DeepX Assignment

WANG BIYUAN

1 Deep Learning Programming

Network Architecture

As it required in the Task Specification, I constructed two networks, convolutional neural network (CNN) and multilayer perceptron (MLP) respectively to perform the image classification on CIFAR-10 database.

I built both networks in a small scale in order to train them on CPU. The detailed structures are shown in Figure 1 and 2:

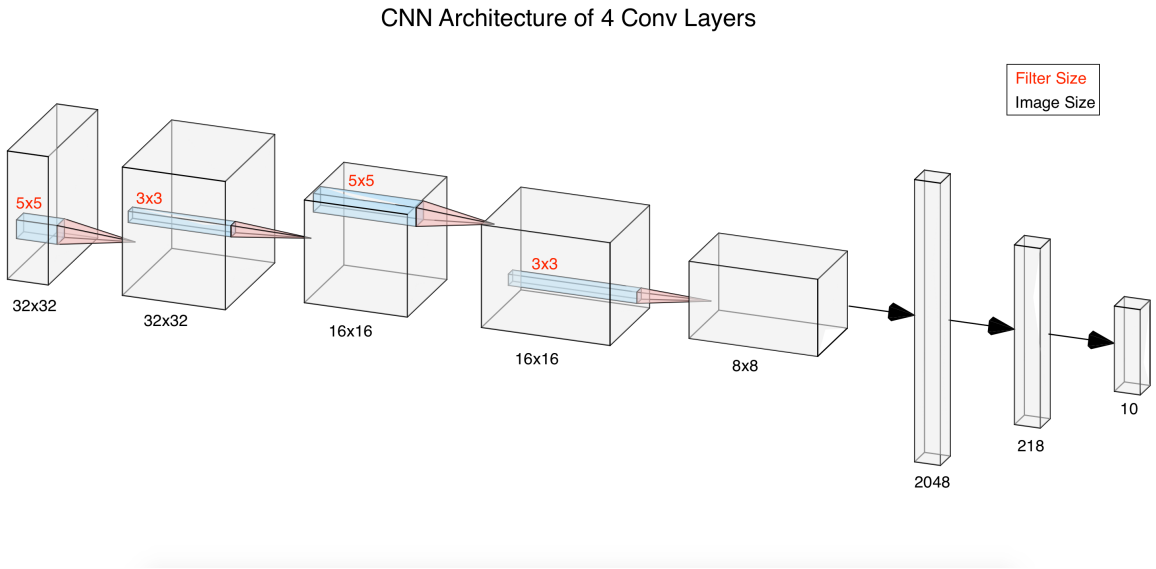

Figure 1: The structure of CNN contains 4 convolutional layers and two fully connected layers. The input is natural images from CIFAR-10 database of size 32x32x3. The output is the predicted probability of each image belongs to one of the ten classes.

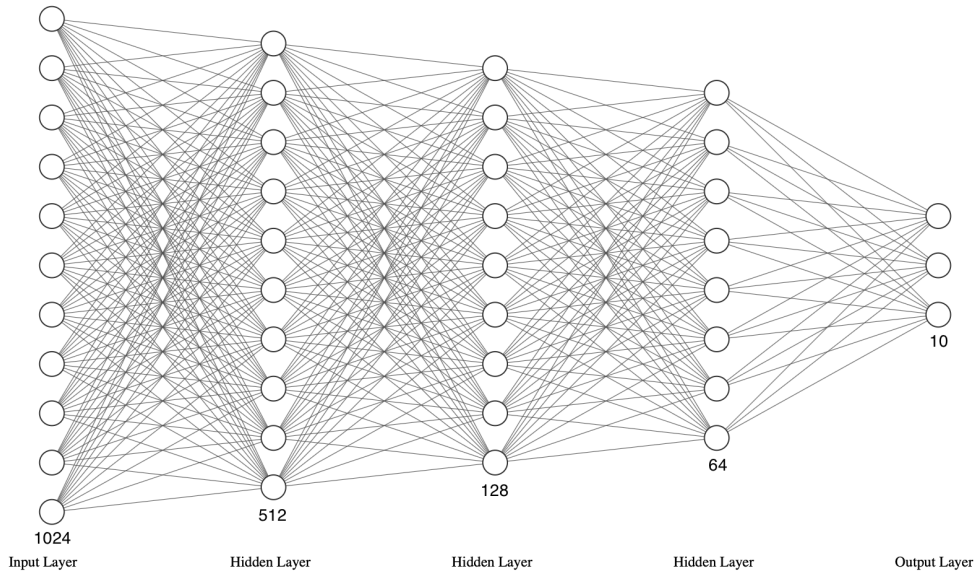

Figure 2: The structure of MLP contains 5 fully connected layers. Each neuron is using ReLU activation function to detect the feature which mostly stimulates the neuron. The input is also natural images from CIFAR-10 database of size 32x32x3. The output is the predicted probability of each image belongs to one of the ten classes.

Training and Testing Results

The training dataset (50000 images) is divided into many batches, I chose 128 as the batch size therefore there are 391 batches in total. In each epoch, all the 391 batches will be trained.

CNN model has different numbers of filters with different size in each convolutional layer. Each filter will do convolution on each pixel and the output of each layer will be processed by activation functions. The training and testing results of CNN model are shown in the following figures:

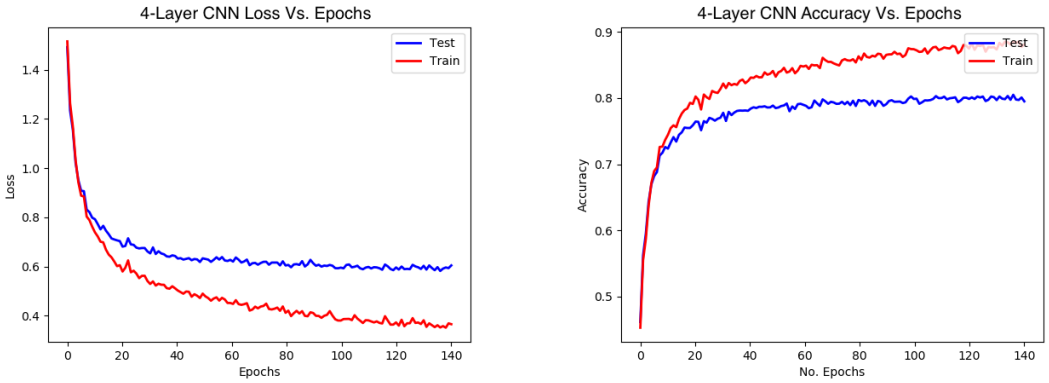

Figure 3: The Loss Vs. Epochs of CNN model shows the loss in both training and testing through 140 epochs. Although the training loss keeps decreasing even to the end of training, the testing loss seems reach to plateau from about 45 epochs onward. The accuracy Vs. Epochs plot has similar trend as the loss plot. When the training accuracy has reached to 90%, testing accuracy is still remaining around 80%. This shows a bit overfitting of the model.

The hyperparameters I used for this CNN network is shown in the following table:

Batch Size	Learning Rate	Epochs	Augmentation
128	0.001	140	None

When I tuned my hyperparameters for the first time, I set learning rate to 0.01 but the accuracy for testing kept at a very low percentage. Then I tried a smaller value 0.001 and the results improved a lot. Comparing to CNN, MLP has worse performance of the image classification even though it used way more epochs to train the network. The hyperparameter settings of my MLP model is shown in the following table:

Batch Size	Learning Rate	Epochs	Augmentation
100	0.001	300	None

The plot of loss and accuracy after each epoch of MLP is shown in the following figure:

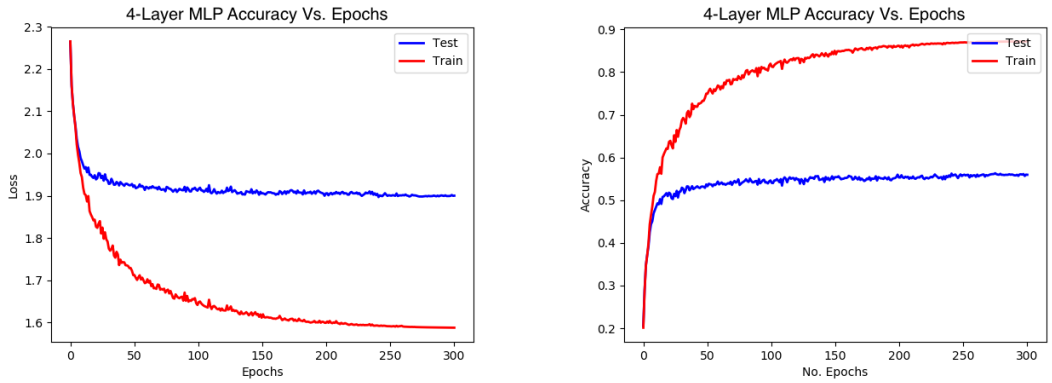

Figure 4: The Loss Vs. Epochs of MLP model with 4 layers of fully connected layers shows the loss in both training and testing through 300 epochs. The general trend is the same as CNN. It also shows the trend of overfitting and hard to improve the results on test set. Moreover, its performance is much worse than the CNN model.

After looking at the results of MLP, I doubted it was because there are not enough number of neurons. I trained another model by using the same hyperparameters but one more fully connected layer. The results are shown in the following figure:

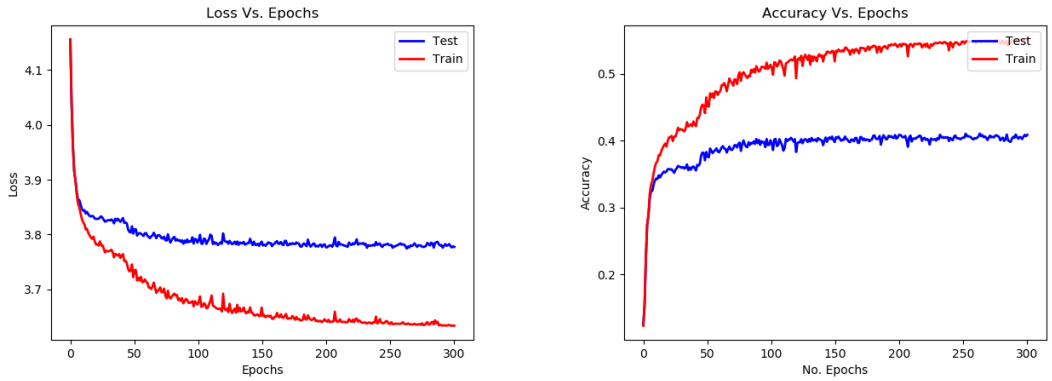

Figure 5: The Loss Vs. Epochs of MLP model with 5 layers of fully connected layers shows the loss in both training and testing through 300 epochs. However, this proved that simply increasing the number of neurons does not necessarily help improve the performance.

Data Augmentation

The limited image dataset can be augmented by simple image transformation such as rotation, crop, flip or change colour channels etc. I also applied some transformations just by setting the transformation parameters in Pytorch. My augmentation settings include random horizontal flip, random rotation of 20 degrees in both left and right, adding colour jitter. The same plots of both models are shown in the following figures:

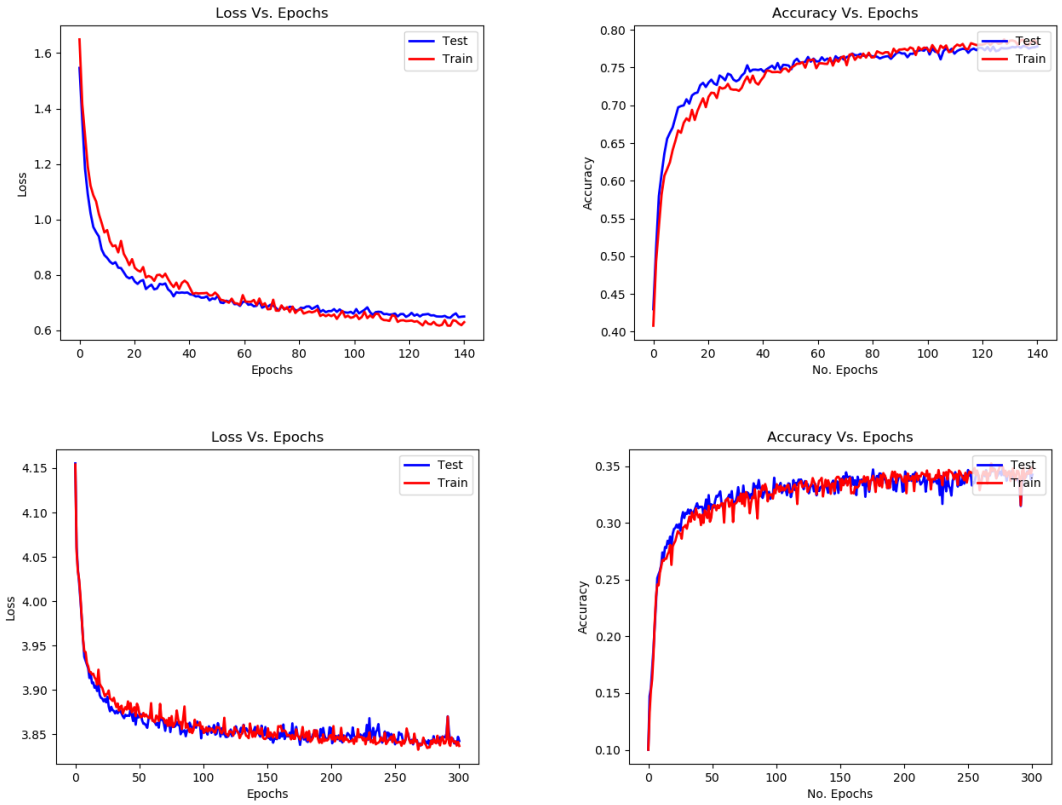

Figure 6: The actual accuracy and loss of both models did not improve obviously comparing to the one without augmentation. However, the difference between training and testing accuracy and loss have reduced a lot. There is not such big gap between the two curves as the plot without transformations.

In the augmentation experiments, all the other settings are kept the same before. In my case, the accuracy does not improved dramatically and it even seems remaining at the same level as before. However, the trend of overfitting reduced a lot and during the training process there were a lot of oscillations. I think the improvement of generalization is due to the augmentation of database which provide more samples for training but due to the structures of the models, there were not enough filters and neurons to learn more features of the images, thus the accuracy did not improve.