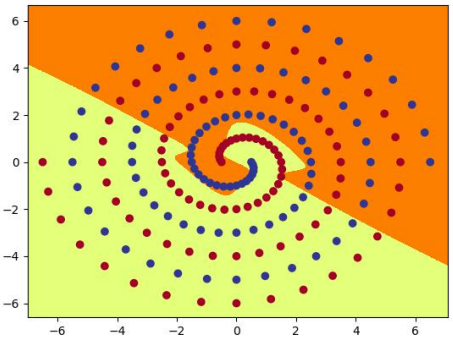
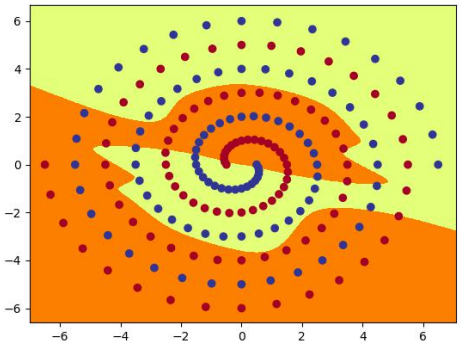
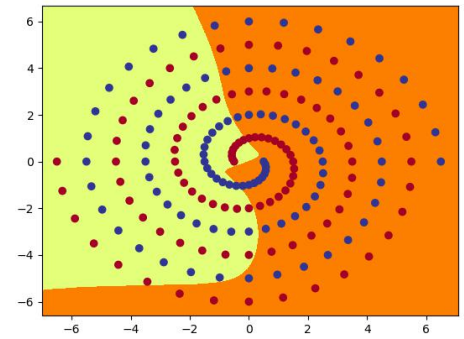
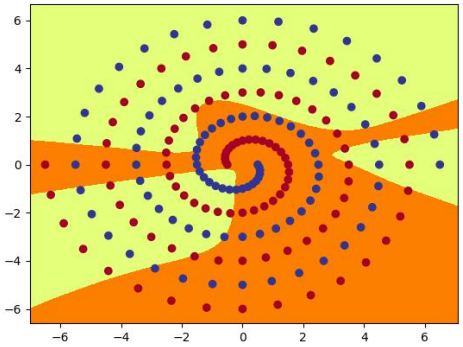
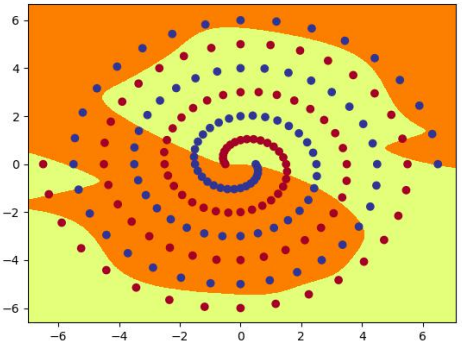
Hid 1:

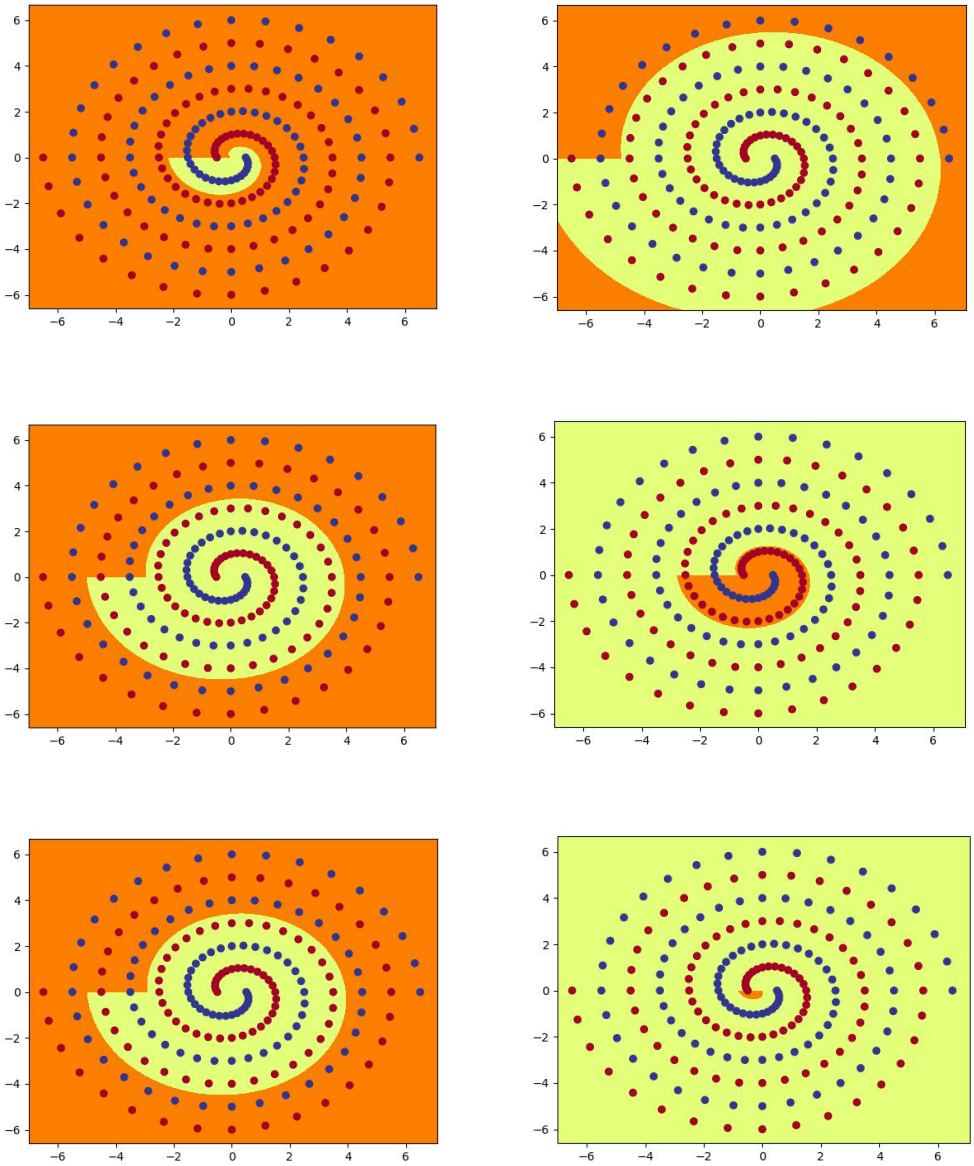







Hid 2:

PolarNet(hid = 6):



Q8.

a) For the model RawNet and ShortNet, the output of the first hidden layer shows it is classified by line, and the output of the second hidden layer shows it is classified by irregular curve. For the model PolarNet, it is classified by the regular curve. All these curves will be synthesized to the final output by weight recombination.

b) It seems that the value for initial weight almost not influence the speed and success of learning. When the value is under 0.1, such as 0.01, many times it could success learn and the speed is good, but I found sometimes it cannot correctly classify within 20000 epochs. If the weight is too small, such as 0.001, most of my test is fail. The value between 0.1 and 0.3 seems good, most of my test could correctly classify all data within 20000 epochs, but the speed is random, it is hard to judge the relationship between the value for initial weight and the speed.

c) The naturalness of the model PolarNet is better than the model RawNet and ShortNet, I learned that if layer is enough, these two model may classify correctly for all of the train data, however, it may lead to the overfitting, when it need to test, the accuracy may be not better than the model

PolarNet. Therefore, we need to select different model for different kind of data set.

d) I changed tanh to relu, and used the same parameter, I found it is hard to get 100% accuracy within 20000 epochs, it is just around 72% and seems that tanh is better.

```
ep:19900 loss: 0.3286 acc: 72.68
ep:20000 loss: 0.3262 acc: 72.68
ep:20100 loss: 0.3296 acc: 73.20
ep:20200 loss: 0.3302 acc: 73.71
ep:20300 loss: 0.3307 acc: 71.65
```