EE569 Digital Image Processing
Name: Shuo Wang
USC ID: 8749390300
Email: wang133@usc.edu
Date: Apr. 23, 2017

EE 569: Homework #4

Issued: 3/26/2017 Due: 11:59PM, 4/23/2017

Problem 1: CNN Training and Its Application to the CIFAR-10 Dataset (50 %)

(a) CNN Architecture and Training (15%)

The non-programming question

The convolutional neural network is one of the revised multilayer perception neural network. It can be used to construct the connection between the images and the classifier labels by training a series of input (images) and output (labels) pairs. As a result, we can use it to do the classification for a group of pictures or justify the label of an image.

First and second questions: Describe CNN components in your own words: 1) the fully connected layer, 2) the convolutional layer, 3) the max pooling layer, 4) the activation function, and 5) the soft max function; What are the functions of these components?

The CNN can be divided into several layers in order: convolution layers; max pooling layer; fully-connected layer; soft-max layer. Those layers have different functions.

1. Fully connected layer

The fully connected layer is a one-dimension layer, which is between the max pooling layer and the soft-max layer. For the fully connected layer linking to the max pooling layer, it can convert the 2-D spatial information into 1-D message by weighted sum of the all elements in the max pooling layer. For the other fully connected layer, it can decrease the dimension of the output until to the number of classification. The more layer the fully connected is, the more complex we can simulate by the neural network.

2. The convolutional layer.

The convolutional layer belongs to the image processing layer. In this process, we will use a series of the filters to do the convolution to the input image. For the RGB image, if the size of the filter is five, the number of elements for calculation of one pixel is $3 \times 5 \times 5 + 1(bias) = 76$. It can be used to extract the key elements including the edge and shape, which can make the image feature among different classes discriminant. The more dimension it is, the more features we can extract from the image.

3. Max pooling layer

The max pooling layer is to decrease the dimension of the image. At first, we partition the image into a group of 2×2 patches. For each 2×2 patch, we will find the maximum value which will be the final output for the pixel corresponding to the 2×2 patch. It can be useful to decrease the complexity of the network.

4. Activation function

The activation function can let us get the output result for each layers given a series of input areas. For

each node in a layer, a series of input numbers will be saved in the node. Then, the node will do the weighted sum for those numbers. Then, node will finally give the output by the activation function which can do the classification for a group of given input numbers.

5. Soft max layer

The soft max layer is a 1-D with 10 elements represent the 10 classes. If one element representing one class is the maximum among those elements, we can regard the input image as this class.

Third and fourth question: What is the major difference between a CNN and the traditional multi-layer perceptron (MLP)? Why CNNs work much better than other traditional methods in many computer vision problems?

When talking about the CNN, we have to discuss another network: MLP. Those two networks are similar, which has the fully connection layer, soft max layer. However, the major difference between the MLP and CNN is that the CNN has a series of convolution layers and max-pooling layers while the MLP does not have. As a matter of fact, the CNN is much better than other networks including the MLP just because of those two layers.

Take the texture classification for example. By the convolution calculation, we can extract the specific and discriminant features while other network cannot make the discriminant among different classes clearly. In addition, the CNN can use the max-pooling to make the calculation of complexity much less than MLP whose complexity is really huge. In addition, the decrease of the complexity requirement can enable us to apply more fully connected layers in order to deal with more complex classification.

Fifth question: Explain the loss function and the classical backpropagation (BP) optimization procedure to train such a convolutional neural network.

The training process can be written as follow:

1. Initialize the weight for each node. Then use the train data to do the calculation from the input to the output.
2. Calculate the error between the actual output and the desired output by the loss function.
3. Put the error value back to each node. For each node, the error value will multiply its input value, which represents the gradient of loss function (Backpropagation).
4. Use the gradient to update the weight number for each node.
5. Repeat from 1 to 4 steps

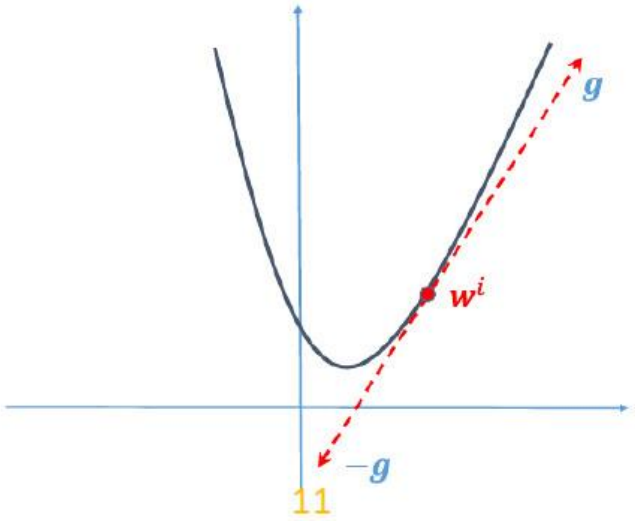
In the process above, we will use the BP (Backpropagation) to make the weight numbers which is used to do the classification optimal so that we can make the classification with little error. In addition, the loss function can represent the difference between the desired output and the actual output. The formula to acquire the loss value is shown as follow:

$$H(p,q) = \sum_{x} p(x) log q(x)$$

The less the value of loss function is, the more optimal the network model is. By the loss value, we can acquire the gradient of the loss value and use the function below to update the weight value for each node:

$$w = w - \eta \frac{dH}{dw}$$

As a result, we can make the weight number close to the local optimal result for classification, the theory can be shown as follow:



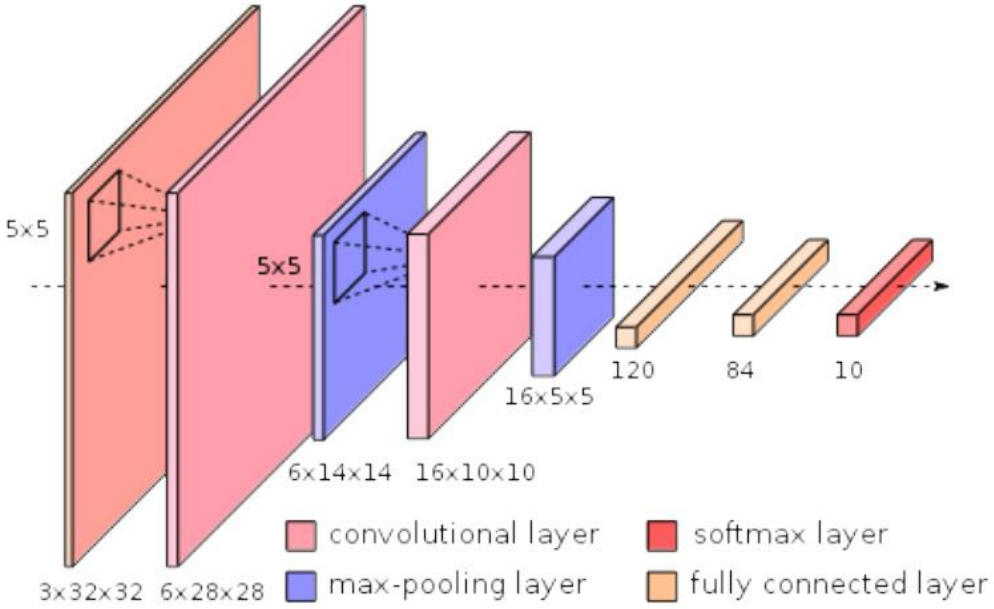
(b) Application of the CNN to CIFAR-10 Dataset (15%)

Motivation

In this section, I will choose the TFlearn to do the CNN network simulation experiment. The dataset is the CIFAR-10 image dataset.

Methods and procedures

A basic convolution neural network can be shown as follow:



At first, I will use the code below to extract the training and testing data into the system:

```
from tflearn.datasets import cifar10
(X, Y), (X_test, Y_test) = cifar10.load_data()
X, Y = shuffle(X, Y)
Y = to_categorical(Y, 10)
Y_test = to_categorical(Y_test, 10)
```

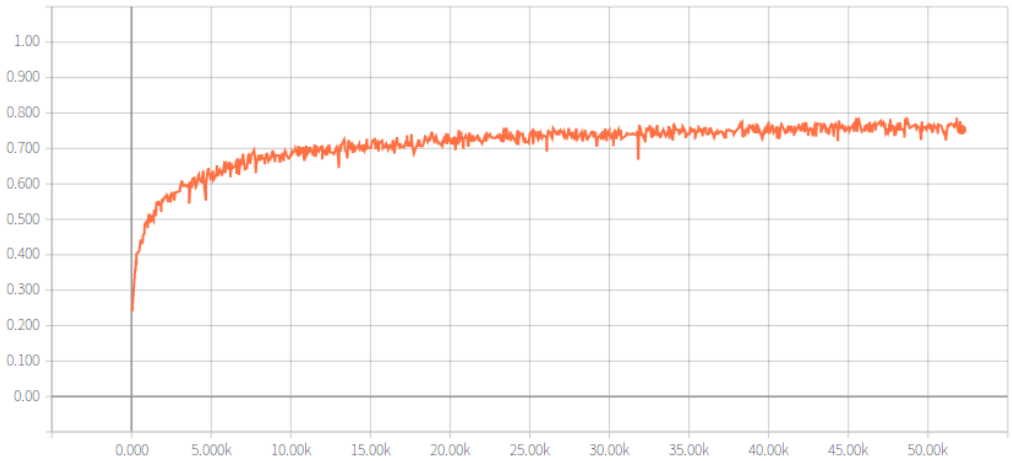
And then, I will build the convolution network.

1. Before entering the data, we can do some preprocessing. One is to randomly flip the image; Another is to randomly rotate the image, and the maximum rotation angle is 25 degrees.
2. Input the image data into the tensor.
3. Six 2-D convolution filters will be applied to extract the data message. For each filter, we will extract the 5×5 message whose center is the output pixel points for each RGB channel. In addition, we will use one bias number to coordinate the final output for each output pixels. As a result, the number of weight numbers for each pixel is $3 \times 5 \times 5 + 1(bias) = 76$. After calculating one output, we move the neighbor pixel, which means that the stride is one.
4. Use the maximum pool to make the size of the image smaller. At first, we can divide the image into a series of 2×2 patches. Then, for each patch, we regard the maximum number as the final output of pixels corresponding to the patches.
5. Use 16 2-D convolution filters to deal with the data. The size of filter is same, which is 5×5, while the depth of filters is different, which is 6. As a result, the final number of the weight numbers is $6 \times 5 \times 5 + 1(bias) = 151$. Also, the stride number is one.
6. The same maximum pool processing will be applied.
7. Build the fully connected layer which has 120 nodes. For each node in the fully-connected layer, we link it with all nodes in the matrix obtained from the step 5.
8. Decrease the fully-connected nodes from 120 to 84. Also, we link it with all nodes in the fully-connected layer obtained from the step 6.
9. Build the soft-max layer which has 10 nodes (the kinds of labels). Also, we link it with all nodes in the fully-connected layer obtained from the step 7. For the layer, the output is the location of the node with the maximum value.

For the initialization, I use the random initialization method to set both the bias and the weight number. In this question, the weight initialization is Truncated normal distribution with zero mean, and the bias is set zero.

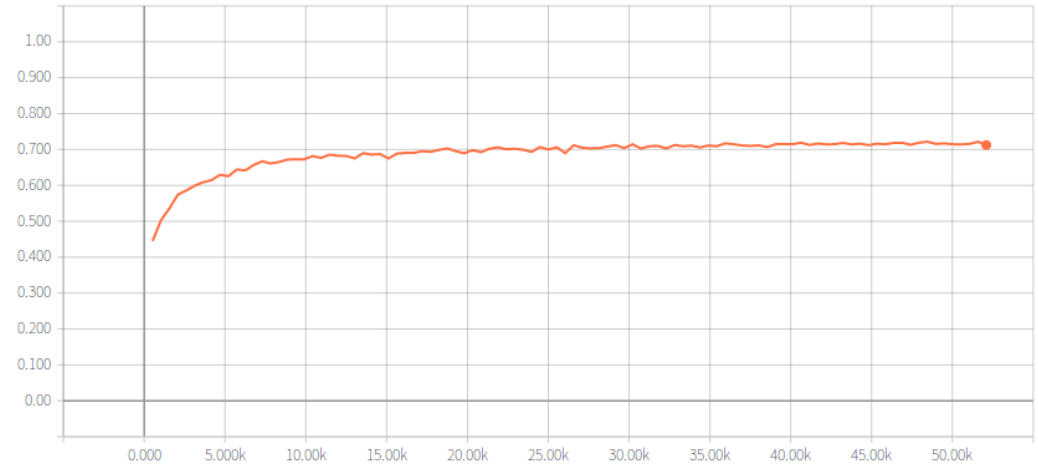
Experimental results

The training result above can be shown as follow (After 100 epochs):



From the result, we can find that the accuracy of the training process can reach 77%. Also, we can find that the increasing gradient of the training accuracy is smaller when the number of epochs increases.

The testing result of the network above can be shown as follow:



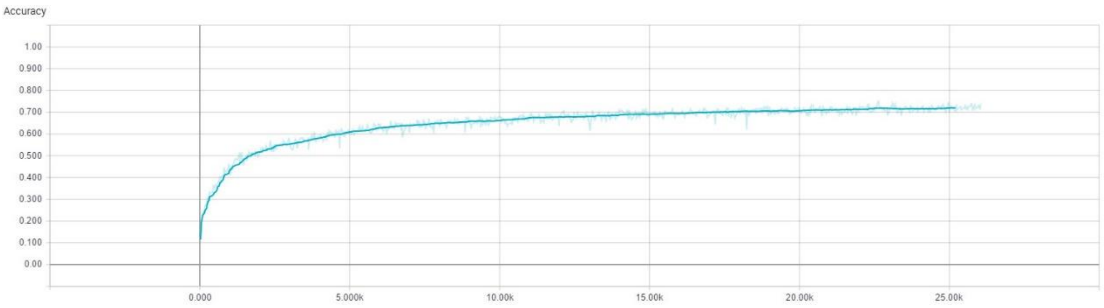
From the result, the testing accuracy is around 72%. Compared with the training accuracy, the testing accuracy is a bit smaller than the training one. In addition, the increasing tendency of the training and testing accuracy is similar.

Discussion

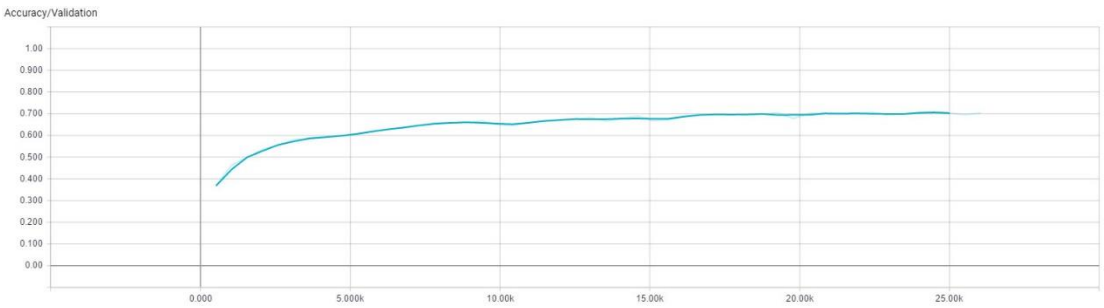
At first, we can discuss **preprocessing step and the random network initialization scheme**. In the question, I use the zero center and standard deviation normalization as the image processing method in order to make the training data without DC value, which can make the training process easier.

We can verify it by deleting those parameters.

The result of training when deleting the zero center and standard deviation normalization can be shown as follow:



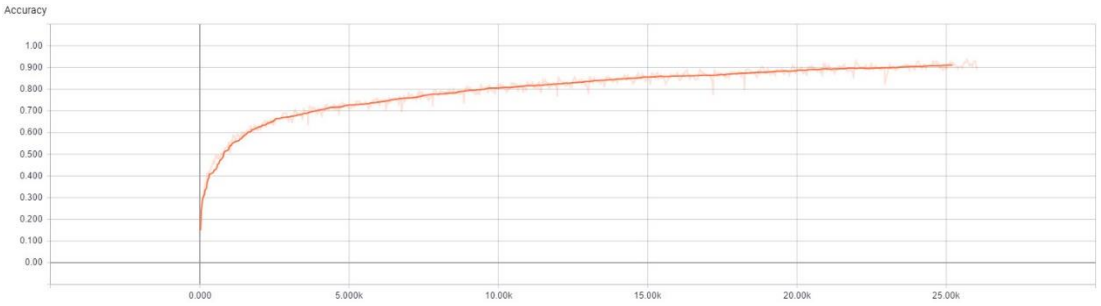
The result of testing when deleting the zero center and standard deviation normalization can be shown as follow:



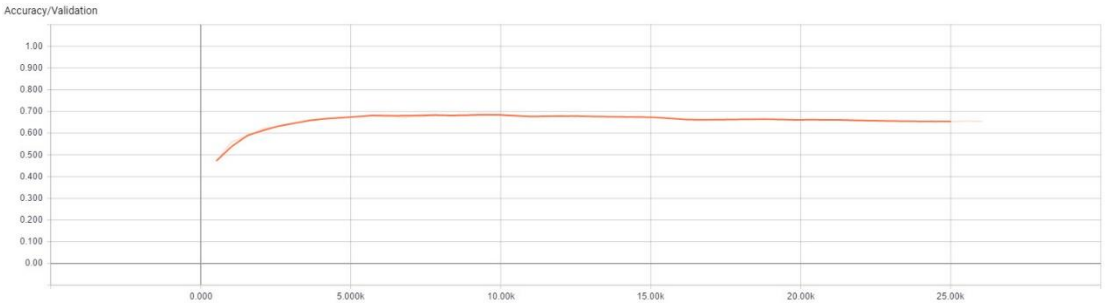
Therefore, compared with the original result in experimental result section, the zero-mean processing

can slightly improve the quality of training procedure significantly.

In addition, I tried the random flip and rotation process in the image augmentation step. By this method, we can get more flexibility on the input data, which increase the accuracy of the network.
The result of training when deleting the random flip and rotation can be shown as follow:

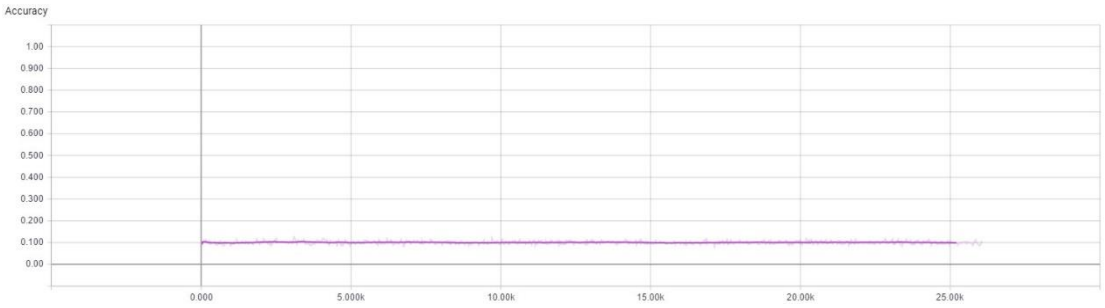


The result of testing when deleting random flip and rotation can be shown as follow:

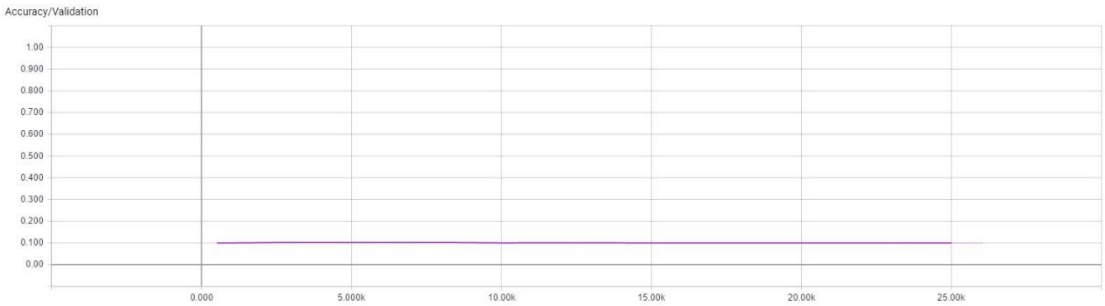


Therefore, compared with the original result in experimental result, we can find that the rotation and flipping processing can decrease the possibility of overfitting and prevent the accuracy from decreasing significantly.

Also, I tried the Xavier random distribution as the method to initialize the weight, which can get better result compared with the uniform random distribution. We can try it by processing the uniform random distribution.
The result of training when applying the uniform random number can be shown as follow:



The result of testing when applying the uniform random number can be shown as follow:

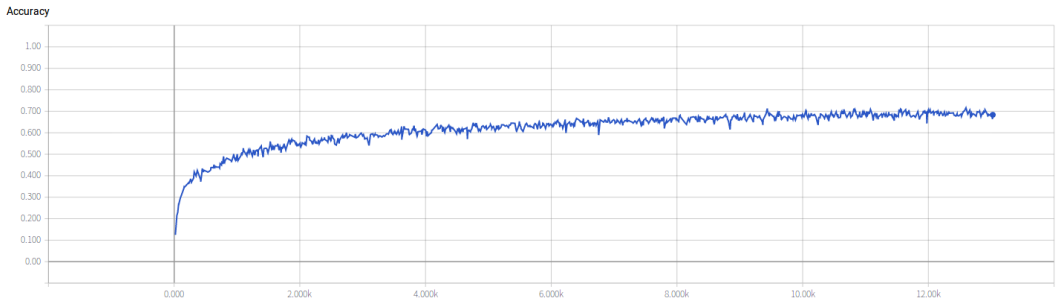


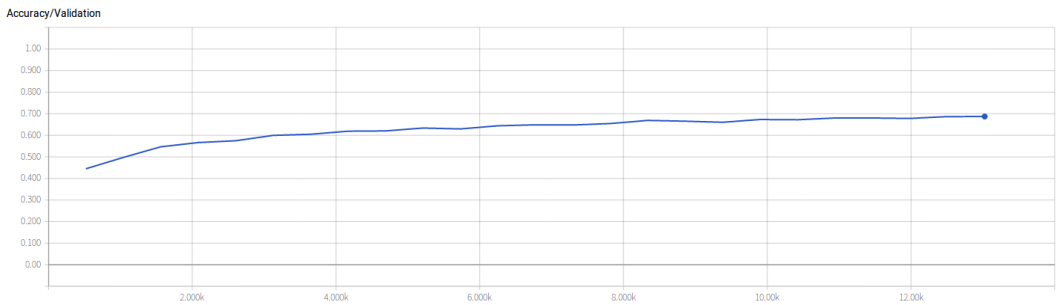
Therefore, compared with the Xavier result in experimental result, we can find that the Xavier random number can improve the accuracy of both training and testing significantly.

In addition, when using the python and tflearn to build the CNN, we can find that there are a series of parameters to change. Therefore, we can discuss the **effect of those parameters**.

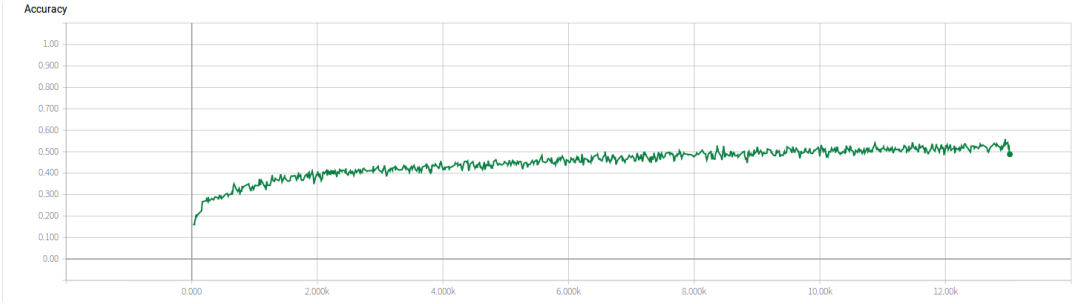
The first parameter is the learning rate for the network. In this question, the default learning rate is 0.001. In this question, we can evaluate different effect when applying different learning rate (The dropout is 1).

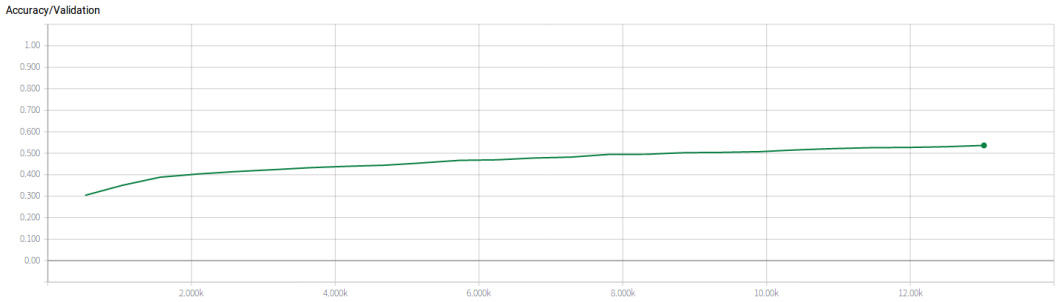
When the learning rate is 0.001, we can get the result (First is training accuracy, second is the testing accuracy):



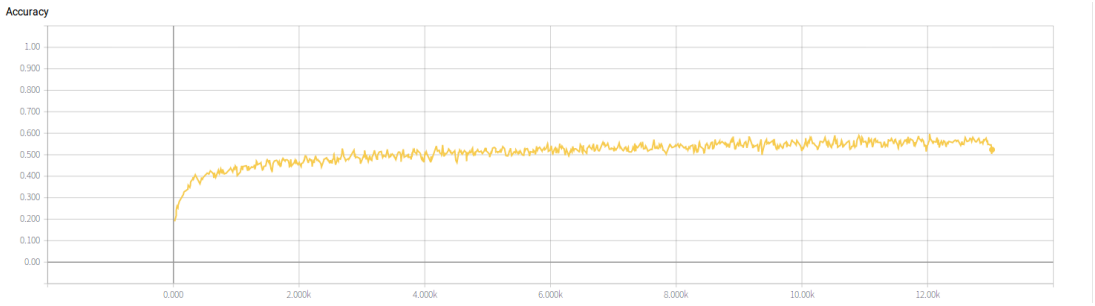


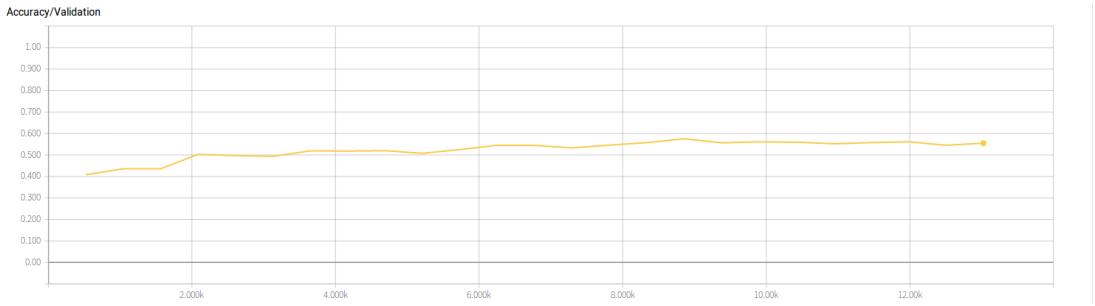
When the learning rate is 0.0001, we can get the result (First is training accuracy, second is the testing accuracy):





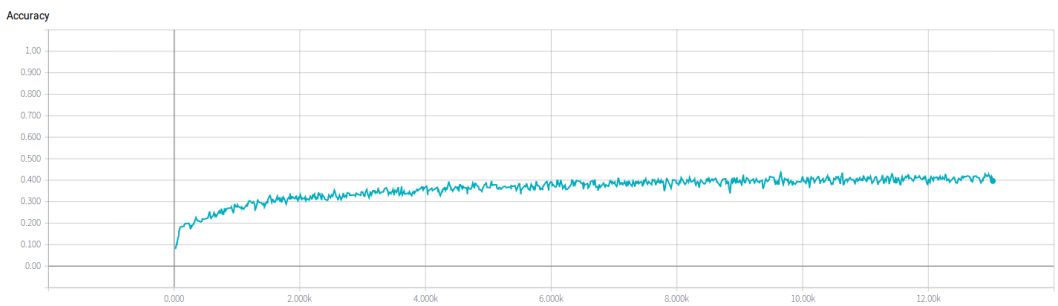
When the learning rate is 0.01, we can get the result (First is training accuracy, second is the testing accuracy):

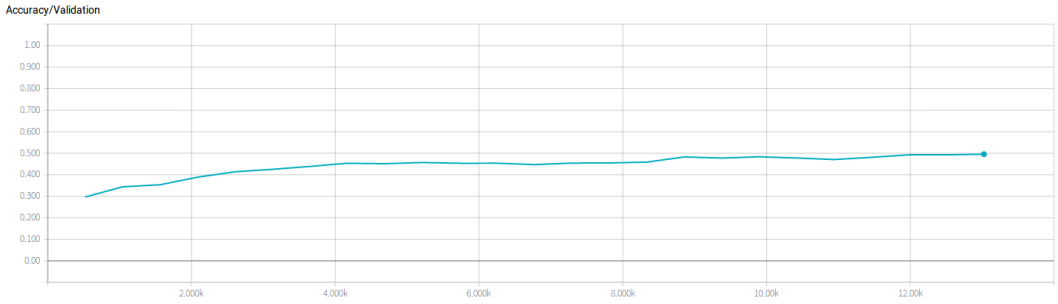




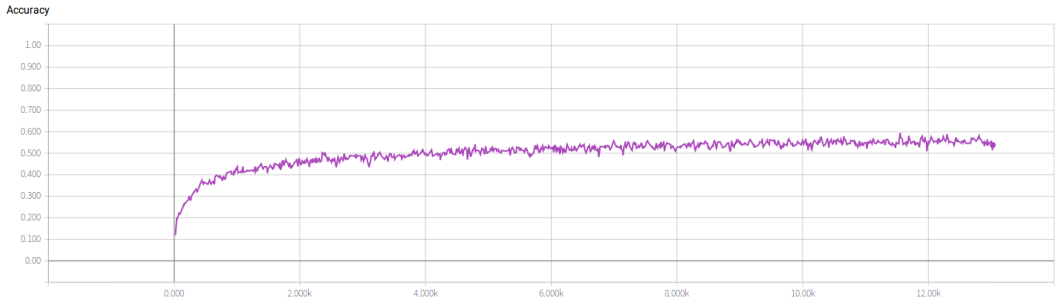
From the result above, we can find that the training rate should not be too large or too small since the accuracy for both the input and output is much more less than the default training rate.

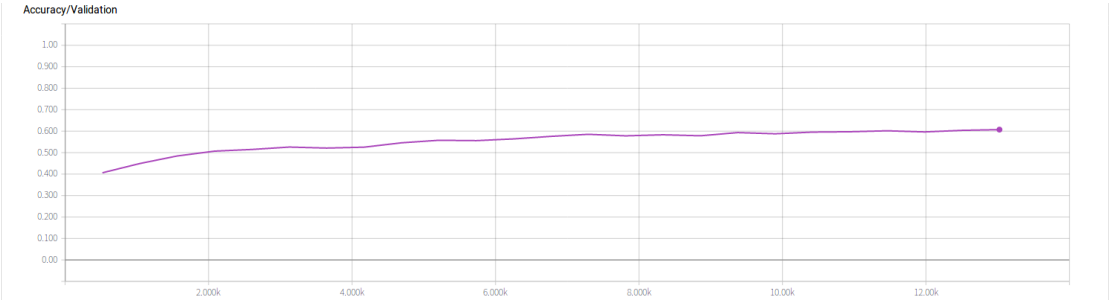
The another parameter is the probability of dropout, which is a float representing the probability that each element is kept. As a result, we can try three values: 0.2, 0.5 and 1 (the learning rate is 0.001). When the parameter is 0.2, the accuracy (first is training, second is testing):



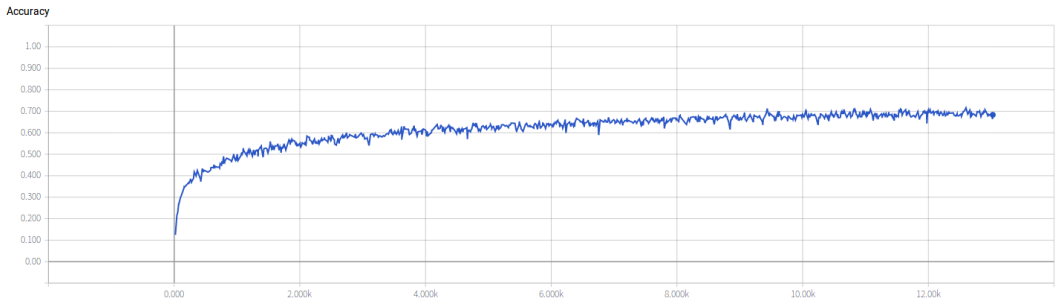


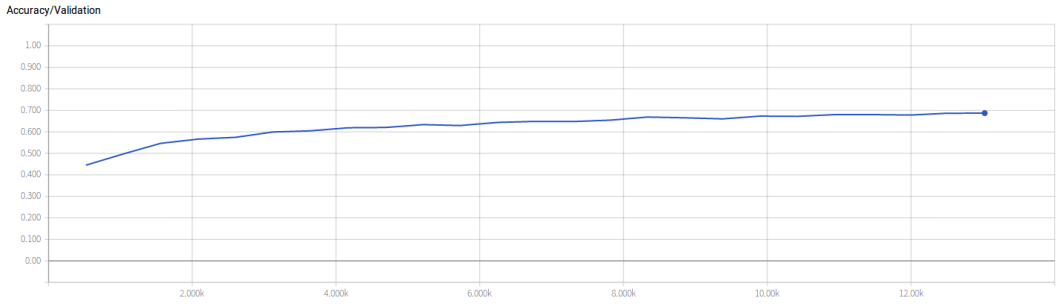
When the parameter is 0.5, the accuracy (first is training, second is testing)





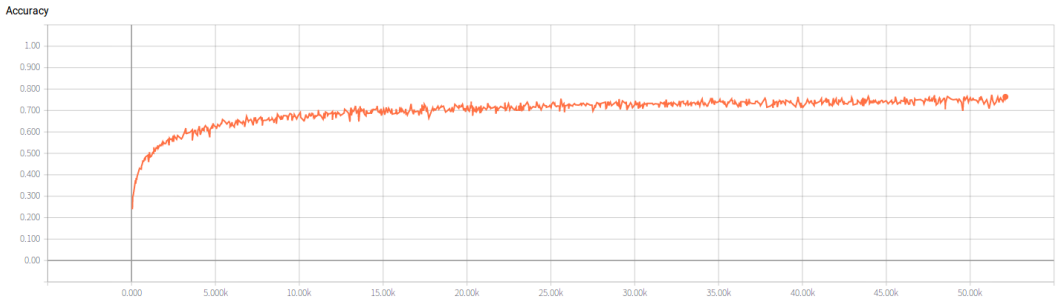
When the parameter is 1, the accuracy (first is training, second is testing):

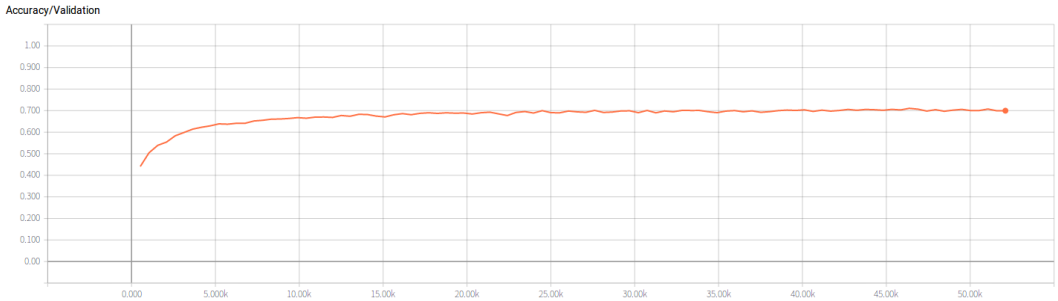




From the result, we can find that the larger the probability of dropout is, which is the probability to preserve the output, the larger the accuracy will be.

According to the discussion above, we can find that we should choose the default learning rate and the probability of dropout is 1. The result when running 100 epochs is:





(c) K-means with CNNs (20%)

Motivation

In the process of training, we should set the parameters for the edge between each linked nodes, which is usually conducted by setting a series of random numbers with some specific distributions. However, we can change such distribution to be a specific initialization according to the input images for training, which is the K-means solutions.

Methods and procedures

For the K-means solutions, we should acquire the input training elements of the SIFAR-10 firstly. Therefore, I use the opencv and the C++ to open the binary package of the CIFAR and acquire the image data. Then, we will do as follow:

1. According to the size of the convolution filter $(5 \times 5 \times 3)$ in the first convolution layer, I will acquire a series of patch whose size is the same as the size of the convolution filter $(5 \times 5 \times 3)$ and we regarded each patch as a sample.
2. For those samples acquires from all training data, I use the K-means to divide them into 6 parts and obtain 6 centroids. Those centroids can be regarded as the initialized parameter of the convolution layer.
3. Use the initialized parameters and the max pool, we calculate the input of the second convolution layer. Then, use the same process to get the initialized parameter of the second convolution layer whose size is $(5 \times 5 \times 6)$.
4. the Process for initializing the first and second fully-connected layer is similar, the size of the weight matrix for those layers is $(5 \times 5 \times 16 \times 120)$ and (120×84).

Experimental results

In this section, I will discuss fifth kinds of initializations:

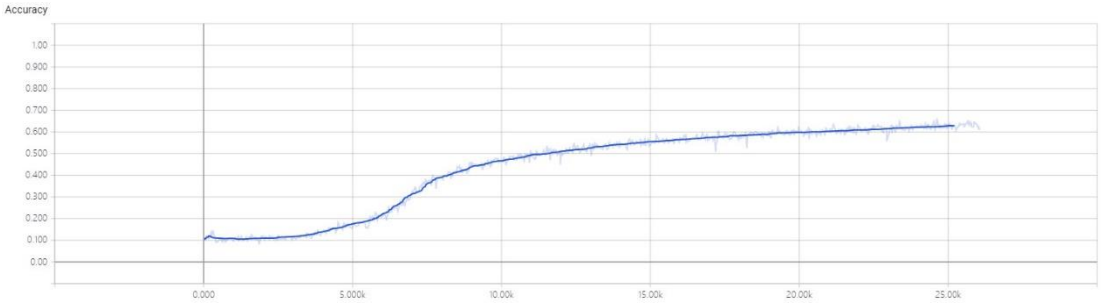
1. The first convolution layer is random initialization with uniform distribution, the second is the

random initialization with uniform distribution

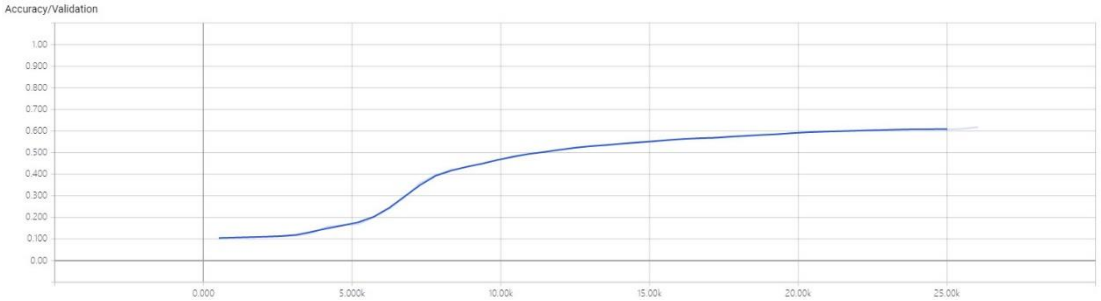
2. The first convolution layer is random initialization with Xavier distribution, the second is the random initialization with Xavier distribution
3. The first convolution layer is K-means initialization with uniform distribution, the second is the random initialization with Xavier distribution
4. The first convolution layer is K-means initialization, the second is the K-means initialization
5. The first convolution layer is K-means initialization, the second is the K-means initialization, and the third is the K-means initialization.

The result of first initialization can be shown as follow (all run 50 epochs):

Training result

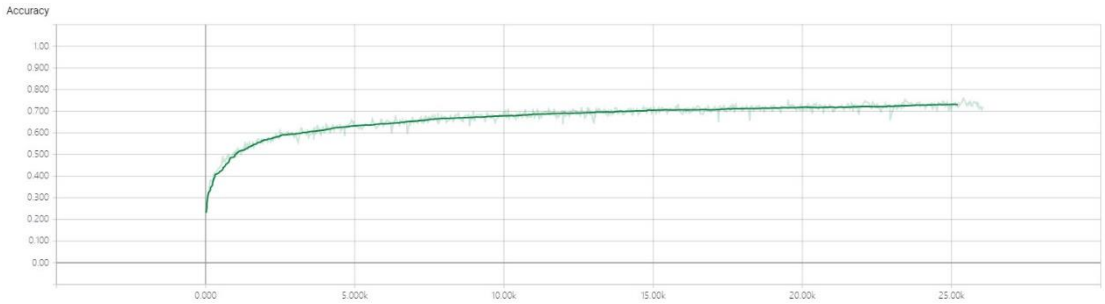


Testing result

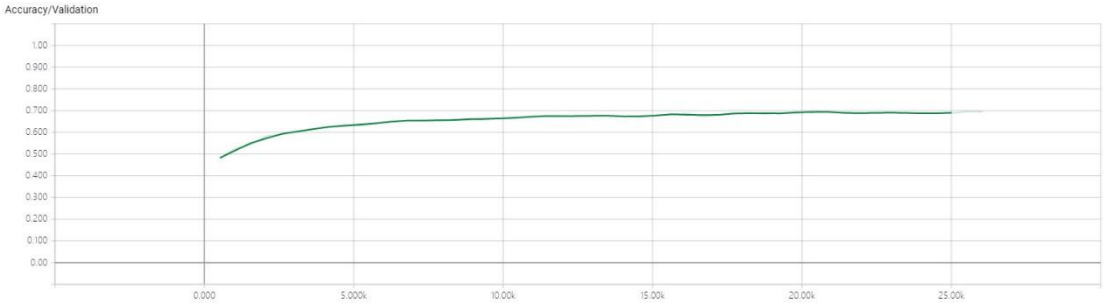


The result of second initialization can be shown as follow:

Training result

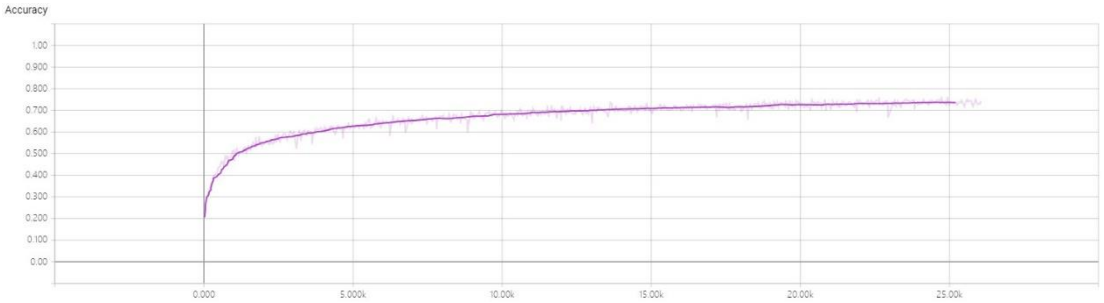


Testing result

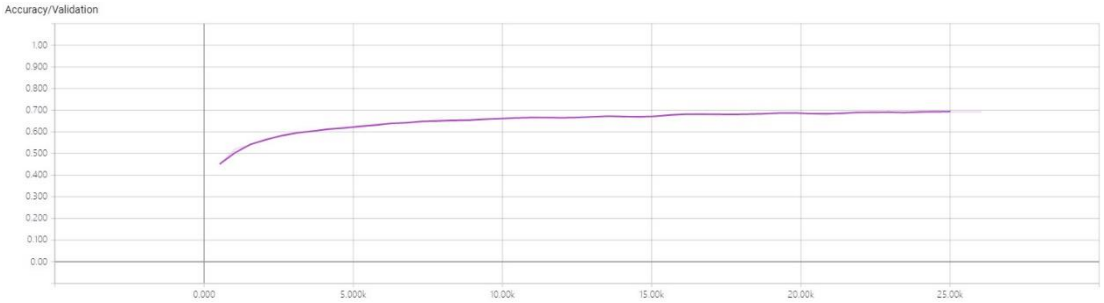


The result of third initialization can be shown as follow:

Training result

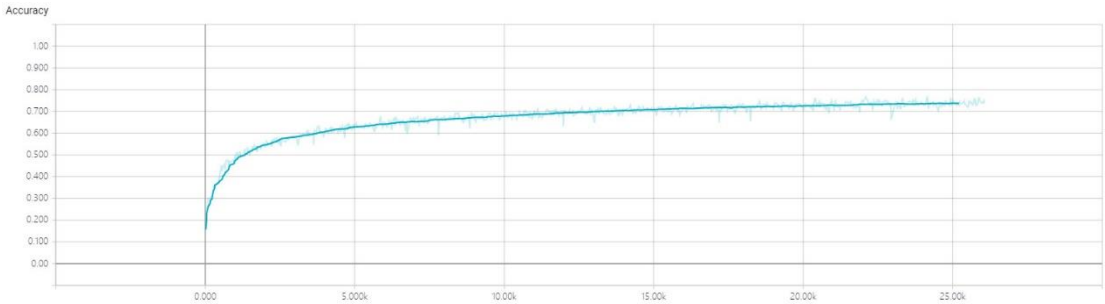


Testing result

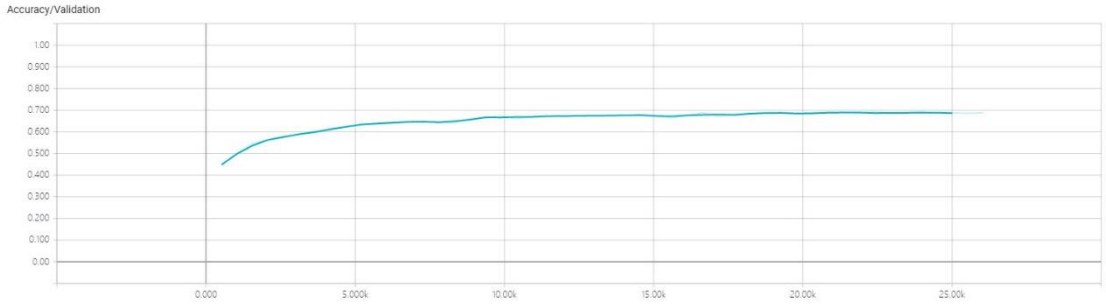


The result of fourth initialization can be shown as follow:

Training result

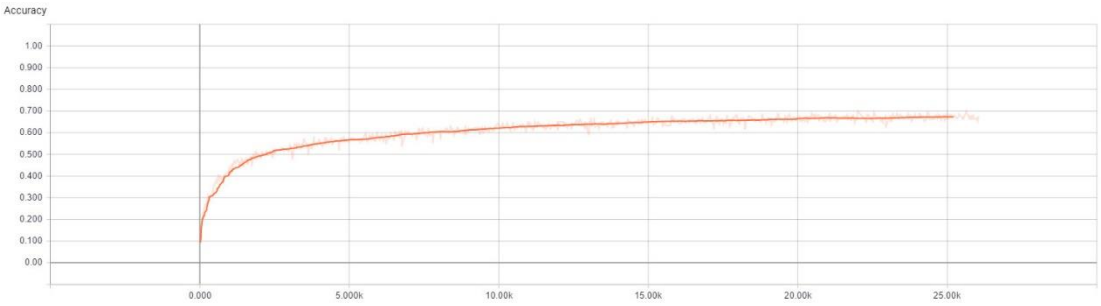


Testing result

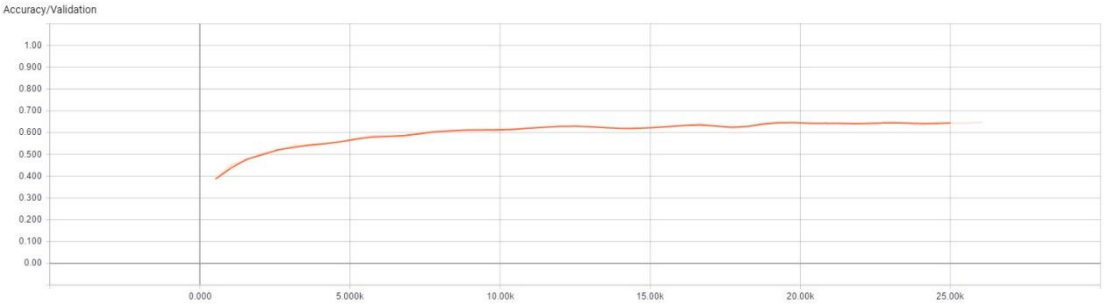


The result of fifth initialization can be shown as follow:

Training result



Testing result



Discussions

The object of the K-means is to acquire a series of characteristics which can classify those images significantly. In the K-means operations, we will acquire the classification and the centroid result of those training images, which means that those images belong to those classifications. As a result, if we use the relative parameters transformed from those K-means centroids, we can make one of the output of the image patch greatly larger than those other. Therefore, the patch with different pattern can be divided in a large distance, which can make the classification of the CNN effectively.

From the experimental result, we can find that the K-means initialization can make a better result compared with the uniform random initialization. In addition, there seems to be little difference between the network whose first layer is the K-means initialization and the one whose both first and second layer are the K-means initialization methods. However, we can find that there is little improvement compared with the network whose initialization method is the Xavier. In addition, if we apply the K-means to initialize the fully-connected layer, there will be a significantly decrease in the first epoch. Therefore, the effect of the K-means is not as good as the random initialization method.

Problem 2: Capability and Limitation of Convolution Neural Networks (50%)

(a) Improving Your Network for CIFAR-10 Dataset (Advanced: 25%)

Motivation

Modify the structure of basic CNN constructed in Problem 1. Find an optimal solution to increase both the training and testing accuracy for the classification CIFAR-10.

Methods and procedures

For this question, Erli Wang, Tian Ye, Weidong Chen and I have built a feasible structure of the convolution neural network. The reason to cooperate with others is that the training process of the network cannot conduct efficiently in my computer. For instance, the running time of utilizing my computer to train the network with 10 layers in 200 epochs will be more than 13 hours. The basic structure can be shown as follow:

Data Augmentation	
Random Flip	
Random Rotation	
Network structure	Layer
3 x 3 conv. 96 ReLU Init.: Xavier	1
batch normalization	
3 x 3 conv. 96 ReLU Init.: Xavier	2
batch normalization	
3 x 3 conv. 96 ReLU Init.: Xavier with stride r = 2	3
batch normalization	
3 x 3 conv. 192 ReLU Init.: Xavier	4
batch normalization	
3 x 3 conv. 192 ReLU Init.: Xavier	5
batch normalization	
3 x 3 conv. 192 ReLU Init.: Xavier with stride r = 2	6
batch normalization	
3 x 3 conv. 192 ReLU Init.: Xavier	7
batch normalization	
1 x 1 conv. 192 ReLU Init.: Xavier	8
batch normalization	
1 x 1 conv. 96 ReLU Init.: Xavier	9
batch normalization	
1 x 1 conv. 10 ReLU Init.: Xavier	10
batch normalization	
global averaging over 6 x 6 dimensions	11
batch normalization	
1 x 1 conv. 10 ReLU Init.: Xavier	12
batch normalization	
10-way softmax	13

At first, we will conduct some basic preprocessing procedure: randomly flip the image and rotate the image by a random angle.

Then, we will do the convolution procedure and the batch normalization several time. In this section, we will use ReLU function as the activation; the Xavier function will be utilized as weight initialization function.

1. Use 96 3×3 filters to apply the convolution. The strides for the convolution is 1. Then, the batch normalization will be applied to do the preprocessing for the next convolution layer. This procedure will be repeated twice.
2. Use 96 3×3 filters to apply the convolution. The strides for the convolution will be changed to be 2. Then, the batch normalization will be applied to do the preprocessing for the next convolution layer.
3. Use 192 3×3 filters to apply the convolution. The strides for the convolution is 1. Then, the batch normalization will be applied to do the preprocessing for the next convolution layer. This procedure will be repeated twice.

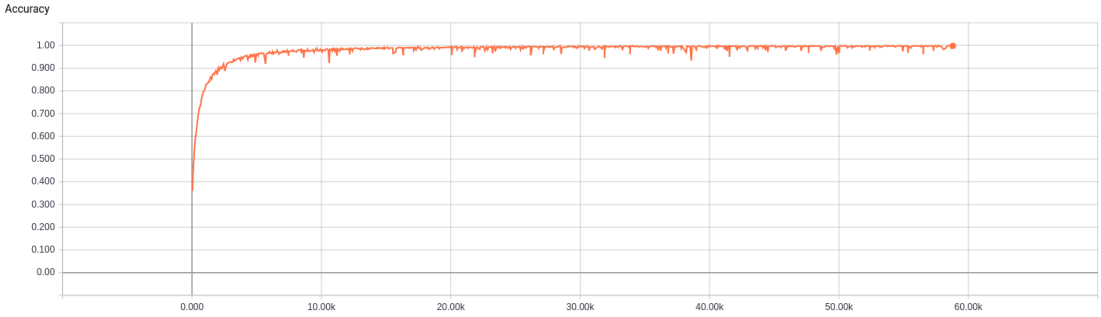
4. Use 192 3×3 filters to apply the convolution. The strides for the convolution is 2. The batch normalization will be applied to do the preprocessing for the next convolution layer.
5. Use 192 3×3 filters to apply the convolution. The strides for the convolution is 1. Then, the batch normalization will be applied to do the preprocessing for the next convolution layer.
6. Use 192 1×1 filters to apply the convolution. The strides for the convolution is 1. Then, the batch normalization will be applied to do the preprocessing for the next convolution layer.
7. Use 96 1×1 filters to apply the convolution. The strides for the convolution is 1. Then, the batch normalization will be applied to do the preprocessing for the next convolution layer.
8. Apply 10 1×1 filters to apply the convolution. The strides for the convolution is 1. Then, the batch normalization will be applied to do the processing for the average-pool layer.
9. Utilize the average pool and cascade it with the soft max layer finally.

In this question, we will use 0.001 as the learning rate for regression and the batch size is 256.

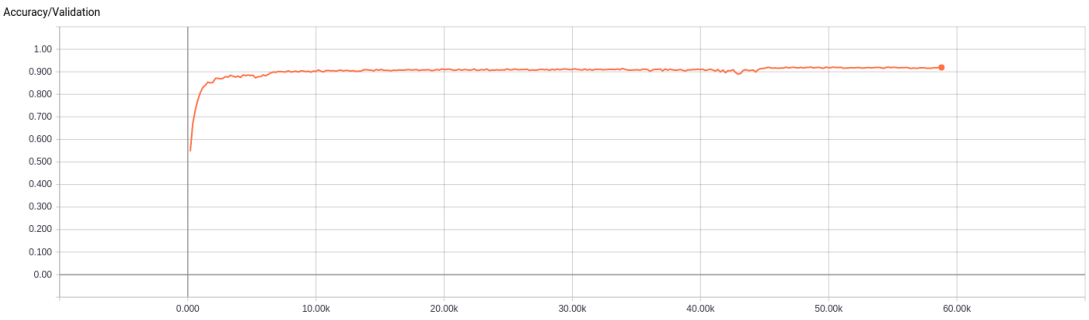
Experimental results

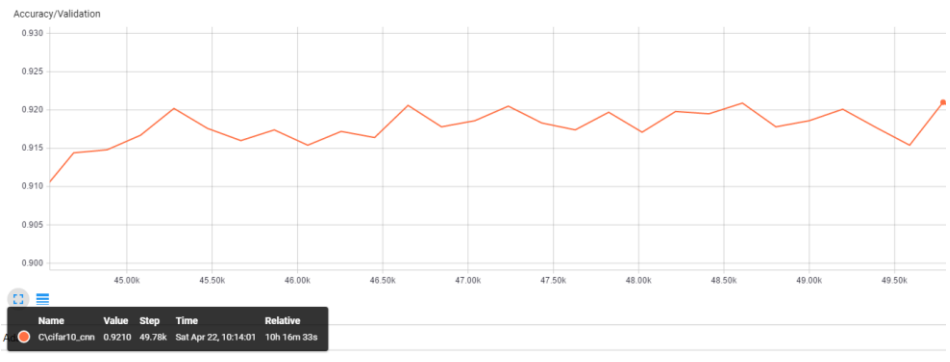
The result of training and testing accuracy graph can be shown as follow (the epoch is 300, batch size is 256):

Training Accuracy



Testing Accuracy





From the result, we can find that the accuracy of the network proposed above can increase the testing and training accuracy significantly compared with the old network in Problem 1. In addition, the maximum testing accuracy in this network is 92.10%

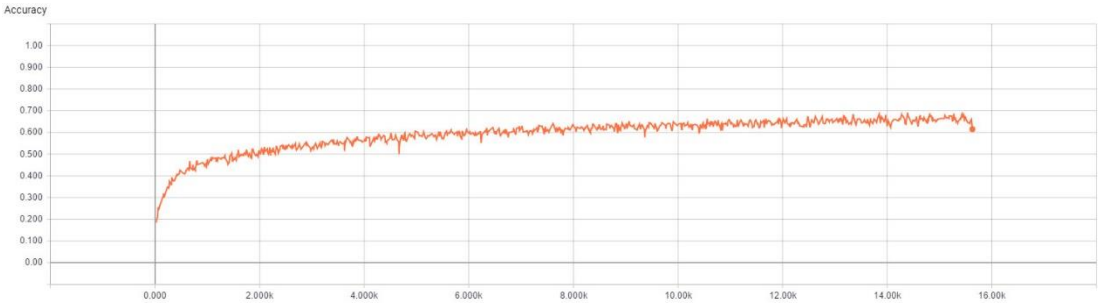
Discussion

In this section, we began to justify the effect when changing the parameter of the convolution layer and fully-connected layer. In this section, we will use the basic network in Problem 1 to justify.

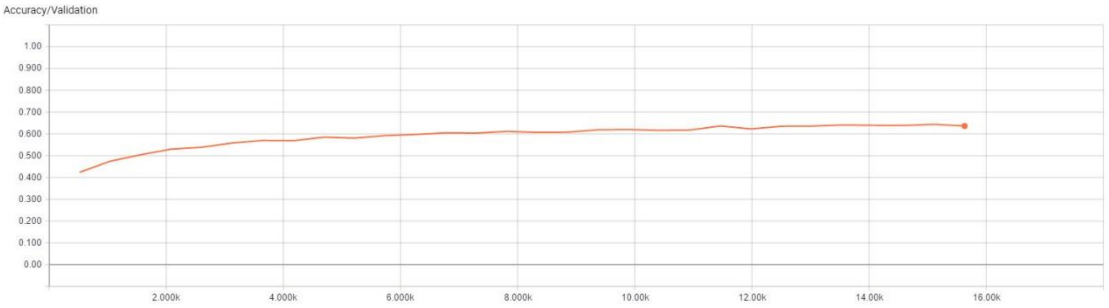
At first, we will choose the first convolution layer as the experimental layer. Then, we can try different number of the filters in the convolution layer, we can try 3, 6 and 18. The result can be shown as follow:

Number of filters = 3:

training result:

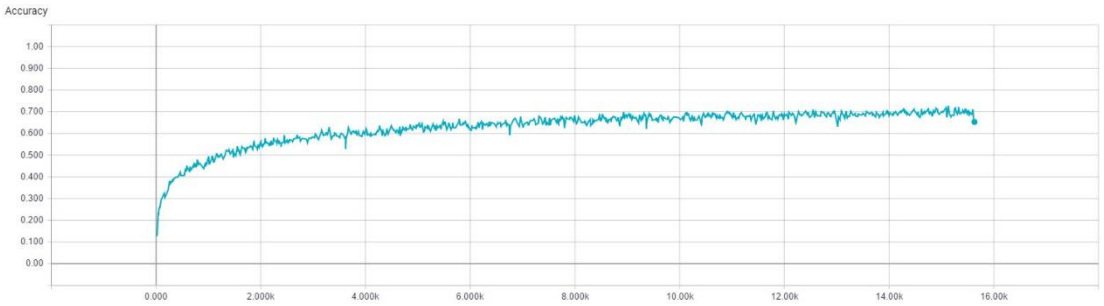


testing result:

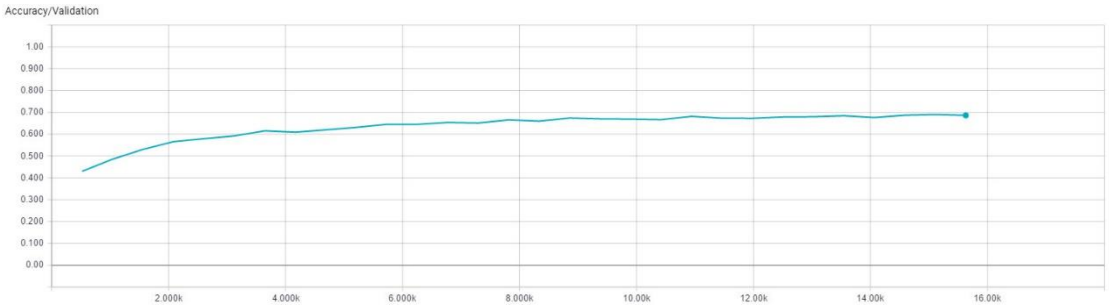


Number of filters = 6:

training result:

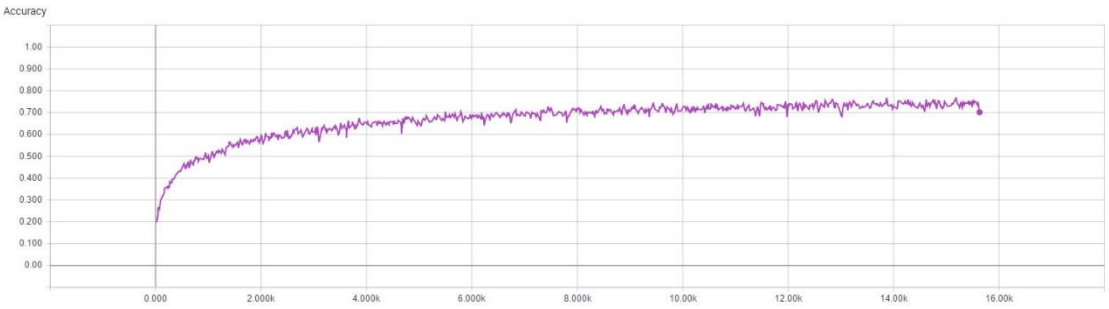


testing result:

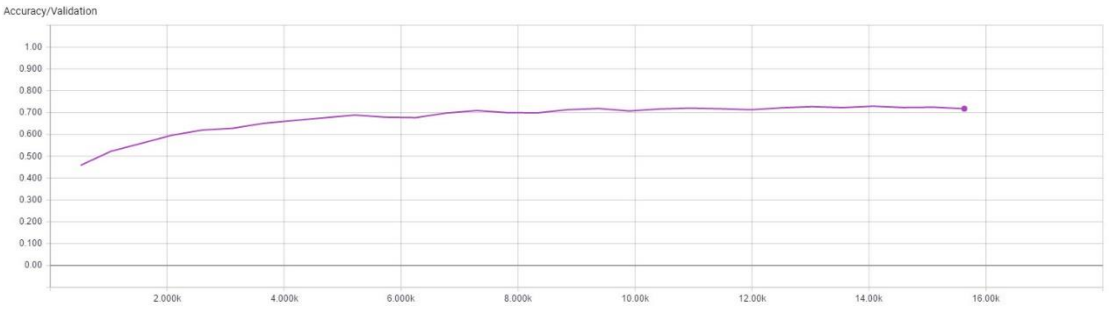


Number of filters = 18:

training result:



testing result:



From the result above, we can find that the larger the number of the filter is, the better the accuracy of the network will be. The reason is that the filter is used to extract the key features for images and the number of features will be too many to use only several filter to extract. As a result, we should increase the number of the filters for the convolution process. According to our experiment, we use 96 and 192 as the number of the filters in this question. In addition, from the discussion in Problem 1 (a), we can find that the deeper the network is, the higher the testing accuracy will be. As a result, we finally built 13 layers for this network.

According to the paper by Jost Tobias Springenberg et al. (2015), we decide to eliminate the fully-connected layer and use more convolution layers to do the training, which can make us get better result than the one with fully-connected layer. The reason is that the fully-connected is one part of the MLP, which is useful for the 1-D message instead of the 2-D message like the images. However, the convolution and max-pool will have better performance than the fully connected since the convolution is designed to process the 2-D message, which could make a better result compared with the MLP.

In addition, we applied the batch normalization algorithms. The batch normalization is the processing for the output data of the convolution layer. After passing through the activation function, the data can

be the result of the output. However, the output will have different means and deviations for each mini-batch, which may have a negative effect on classifications. As a result, we apply the batch normalization to make the output for each mini-batch be set as 0 mean and 1 standard deviation. As a result, we will use the output of batch normalization as the final output of the convolution layer.

Also, we can use the convolution layer with 2 strides to replace the max-pooling process, which can get better result for the accuracy. The reason is that when computing the max-pool, we just choose the elements with maximum value. However, for the computing the convolution layer, we can utilize not only the maximum value, but other values, which can make us consider the result completely with little loss and make the network complex. As a result, the accuracy may increase.

In addition, we applied the batch normalization algorithms. The batch normalization is the processing for the output data of the convolution layer. After passing through the activation function, the data can be the result of the output. However, the output will have different means and deviations for each mini-batch, which may have a negative effect on classifications. As a result, we apply the batch normalization to make the output for each mini-batch be set as 0 mean and 1 standard deviation. As a result, we will use the output of batch normalization as the final output of the convolution layer.

Finally, we can use the convolution layer with 2 strides to replace the max-pooling process, which can get better result for the accuracy. The reason is that when computing the max-pool, we just choose the elements with maximum value. However, for the computing the convolution layer, we can utilize not only the maximum value, but other values, which can make us consider the result completely with little loss and make the network complex. As a result, the accuracy may increase.

As a matter of fact, we have tried a series of preprocessing that can improve the accuracy of the network. However, they cannot improve the accuracy significantly, and they may improve the complexity of the significantly. For example, the ZCA may not improve the accuracy of the network since the process of PCA is to filter the irrelevant message and preserve the vital messages, which means that the volume of the messages decreased. In addition, the ZCA whitening has a large running time since the ZCA process can only be conducted by CPU, which requires a large amount of time to process. What is more, the blur may decrease the quality of the networks because the image size in CIFAR-10 is only 32×32, which is really tiny and unclear while the blur can simulate the image in a large distance. Therefore, it is useless to apply the blur process.

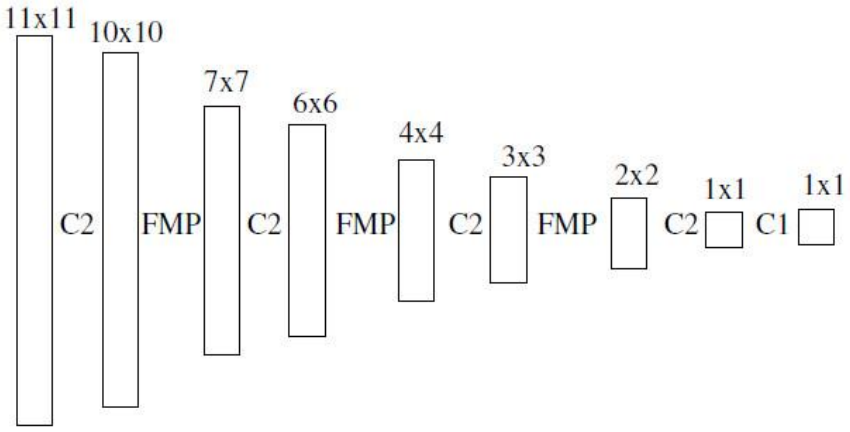
(b) State-of-the-Art CIFAR-10 Implementation (Advanced: 25%)

Motivation

In this section, we will find some networks proposed by other researchers and analyze the neural network. Then, we can compare the result among the baseline CNN, our improved CNN and the network we find in the paper.

Methods and procedures

In this section, we choose the network in the paper **Fractional Max-Pooling** proposed by Benjamin Graham. One of the sketch map for the fractional max pooling can be shown as follow:



In this question, the number means the size of the image in each layer. C2 is the convolution in 2-D with $stride = 1$ and $size_{filter} = 2$; FMP is the fractional max pooling layer. In each FMP layer, the ratio of the size of input to the one of output is $\sqrt[3]{2}$. In addition, for each output node of max pooling layer, the value is obtained from the original layer randomly.
Therefore, the author proposed a network by the fractional max pooling:

$$\left(160nC2 - FMP\sqrt[3]{2}\right)_{12} - C2 - C1 - output$$

In this section, the structure of cascading convolution filter and FMP which will decrease the size by $\sqrt[3]{2}$ will be cascaded 12 times. In addition, the number of the filters in each convolution layers are 160. Then, the next convolution 2-D filter has 160 filters, and the next is the soft-max layer. As a result, the image will be decreased from 32×32 to 2×2 in the $\left(160nC2 - FMP\sqrt[3]{2}\right)_{12}$ and the final image output will be 1×1.

Experimental results

One of key points that the author **achieves a good result** is the application of fractional max pooling. As is shown as follow, the FMP can decrease the size of input image into the size with the size of $\sqrt[3]{2}$ less than original image. As a result, for each FMP, the relative size of the output is significantly larger than the size of the output by the max-pooling, meaning that we can preserve much more information than the max-pooling approach when doing the down-sampling. Therefore, we may acquire more accurate classification relationship. In addition, the depth of 12 means that we can simulate really complex relationship significantly.

Another point to improve the accuracy is to attempt to build a series of FMP networks by different random FMP method. Then, apply the testing images into each network and vote for the most output results in each CNN as the final result of the classification. Theoretically, we can improve the testing accuracy significantly if the number of networks is large.

The best result when applying the FMP algorithm, baseline CNN and our approach can be shown as follow:

Methods	FMP	baseline CNN	Our approach
Accuracy	96.53% (100 tests)	75.01% (100 epoch)	92.10% (300 epoch)

Discussion

In this section, we can discuss the effect among three networks in this assignment.
From the result above, we can find that the accuracy of FMP is much higher than the baseline CNN and our approach. In addition, from the structure of the three networks, we can find that both Out approach and the FMP CNN have really sophisticated structure including large number of filters in each layer and all 2-D layers and great depth while the baseline CNN has only a few filters in each convolution layer and swallow depth of the network.

Then, we can discuss the **pros and cons** for three methods:

FMP:

The advantage of the FMP is the high accuracy. As matter of fact, one of the reason why the base line CNN is not as good as the FMP is that the number of filters and the depth of the network in the basic CNN is much less than the ones in and the FMP. In addition, the FMP's accuracy is higher than the one of our approach since the FMP has the voting process while our approach does not.
However, the FMP has a serious disadvantage: the calculation complexity is really high. In fact, we compared the running time among FMP algorithm, baseline CNN and our approach, we can find that the running time among them are 500s, 15s and 150s for each epoch. The reason is that the fractional max-pool acquires the output by picking the input pixel randomly, which will spend a plenty of time finishing it. In addition, the number of filters is much more than other two approaches.

Base CNN:

The advantage of basic CNN can be shown above clearly: least calculation complexity. Therefore, we can acquire the final result of the basic CNN in less than one hour. However, the shortcoming of the baseline CNN is significant: least testing accuracy. The reason is that the depth of the network and the number of filters are too low. In addition, there is a series of fully-connected layers instead of the 2-D convolution layer, which can get poor performance when dealing with the 2-D images.

Our approach:

As a matter of fact, our approach is really feasible in both the testing accuracy and the running time. The reason to get an applicable accuracy is the adequate depth of networks and the number of filters which makes it possible to acquire more features and build more complex relationships. In addition, we just use the convolution layer with 2 strides to replace the max-pooling layer so that the running time will be sharply decreased compared with the FMP.
However, there is small shortcoming of our approach: The accuracy cannot increase significantly in the future. The reason is that we only use one network in the question to process the CIFAR-10 while the FMP can allow us to use a series of networks and vote for the most result as the final analysis result.