Lab4

Name: Zeng Yuhang

ID: 222320008

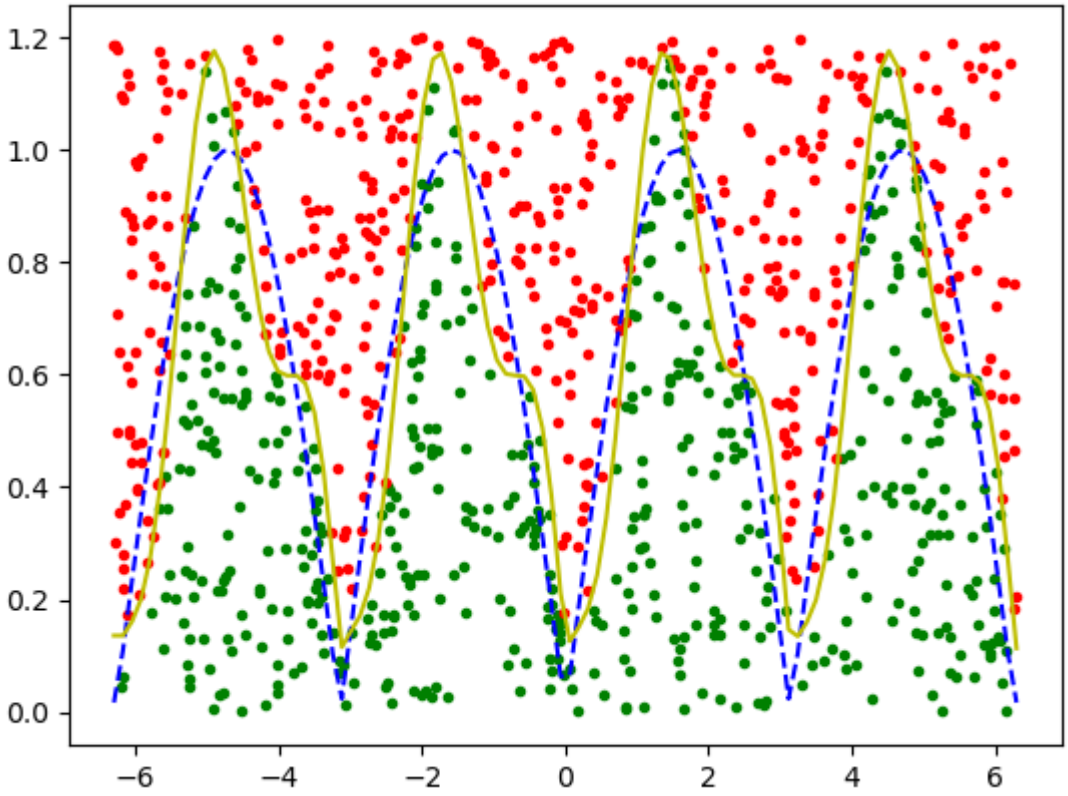
Variant: 1

Variant

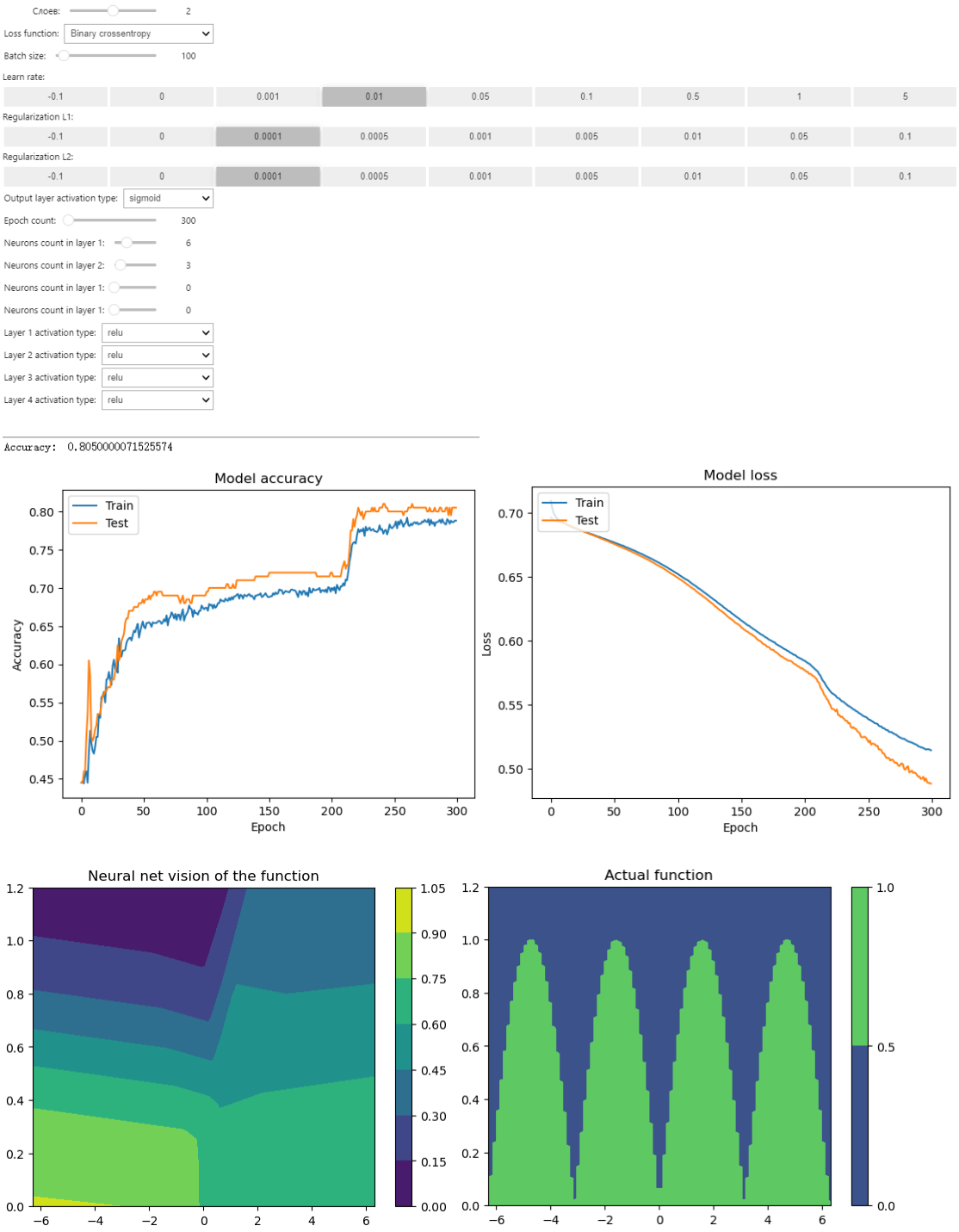
Var	Part1 func	Part2 data	Hyperparameters*
1	Absolute(Sin(x)) X: -6.3..6.3 Y: 0..1.2	CIFAR10	Layers count, neurons count per layer

Part1

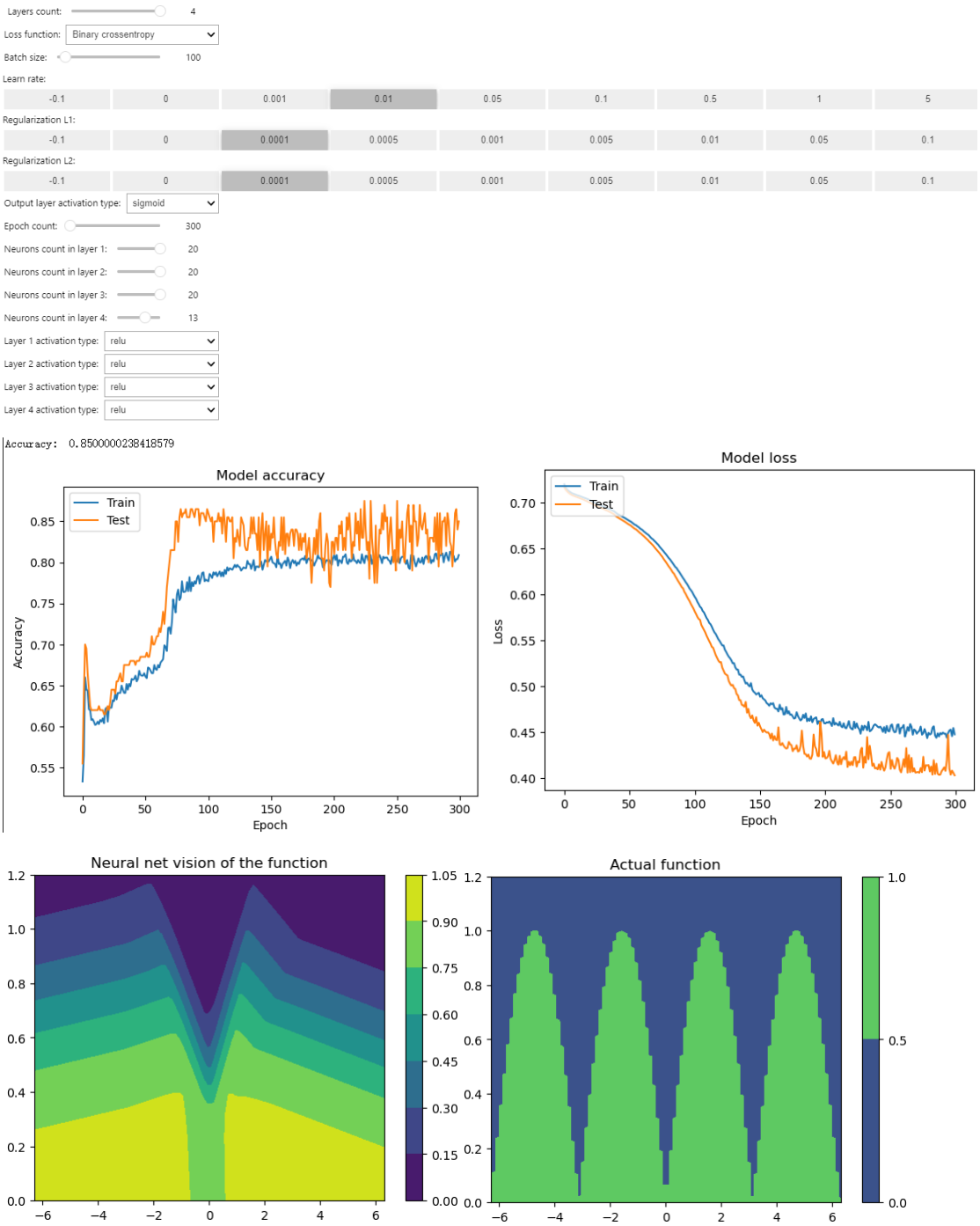
The problem is a binary classification. We create data with some noisy to simulate the real-world problem. The distribution of data is shown below. The red and green dots infers two different class. Blue line is the function we want the model to fit, while yellow line is the exact boundary of two class.



The accuracy with default parameters is 0.805.



Then, I change the Layers count and neurons count per layer. Try to make the accuracy close or reach 0.95 but failed. Only reach 0.85:



I find that the loss decrease trend is not obvious after 300 epochs training, but the model only learned two peaks out of four and the loss trembles a lot. So, I decrease the learning rate and tried to change other parameters to make model more complex. However, in practice, the lower learning rate turns out to track into local optimal solution. Finally, reaches 0.96.

Layers count: 4

Loss function: Binary crossentropy

Batch size: 100

Learn rate:

-0.1	0	0.001	0.01	0.05	0.1	0.5	1	5

Regularization L1:

-0.1	0	0.0001	0.0005	0.001	0.005	0.01	0.05	0.1

Regularization L2:

-0.1	0	0.0001	0.0005	0.001	0.005	0.01	0.05	0.1

Output layer activation type: sigmoid

Epoch count: 3000

Neurons count in layer 1: 20

Neurons count in layer 2: 20

Neurons count in layer 3: 20

Neurons count in layer 4: 10

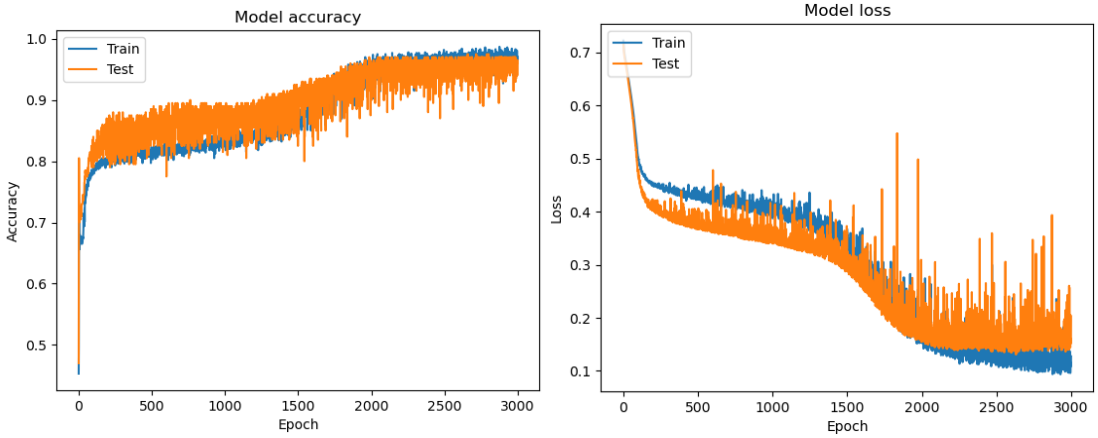
Layer 1 activation type: relu

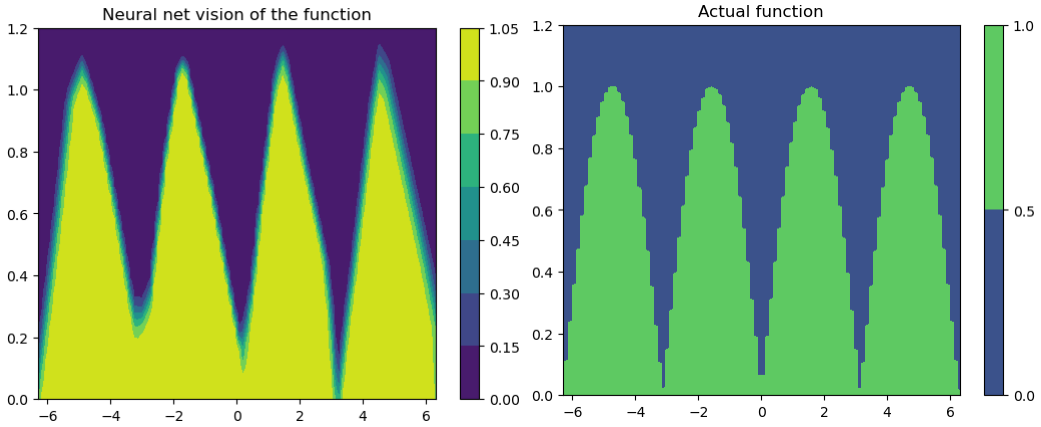
Layer 2 activation type: relu

Layer 3 activation type: relu

Layer 4 activation type: relu

Accuracy: 0.9599999785423279





Part2

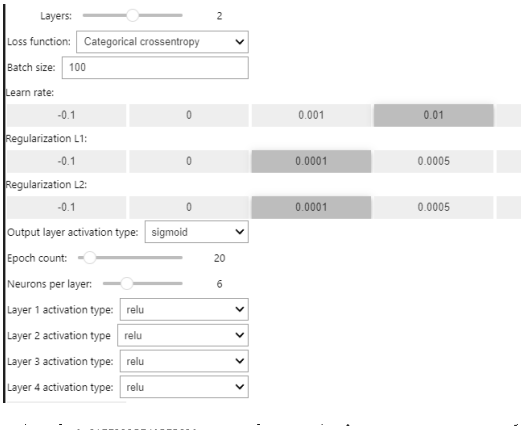
There are represented such hyperparameters as

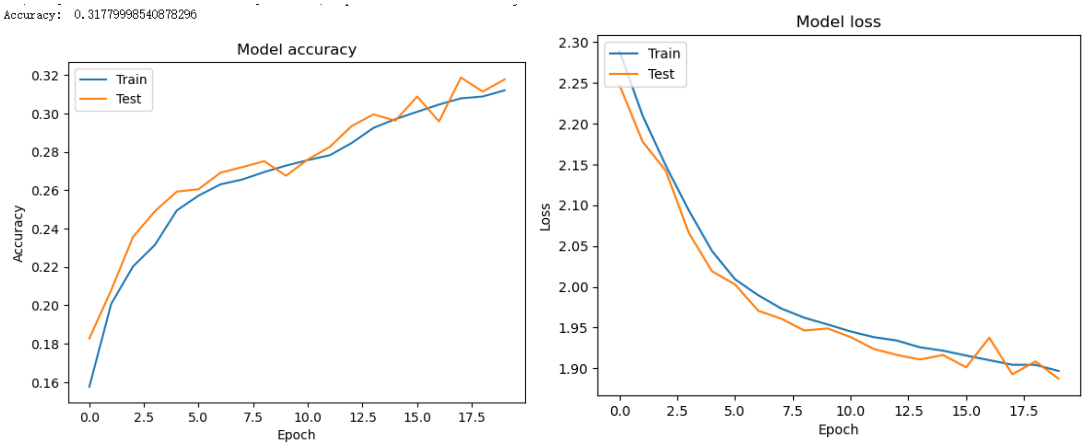
- Layer count: dense of the model
- Neurons count per layer (actually it's not hyperparameter but structure parameter): width of the model
- Learn rate: the length of each step
- Regularization L1 and L2: punish the model if it is too complex

- Output layer activation type: the function to get the result
- Layer activation type: the function to calculate value inside the network at each neuron
- Loss function type: the function to calculate loss which is used to bp to optimize the network
- Epoch count: the times of training

The CIFAR10 is a 10-classification problem. And the original code has two serious problems which will make the network can't learn any thing from data. First, the structure of CIFAR10 dataset is different from mnist. The given one_hot function is not suitable for CIFAR10, we need to change the format of CIFAR10 to fit mnist. Second, each input is a set of integer range from 0 to 255, which is too large for the model to learn. We need to do normalization before start training.

After processing the data, the accuracy with the default parameters is 0.32.

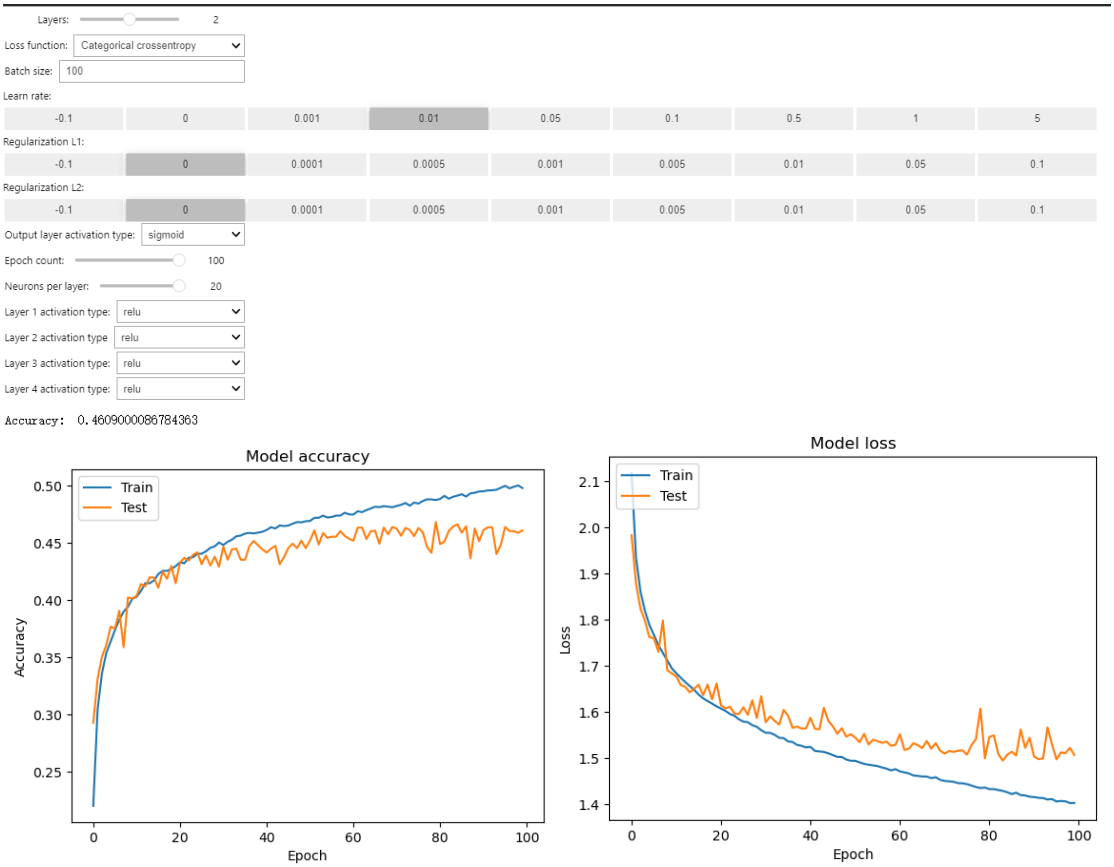




Then, I change Layers count and neurons per layer. The accuracy of each set of parameters is shown in the table below. The rows refers to the layers count, and the coloums refers to the neurons per layer:

	1	**2**	**6**	**10**	**15**	**20**
1	0.190	0.262	0.368	0.404	0.421	0.425
2	0.100	0.182	0.327	0.371	0.396	0.426
4	0.100	0.183	0.294	0.367	0.390	0.409

Only using fully connected network to solve CIFAR10 problem is really hard to get good performance. I change other parameters, only reach accuracy of 0.461.



The max 100 epochs are not enough for model to convergent on training set. Larger learning rate cannot fix this problem, because it will also lead to the same trouble through a different way.

Summary

Describe impact of each hyperparameter on accuracy:

Layers Count:

Layers count needs to be appropriate. Too small will lead to bad performance. Meanwhile, too large will make the network to complex so that increase the probability of overfit.

Neurons count per layer:

Too much will lead to overfit, too little will lead to underfit.

I can't make the accuracy up to 0.95 in part2, I think maybe I didn't find the best hyperparameters or there are some better solutions to process data to extract feature before training.

In conclusion, I find it's really hard to tune the hyperparameters and difficult to understand network, I really need to learn more.