Report

Part 1

For part 1, linear model, 2-layer fully connected model and convolutional network are trained on the same dataset KMNIST to classify 10 kuzushiji characters. Obviously, the performance of convolutional network is better than others while the simplest linear model has the lowest accuracy. Here are the confusion matrices of these three models successively:

Confusion Matrix of NetLin

	o	ki	su	tsu	na	ha	ma	ya	re	wo	
	0	1	2	3	4	5	6	7	8	9	chosen
0	735	9	7	3	84	11	5	15	16	29	
1	6	647	49	32	49	35	53	21	41	62	
2	6	73	602	39	53	100	109	25	48	65	
3	35	24	50	770	22	20	16	17	66	10	
4	34	36	25	22	601	16	39	139	10	62	
5	52	25	29	36	19	695	22	15	30	33	
6	3	74	65	19	38	36	698	80	57	18	
7	75	26	49	27	40	19	29	535	8	54	
8	31	27	59	37	29	51	16	109	703	36	
9	23	59	65	15	65	17	13	44	21	631	
target											

Test set: Average loss: 3.1118, Accuracy: 6617/10000 (66%)

Confusion Matrix of NetFull

	o	ki	su	tsu	na	ha	ma	ya	re	wo	
	0	1	2	3	4	5	6	7	8	9	chosen
0	844	4	8	4	62	8	5	18	8	7	
1	3	806	23	14	32	23	26	19	31	28	
2	1	28	789	25	25	72	54	19	18	50	
3	3	5	55	906	8	17	13	7	48	4	
4	35	20	11	4	772	15	23	39	4	39	
5	26	10	17	16	12	806	11	3	10	9	
6	4	65	38	6	24	23	840	44	26	23	
7	46	10	14	9	15	4	11	768	7	23	
8	28	21	24	6	27	20	7	49	843	16	
9	10	31	21	10	23	12	10	34	5	801	
target											

Test set: Average loss: 2.5810, Accuracy: 8175/10000 (82%)

Confusion Matrix of NetConv											
	o	ki	su	tsu	na	ha	ma	ya	re	wo	
	0	1	2	3	4	5	6	7	8	9	chosen
0	942	3	11	3	24	4	3	4	4	7	
1	3	905	6	3	10	9	6	10	11	7	
2	1	11	870	13	5	39	18	3	7	9	
3	2	0	48	970	6	7	2	2	12	3	
4	19	11	6	2	905	6	5	10	2	12	
5	8	1	7	4	5	909	2	0	6	1	
6	2	44	16	2	14	14	958	10	7	4	
7	15	4	15	1	11	5	2	935	2	5	
8	4	7	11	1	13	5	2	10	947	1	
9	4	14	10	1	7	2	2	16	2	951	
target											

Test set: Average loss: 2,2441, Accuracy: 9292/10000 (93%)

The highlighted cells stand for the characters which are more likely to be mistaken, the deeper the colour, the higher the probability.

There are several characters that are likely to be misclassified by all three models, which are listed below in the (output, target) form:

(0, 4), (2,5), (2, 6), (3, 2), (4, 0), (5, 0), (6, 1), (7, 1), (8, 7), (9, 1), (9, 2), (9, 7)

After careful investigation, the reason for the higher error rate is found: firstly, two characters may have similar structure, secondly, they may contain same subparts.

For example, お is most likely to be misclassified as な, it is obvious that they both have a cross, a dash and an ellipse in the lower right corner. Similar structure and identical subparts are the main reasons of mistake especially when they are written in cursive style.

Another example for structure is れ and や, basically they don't share any subparts in common, but the error rate for them is substantially high in the first two models. The reason is clear: NetLin and NetFull mainly look at structures as the layers are simpler, but convolution network may look at relatively complex patterns of subparts. These two characters both concentrate on top left corner and extend the stroke to right. When NetConv finds that the character doesn't hold certain subparts, even its structure strongly suggest that it is some other character, the network still refuses that proposal. Thus, the likelihood of NetFull recognizing れ as や is not significant.

All the pairs listed above follow the same rule, even a convolution network considers structure and subparts at the same time, wrong classifications still exist because the dataset is extracted from cursive human written samples which is not that standard as expected.

Also, a lot of experiments were made on the architecture of the models, especially on CNN models. Number of filters, size of filter, stride size, paddings, dropouts, maxpool, nodes for fully connected layer are tested. The result was surprising but reasonable: firstly, size of

layers or number of nodes and accuracy is not positively correlated, secondly, simplification (maxpool, dropout) does not always lead to worse result, finally, there is no negative correlation between speed and performance.

At first, several tests were performed to derive the number of nodes in the hidden layer of NetFull to achieve a local minimum for error rate. The relation between number of nodes and error rate looks like a tick, in which there is a local minimum. For this dataset, the local optimum hidden size is 110. Also, a lot of experiment were made on the parameters of the convolutional network. Surprisingly, the accuracy doesn't grow along with number of filters or filter size consistently, which is because when the filters expand, more noise are considered and then it leads to overfitting.

Secondly, stride and maxpool also played the role of avoiding overfitting, it reduces the size of output filter to eliminate some of the noise. It is interesting to find that maxpool can improve the network when it is placed between two convolutional layers, however, it can also ruin the whole model if it follows the second convolutional layer. The reason should be oversimplifying: the output of the second layer is accurate enough and doesn't contain significant amount of useless information, thus maxpooling may eliminate nearly 75% of useful information (when using 2x2 maxpool). The extent of simplification should be adjusted elaborately to achieve a high efficiency of dropping out distracters come from the dataset.

In the end, during the experiment, it is concluded that there is no visible relationship between speed and accuracy. This is deduced from the first two findings: Larger layers require more time to complete, but obviously that bigger is not always better. Simplification also accelerates the network to some extent without reducing accuracy. Usually, an optimal model does not require much more time to run, when the result reaches optimum, the efficiency often reaches optimum as well. Speed is always a important topic, reducing time cost without worsen the model is also the last but crucial step after finding the optimum parameters. Sacrifice of accuracy is not necessary if appropriate techniques are used.

Moreover, meta-parameters also have significant influence on the performance which will be discussed separately.

--lr: learning rate determines the efficiency of learning, the higher the rate, the faster the model learns. Faster does not refer to the speed of program, rather the amount of adjustment made based on each training example. Although the time spend is fixed, typically a higher learning rate lead to a higher accuracy. However, when it is too big (depend on the model and dataset), the effect would be negative: accuracy may decline instead. The reason is straight forward because the model may "over learn" thus passes over the optimal answer. The learning rate suggested for the netConv is 0.15, final accuracy of 96% could be achieved with which.

--mom: momentum improves the process of approaching better answer by accelerating. That leads to a faster speed which saves time and make it easier to jump over a local minimum. The model has a better performance with mom equals to 0.75.

--epochs: epoch dose not affect the performance a lot, once the number of epochs is enough, no obvious change would be made, rather, overfitting and overtime may occur as consequence. 10 epochs are fairly enough for KMNIST.

--cuda: CUDA is not a part of the model, so it's not discussed here. The role of CUDA is using power of GPU to speed up the program.

Part 2

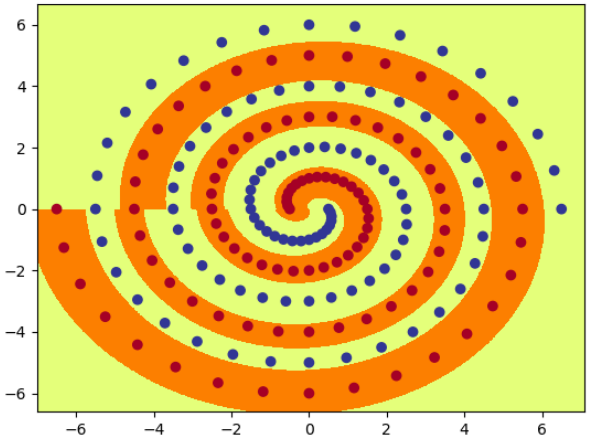
The three models construct very different hidden layers which vary from shape and usage. The first layer of Polar net has distinct hidden value comparing to the other two because of the unique coordinate system used. It generates spiral shaped masks while others have linear separated hidden layers. Another difference is that PolarNet generates the final answer directly from the first hidden layer by overlapping them while RawNet and ShortNet need to first combine those linear layers to form some curves in second layer to cover the spiral shape. There is no significant difference between the layers of RawNet and ShortNet.

Some experiments were made on initial values for RawNet and ShortNet. It is found that initial value of 0.2 gives quite satisfying performance with 7000 expected epochs. Although sometimes it stuck on some point and epochs needed approach to 20000, it is still much better than those values in both directions. A larger initial weight could save time for approaching to optimal answer however the network may go far away if the initial weight is too large.

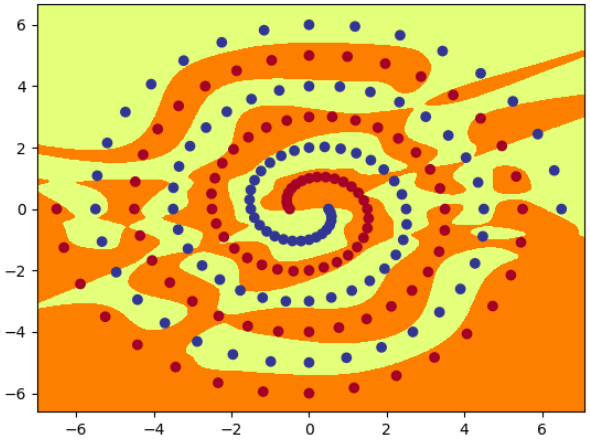
Apparently, the output of PolarNet is more natural than the models using Cartesian coordinate system. Meanwhile, the naturalness of the output of ShortNet is relative higher than that of RawNet. The reason for high naturalness of PolarNet is the spiral shape which can be represented approximately in a linear form with polar coordinates. Thus, the regression for this model is more likely to approach the real function of the spiral shape. However, only approximate shape can be imitated by networks using Cartesian coordinates. Representation is regarded as a fixed “layer” to filter data which is designed by human, with an appropriate representation, the model can fit the samples with higher accuracy. The purpose of elaborating representation of data is to cover as much features which are fixed, obvious and do not require machine learning to find out as possible. With a pertinent representation, elevation of both accuracy and speed of a model is possible.

In the end, some experiments were made on the parameters and architectures. The first finding is that larger batch size does not always gives a better performance. Typically, increase of batch reduces time needed and may increase the efficiency of learning. 1, 97 and 194 were chosen as batch size, and it is found that when batch size is 1, the training process is considerably slow. Even it is faster to train each epoch when 194 is chosen, the efficiency is not satisfying: data can not be fitted within 20000 epochs. So, choosing an appropriate batch size can lead to higher accuracy and speed. Also, different activations were tested and it is found that ReLU is not suitable for this task. Because the spiral shape is smooth and divided equally, and Tanh fits the features well. However, the shape of ReLU is too sharp and it does not give symmetric output, some points would be hard to be fitted as it may be ignored by ReLU.

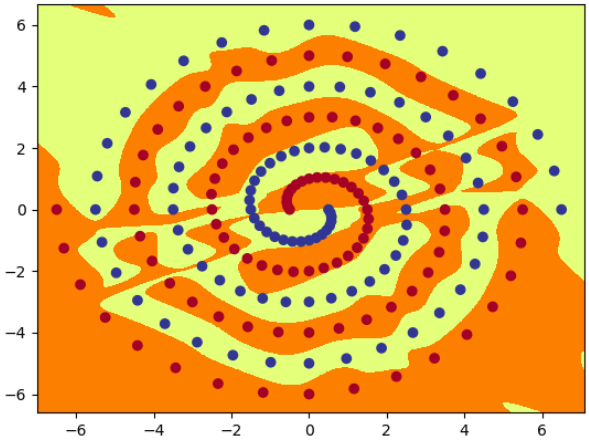
Appendix: Output Functions



Output for PolarNet

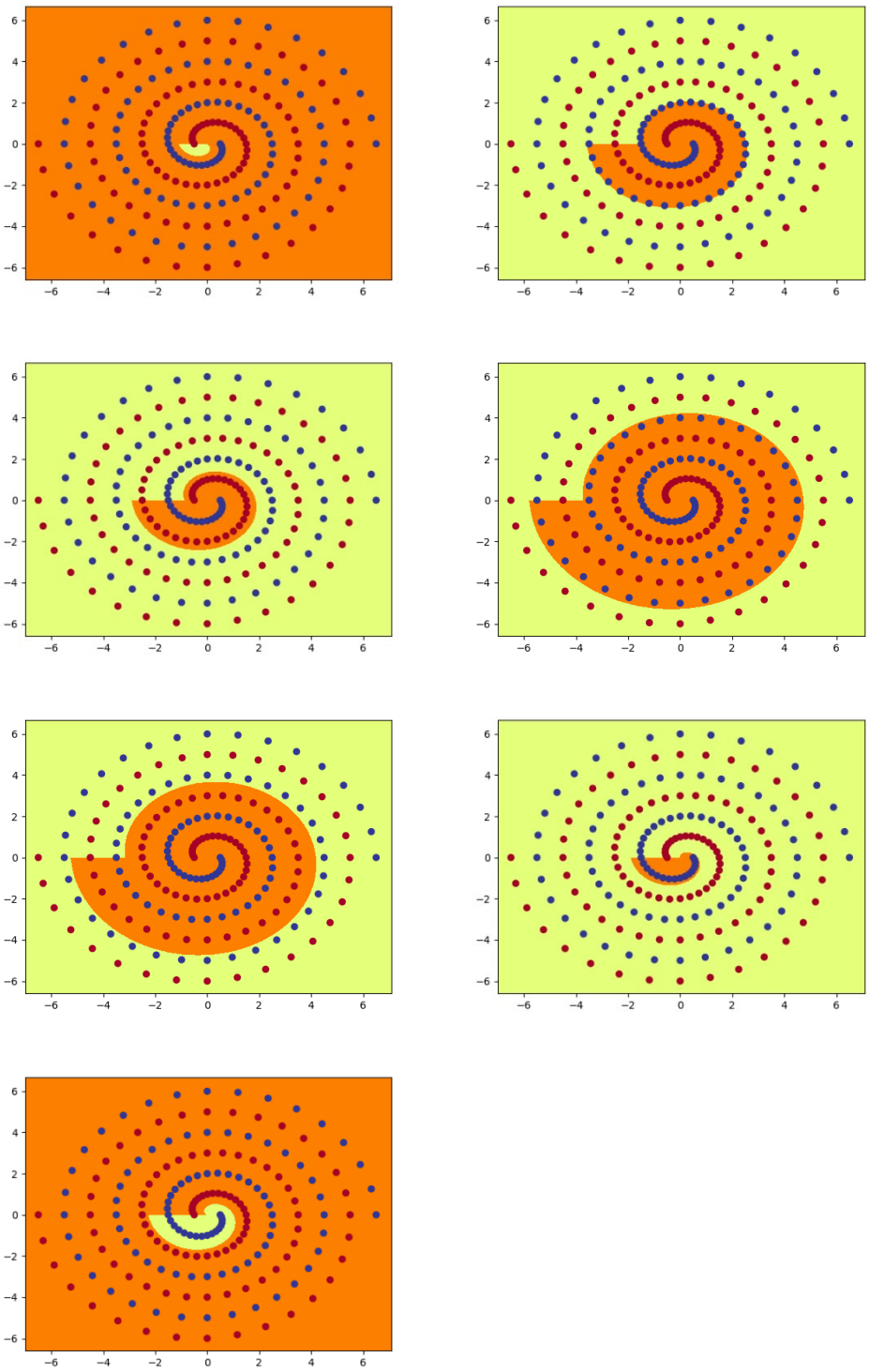


Output for RawNet

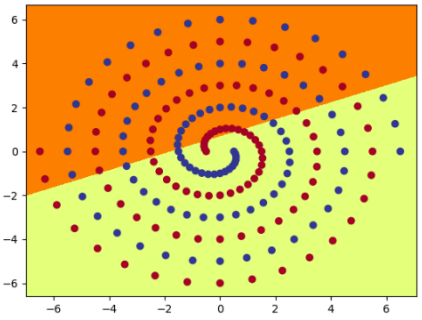
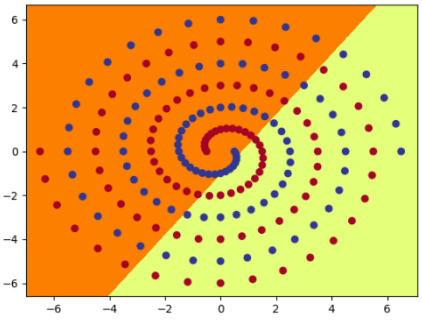
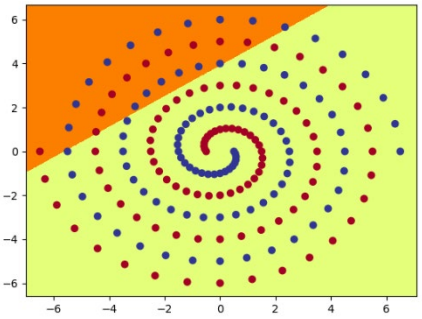
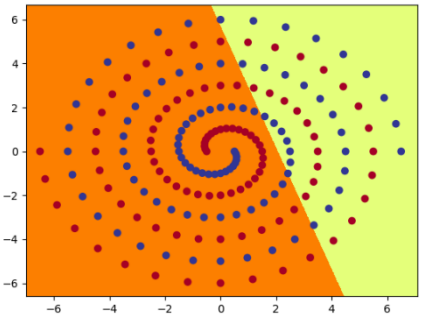
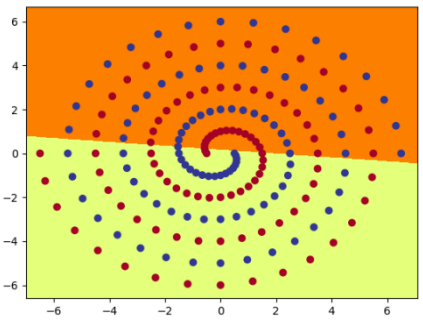
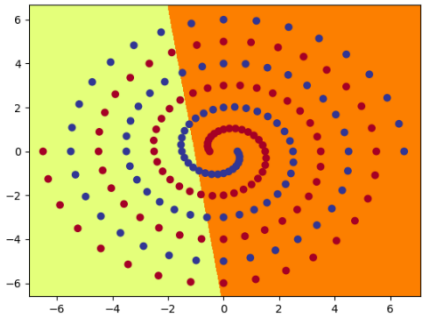


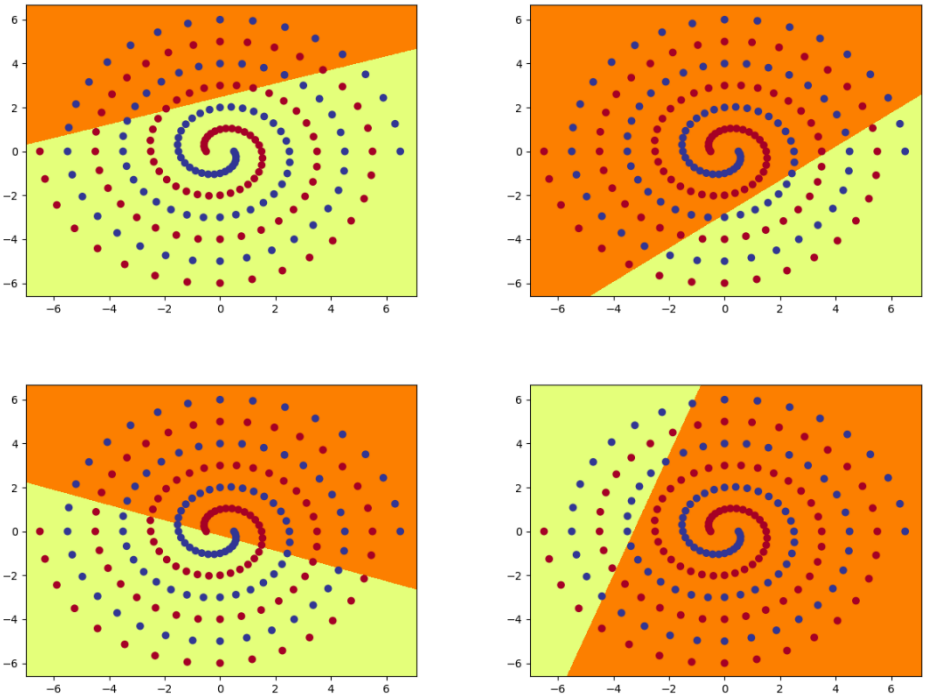
Output for ShortNet

Appendix: Hidden Layers

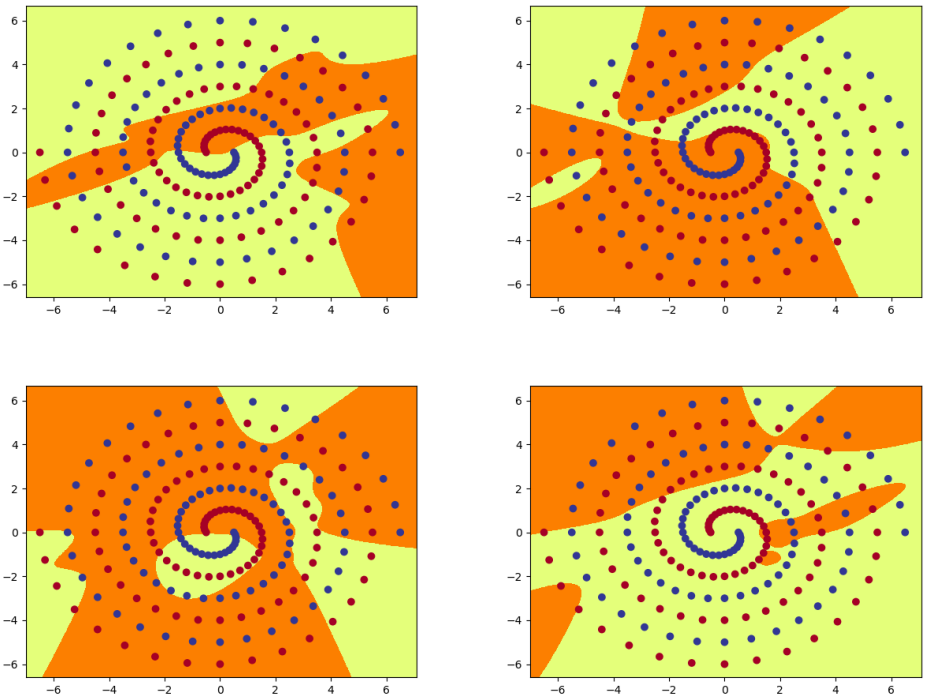


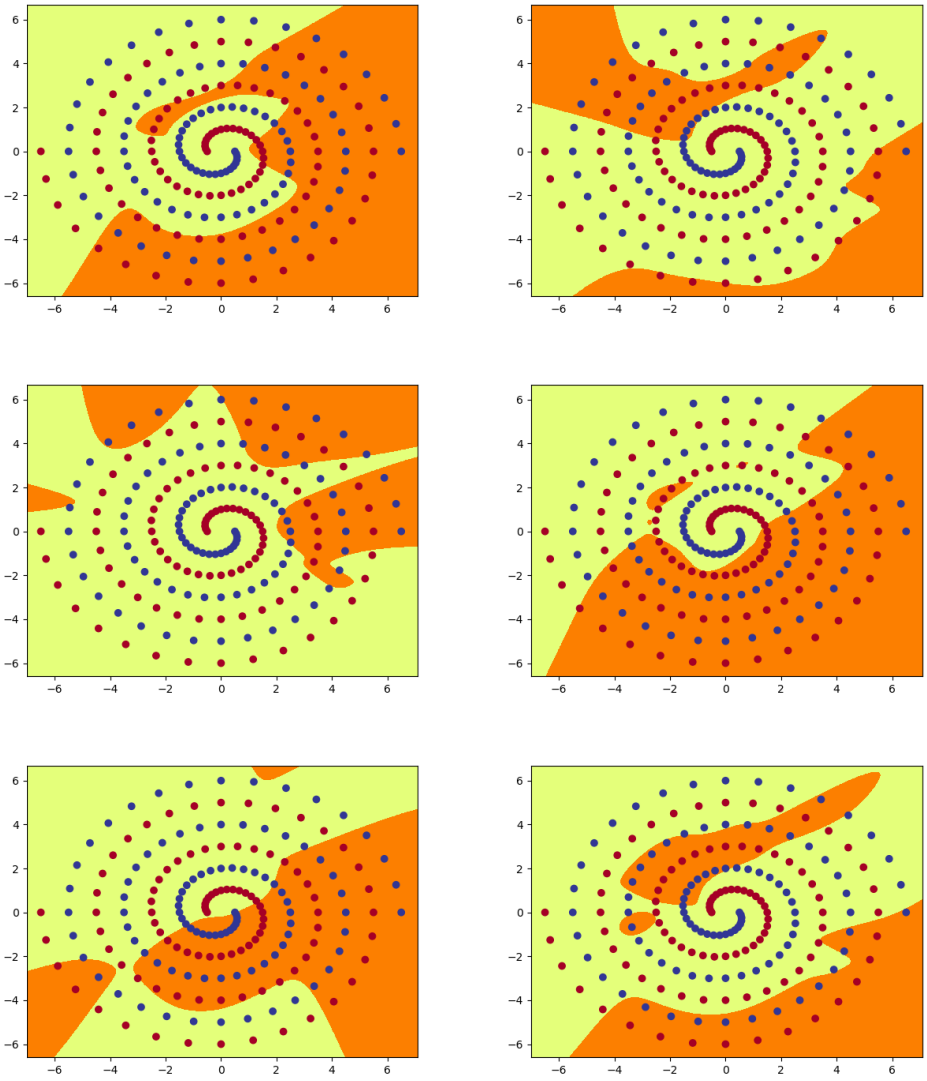
Hidden Layer for PolarNet

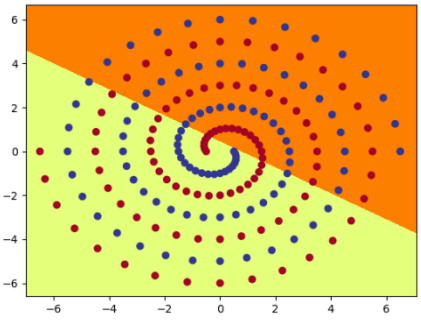
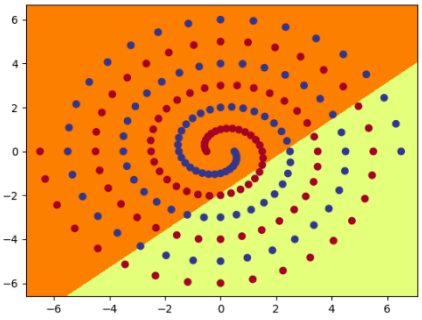
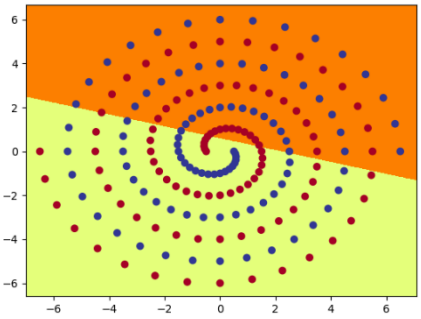
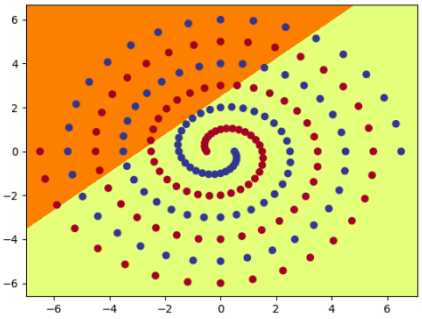
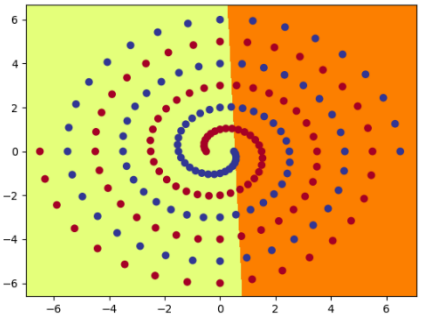
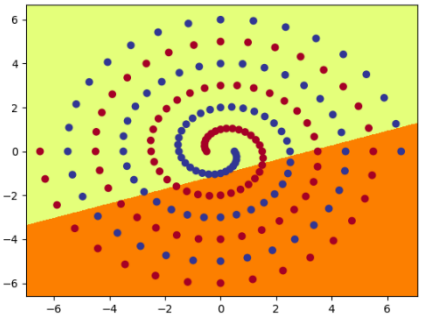


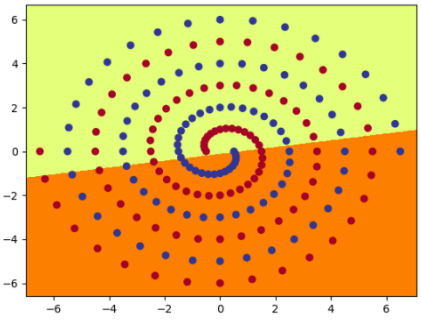
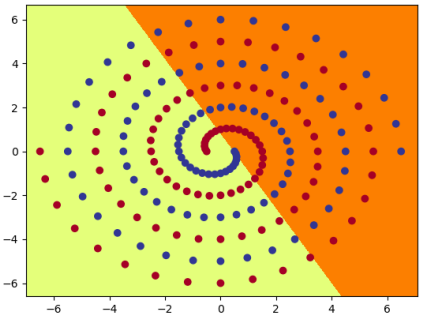
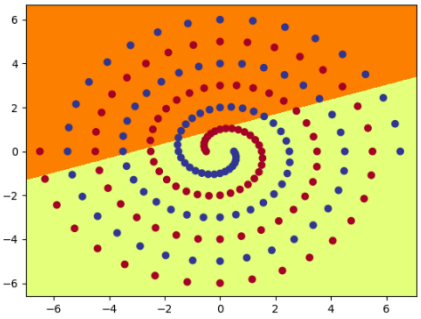
Layer 1 for RawNet



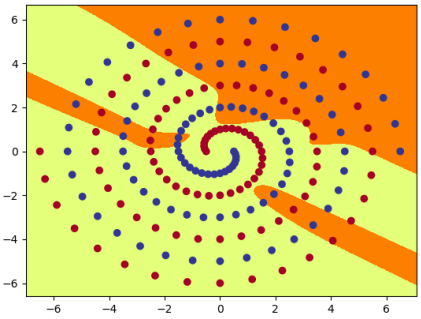
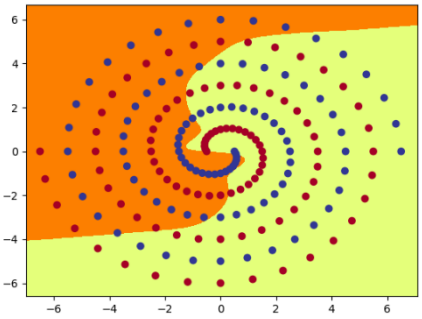
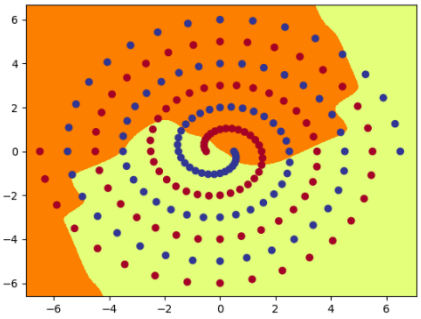
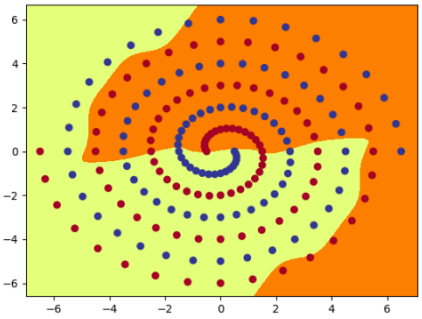


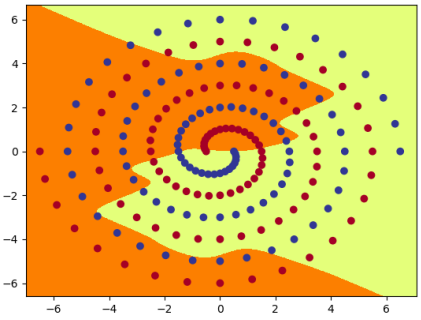
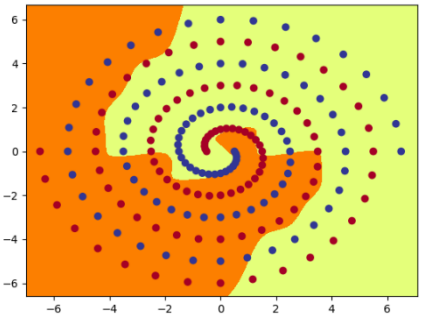
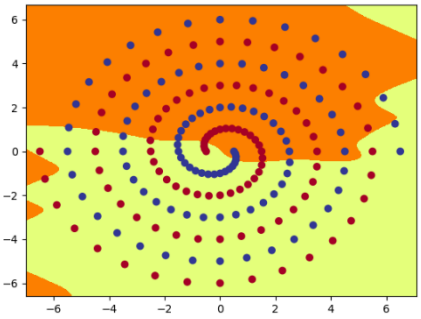
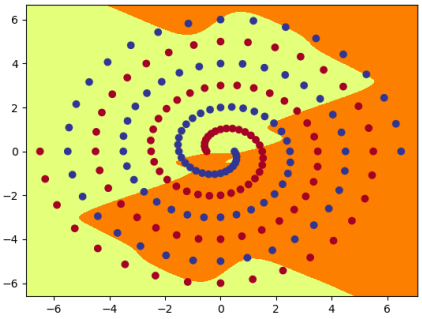
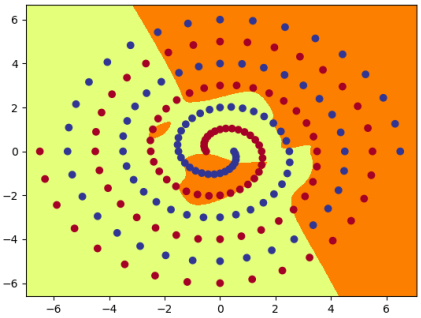
Layer 2 for RawNet

Layer 1 for ShortNet

Layer 2 for ShortNet