2019 Spring COM526000 Deep Learning - Homework 3 Report

Convolutional Neural Network for Image Recognition

105061110 周柏宇

Problem 1

Please describe in details how to pre-process images because of the different resolution images in dataset and explain why.

I resize all the images to 128x128 for two reasons. One, it is easier to calculate the output size and it facilitate the design of the model. The other reason is that my GPU cannot handle too many images at the same time. After some experiments, I found out that bigger images doesn't boost the performance a lot, and smaller images makes the training session faster. Furthermore, I transform all the images to grayscale. It is because that there are RGB and grayscale images mixing in the dataset. In order to get accepted by the model, we need to unify the channel. And again by several experiments, training in grayscale doesn't kill much of the performance, and it also reduce the number of parameters to train.

Problem 2

Please implement a CNN for image recognition. You need to design at least two layers of convolutional layers and analyze the effect of different settings including stride size and filter size.

In my design, I construct two convolution layers, one dense layer and the output layer:

Convolution layer 1:

- Kernel size: 5x5
- Strides: 1 for horizontal and vertical.
- Zero-padding
- Activation function: ReLU
- Output feature maps: 32

Max pooling 1:

- Kernel size: 4x4
- Strides: 4 for horizontal and vertical.
- Zero-padding

Convolution layer 2:

Kernel size: 5x5

Strides: 1 for horizontal and vertical.

Zero-padding

Activation function: ReLU

Output feature maps: 64

Max pooling 2:

Kernel size: 4x4

Strides: 4 for horizontal and vertical.

Zero-padding

Fully connected layer 1(Dense):

Output units: 1024

Activation function: ReLU

Fully connected layer 2(Output):

Output units: 101

Activation function: Softmax

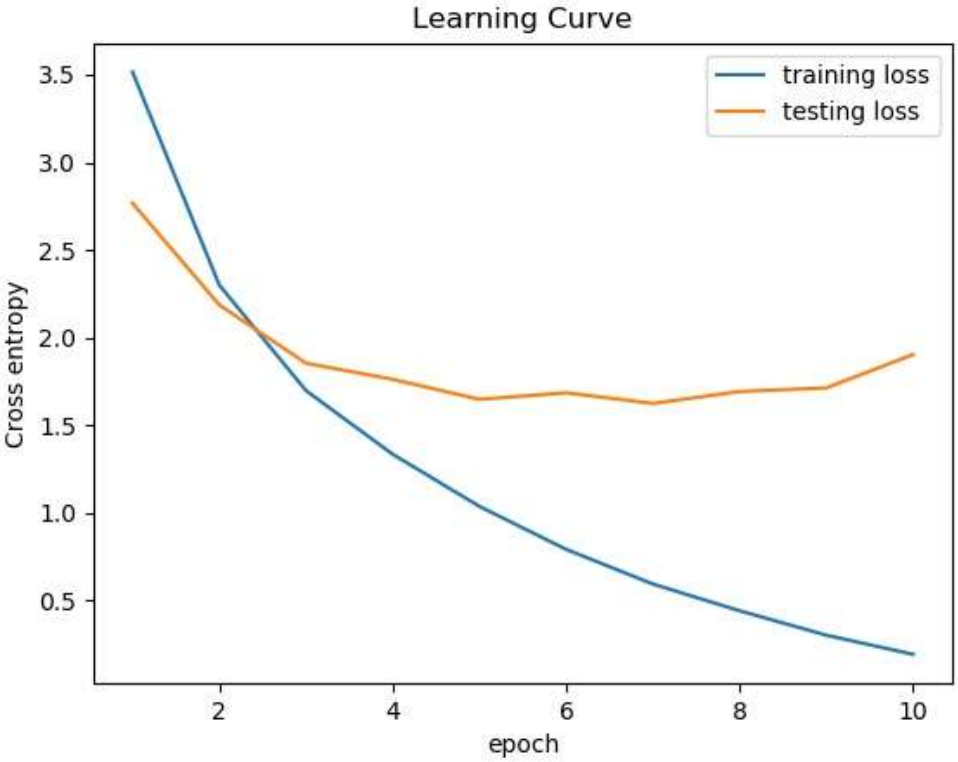
Optimizer: Adam, with learning rate 4e-4.

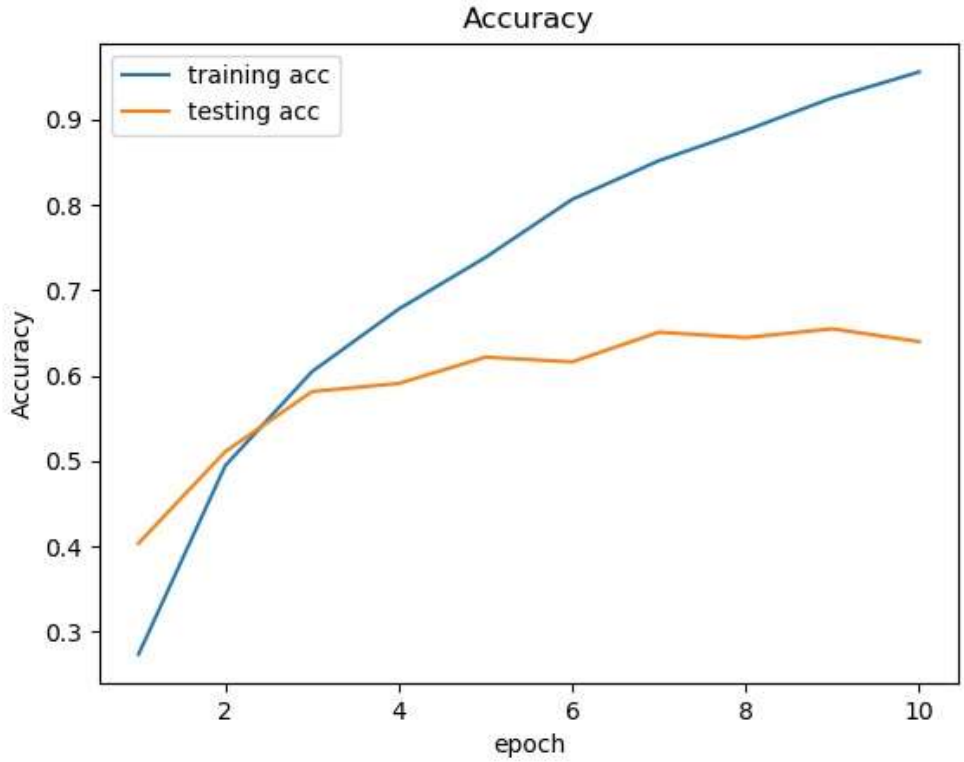
Epochs: 10

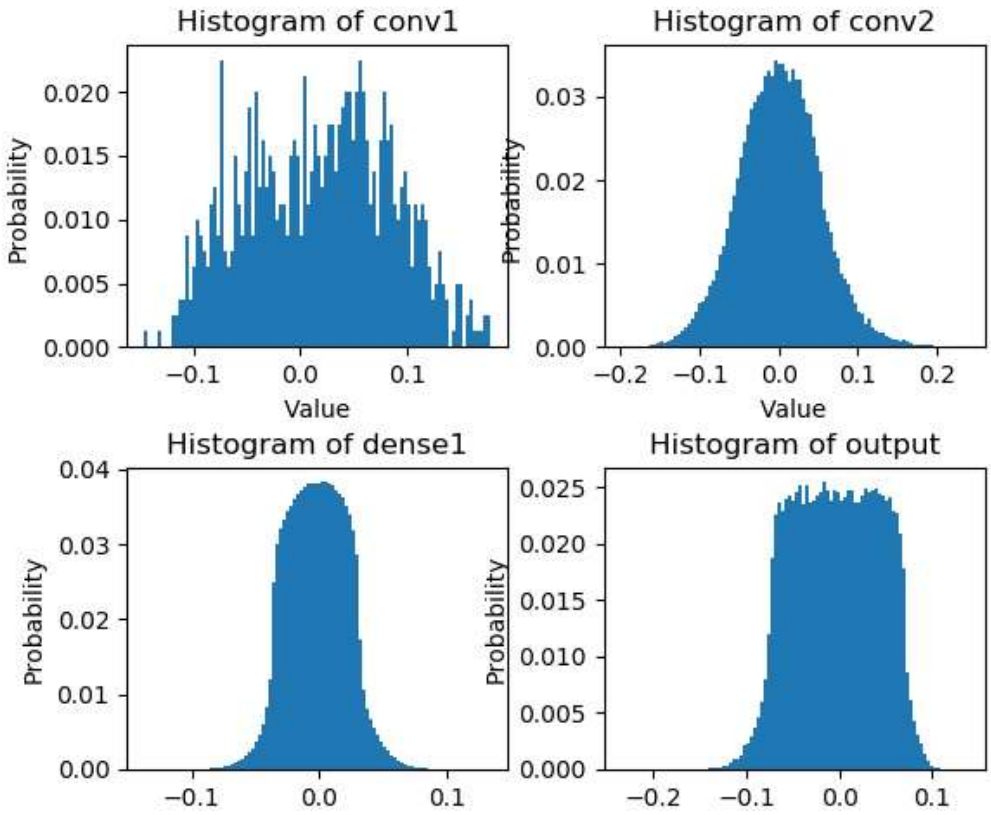
Batch sizes: 64

Problem 3

Plot the learning curve, accuracy rate of training and test sets as the example found in Figure 1, and Figure 2. Plot distribution of weights as illustrated in Figure 3.

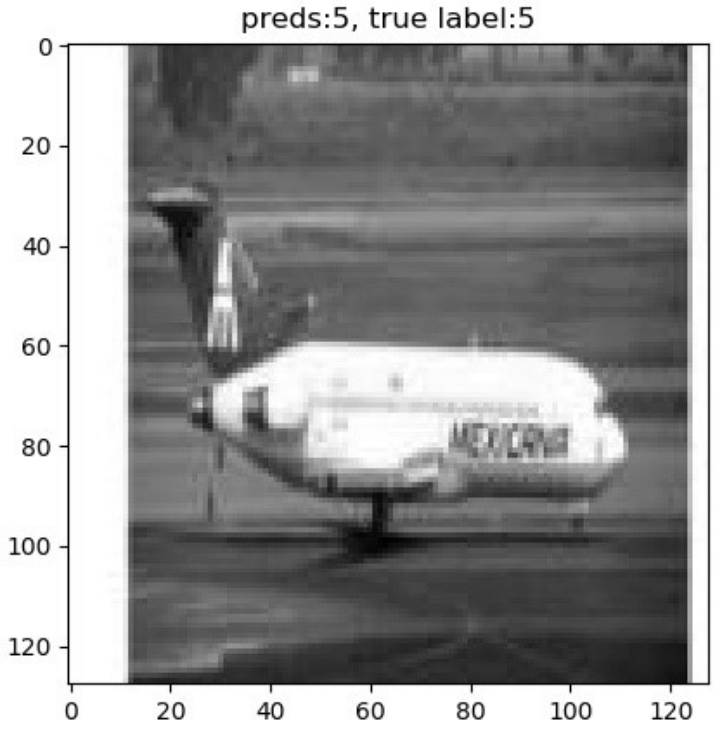

Learning Curve
training loss
testing loss
Cross entropy
epoch
3.5
3.0
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10

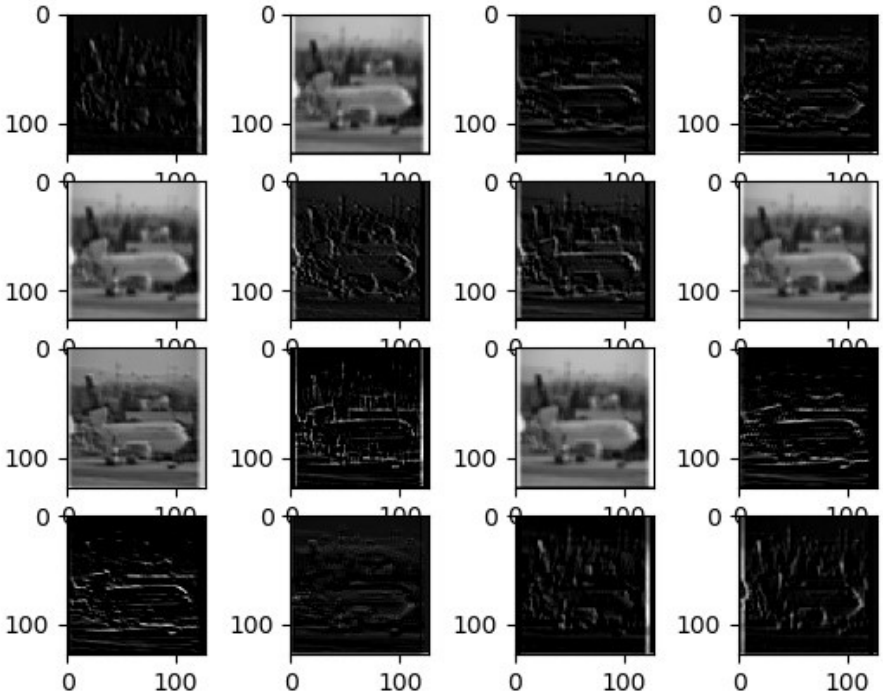

Accuracy
training acc
testing acc
Accuracy
epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2
4
6
8
10



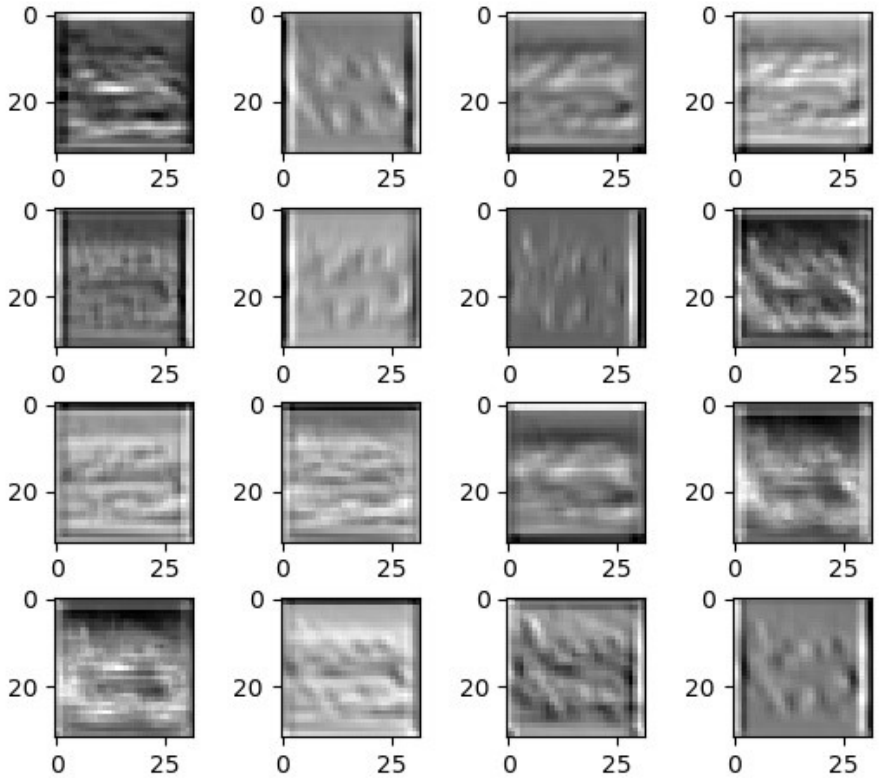
Problem 4

Please plot activations of the first, and second convolution layers as illustrated in Figure.5, Figure.6. Please also plot the corresponding image with your prediction and label (see Figure 4) and explain what you observe.





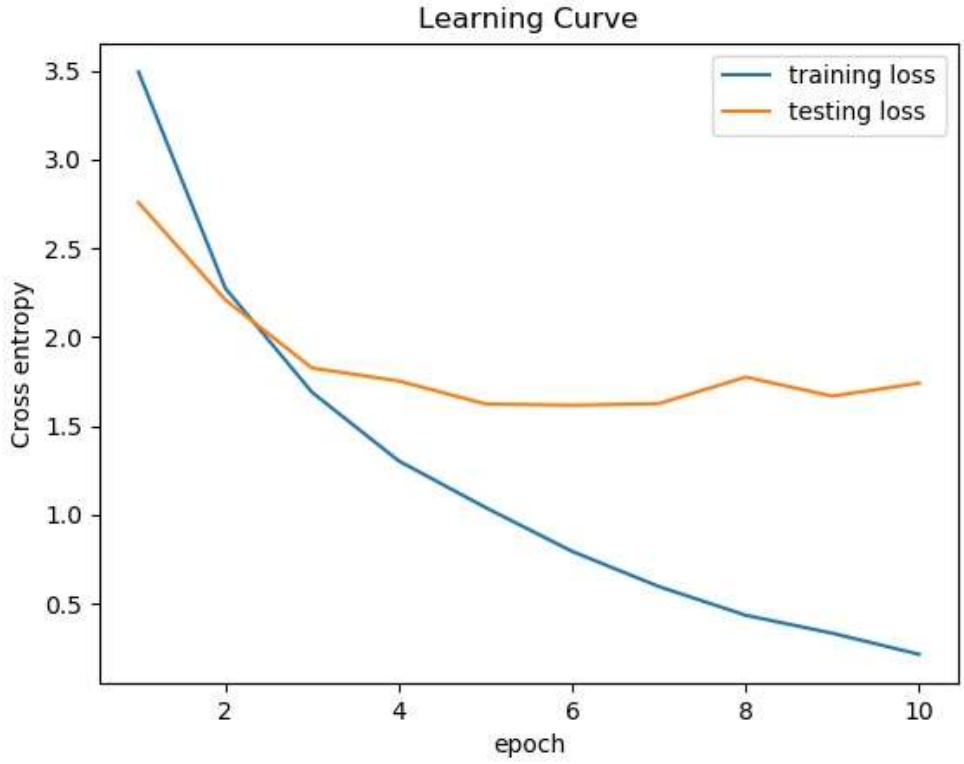
In the first activation layer, we can see the contour of the airplane in some of the plots. Other plots is not so intuitive to interpret.

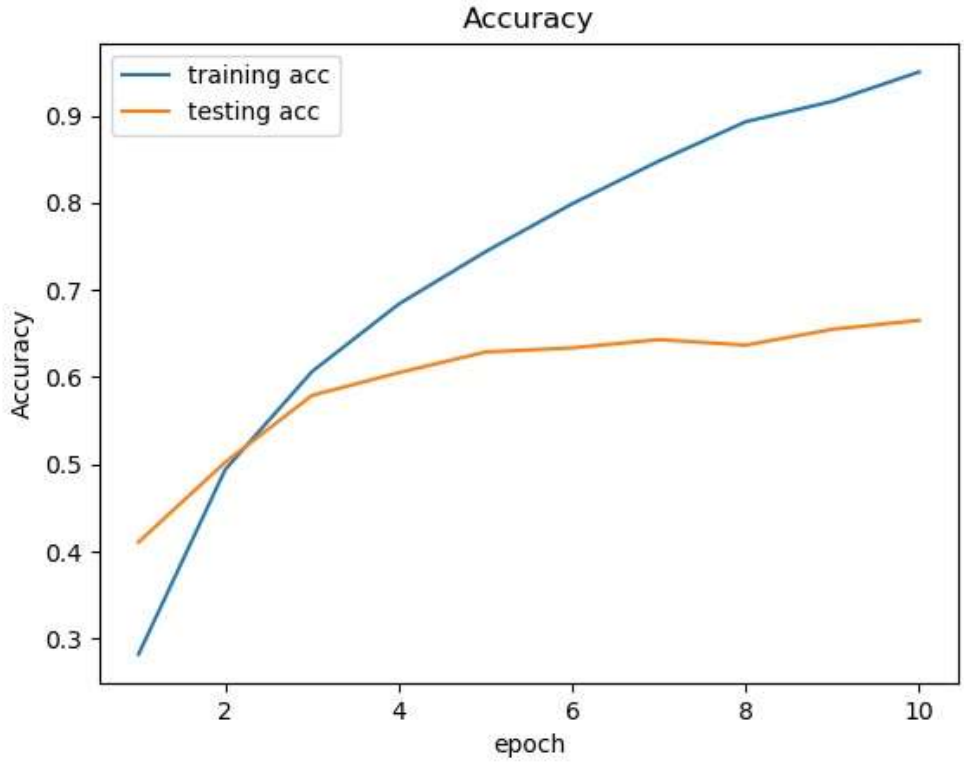


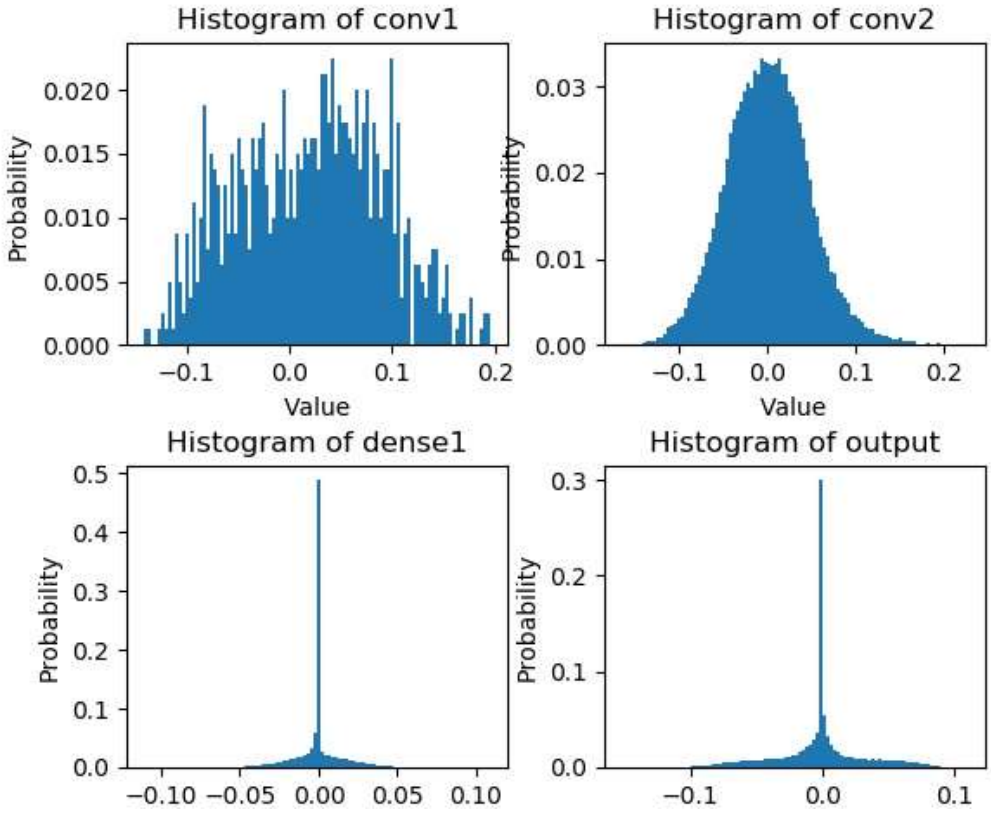
In the second activation layer, the result is a lot different from the target image. But I do notice that all the plots seem smoother than those from first activation layer.

Problem 5

Repeat 3 and train a CNN with L2 regularization. Discuss about the difference between with and without regularization.


Learning Curve
training loss
testing loss
Cross entropy
epoch
3.5
3.0
2.5
2.0
1.5
1.0
0.5
2
4
6
8
10


Accuracy
training acc
testing acc
Accuracy
epoch
0.9
0.8
0.7
0.6
0.5
0.4
0.3
2
4
6
8
10



In terms of performance, the boost is not so significant. But we can clearly see the effect of the regularization term. Especially for the fully connected layers' parameter, most of them are very close to zero. I choose the regularization constant to be 5e-4 and I found it very hard to be fine-tuned. I can use bigger regularization constant to make the convolution layers' parameters closer to zero, but the model is usually under-fitting, which kills the accuracy.

Problem 6

Please give three examples of ways to solve overfitting and try to use them to improve your model, e.g. using data augmentation, batch normalization, and dropout.

For data augmentation, I use 3 methods:

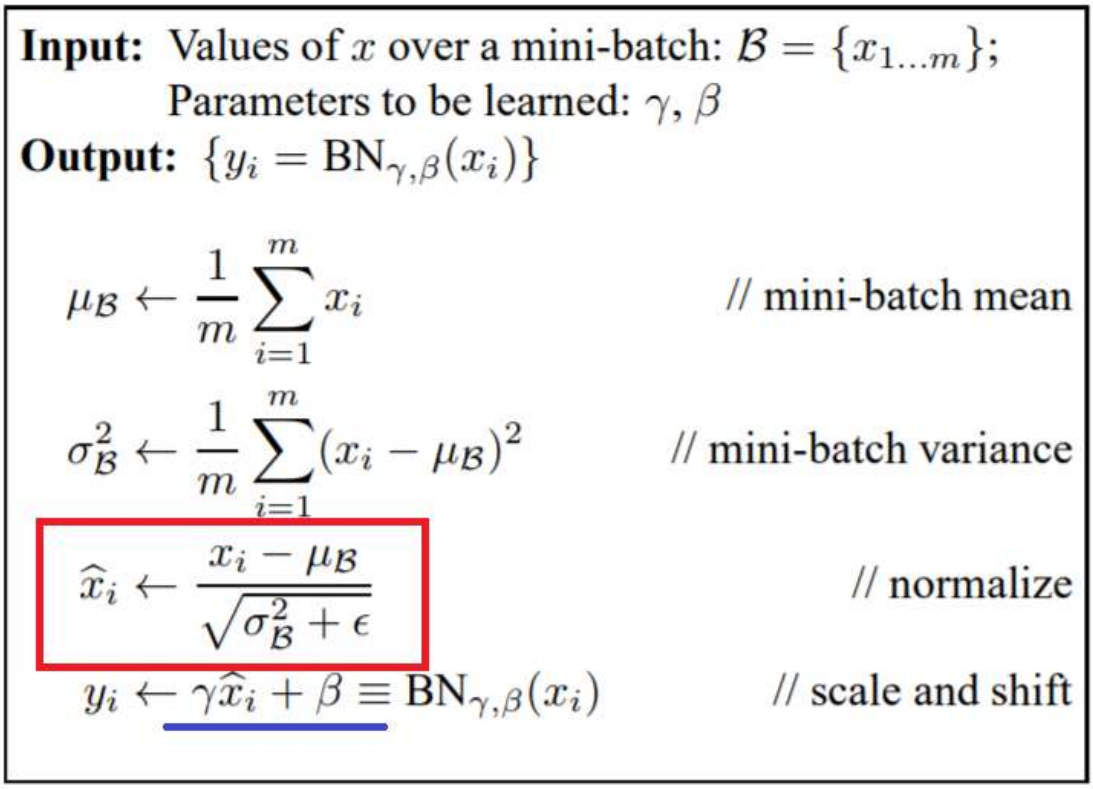
Flip: the left-to-right flip

Rotate: rotate the picture from degree -90~+90 uniformly randomly.

Noise: Gaussian noise with zero mean and 10 variance or salt-and-pepper noise with probability 0.01. The two types of noise are chosen uniformly randomly.

Batch normalization:

Use the formula:



Drop-out layer: It is already incorporated in the building of the model. For the previous problems, I simply set dropout_rate to 0. In this case, I set dropout_rate to 50%.