ZZEN9444 Hexamester 2 2022

Assessment 2

Written Report

Sze Chan

Z5061978

Part 1 – Japanese Character Recognition

Step 1

Final accuracy:
Test set: Average loss: 0.9975, Accuracy: 6828/10000 (68%)

Confusion matrix:

		Predicted									
		o	**ki**	**su**	**tsu**	**na**	**ha**	**ma**	**ya**	**re**	**wo**
		0	1	2	3	4	5	6	7	8	9
Target	**0**	757	4	7	4	72	11	4	14	11	5
	1	5	655	61	42	48	33	23	28	41	60
	2	5	113	690	70	74	152	154	24	76	80
	3	10	14	35	757	19	18	13	15	49	2
	4	28	32	22	14	630	22	27	110	9	71
	5	59	18	16	41	14	690	18	13	35	20
	6	3	76	56	11	27	33	726	76	45	23
	7	74	10	32	20	25	7	19	554	7	38
	8	39	18	40	29	21	23	7	109	704	36
	9	20	60	41	12	70	11	9	57	23	665

Step 2

Final accuracy:
Test set: Average loss: 0.5112, Accuracy: 8455/10000 (85%)

		Predicted									
		o	**ki**	**su**	**tsu**	**na**	**ha**	**ma**	**ya**	**re**	**wo**
		0	1	2	3	4	5	6	7	8	9
Target	0	846	4	7	3	39	11	3	21	11	5
	1	5	820	13	10	30	16	15	12	31	17
	2	1	32	836	27	19	76	46	24	26	41
	3	6	6	44	920	9	7	10	4	52	7
	4	31	13	9	1	818	13	13	20	4	29
	5	35	11	16	14	6	822	5	10	6	5
	6	2	63	26	8	29	28	896	32	31	28
	7	42	7	14	1	19	1	6	825	3	10
	8	28	18	21	7	18	21	1	26	828	14
	9	4	26	14	9	13	5	5	26	8	844

Step 3

Final accuracy:
Test set: Average loss: 0.4916, Accuracy: 8498/10000 (85%)

		Predicted									
		o	ki	su	tsu	na	ha	ma	ya	re	wo
		0	1	2	3	4	5	6	7	8	9
Target	0	855	4	7	3	35	10	3	22	11	4
	1	5	825	13	9	30	13	13	14	30	17
	2	1	32	838	28	17	72	46	23	27	41
	3	6	5	44	923	8	8	10	4	51	7
	4	28	12	9	1	823	11	12	18	4	28
	5	29	10	16	13	5	831	5	9	6	5
	6	2	61	24	8	30	27	899	30	30	28
	7	42	5	14	1	19	1	7	829	3	11
	8	28	20	20	7	20	20	1	26	830	14
	9	4	26	15	7	13	7	4	25	8	845

Step 4

This first part was an excellent way see how we can apply various Neural Networks (NNs) in real life. Moreover, I was able to comprehend the ways various models absorb the same dataset in their unique ways. NetLin, NetFull and NetConv were had the same set of data but as we can see now, their predictions of letters varied in terms of precision.

- Relative accuracy:

  NetLin had an accuracy of only 68% - a lot of wrong predictions with significant probabilities compared with the latter two, which have fewer incorrect predictions that have small probabilities.

- Confusion matrix for each model:

  As an example, we observe in the first row for all models, the second largest number apart from cell 0,0 is 0,4. This means that across all models, 0(お) was incorrectly predicted as 4(な). Another observation is the similarities in the two characters with some certain features – like the dash in the top right and the horizontal strokes for example. One more example would be 6(ま) being incorrectly predicted as 1(き) across all models for the same reason – having the largest number (in cell 6,1) apart from cell 6,6 and having two horizontal strokes.
  Something to note is the accuracy isn't perfect due to the lower resolution of the handwritten characters (28x28) – images with higher resolution can help with better results.

- Experiment with other architectures and/or metaparameters:

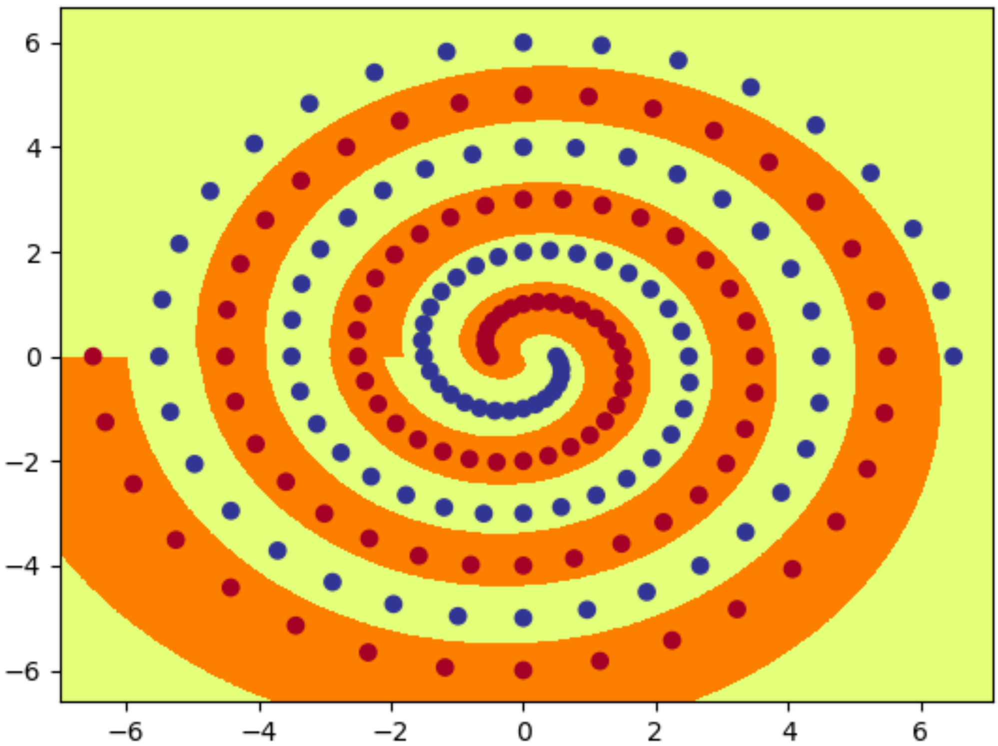
  I experimented around with the number of layers, max pooling and number of hidden nodes to check any differences in final accuracy of different choices. Increasing the filters did nothing much to accuracy, but accuracy was impacted with max pooling to get rid of worthless information in dataset. A 2-layer convolution neural network along with Max pooling improved the results.

In terms of the metaparameters:

- Epochs: Accuracy started to level off at Epoch 7.
- Momentum: I started off putting the default 0.5 as momentum but realised that setting it to 0.81 achieved the answer faster (getting to the local minimum).
- Learning Rate: As mentioned in the course, the learning rate has a big impact and I needed to be cautious of setting it too low or too high. I started experimenting around with 0.01 and gradually worked my way up to 0.1, and eventually noticed that a learning rate of 0.15 got 96% accuracy.

Part 2 – Twin Spirals

Step 2

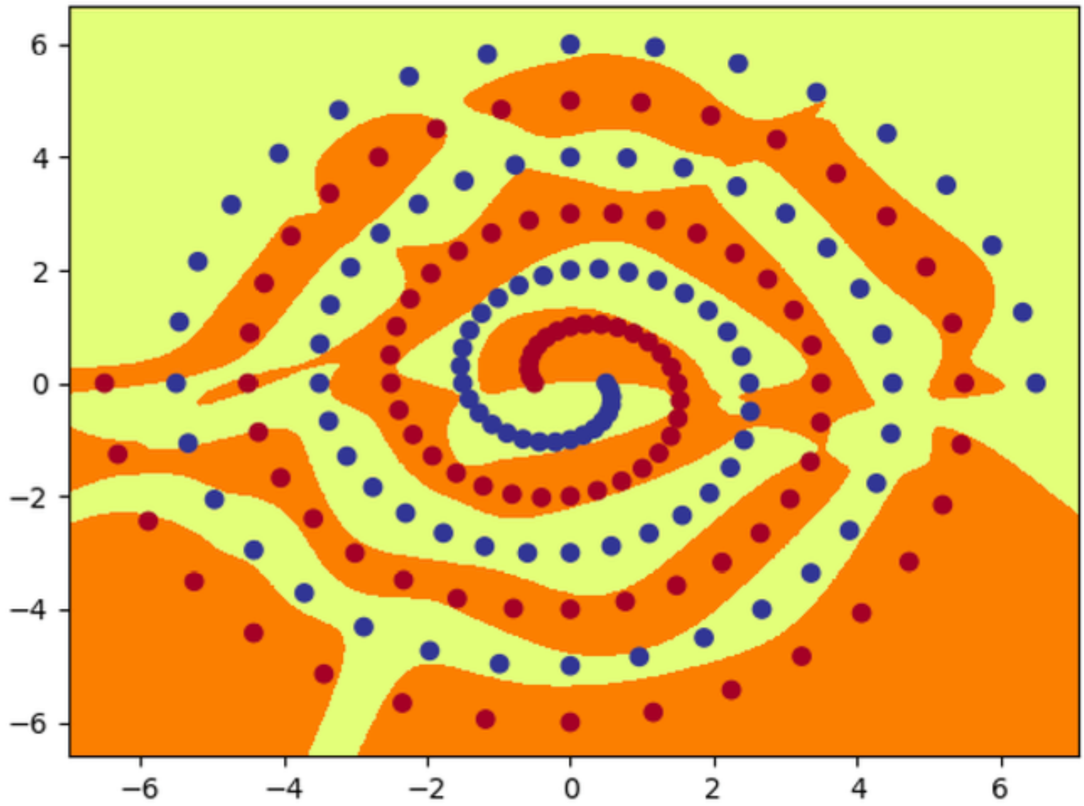


(polar_out.png)

By visual assessment, the minimum number of hidden notes is 7.

Step 4

I have set initial weights = 0.121, hidden nodes to be 7 with all other parameters set to default. And double checked so that RawNet learns to correctly classify all training data within 20000 epochs, on almost all runs:



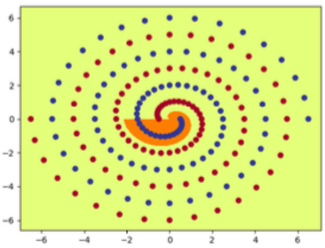
(raw_out.png)

Step 5

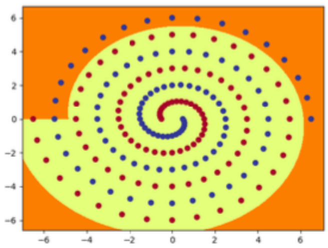
(Plots of all the hidden nodes in PolarNet, and all hidden nodes in both layers of RawNet):

PolarNet Images:

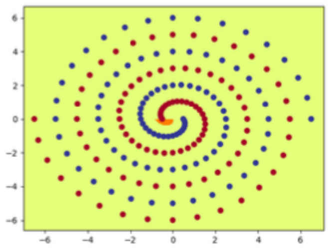
polar1_0.png



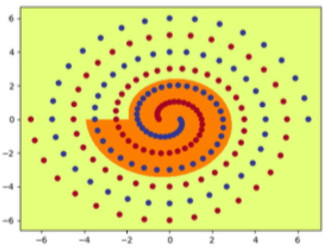
polar1_1.png



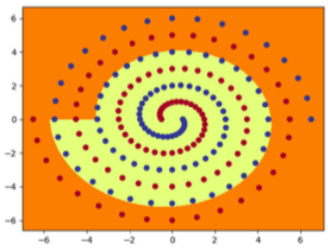
polar1_2.png



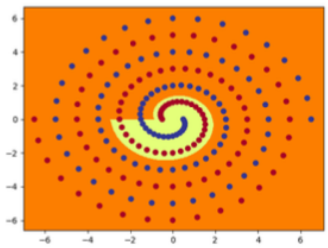
polar1_3.png



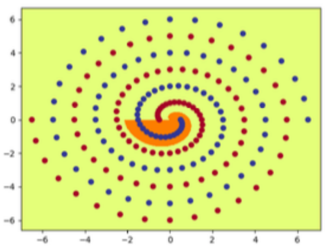
polar1_4.png



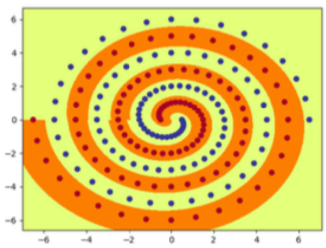
polar1_5.png



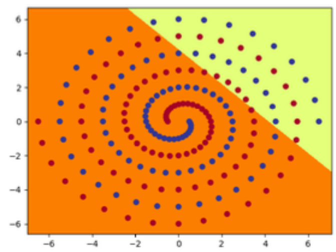
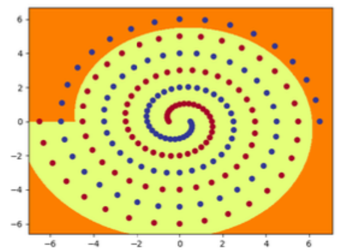
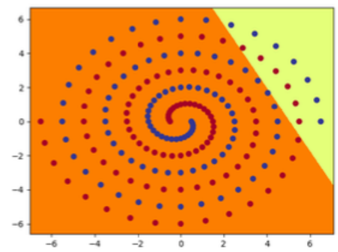
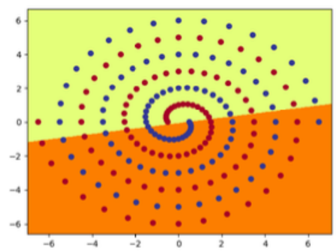
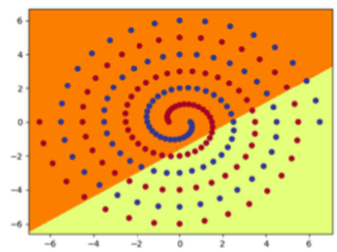
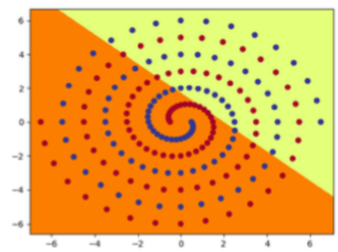
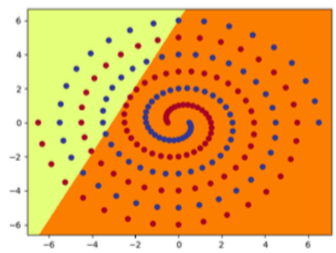
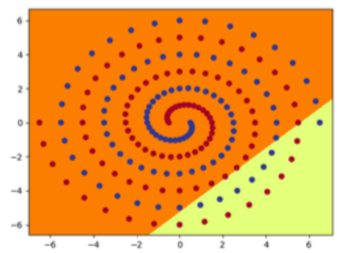
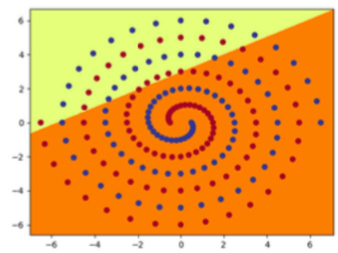
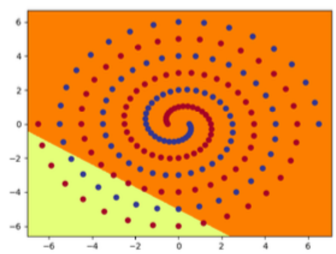
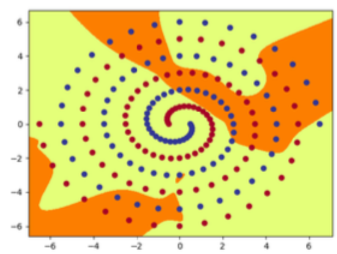
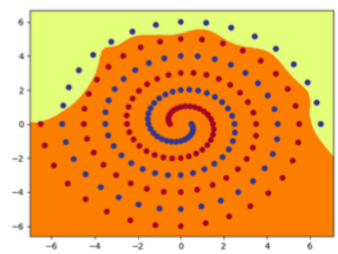
polar1_6.png

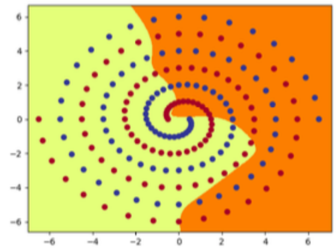
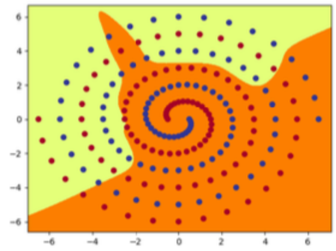
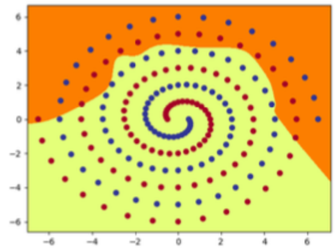


polar_out.png

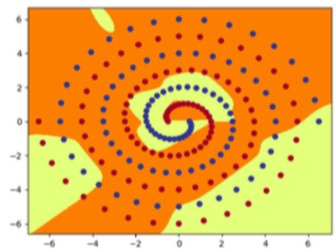
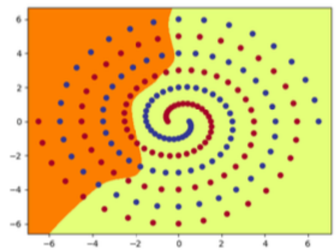
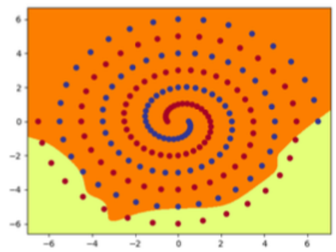


RawNet Images:

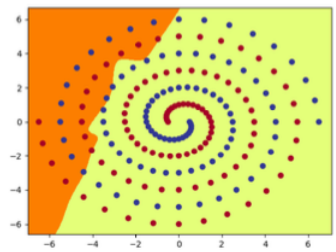
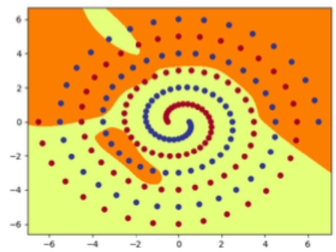
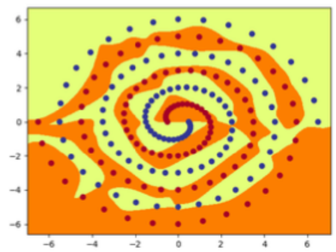
raw1_0.png	raw1_1.png	raw1_2.png
		
raw1_3.png	raw1_4.png	raw1_5.png
		
raw1_6.png	raw1_7.png	raw1_8.png
		
raw1_9.png	raw2_0.png	raw2_1.png
		
raw2_2.png	raw2_3.png	raw2_4.png

raw2_5.png | raw2_6.png | raw2_7.png

raw2_8.png | raw2_9.png | raw_out.png

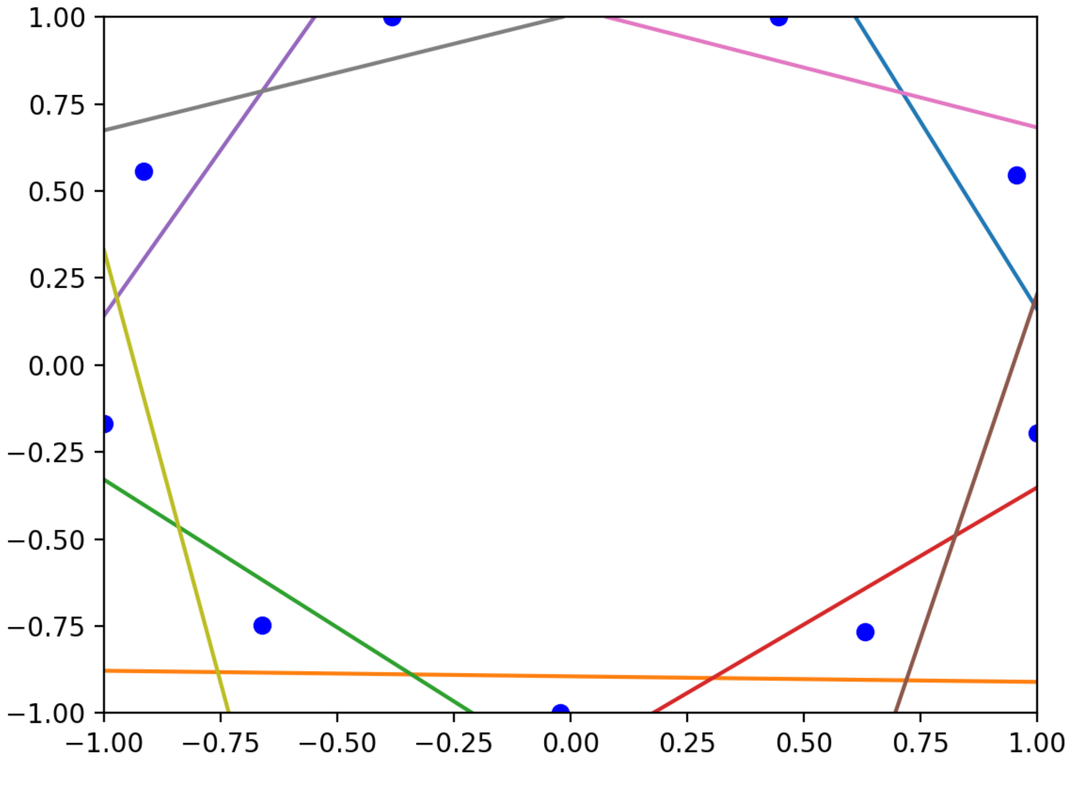
Step 6

- In PolarNet the input is converted to polar coordinates with one hidden layer, whereas RawNet has 2 hidden layers. Moreover, the first layer of PolarNet contains unique values which generates spiral designed layers, but RawNet produces linearly separated layers.
- PolarNet's input is different to RawNet and the output's shape understood by the system using the filters is compared to the flawless shape given PolarNet using polar coordinates.
- The network was trained with initial weights between 0.1 and 0.7. At first, the model trained faster (<6000 epochs), but won't train with 30000 epochs. The weight of 0.121 was found with the seed set to zero, and RawNet trains within 20000 epochs. If the weight is less than 0.121, the network never diverts from roughly 50% accuracy also not improving after 20000 epochs. But if the weight was set to more than 0.121 but less than 0.29, network learns faster, and if it goes beyond 0.3, the behaviour goes back to that of weight less than 0.121.
- Increasing the batch size from 94 to 197 had a good impact with better performance regarding time and number of epochs needed to train the program. Using SGD instead of Adam meant it took a long time to converge with batch size =197, hidden nodes = 10 and lr=0.01, taking up to 20000 epochs to converge. Thus, it is important

to set other metaparameters for each optimiser. Adding more layers has no real impact on output/learning task. Finally, ReLU() ignores some outputs and is asymmetric – which means it is not okay to use ReLU in this spiral problem, clearly.

Part 3 – Hidden Unit Dynamics

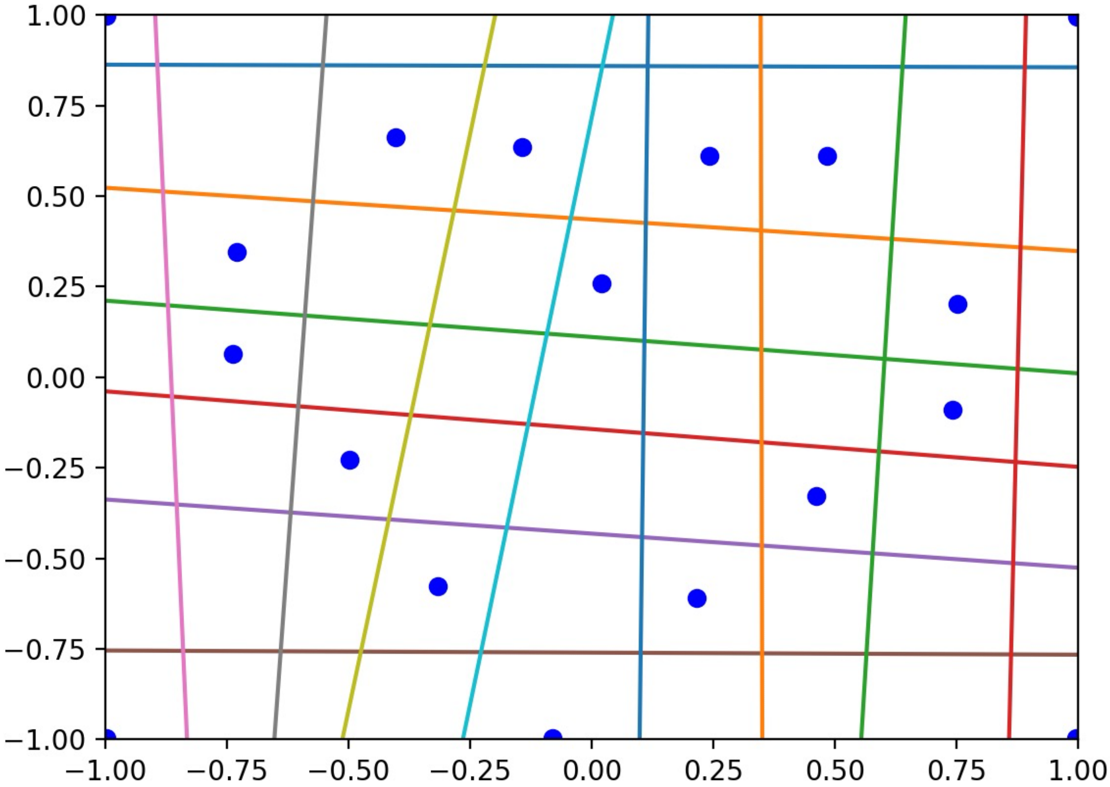
Step 1



Step 2

The hidden unit activations become progressively packed closer together towards more of a heart formation as the training progresses and the output boundaries increase in numbers and become vertically straight.

Step 3



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