**Project 1 - Characters, Spirals and Hidden Unit Dynam**

**YOUR NAME HERE**

**Part 1. Japanese Character Recognition**

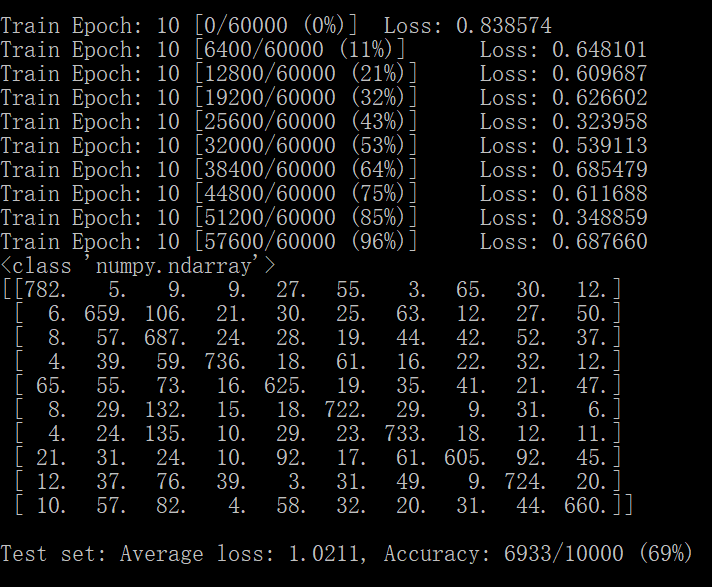
When debugging, I found that the number of samples cannot be exactly split into batches of size 64, that is, len(dataset) % batch\_size is not zero.

There are two possible solutions: First, use drop\_last to drop samples of the last incomplete batch; Second, change the value of batch\_size so that len(dataset) % batch\_size = 0.

The results below are of the second solution.

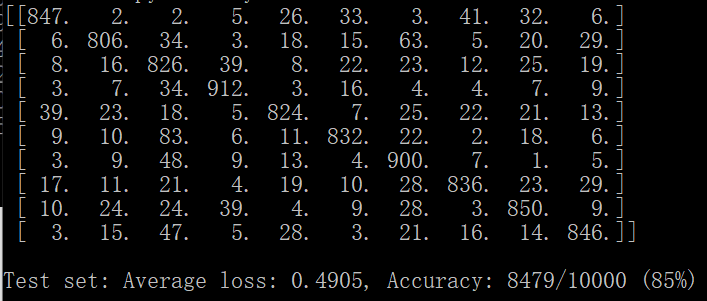
1. A linear function followed by log softmax.

Batch\_size = 64 and drop\_last = True in dataloader:

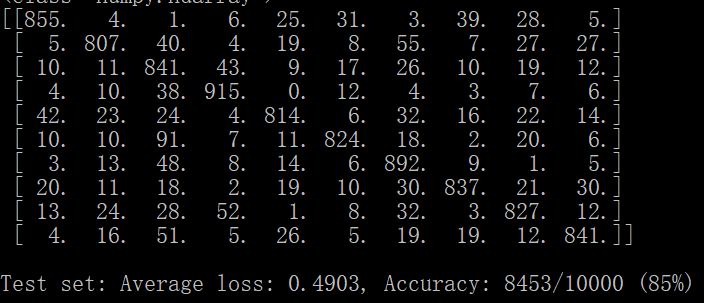


1. A fully connected 2-layer network, using tanh at the hidden nodes and log softmax at the output node.

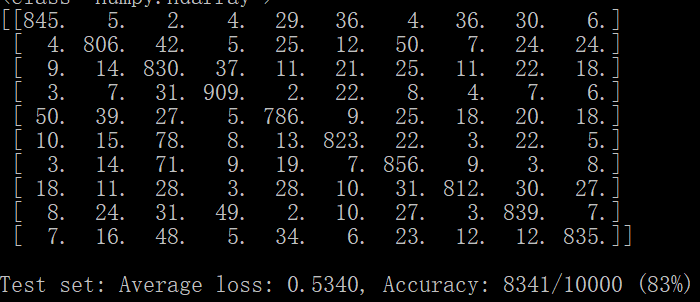
Hidden\_size = 500:



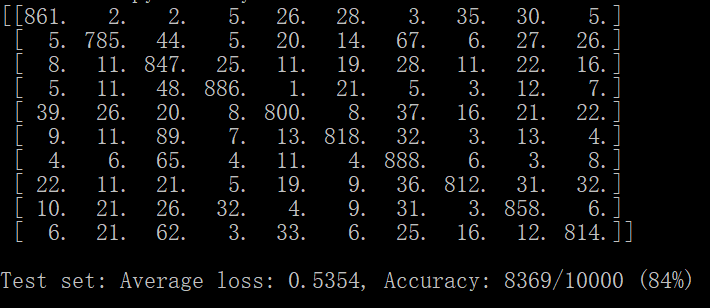
Hidden\_size = 300:



Hidden\_size = 100:

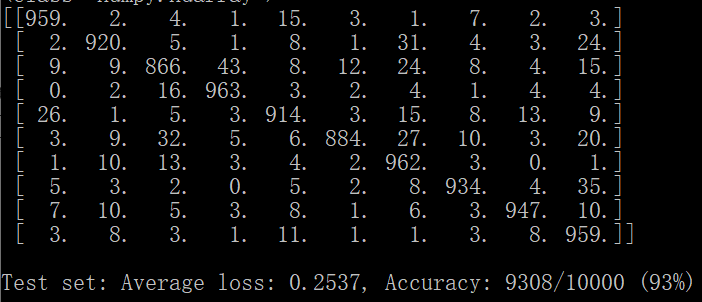


Hidden\_size = 50:



Conclusion: The value that achieves high accuracy on the test set should be relatively large, such as 300 or 500.

1. Two convolutional layers plus one fully connected layer layer, all using relu activation function, followed by the output layer.



1. Discuss what you have learned from this exercise, including the following points:
2. **the relative accuracy of the three models,**

Obviously, the accuracy of convolutional layer is much better than linear function, and the performance of fully connected network (or simple forward neural network) is between them.

It seems like that the more complex the model is, the more accurate it is. This is because all the three models are relatively easy compared to really complicated models, so they are not likely to face overfitting problems. Therefore, a more complicated model, such as convolutional neural networks, can be more expressive for picture information and suits quite well for image processing.

1. **the confusion matrix for each model: which characters are most likely to be mistaken for which other characters, and why?**

0="o", 1="ki", 2="su", 3="tsu", 4="na", 5="ha", 6="ma", 7="ya", 8="re", 9="wo"



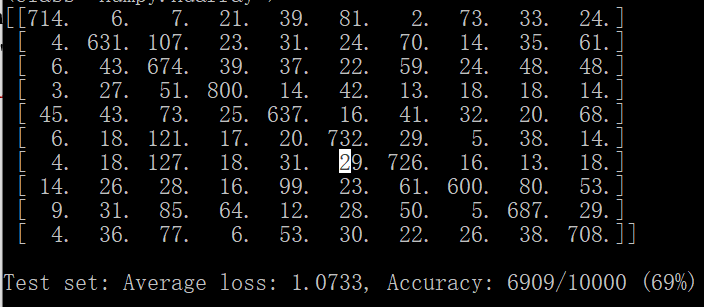
According to all the confusion matrix above, the entries: (1,2), (4,0), (5,2), (6,2), (9,2) in confusion matrix is relatively high except diagonal values. That is, “su” is likely to be mistaken for “ki”, “ha”, “ma”, and “wo”, and “o” is likely to be mistaken for “na”.

It is consistent with human being’s visual sense: “su” is similar to “ki”, “ha”, “ma”, and “wo” because they are more complex than other characters in structure. “o” is similar to “na” with a point on the up-right corner and one and a half-open cycle in the bottom.

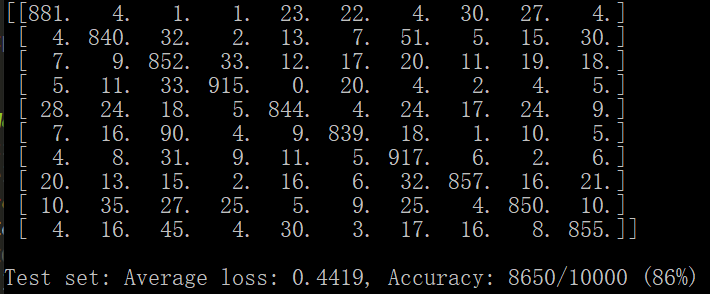
1. **you may wish to experiment with other architectures and/or metaparameters for this dataset, and report on your results; the aim of this exercise is not only to achieve high accuracy but also to understand the effect of different choices on the final accuracy.**

In Task 1, I try to change batch\_size to 16 (same as the first solution mentioned at the beginning) and find that there is no big difference between the two solutions.

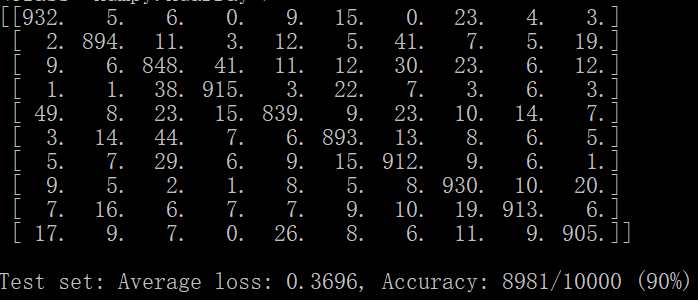
Batch\_size = 16:



In Task 2, I explore the difference between tanh and ReLU (hidden\_size = 300), the performance is higher than tanh.



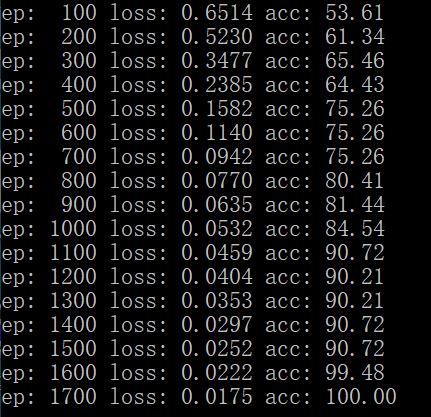
In Task 3, I remove the second convolutional layer, and the accuracy becomes lower than two-convolutional-layer model.

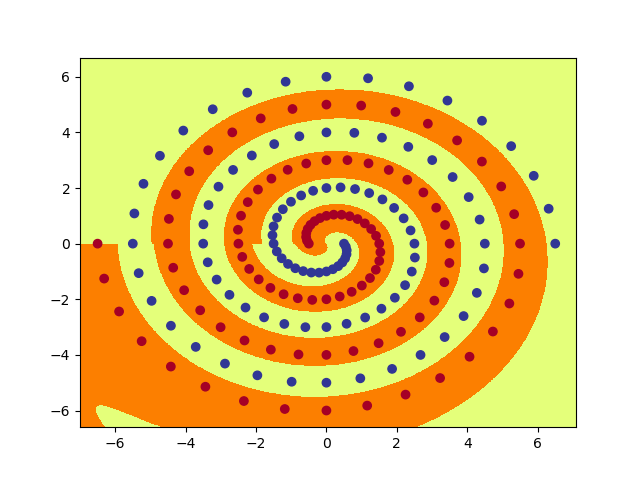


**Part 2: Twin Spirals Task**

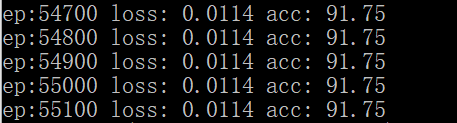
2.

Firstly, try hid = 10: accuracy can reach 100 with less than 2000 epochs.

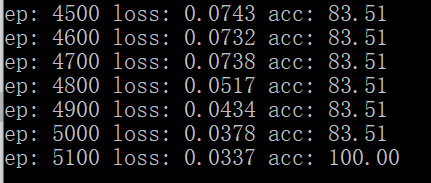


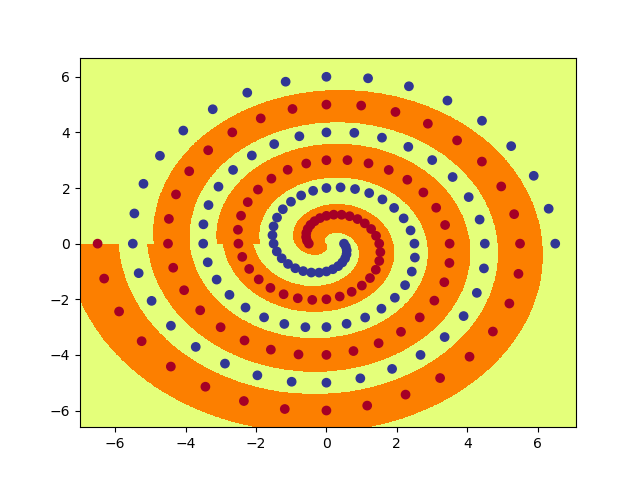


Then, try a small hid. When hid = 5: Accuracy cannot be higher than 91.75 even with more epochs. Stop training so there is no output picture. So, we can slowly decrease the value of hid from 8, and find the minimum value.

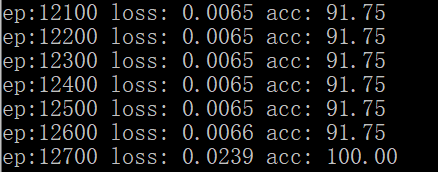


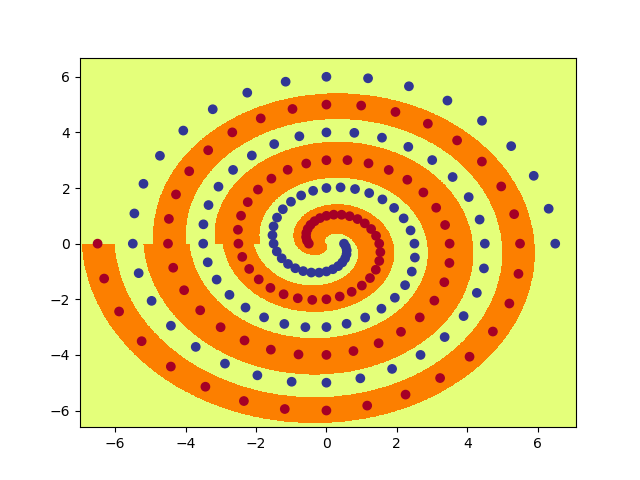
Hid = 8: Correctly classify all of the training data within 6000 epochs.

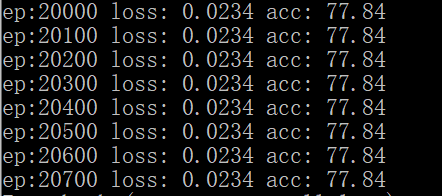




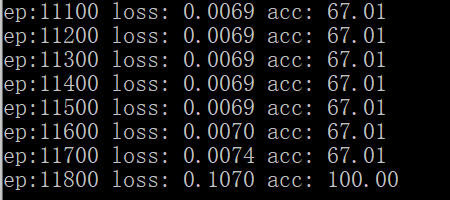
Hid = 7:

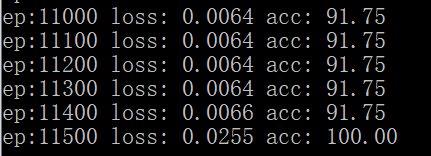


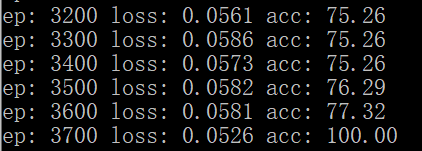


Hid = 6: Accuracy is less than 80 when epoch reaches 20000.  


It seems that **7** is the minimum number of hidden nodes required so that this PolarNet learns to correctly classify all of the training data within 20000 epochs, on almost all runs. To make this conclusion more robust, I run code for several more time, and all the results show that: when hid is 7, PolarNet can reach accuracy of 100 within 20000 epoches.

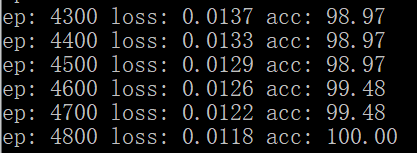




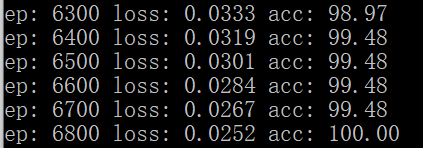


4.

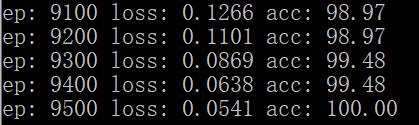
When hid = 15, init = 0.2, it only needs less than 5000 epochs, so we can try smaller value of hid or init.



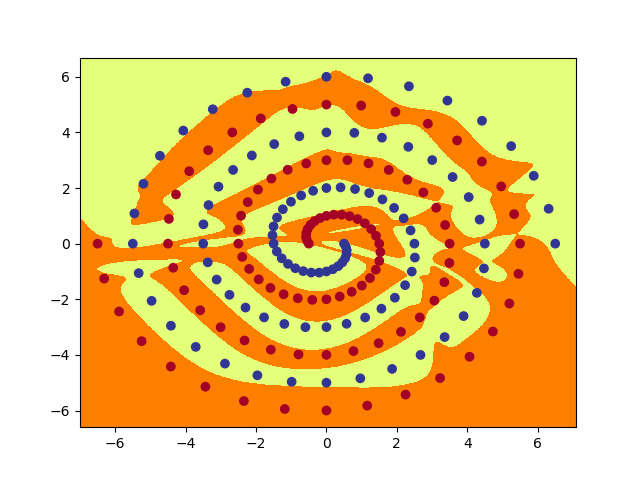
When hid = 12, init = 0.15, it needs around 7000 epochs to reach accuracy of 100, continue exploring smaller values.



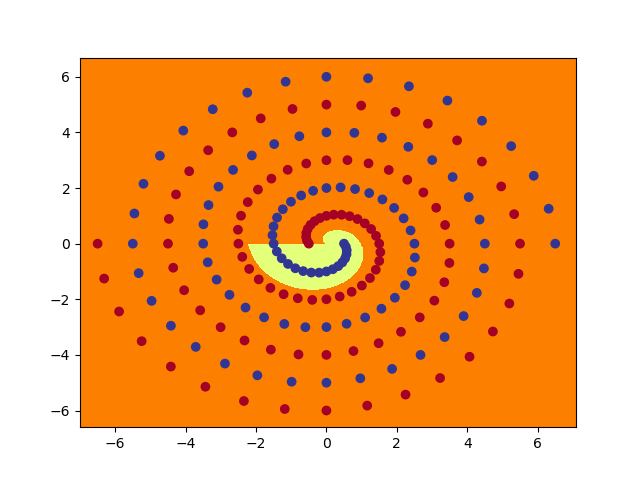
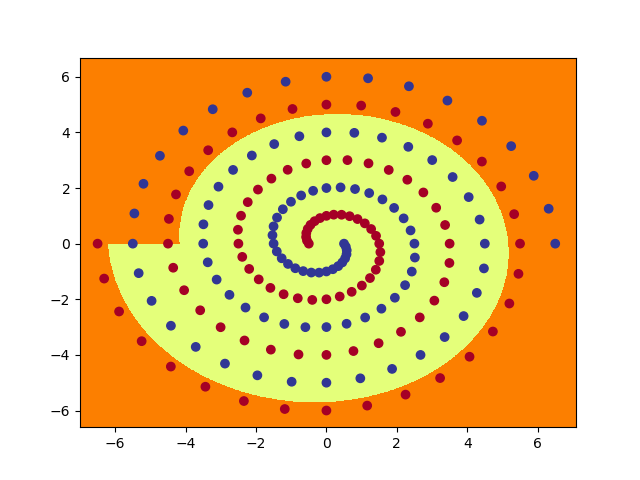
When hid = 10, init = 0.12, it needs around 10000 epochs. As we do not need find the minimum value for the number of hidden nodes, we can stop here with parameters hid = 10, init = 0.12, other parameters are using default values.

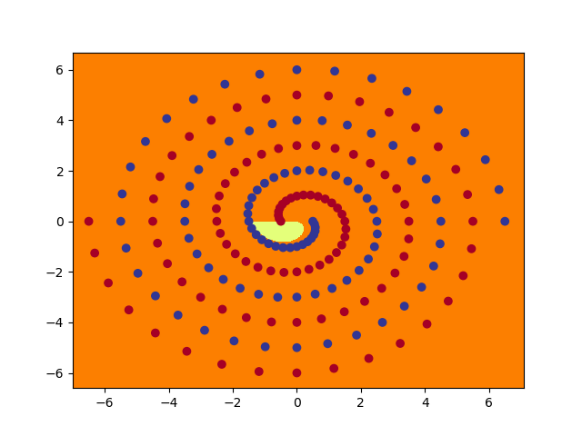
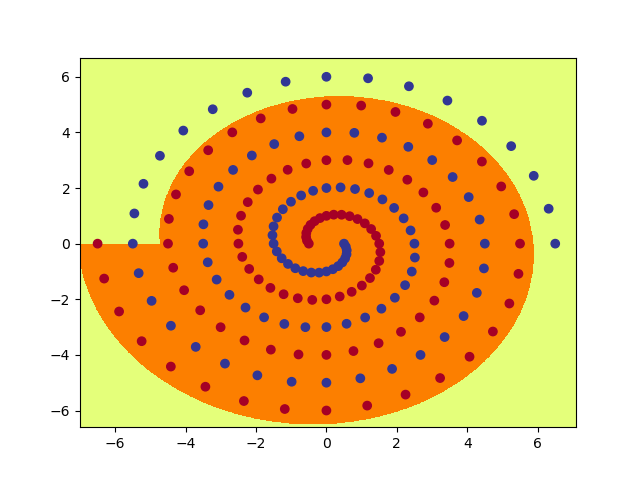


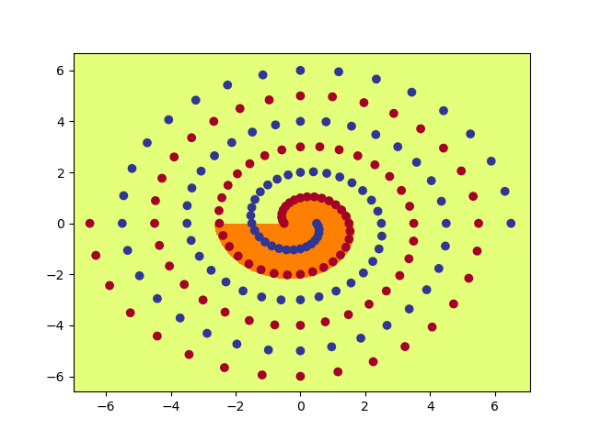
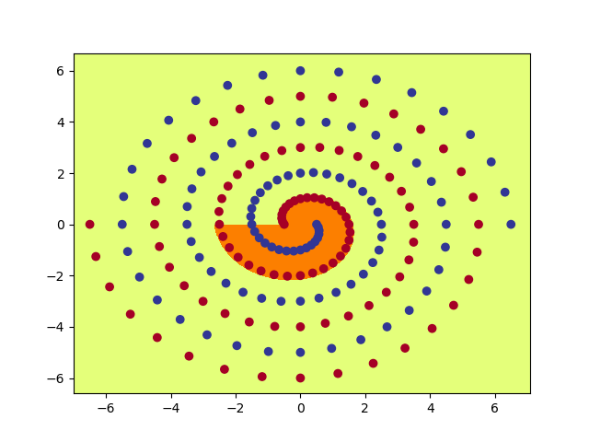
Raw\_out.png looks like this:

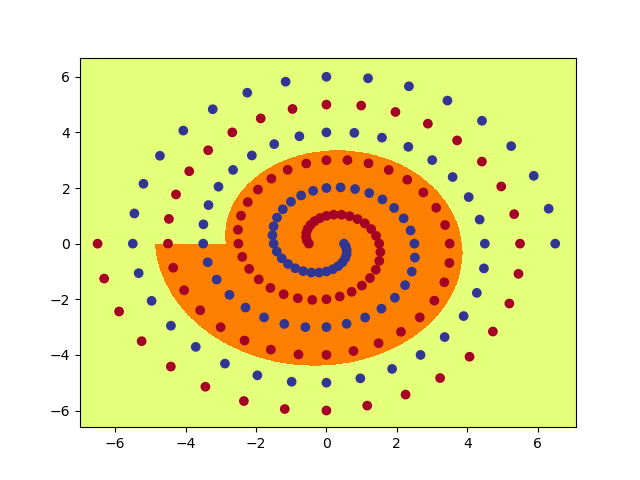
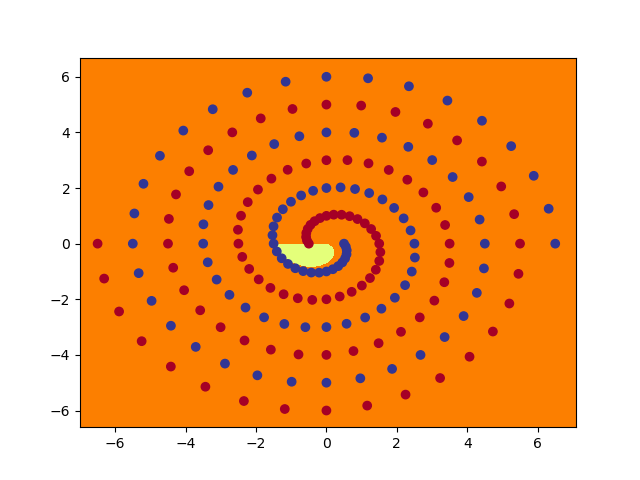


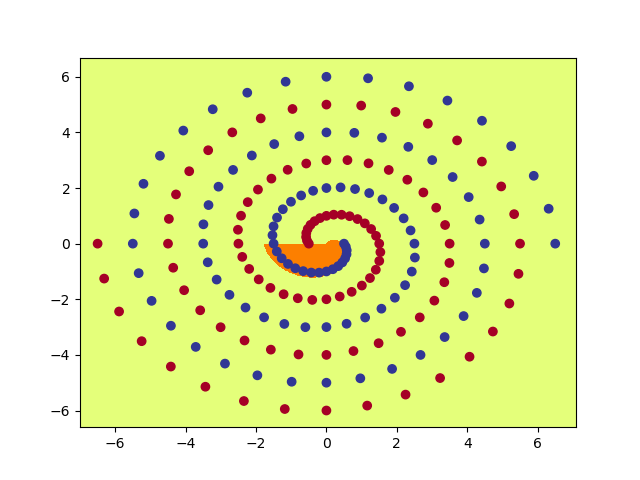
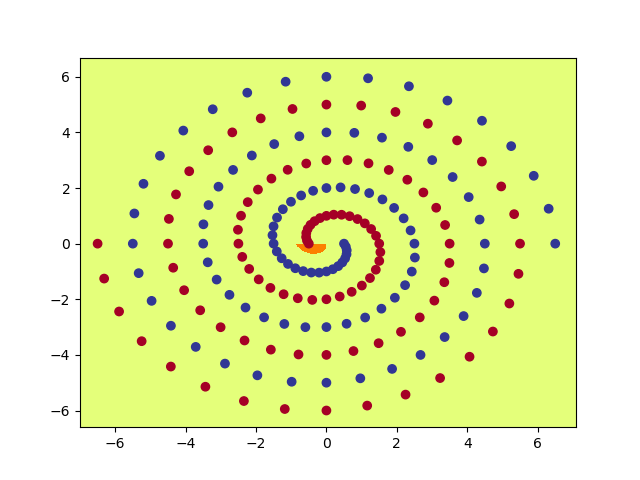
1. PolarNet (only one hidden layer): from node 0 to node 9

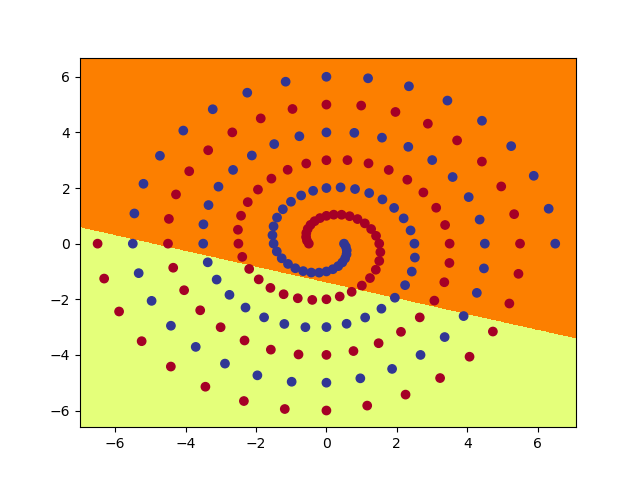
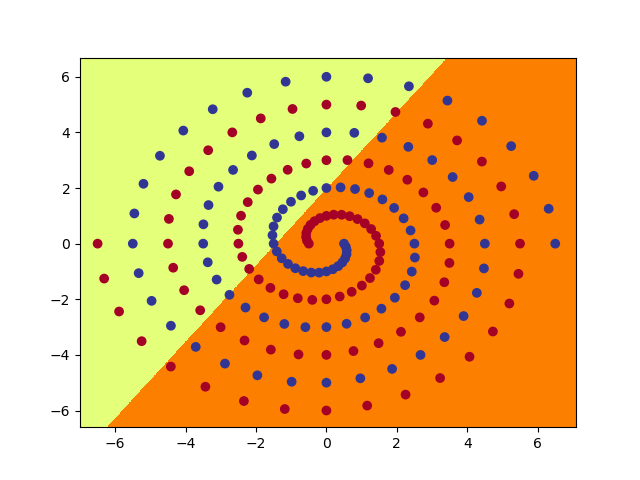


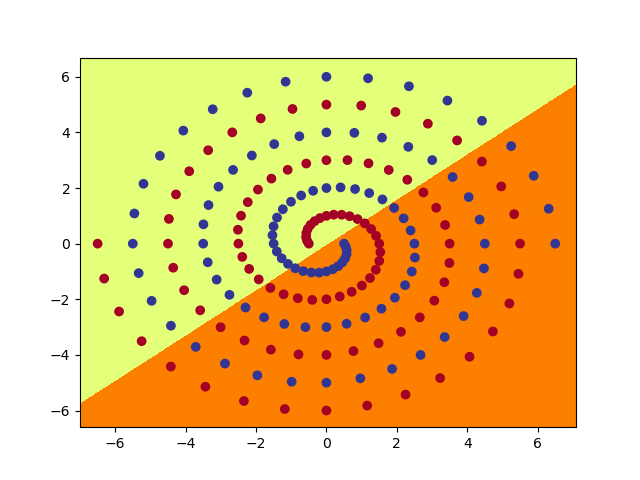
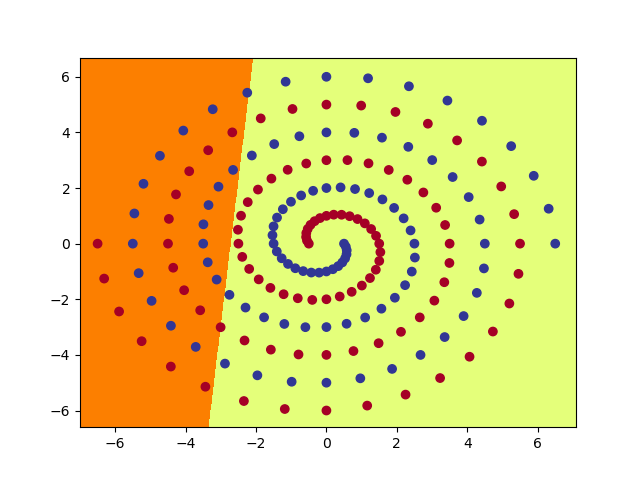


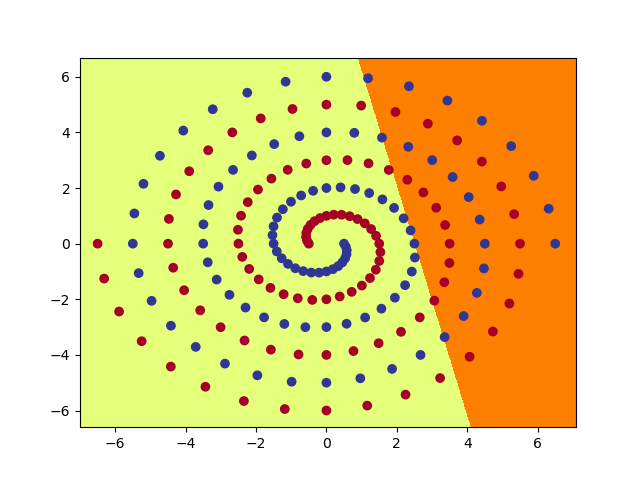
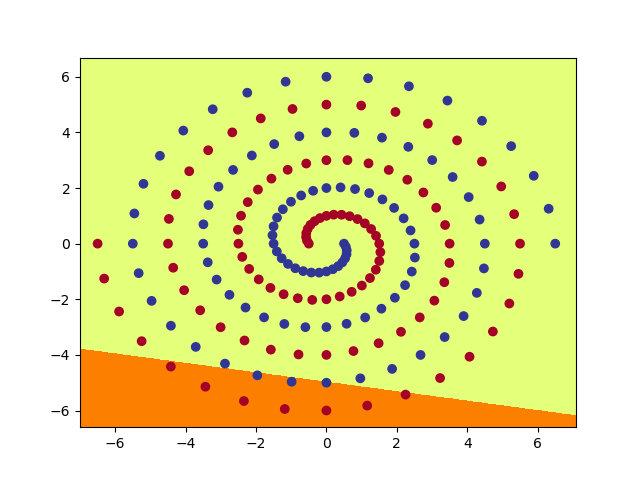


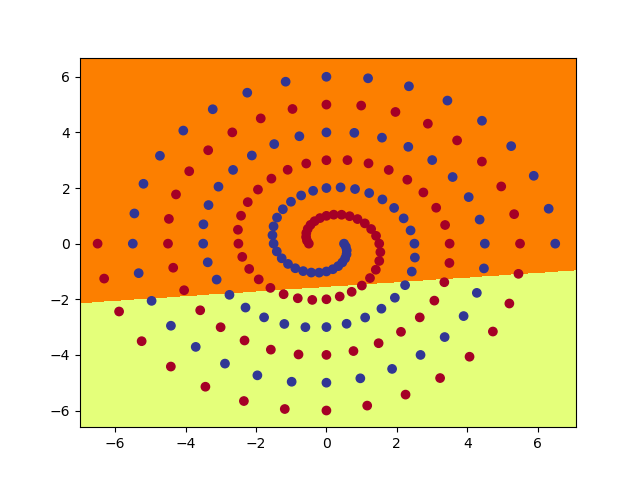
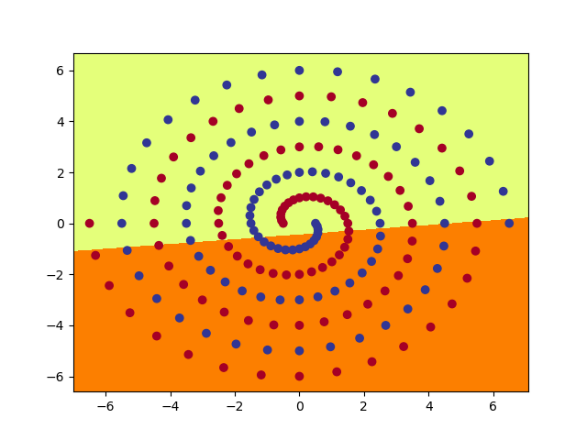


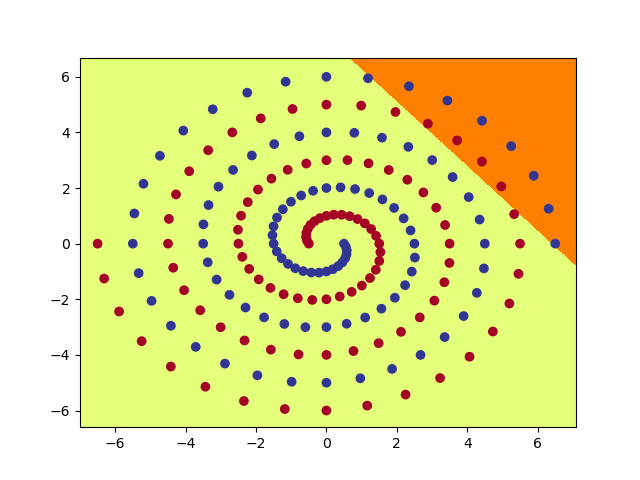
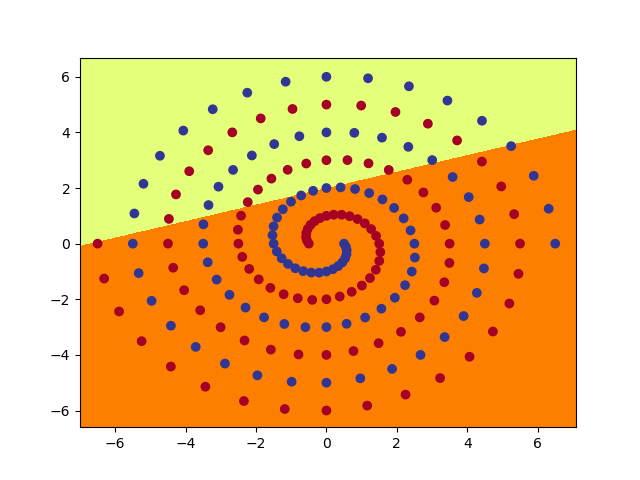


RawNet, hidden layer 1, node from 0 to 9:  
 

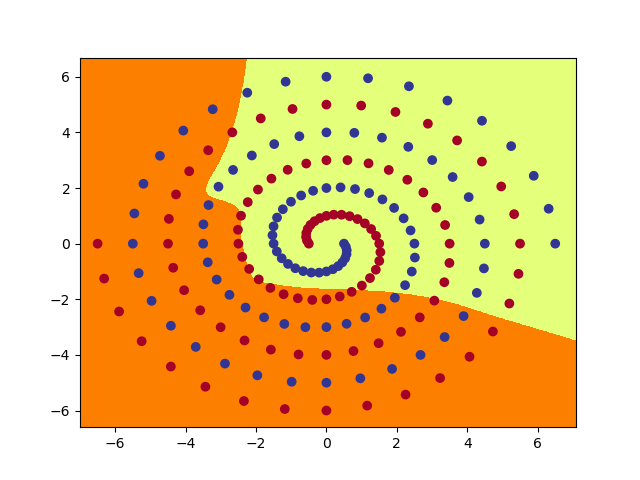
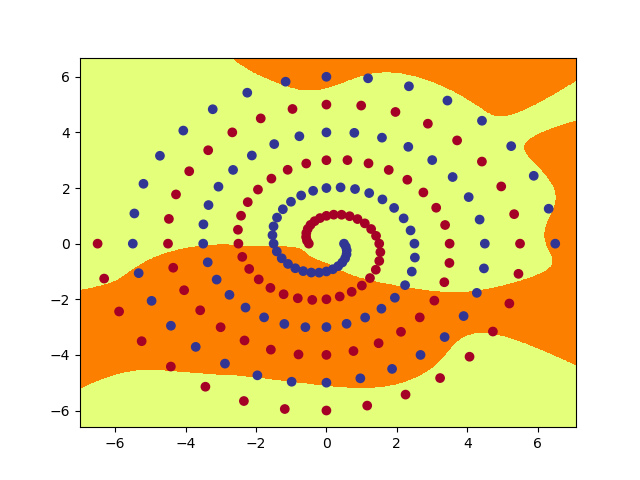
 

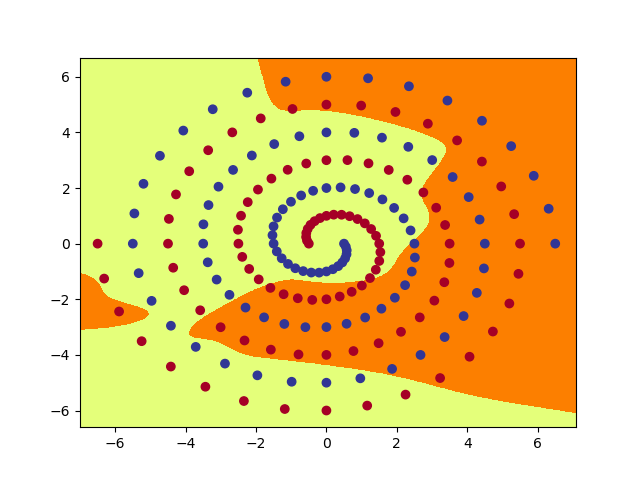
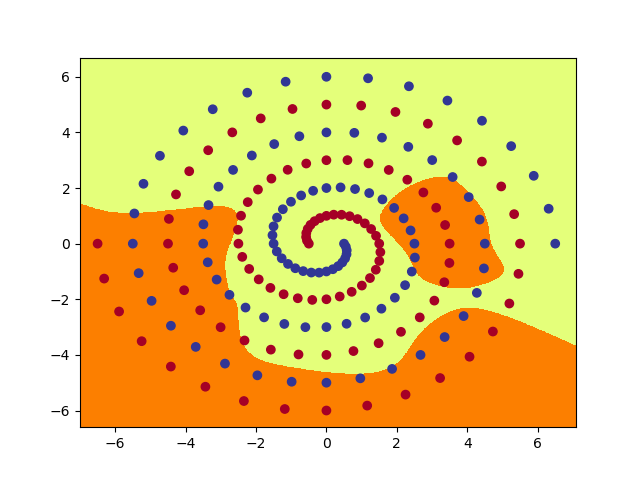
 

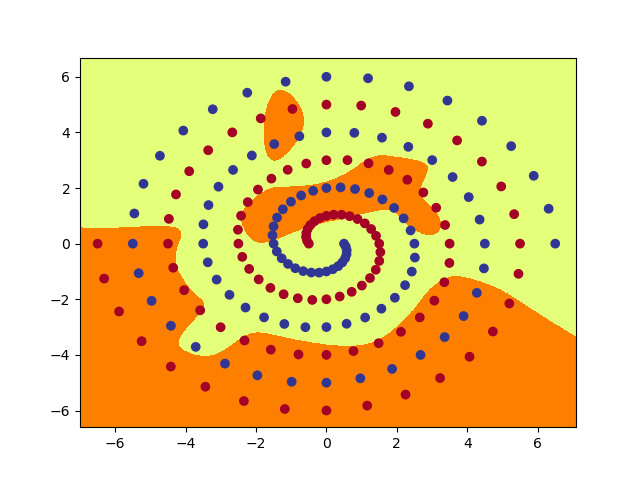
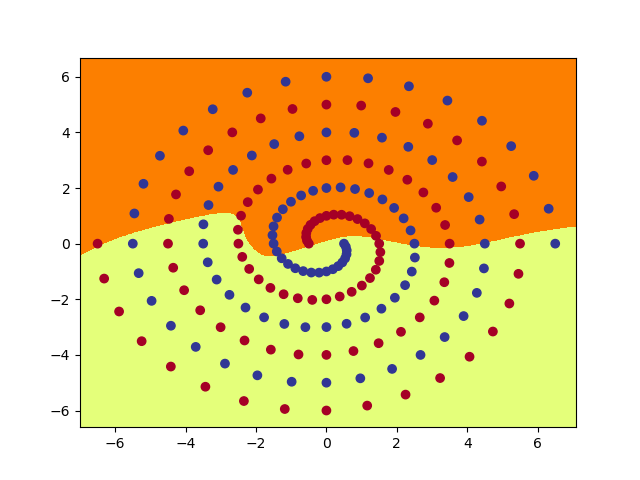
 

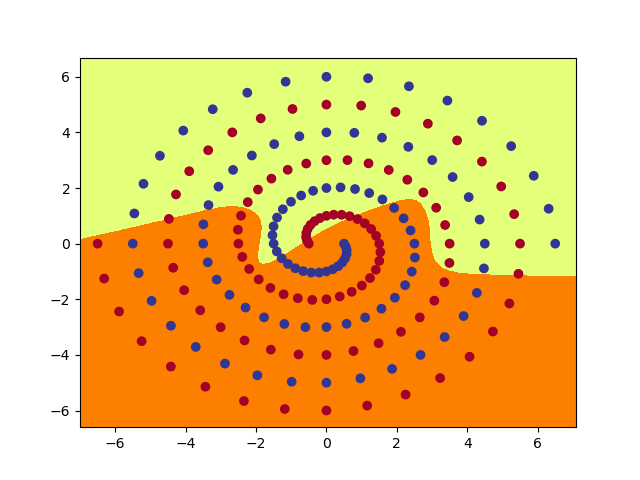
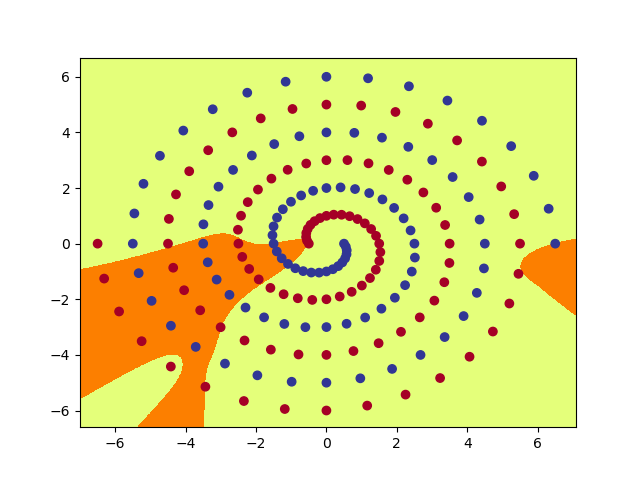
 

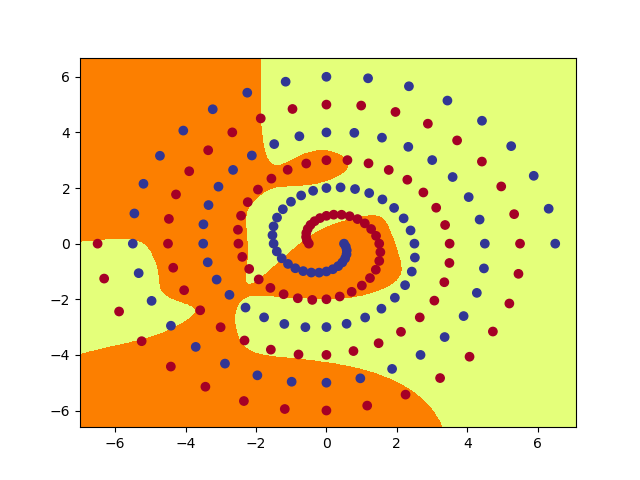
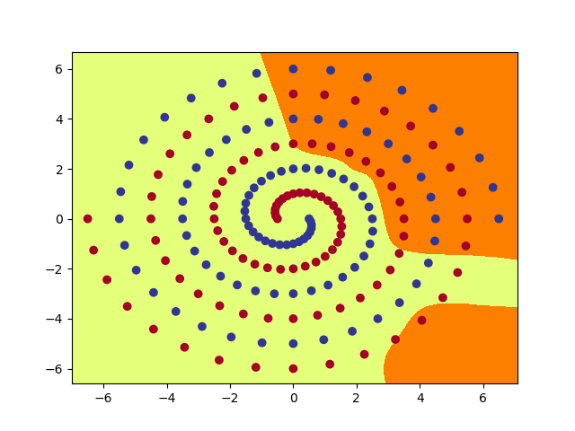
RawNet, hidden layer 2, node from 0 to 9:

1. Discuss what you have learned from this exercise, including the following points:
2. 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.

PolarNet only has one hidden layer, which learns linearly separable features. While RawNet has two fully connected hidden layers, in which the second hidden layer learns “convex” features. The output layer in RawNet kind of combines these to produce “concave” features, while the output layer in PolarNet is not so representative for non-linear features.

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

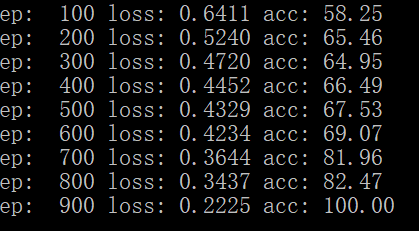
In my experiments, I find that the larger the initial weight size, the faster it converges. So, when plotting hidden nodes, I will choose a relatively large init value 0,12. However, if the initial weight size becomes too large for the model, it is likely to fail because the network is not stable anymore.

Theoretically, a small value of weight size is better for training, but it may be pretty slow. It is a trade-off, we need to choose the suitable weight size to balance speed and accuracy.

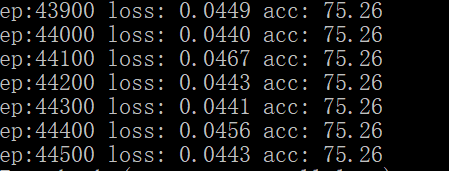
1. you may like to also experiment with other changes and comment on the result - for example, changing batch size from 97 to 194, using SGD instead of Adam, changing tanh to relu, adding a third hidden layer, etc.

To focus on batch\_size or other changes, I use PolarNet with hid = 10 to do the experiments.

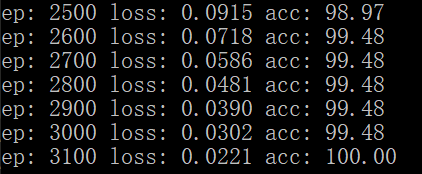
Changing batch\_size to 194, it trains much faster:



Changing tanh to relu, it becomes very slow and almost stagnates. Maybe the half-linear activation function relu is not suitable for PolarNet:

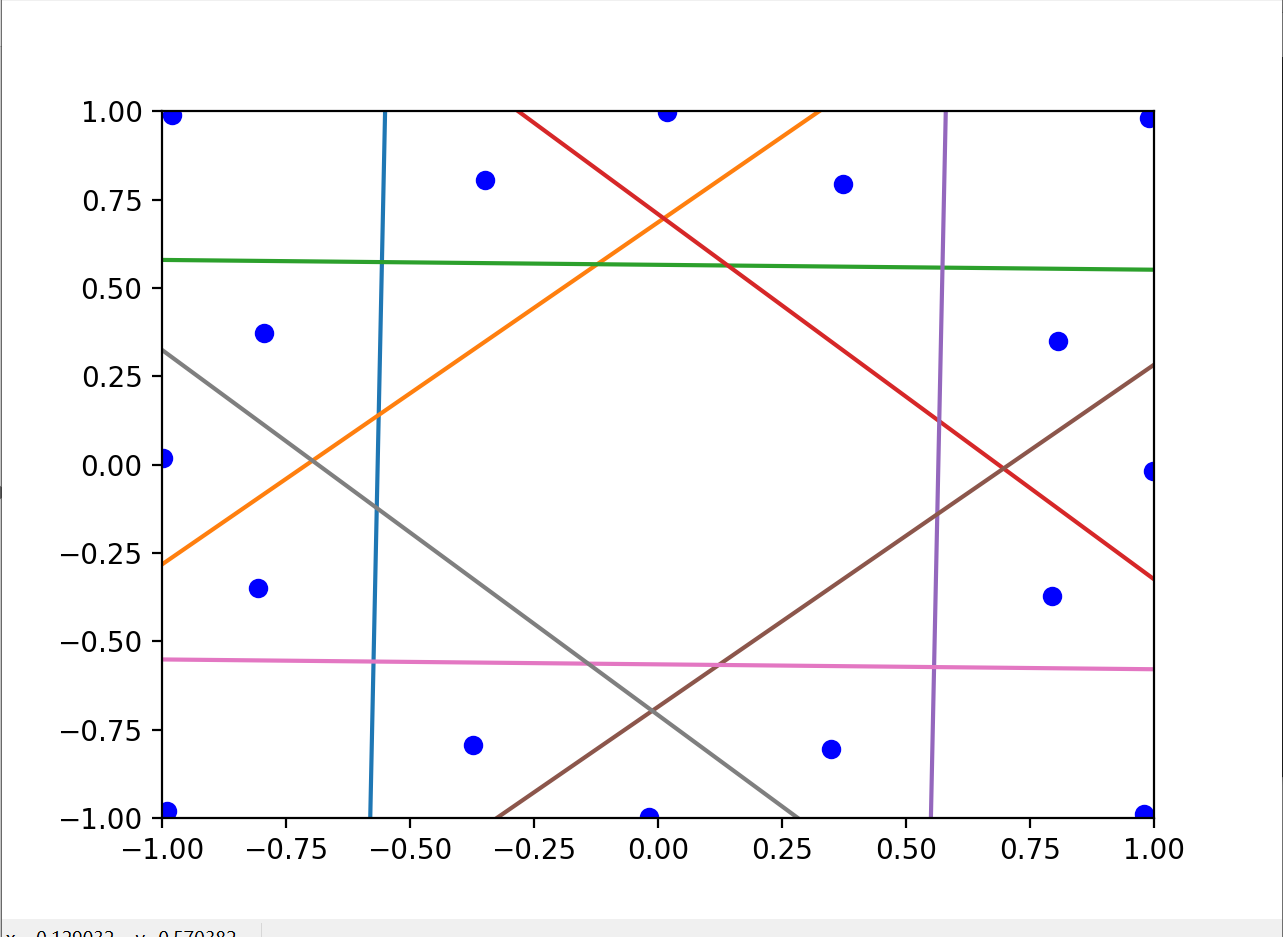


Add a third layer for RawNet, with parameter hid = 10 and init = 0.12, it becomes faster because the neural network is more complicated to represent more information, so can learn the model in a higher speed:

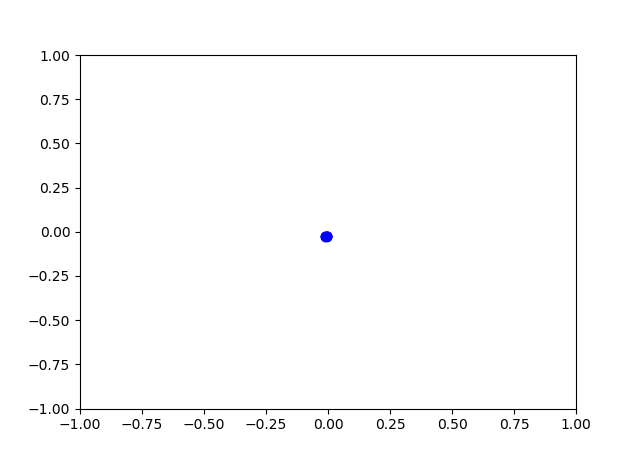
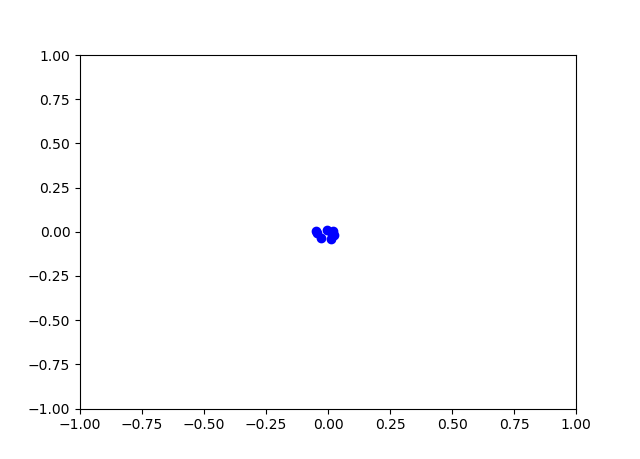


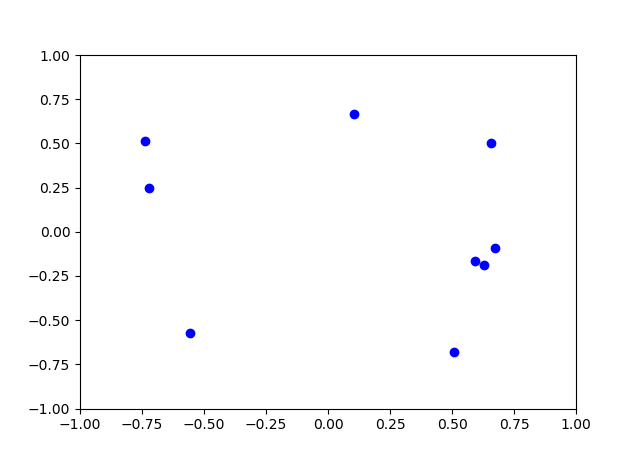
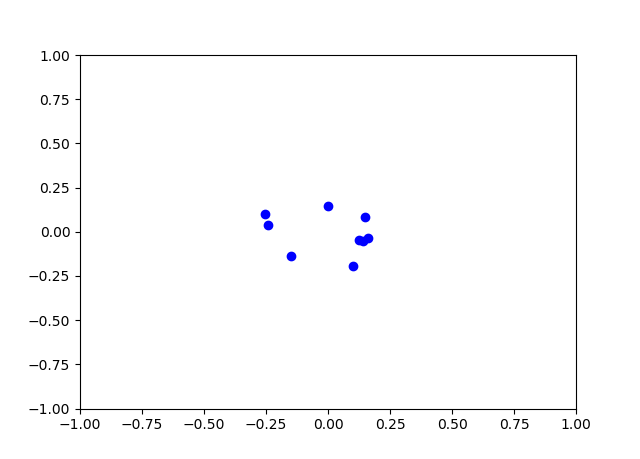
**Part 3: Hidden Unit Dynamics**

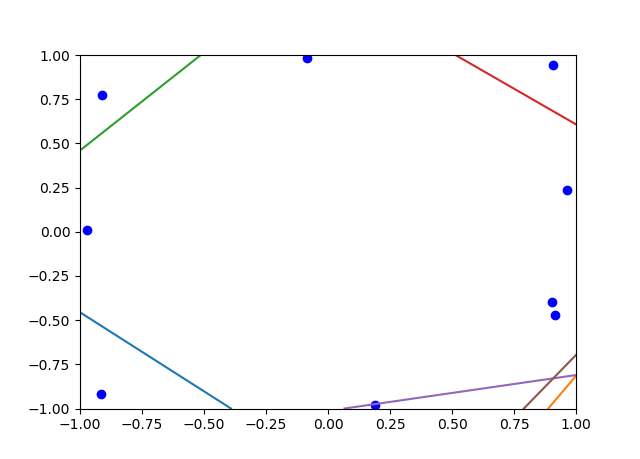
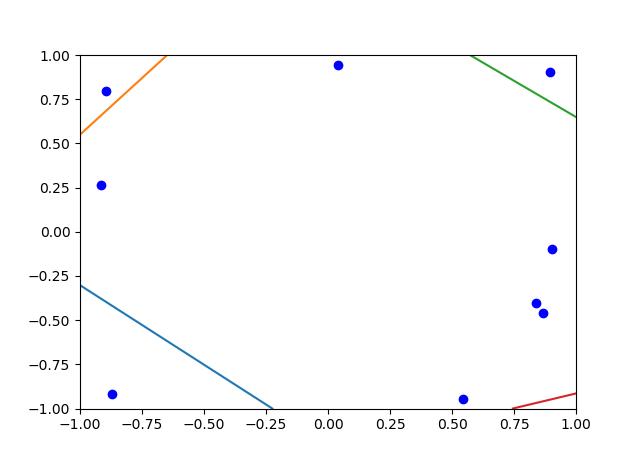
1. Final image is shown below:

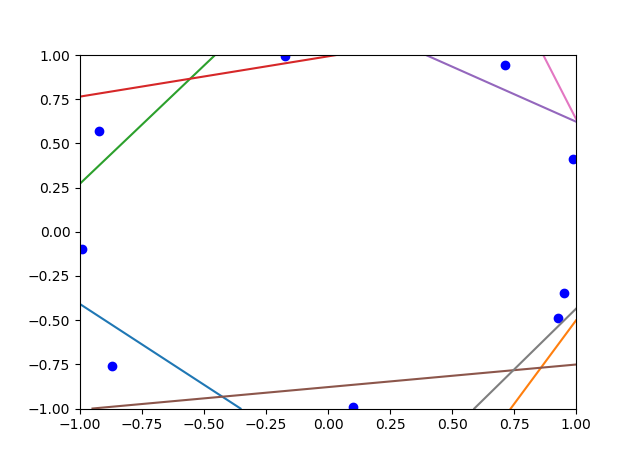
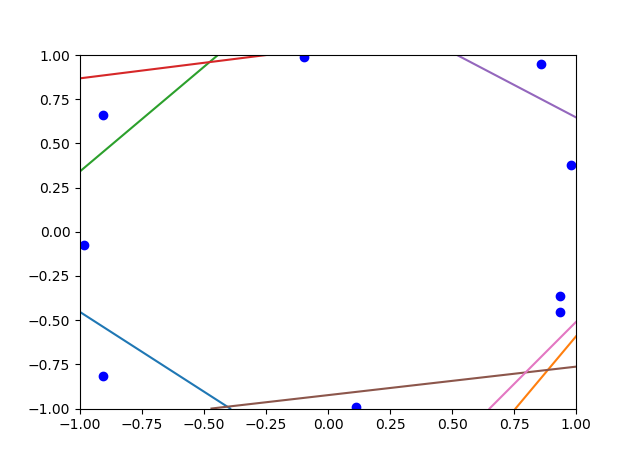


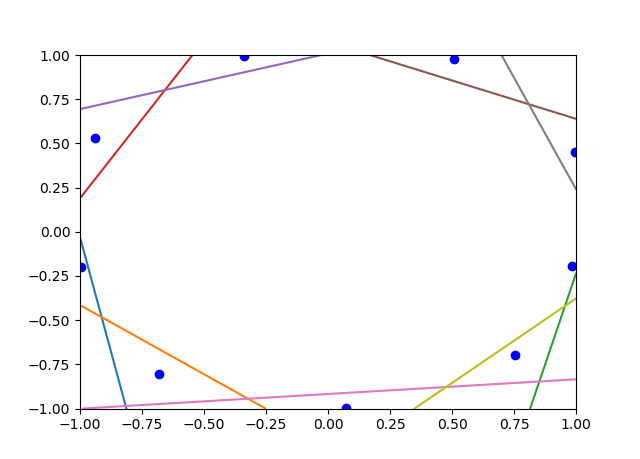
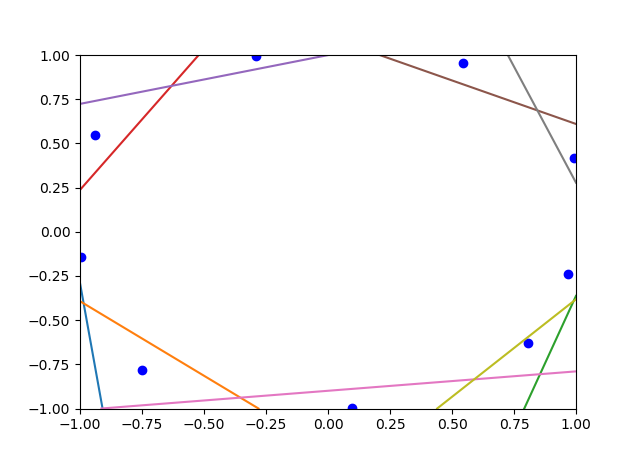
1. Pictures shown below, the first eleven pictures and the last one:

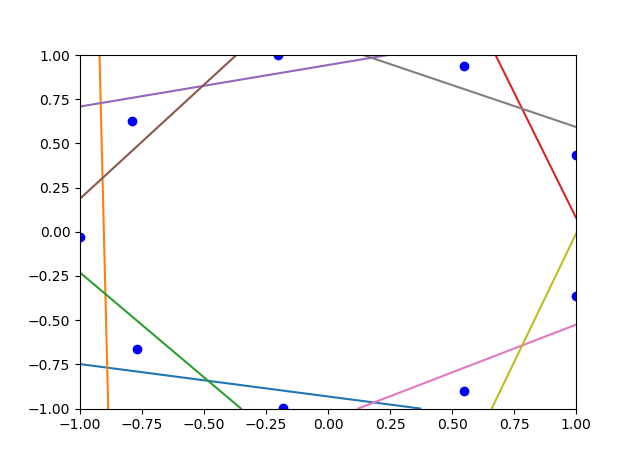
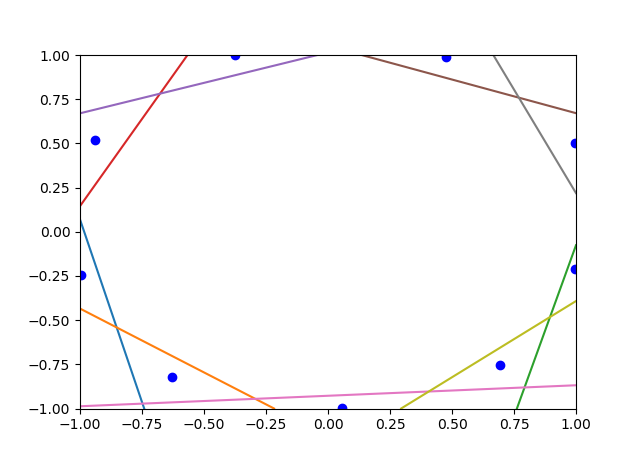
 











The hidden unit dots are gathering together at first, and then slowly spread out. The lines move in the opposite directions: from outside to inside. At last, it forms like a rough cycle.

1. 4. 是一个套路，就是想一个tensor组成一个爱心，我暂时还没想出一个完美的tensor。。如果没想出来的话我在发你一个相似的tensor拿点分吧。