# HW3

CeFang N14871039

March 27 2017

## 1 HW3.1

For the first question, I design a fully connected neural network as follow:

The network is design as figure1. I use 10 linear units and 10 sigmoid units, then, concatenate with 4 linear units and 4 LogSoftMax units. Then, the index of maximum value of four outputs is the predict label.

Here, I set the mini-batch size is 200 and iteration time is 15.

By comparing with the real label, I obtain the accuracy for train and test data: Accuracy for train data is : 0.92777777777778
Accuracy for the test data is : 0.9275

The test loss and train loss is shown in figure2.

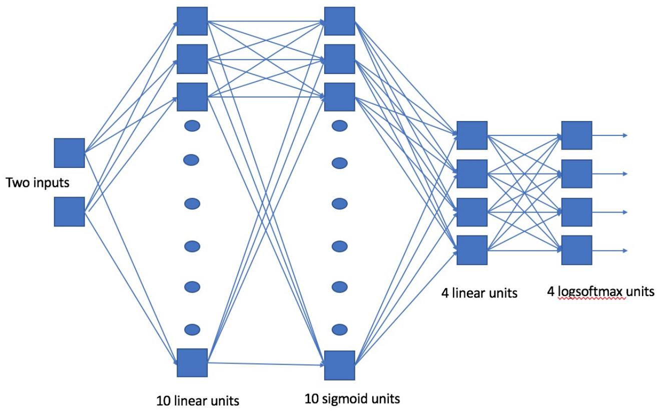


Figure 1: Architecture of my design

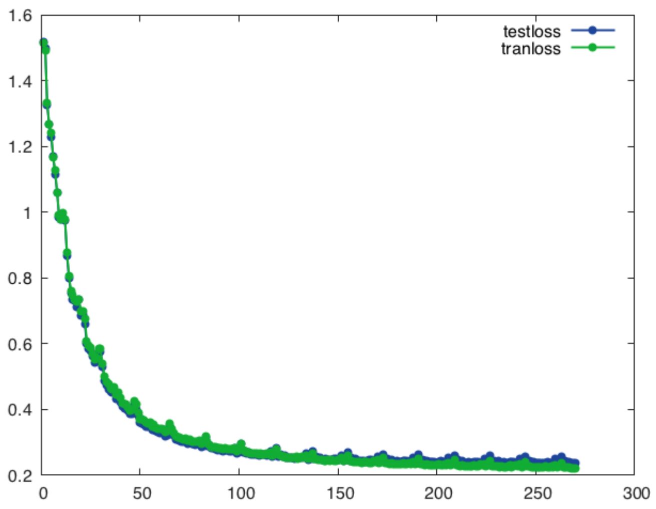


Figure 2: Train and test loss for classification

## 2 HW3.2

For the second question, I design a fully connected neural network as follow:
I use 2 units for input layer and 10 linear units then 10 ReLU units for hidden layer, and 1 unit for output layer.
The structure is shown in figure3.

Here, I set the mini-batch size is 100 and Iteration time is 25.
The loss plot is shown in figure4.

Final training loss is : 79.130469105464
Final test loss is : 99.402951500766

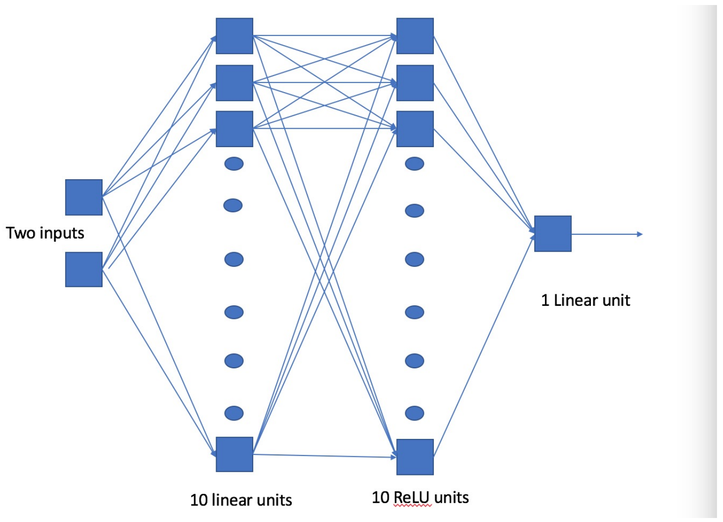


Figure 3: Architecture of my design

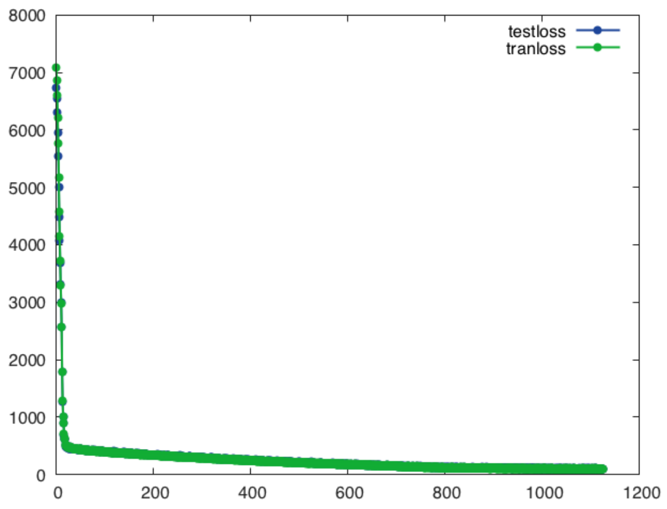


Figure 4: Train and test loss for regression

## 3 HW3.3

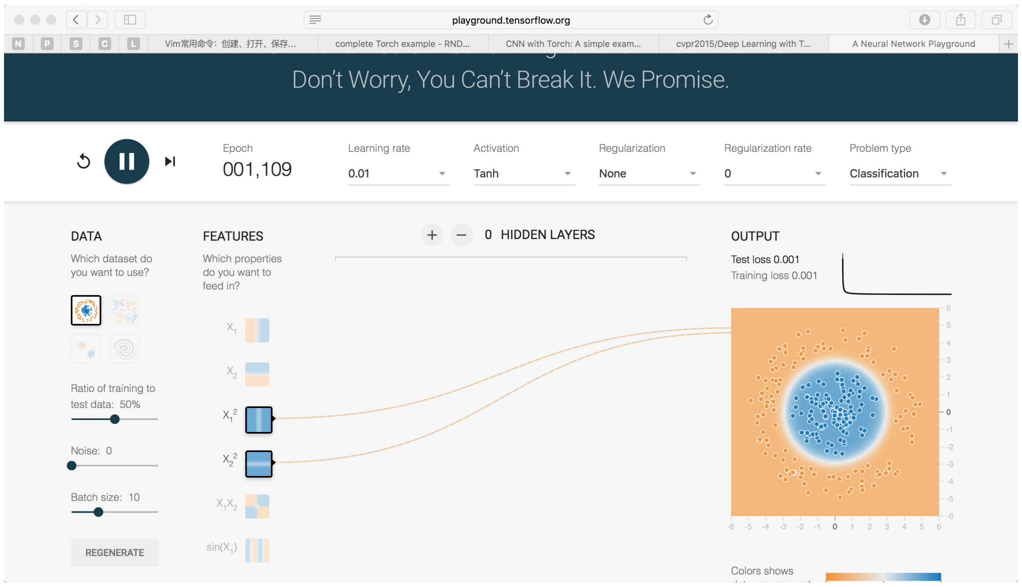
For the first data sets,I only use two neurons:
The result is shown in figure5.

For the second data sets, I only use one neuron:
The result is shown in figure6.

For the third data sets, I only use one neuron:
The result is shown in figure7.

If we want better performance, we can use two neurons:
The result is shown in figure8, from which we can see that the train loss is reduced from 0.001 to 0, and the test loss is reduced from 0.006 to 0.

For the forth data sets, I only use 9 neurons:
The result is shown in figure9.
For this part, I have to try several times for this result with the same parameters, I guess this is happening because our initialization is selected randomly, and our loss function is not a convex function. So, our final solution may fall into an local minimum instead of the global minimal. I got this result after I try 9 times for the same setting.


Don't Worry, You Can't Break It. We Promise.

Epoch 001,109 | Learning rate 0.01 | Activation Tanh | Regularization None | Regularization rate 0 | Problem type Classification

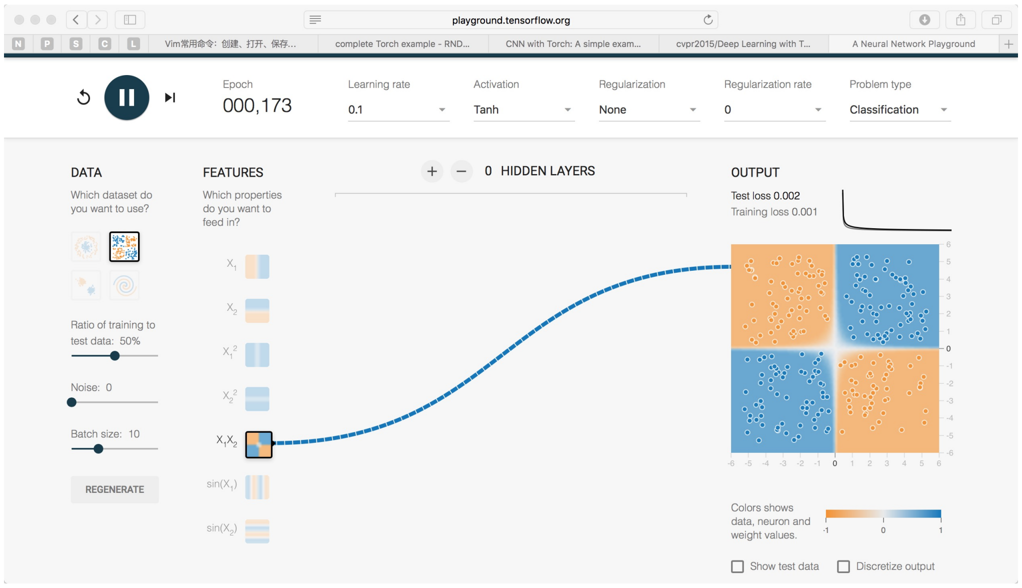
DATA: Which dataset do you want to use? Ratio of training to test data: 50% · Noise: 0 · Batch size: 10

FEATURES: Which properties do you want to feed in?

0 HIDDEN LAYERS

OUTPUT: Test loss 0.001 · Training loss 0.001

Figure 5: For the first data sets


Epoch 000,173 | Learning rate 0.1 | Activation Tanh | Regularization None | Regularization rate 0 | Problem type Classification

DATA: Which dataset do you want to use? Ratio of training to test data: 50% · Noise: 0 · Batch size: 10

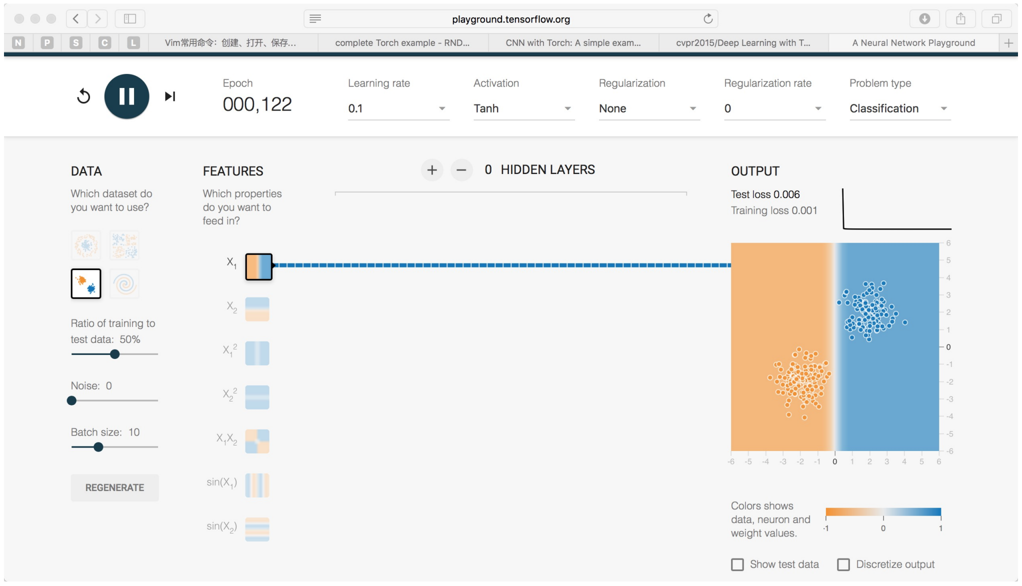
FEATURES: Which properties do you want to feed in?

0 HIDDEN LAYERS

OUTPUT: Test loss 0.002 · Training loss 0.001

Colors shows data, neuron and weight values.

Figure 6: For the second data sets


playground.tensorflow.org
Vim常用命令：创建、打开、保存...
complete Torch example - RND...
CNN with Torch: A simple exam...
cvpr2015/Deep Learning with T...
A Neural Network Playground
DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
HIDDEN LAYERS
OUTPUT
REGENERATE
Colors shows data, neuron and weight values.
Show test data
Discretize output

Epoch 000,122

Learning rate 0.1

Activation Tanh

Regularization None

Regularization rate 0

Problem type Classification

Ratio of training to test data: 50%

Noise: 0

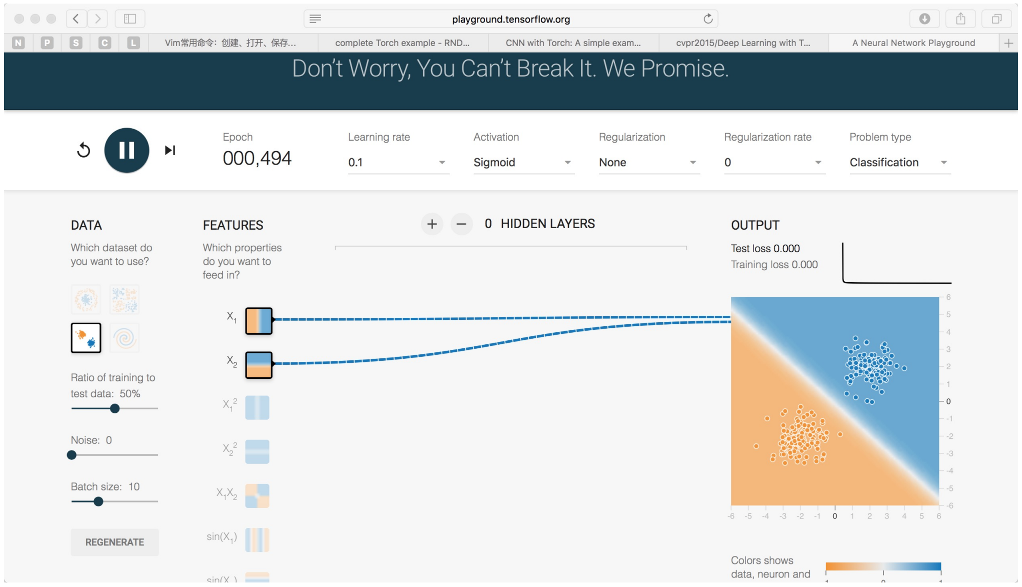
Batch size: 10

0 HIDDEN LAYERS

Test loss 0.006

Training loss 0.001

Figure 7: For the third data sets


playground.tensorflow.org
Vim常用命令：创建、打开、保存...
complete Torch example - RND...
CNN with Torch: A simple exam...
cvpr2015/Deep Learning with T...
A Neural Network Playground
Don't Worry, You Can't Break It. We Promise.
DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
HIDDEN LAYERS
OUTPUT
REGENERATE
Colors shows data, neuron and

Epoch 000,494

Learning rate 0.1

Activation Sigmoid

Regularization None

Regularization rate 0

Problem type Classification

Ratio of training to test data: 50%

Noise: 0

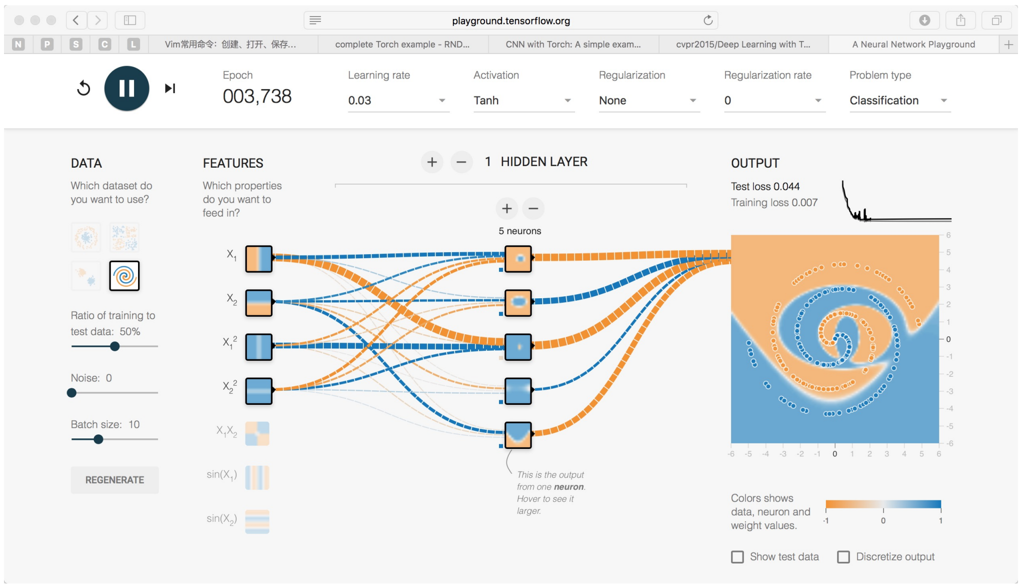
Batch size: 10

0 HIDDEN LAYERS

Test loss 0.000

Training loss 0.000

Figure 8: For the third data sets and better performance


playground.tensorflow.org
A Neural Network Playground
DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
HIDDEN LAYER
OUTPUT
REGENERATE
This is the output from one neuron. Hover to see it larger.
Colors shows data, neuron and weight values.
Show test data
Discretize output

Epoch 003,738
Learning rate 0.03
Activation Tanh
Regularization None
Regularization rate 0
Problem type Classification
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
1 HIDDEN LAYER
5 neurons
Test loss 0.044
Training loss 0.007

Figure 9: For the forth data sets

## 4 HW3.4

For this problem, firstly, I design a model to predict the digit in the image. Then, I design a XOR model to tell if the result from both image is the same. The structure is shown in figure10.

The loss curve of the classification model is shown in figure11.

In my CNN, I use SpatialConvolution units,SpatialMaxPooling units, linear units and tanh units, because its an classification problem, I also add an LogSoftMax unit.

The accuracy of predicting the digits in the image is 0.93

The accuracy of predicting the matching of different image is 0.986

I guess this is happening because my data sets is not big enough and the possibility of selecting two image with exactly the same digit is low. If I have a large data set, I believe my CNN can do better.

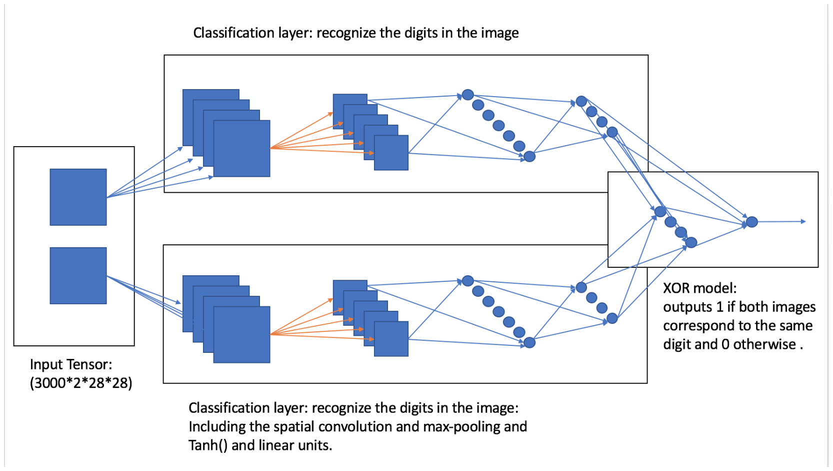

Classification layer: recognize the digits in the image
Input Tensor:
(3000*2*28*28)
XOR model:
outputs 1 if both images correspond to the same digit and 0 otherwise .
Classification layer: recognize the digits in the image:
Including the spatial convolution and max-pooling and Tanh() and linear units.

Figure 10: Architecture of my CNN

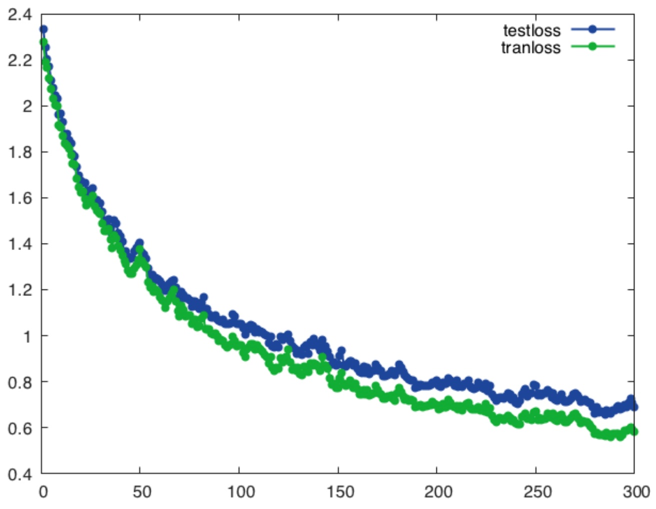


Figure 11: Loss of classification model10 classes