2020

Project 1

JAPANESE CHARACTERS AND INTERTWINED SPIRALS

ANANT KRISHNA MAHALE | Z5277610

Part 1

Question 1.

Final accuracy:
Test set: Average loss: 1.0092, Accuracy: 6963/10000 (70%)

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**	768	7	8	5	58	8	5	16	11	8
	1	6	671	61	35	52	28	20	29	39	51
	2	7	106	689	56	75	122	148	28	93	85
	3	13	17	25	761	21	17	10	13	43	3
	4	30	27	26	15	622	19	25	84	7	51
	5	64	22	22	59	19	724	24	17	33	33
	6	2	57	46	12	33	30	723	53	44	18
	7	62	14	37	19	38	9	20	624	6	32
	8	31	26	46	26	22	33	9	87	702	40
	9	17	53	40	12	60	10	16	49	22	679

Question 2.

Final Accuracy:
Test set: Average loss: 0.4945, Accuracy: 8490/10000 (85%)

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**	864	4	8	4	43	9	3	20	10	2
	1	3	809	14	8	21	10	10	18	28	18
	2	2	41	839	35	19	79	49	19	31	48
	3	4	4	38	920	5	11	10	3	53	4
	4	32	16	11	3	828	14	13	18	3	30
	5	30	9	17	13	6	831	4	11	9	4
	6	2	61	30	4	28	22	897	30	25	19
	7	36	6	12	1	16	2	9	831	3	19
	8	22	19	19	7	20	16	2	24	830	15
	9	5	31	12	5	14	6	3	26	8	841

Question 3.

Final Accuracy:

Test set: Average loss: 0.2489, Accuracy: 9382/10000 (94%)

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**	961	1	12	1	28	5	4	5	13	8
	1	4	938	14	1	16	14	4	6	19	8
	2	1	3	867	9	1	27	5	4	7	5
	3	0	0	35	966	2	6	2	1	8	1
	4	20	5	8	1	908	5	3	2	8	5
	5	2	0	17	6	4	923	4	1	5	0
	6	0	30	24	8	16	13	973	9	4	2
	7	8	4	8	2	8	1	2	963	7	6
	8	1	3	7	2	5	2	2	2	922	4
	9	3	16	8	4	12	4	1	7	7	961

Question 4.

This exercise helped me to understand the application of different Neural Networks in real world examples. It also helped to understand how different models learn the same dataset differently. All the three models (NetLin, NetFull, NetConv) are trained on the same dataset but they predict the letters with different accuracy.

Summary	
Model	**Accuracy**
NetLin	70%
NetFull	85%
NetConv	94%

Summary of Confusion Matrix of all the models.

Predicted

NetLin

Target	**o**	**ki**	**su**	**tsu**	**na**	**ha**	**ma**	**ya**	**re**	**wo**
	0	1	2	3	4	5	6	7	8	9
0	768	7	8	5	**58**	8	5	16	11	8
1	6	671	**61**	35	**52**	28	20	29	39	**51**
2	7	106	689	**56**	**75**	122	148	28	**93**	**85**
3	13	17	25	761	21	17	10	13	43	3
4	30	27	26	15	622	19	25	**84**	7	**51**
5	**64**	22	22	**59**	19	724	24	17	33	33
6	2	**57**	46	12	33	30	723	53	44	18
7	**62**	14	37	19	38	9	20	624	6	32
8	31	26	46	26	22	33	9	**87**	702	40
9	17	**53**	40	12	**60**	10	16	49	22	679

NetFull

Target	**o**	**ki**	**su**	**tsu**	**na**	**ha**	**ma**	**ya**	**re**	**wo**
	0	1	2	3	4	5	6	7	8	9
0	864	4	8	4	43	9	3	20	10	2
1	3	809	14	8	21	10	10	18	28	18
2	2	41	839	35	19	**79**	**49**	19	31	48
3	4	4	38	920	5	11	10	3	**53**	4
4	32	16	11	3	828	14	13	18	3	30
5	30	9	17	13	6	831	4	11	9	4
6	2	**61**	30	4	28	22	897	30	25	19
7	36	6	12	1	16	2	9	831	3	19
8	22	19	19	7	20	16	2	24	830	15
9	5	31	12	5	14	6	3	26	8	841

NetConv

Target	**o**	**ki**	**su**	**tsu**	**na**	**ha**	**ma**	**ya**	**re**	**wo**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	961	1	12	1	28	5	4	5	13	8
1	4	938	14	1	16	14	4	6	19	8
2	1	3	867	9	1	27	5	4	7	5
3	0	0	35	966	2	6	2	1	8	1
4	20	5	8	1	908	5	3	2	8	5
5	2	0	17	6	4	923	4	1	5	0
6	0	30	24	8	16	13	973	9	4	2
7	8	4	8	2	8	1	2	963	7	6
8	1	3	7	2	5	2	2	2	922	4
9	3	16	8	4	12	4	1	7	7	961

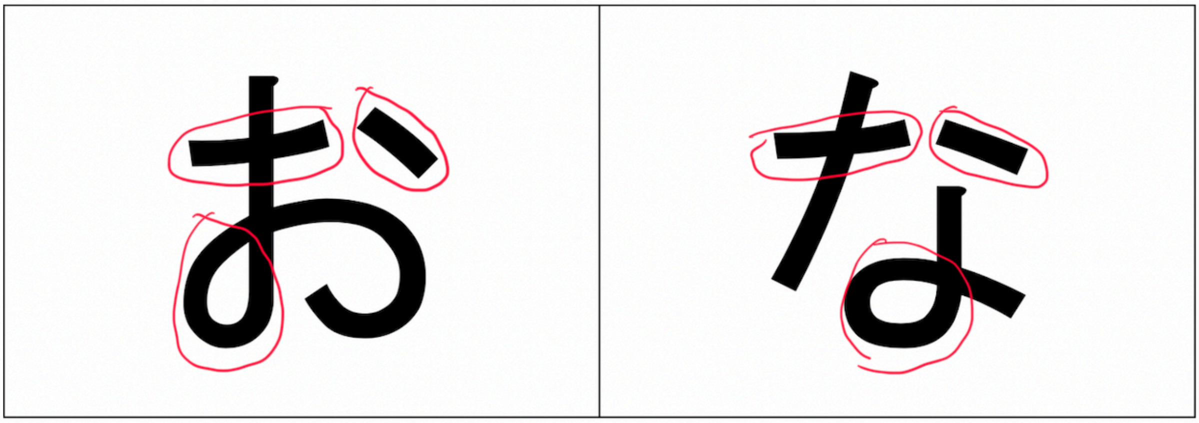
I have highlighted wrong predictions with **Red** colour for (probability >50) and with yellow colour for (probability 20<= X <50)

We can see that NetLin has many wrong predictions with high probabilities compared to Netfull and NetConv. On the other hand, NetConv has few wrong predictions that too, with small probabilities.

Here is the list of letters misclassified in all the three models.

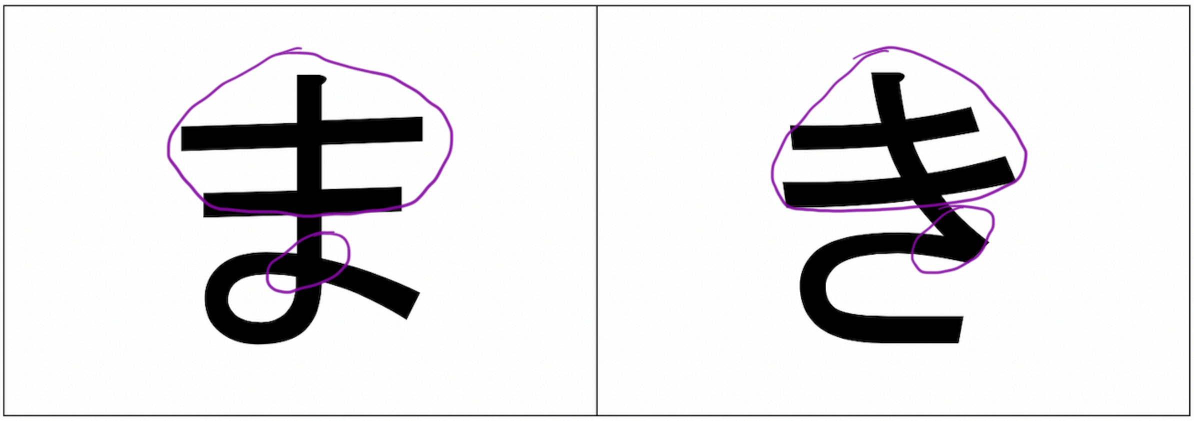
Target	Predicted
0 お	4 な
2 す	5 は, 6 ま
3 つ	2 す
4 な	0 お
5 は	0 お
6 ま	1 き
7 や	1き
8 れ	7 や
9 を	1 き,2 す,7 や

We can see that for the character 0(お) is predicted as 4(な) in all the models.



From the above image we can see that there are certain features which are common in both the characters which is the reason why they can be misclassified by the models.

Another example is, 6(ま) is represented as 1(き)



Again, the reason remains same.

However, there are characters 9(を) and 7(や) which do not look alike but they have been predicted with high probability in NetLin(49) and NetFull(26) models but not in NetConv model. This explains the advantages of CNNs over the other. It not only filters the right part of the images but also makes sure all the parts of the images are in place before predicting. Looking at the matrix, we can see that 9 is predicted as 7 in NetConv as well but, probability with which it has been predicted is very low (7) and can be ignored.

There is one more thing about the accuracy about these models. We can’t completely blame the machine for wrong prediction, we will have to account the fact that these are images of human handwriting and scanned ones with 28x28 resolution. Probably images with higher resolution may yield better results.

Convolution Neural Network can be constructed with different architecture. Considering this fact, I made changes to the number of layers, max pooling, number of hidden nodes. Though, the structure may look very different, the result was relatively same. I always thought that number of layers in CNN are proportional to accuracy. It proved me wrong. To my surprise, the accuracy did not improve with increase in number of filters. The accuracy improved with usage of max pooling as it eliminates the useless information in the data. Among all the experiments, I found the 2-layer CNN with Max pooling yielded better results.

On the other hand, when I add on more linear layer to the NetFull, it improved the accuracy of the model by 3%.

Influence of Metaparameters:

1. Number of Epochs (--epochs).
   I observed that after certain number of Epochs, the accuracy remains constant. For the given dataset, the efficiency mostly constant after the Epoch 8.

2. Momentum (--mom)
   It appeared that increasing the momentum from default 0.5 to 0.79 helped to achieve the answer faster as momentum aid to get the local minimum.

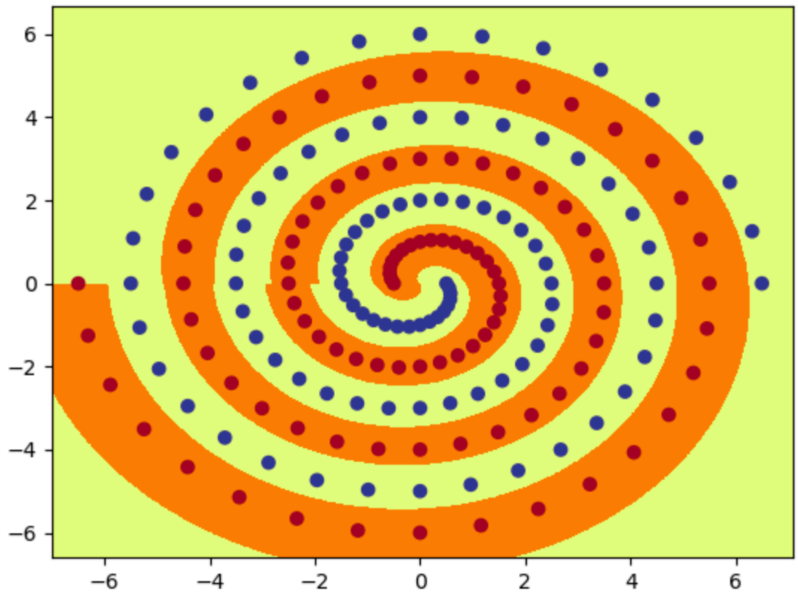
3. Learning Rate (--lr)
   Learning rate has major impact among all other meta parameters. Setting the value too low or too high always impacted the network.
   For convolution network, the lr = 0.15 yielded 96% accuracy.

Part 2

Question 2.

Minimum number of hidden nodes = 7.

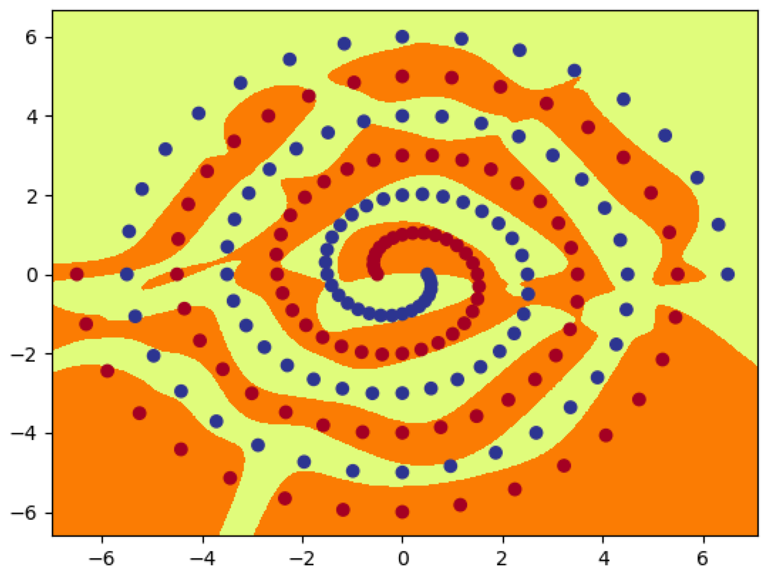
polar_out.png



Question 4.

Initial weights = 0.121.

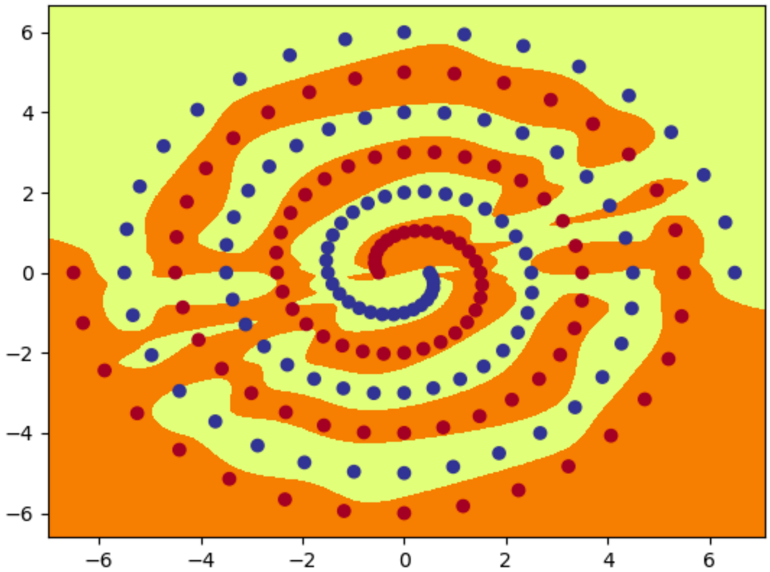
raw_out.png



Question 6.

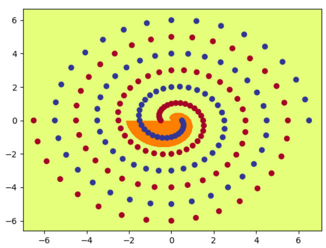
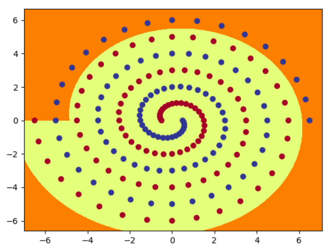
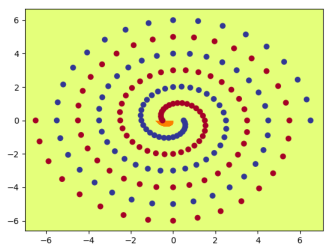
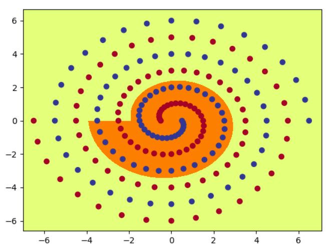
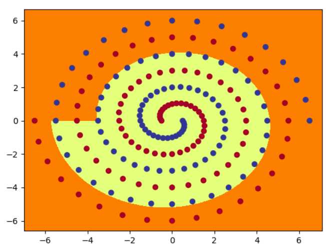
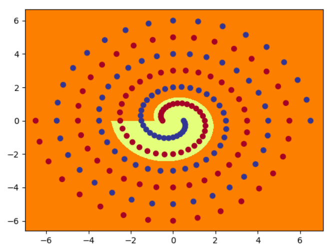
Meta-parameters -> **hid= 7 | init = 0.121 and all other parameters default.**

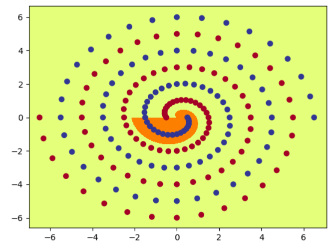
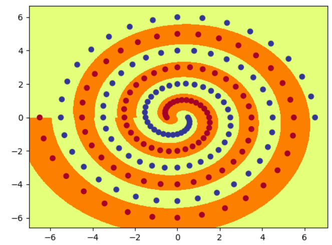
short_out.png



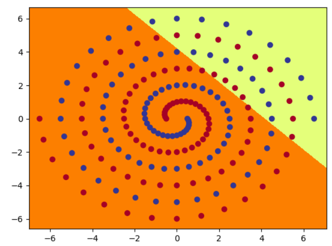
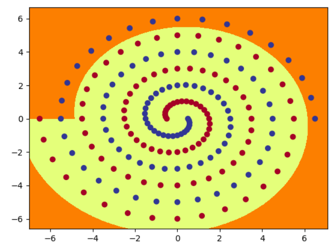
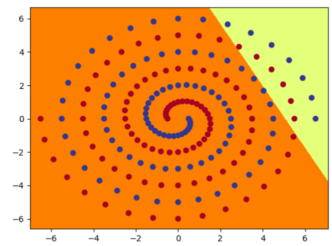
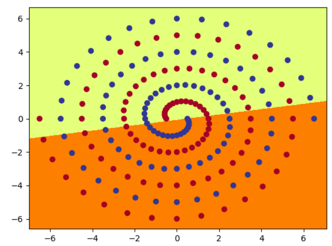
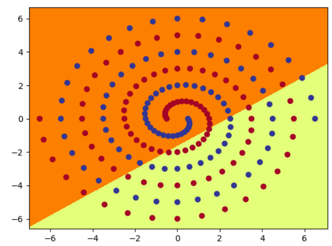
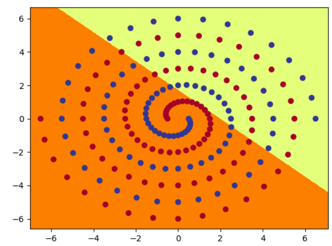
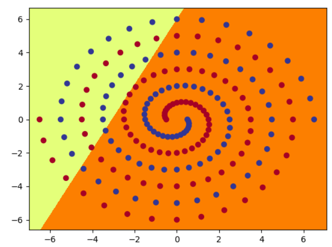
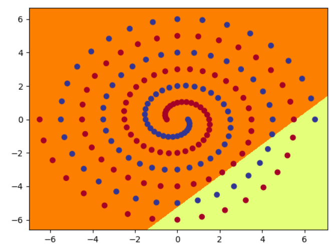
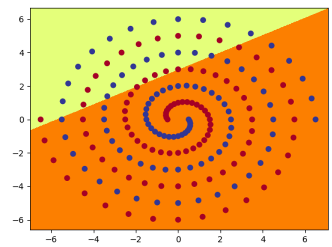
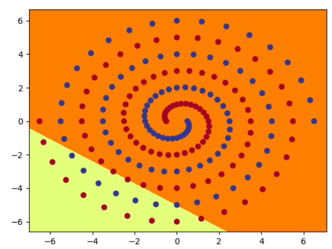
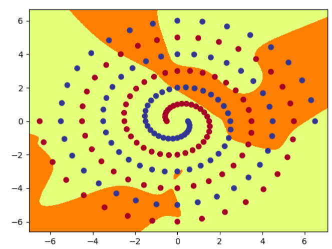
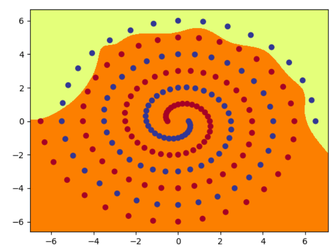
Question 7

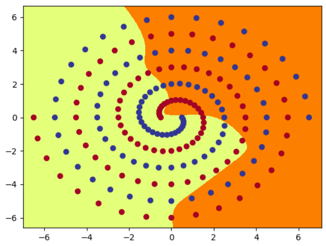
PolarNet Images

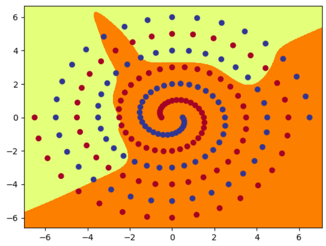
polar1_0.png	polar1_1.png	polar1_2.png
		
polar1_3.png	polar1_4.png	polar1_5.png
		
polar1_6.png	polar_out.png	

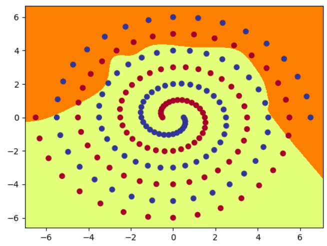
		

RawNet Images

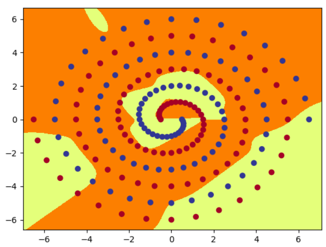
raw1_0.png 	raw1_1.png 	raw1_2.png 
raw1_3.png 	raw1_4.png 	raw1_5.png 
raw1_6.png 	raw1_7.png 	raw1_8.png 
raw1_9.png 	raw2_0.png 	raw2_1.png 
raw2_2.png	raw2_3.png	raw2_4.png



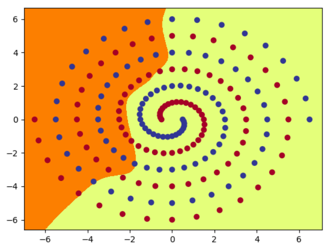




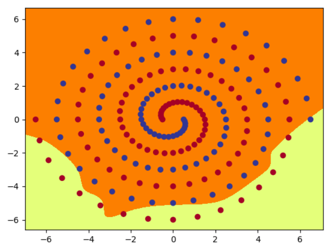
raw2_5.png



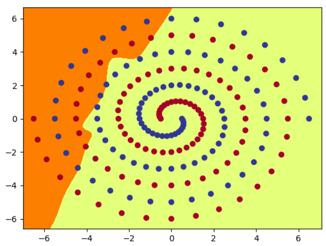
raw2_6.png



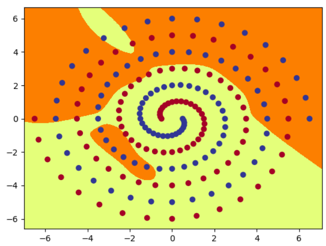
raw2_7.png



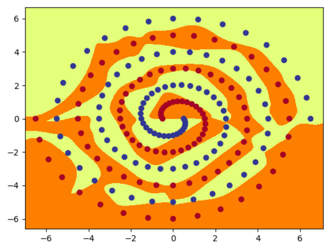
raw2_8.png



raw2_9.png

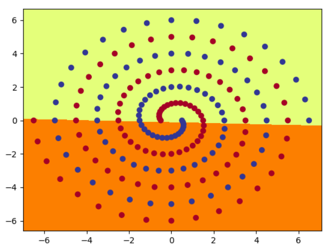


raw_out.png

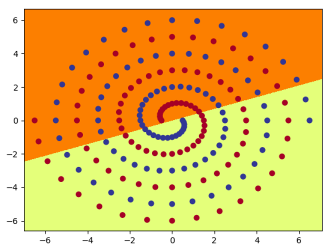


ShortNet Images

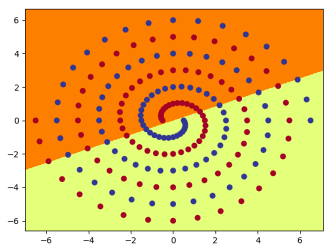
short1_0.png



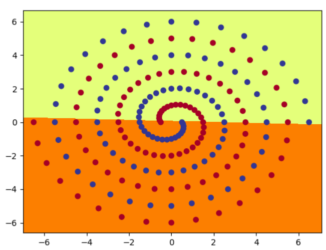
short1_1.png



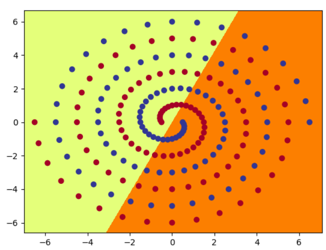
short1_2.png



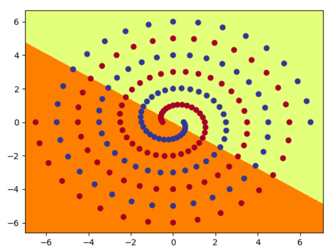
short1_3.png

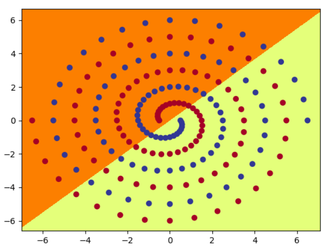
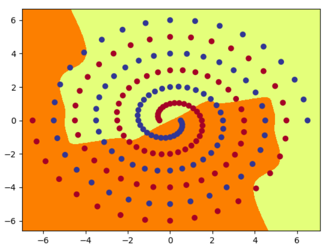
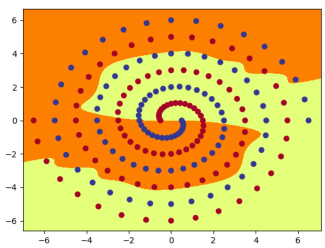
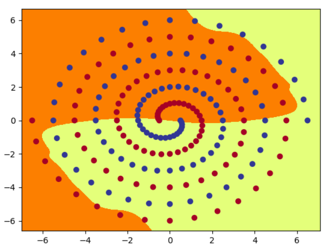
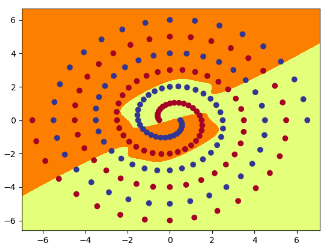
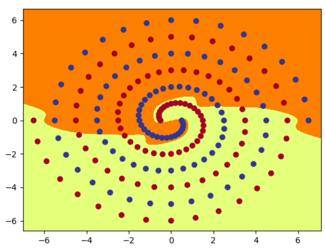
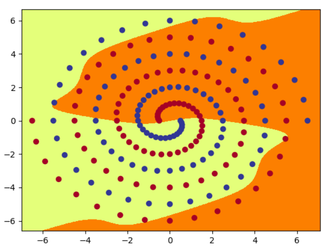
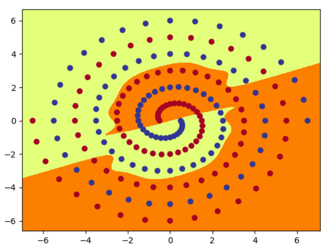
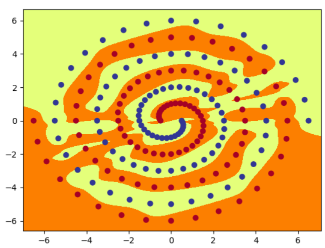


short1_4.png



short1_5.png



short1_6.png 	short2_0.png 	short2_1.png 
short2_2.png 	short2_3.png 	short2_4.png 
short2_5.png 	short2_6.png 	short_out.png 

Question 8

a.
All the 3 models are that are discussed in the exercise vary with respect to their structure. Firstly, in Polar net, the input(x,y) is converted to polar-coordinates and has only one hidden layer compared to other 2 models (RawNet and ShortNet) where raw inputs are used and there are 2 hidden layers. As a result, fist layer of PolarNet has unique values that generates spiral shaped layers compared to other 2 models which generate linearly separated layers. The output of ShortNet and RawNet are similar as they combine the output from the linear layers to form the curves.

b.
I have trained the network with different initial weights ranging from (0.1 to 0.71). Initially it was confusing. The model gets trained faster (within 6000) epochs and suddenly it wouldn't get trained with 30000 epochs. After doing research, I found setting seed is important. After

setting the seed to 0, I started getting the same results. Thus, I found, with the init weight 0.121 both ShortNet and RawNet gets trained within 20000 epochs.
For the init weight < 0.121, the network gets stuck with the accuracy ~50% and does not improve even after 20000 epochs. On the other hand, init weight >0.121 and <0.29, the network learns faster (especially for 0.151) and exhibits similar behaviour of value <0.121 for the value >3.0.
In extreme cases it takes lot of time to converge or might not converge at all.

c.
As discussed earlier, the PolarNet has a different input compared to ShortNet and RawNet. As a result, it looks more natural. Among ShortNet and RawNet, the ShortNet is more natural compared to RawNet as it has Short connections along with 3 layers. However, the shape of the output interpreted by the system using the filters as compared to the perfect shape given the polar net using the Polar Co-ordinates.

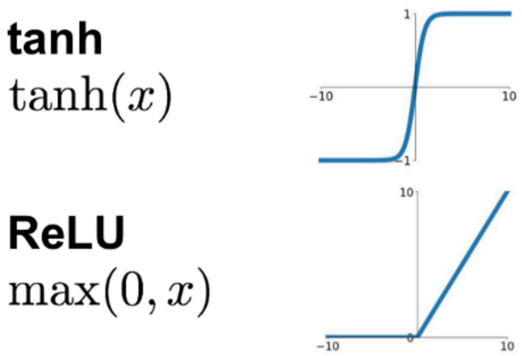
Data representation plays a very important role in deep learning task as everything the model learns or predicts depends on this. For a very good data representation, we need to remove the noise in the data using different techniques and domain knowledge. However, if we are collecting fresh set of unstructured data we can apply unsupervised learning techniques and use the data effectively.

d.
Changing the batch size 94 to 197:
It had positive impact on the execution of the program. It always gave better performance with respect to time and number of epochs required to train the program.

using SGD instead of Adam:
After changing the optimizer to SGD from Adam, the program was taking very long time to converge with batch size 197, 10 hidden nodes and default learning rate. However, after setting the batch size to 10 and learning rate to 0.01, it started to converge within 20000 epochs.

Take away: For each optimiser it is necessary to set other metaparameters.

Changing the tanh() to Relu()



After changing the tanh() to Relu() the model won't achieve 100% accuracy. From the above graph it is clear that ReLU() ignores some the outputs and is not symmetric. Since this program intends to solve the spiral problem, it is evident that ReLU is not suitable for this program.

Adding more layers:
AS discussed in the previous part of the assignment, adding more layers did not have much effect on the output or learning task.

Note:

All the runs were conducted on the CSE Machine Weber.

Even though Weber, Weill, Wagner, Williams have same version of required libraries, processors, the results returned by all these machines were different.

For example, while finding the initial weights for the RawNet with hidden nodes 10, the model never converged on Weill, Wagner, Williams within 20000 epochs for the value 0.121. However, it did converge for the value 0.151.
On the other hand, Weber, local machine (MacBook Pro 2019) and google collab exhibited similar results and the algorithm did converge within 20000 epochs on all the 3 machines.
Results: https://github.com/anantkm/comp9444_results/tree/master/RawNet