Name: 曾钰城

No: 1173710105

1. Environment

Operating System: ubuntu 18

Language: Python 3.6

JetBrains PyCharm Community Edition 2019.2.2 x64

IDE: JetBrains PyCharm Community Edition 2019.1

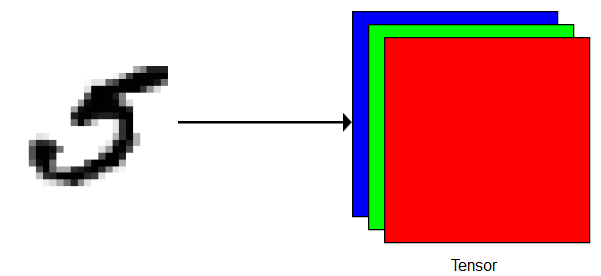
Tools Library：Pytorch, Cuda, Numpy, Visdom

1. Process data
2. download mnist dataset

download url: <http://yann.lecun.com/exdb/mnist/>

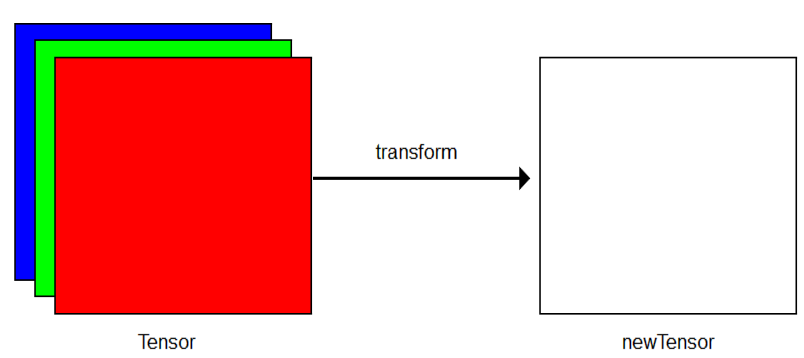
1. convert image to tensor

Original image consists of 28\*28 pixels which consists of RGB. RGB ranges from 0 to 255. Original image can be represented as a 3\*28\*28 tensor.



1. simplify and normalize tensor

3\*28\*28 tensor is not easy to compute. Because most of the pixels in the image are white or black, 3\*28\*28 tensor can be reduced to 1\*28\*28 tensor without losing information.



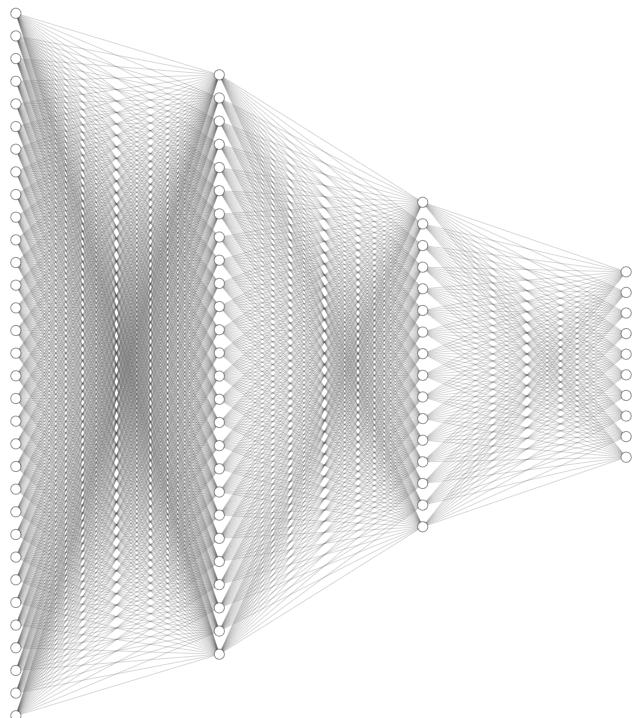
transformation function:



1. dataset divide into train set and test set

7500 for training and 2500 for testing

1. Deep Learning Model--Multi-Perception Model
2. model framework











1. function design

The first linear function: 

The second linear function: 

The third linear function: 

The first activation function: 

The second activation function: 

The third activation function: 

1. forward function



1. backward function

Too complicated, let pytorch do it

1. loss function

We assume the neural network output as q, , which can be view as a probability distribution. The i-th component of q is the probability that the input data is labeled i.

In the same way, we assume the true label of the input data as p, ，which also can be view as a probability distribution.

We can measure the distance between the two probability distributions by cross entropy, so we can choose cross entropy as lose function.

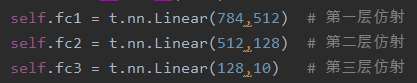


1. predict

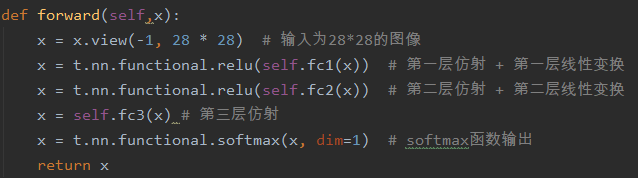
We assume the neural network output as q, . qi is the i-th component of q.



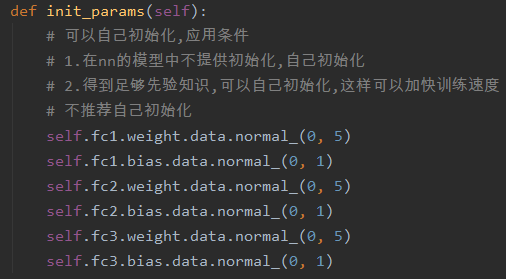
1. Coding
2. Linear function design



1. Forward function



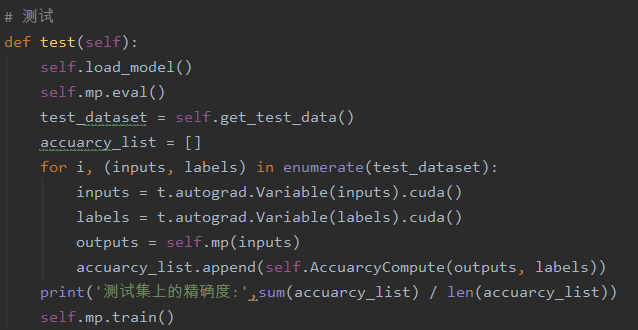
1. Init param



1. Train

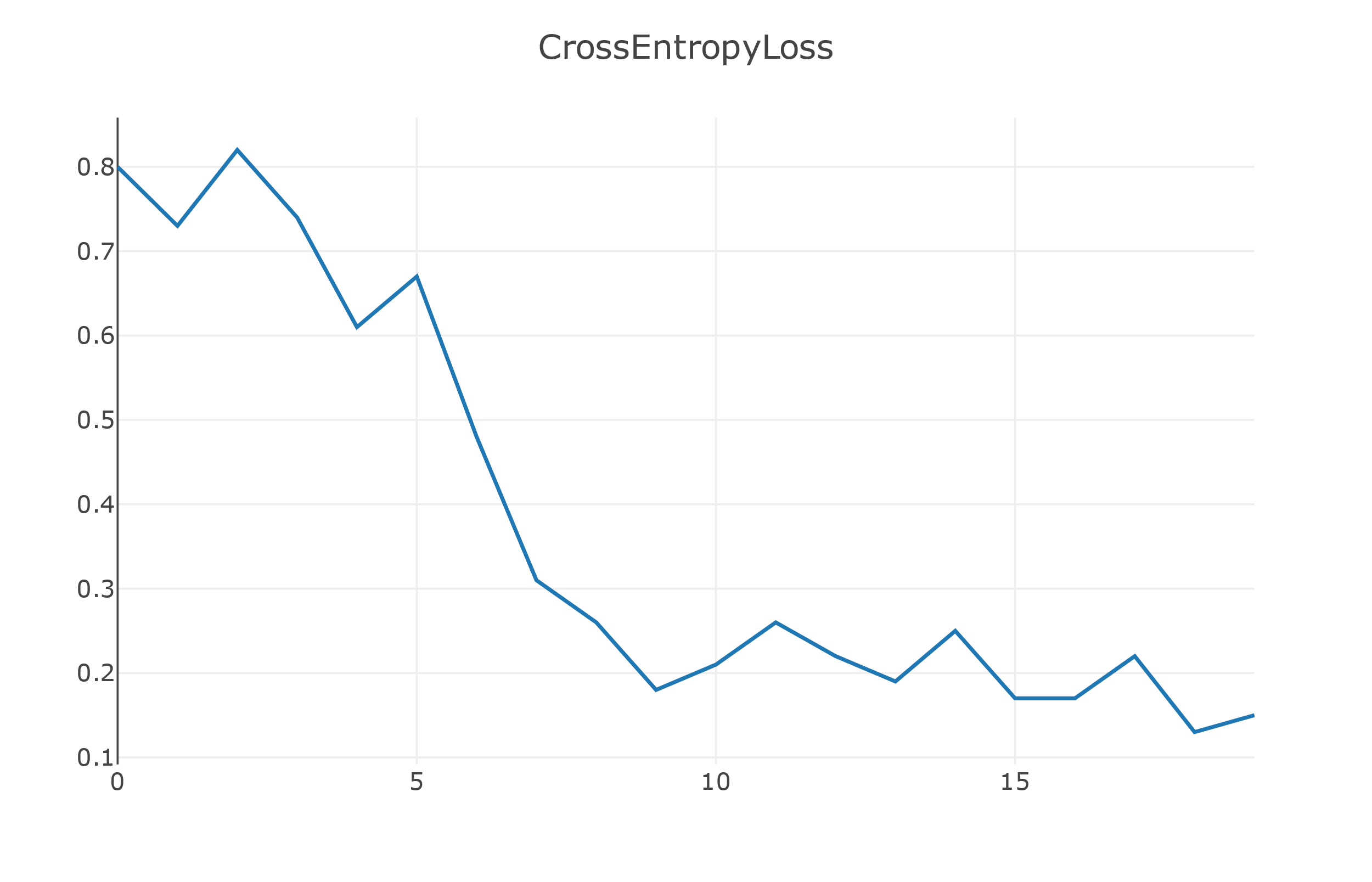


1. Test



1. Classify Original MNIST images

Cross entropy loss changes with the epoch



After 20 epoch, we get a model. Model accuracy on test set is 89.781%

1. Blur image
2. Blur image with Gaussian white noise

Mean = (14, 14)

Variance = [[14,0],[0,14]]

White noise point = 300

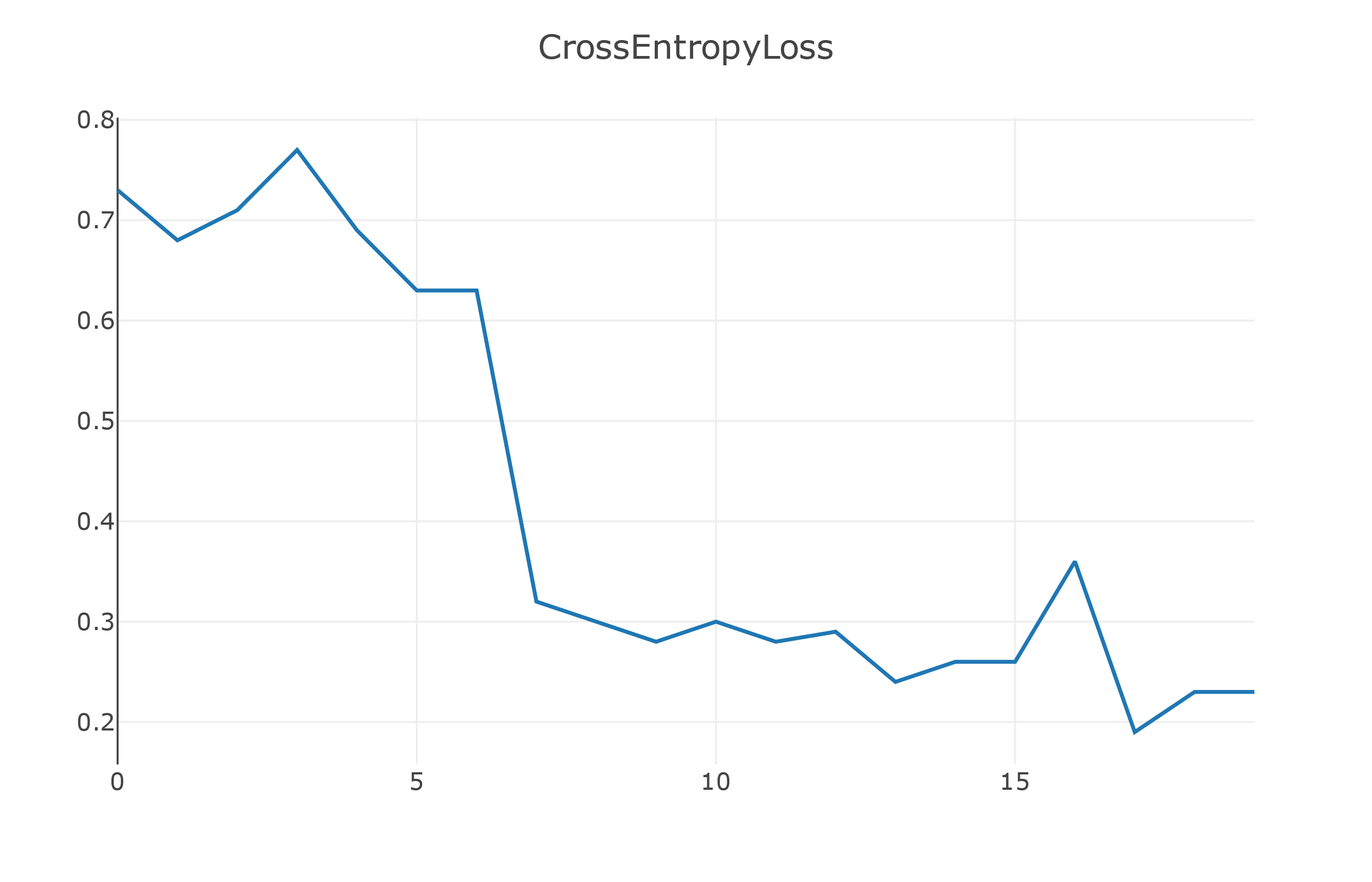


The first row is original image

The second row is blur image

1. Classify Original MNIST image

Cross entropy loss changes with the epoch

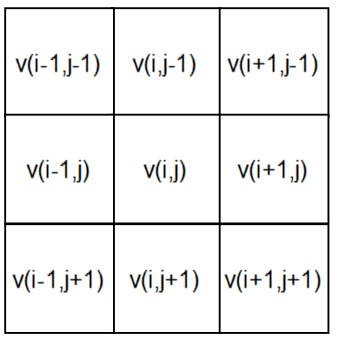


After 20 epoch, we get a model. Model accuracy on test set is 73.011%

1. Optimize the performance of classifying blur images
2. Gaussian filter

Gaussian filter is a linear smoothing filter suitable for eliminating Gaussian noise.

One pixel is equal to the weighted sum of its surrounding pixels.



Function:



1. Blur images after filter



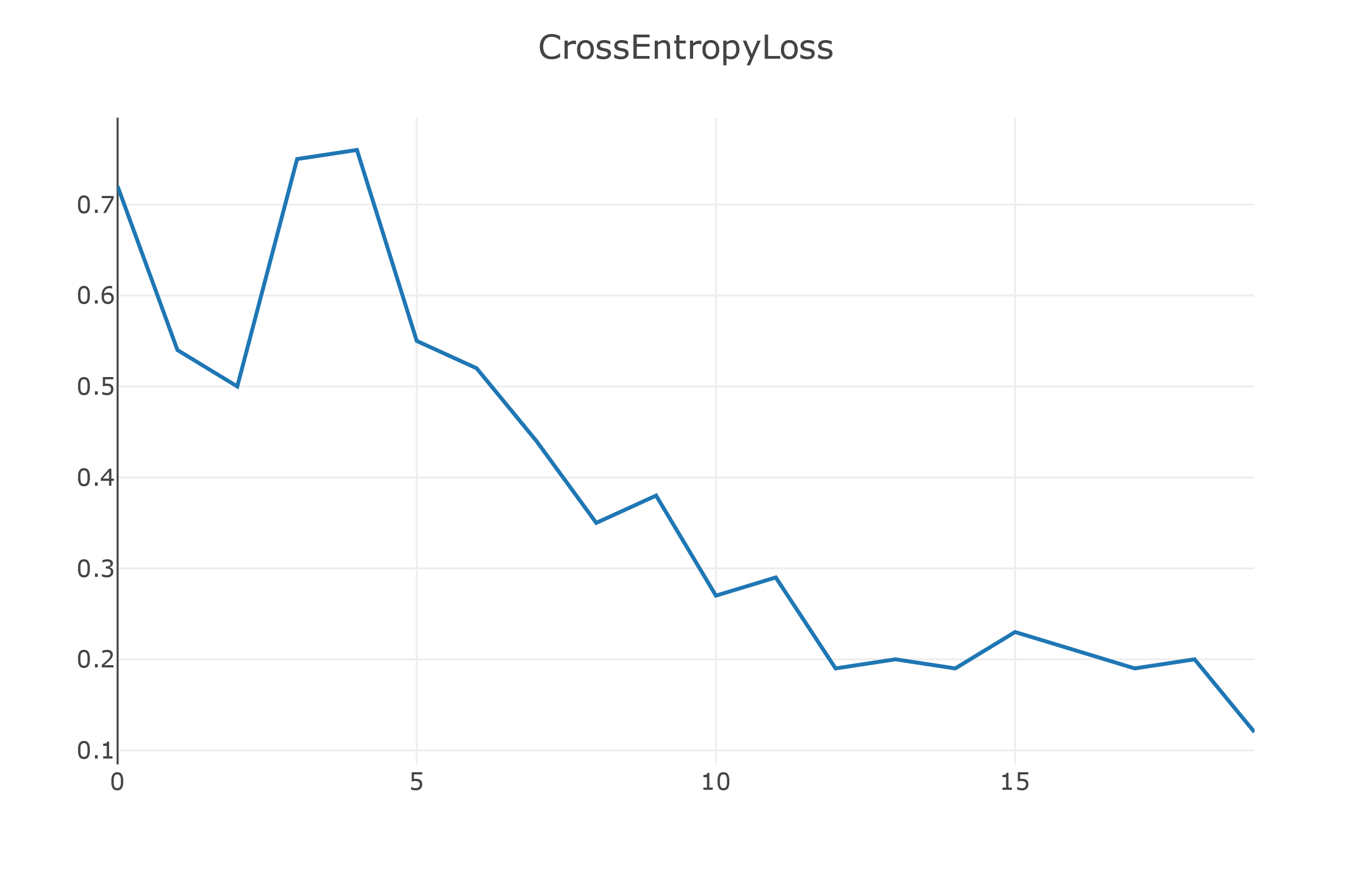
The first row is original image

The second row is blur image

The third row is filtered image

1. Classify filtered image

Cross entropy loss changes with the epoch



After 20 epoch, we get a model. Model accuracy on test set is 83.211%

1. The project has been submitted to Github

url：<https://github.com/1173710105/2020-summer-AIDB-Project>