Capstone Project

Yatin Gupta

Machine Learning Engineer Nanodegree

March 3, 2017

1. Definition

1.1 Project Overview

The improvement on the camera and the widespread of larger capacity hardwares lead us to store more digital pictures. However computers can't understand the meaning of the pictures, so it is essential for us humans to classify or search images. Image recognition(Figure.1) will help us to classify or search pictures without our intervention. These days, because of the development of the computational capacity, we can process large number of pictures with high level accuracy(nearly the human's recognition). In this project, I'll discuss the image recognition algorithm which will be useful in the classification of large number of images.

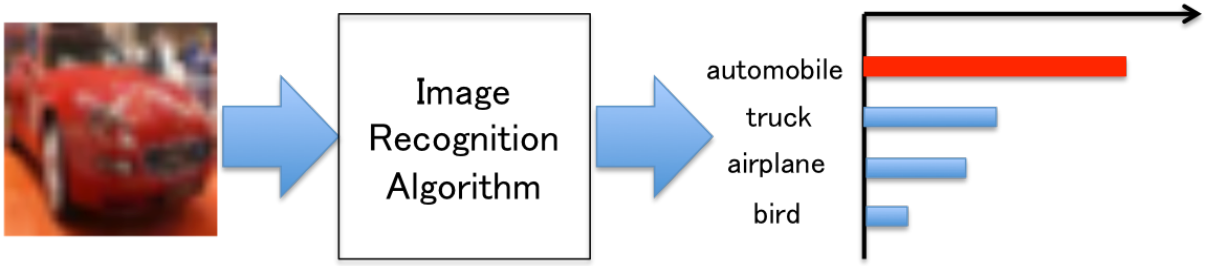

Figure 1: Image of the Algorithm

1.2 Problem Statement

The dataset I use in this task is CIFAR-10 dataset. The description about it is below:
The CIFAR-10 dataset consists of 60,000 32×32×3 (width=height=32 and RGB=3) color images in 10 classes, with 6,000 images. 50,000 images are the training images and 10,000 images are the test images.

The classes are completely mutually exclusive. There is no overlap within each image class.

Many research have been done with the CIFAR-10 dataset. Some of the re- searches use very deep neural networks for training(ranging from 5 to more than 100 layers in the reference [1]). The reference proposes a way of training very deep networks by using adaptive gating units to regulate the information flow. Others have originality in pooling layer which is one of CNN components.The reference [2] proposes fractional max-pooling whose pooling size is fractional. The advantage of fractional max-pooling is to avoid overfitting.

In this task, I'll discuss the image recognition by Convolutional Neural Network(CNN) and its hyper-parameter tuning. When constructing CNN model, hyper-parameter tuning is one of the essential and time-consuming tasks. Therefore, I propose a method, called "Bayesian Optimization", to tune hyper-parameters by not using "grid search" which is not appropriate for this task because of computationally expensive. Especially, I'll tune the number of neurons in the fully connected layer and the learning rate of the optimizer.

1.3 Metrics

The objective in this problem is multi class classification. Therefore,the metrics will be accuracy score. In the deep learning algorithms, the predicted output is probability. The probability is defined as the softmax as shown below.

$$p_j = \frac{e^{u_j}}{\sum_{k=1}^{n} e^{u_k}} \tag{1}$$

The probability is calculated by each class and then the highest probability is the predicted output. In this task, the accuracy score is calculated by the following formula.

$$Accuracy\ Score = \frac{Number\ of\ correctly\ classified\ output}{Number\ of\ the\ data} \tag{2}$$

2. Analysis

2.1 Data Exploration

CIFAR-10 dataset consists of 50,000 images of training data and 10,000 images of test data.For each data, it contains 10 different kind of classes(discussed in the next chapter) and the distribution of each class is the same in both training and test data. That means 5,000 images for each class in the training data and 1,000 images for each class in the test data.(Fig.3 and Fig.4).

2.2 Exploratory Visualization

There are 60,000 of images in total. Figure.2 (50,000 images for training data and 10,000 images for test data) shows the samples of the images. I plotted 10 images for each class.Figure.3 and Figure.4 shows the distribution of the data. For the training dataset, each class has 5,000 images and for the test dataset, each class has 1,000 images. The label 0 to 9 corresponds to 'Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship, and 'Truck'.

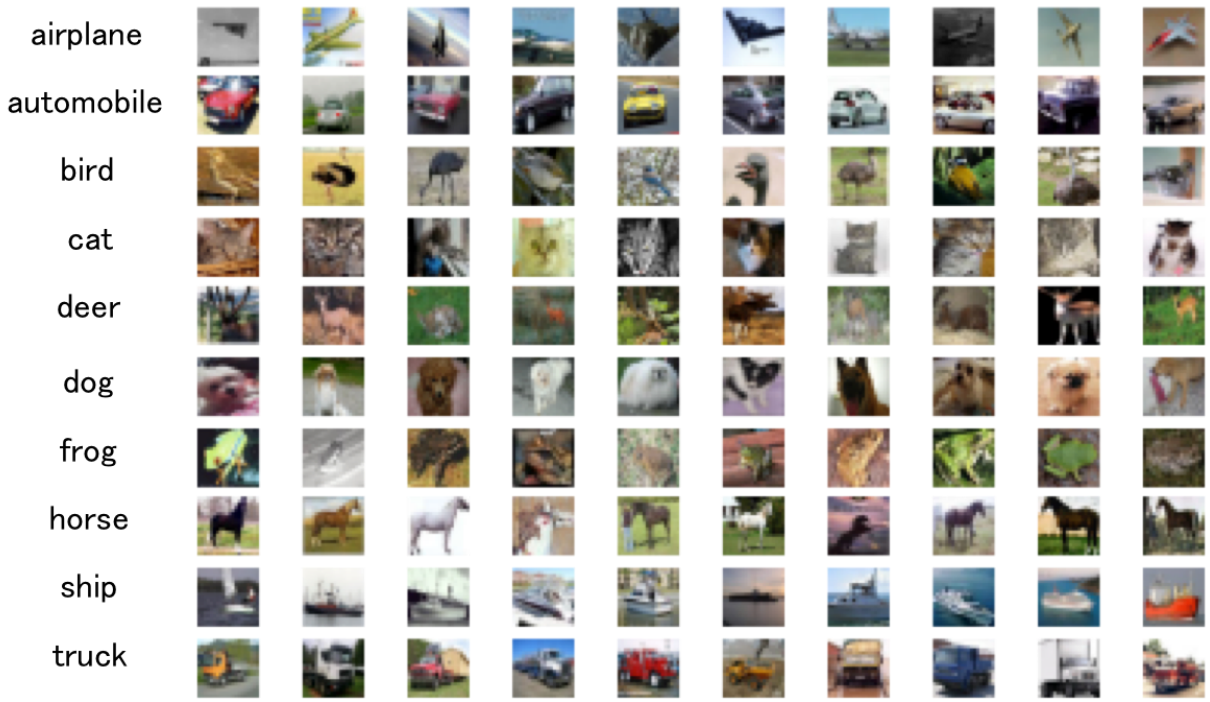

Figure 2: Sample of the Images

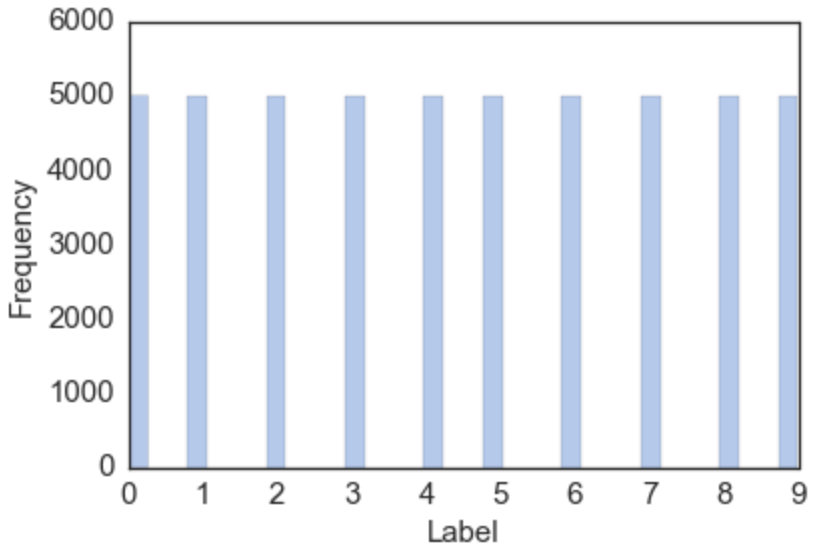

Figure 3: Distribution of Training Data

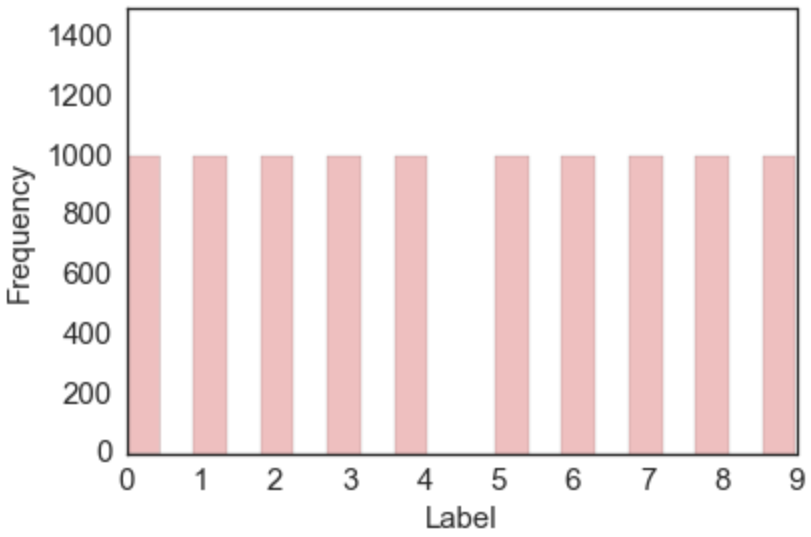

Figure 4: Distribution of Test Data

2.3 Algorithms and Techniques

In this task, I'll use Convolutional Neural Network(CNN). CNN has been successful in practical applications for image recognition.CNN consists of convolution layer, pooling layer and fully-connected layer and sometimes contains local contrast normalization(LCN). In this chapter, I'll discuss the convolution layer and pooling. As the name 'convolution' suggests, the network employs a mathematical operation called convolution. Fig.5 is the example of the architecture of the CNN.Fig.6 illustrates the image of the CNN.[3]

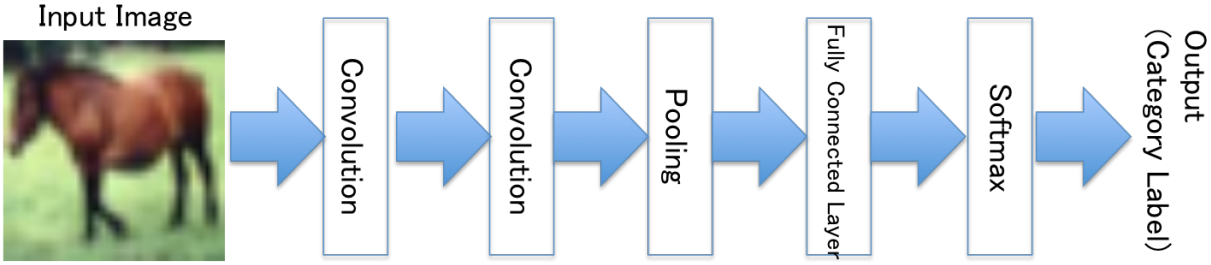

Figure 5: An example of CNN Architecture

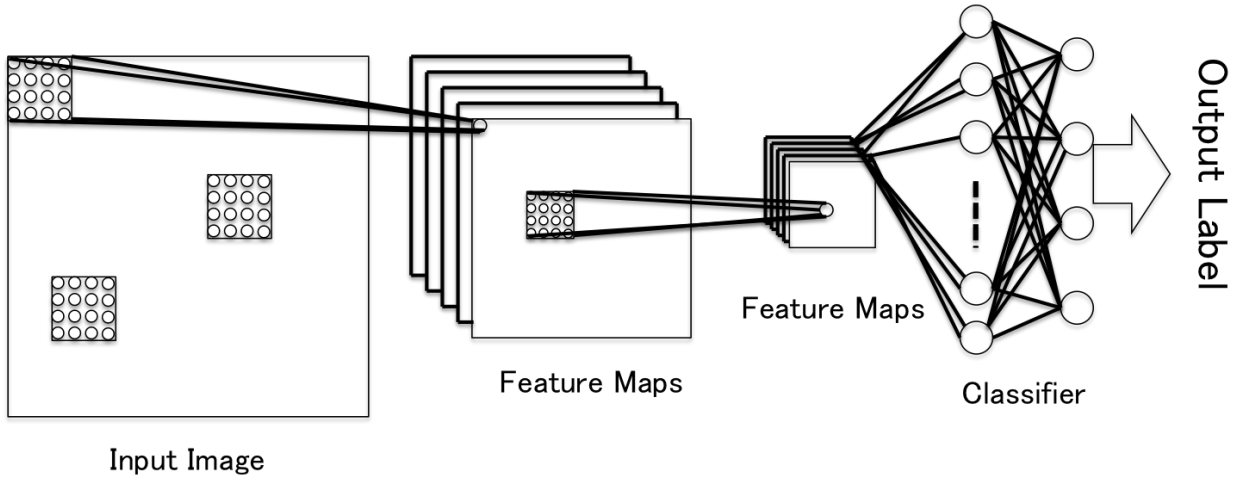

Figure 6: Overview of CNN

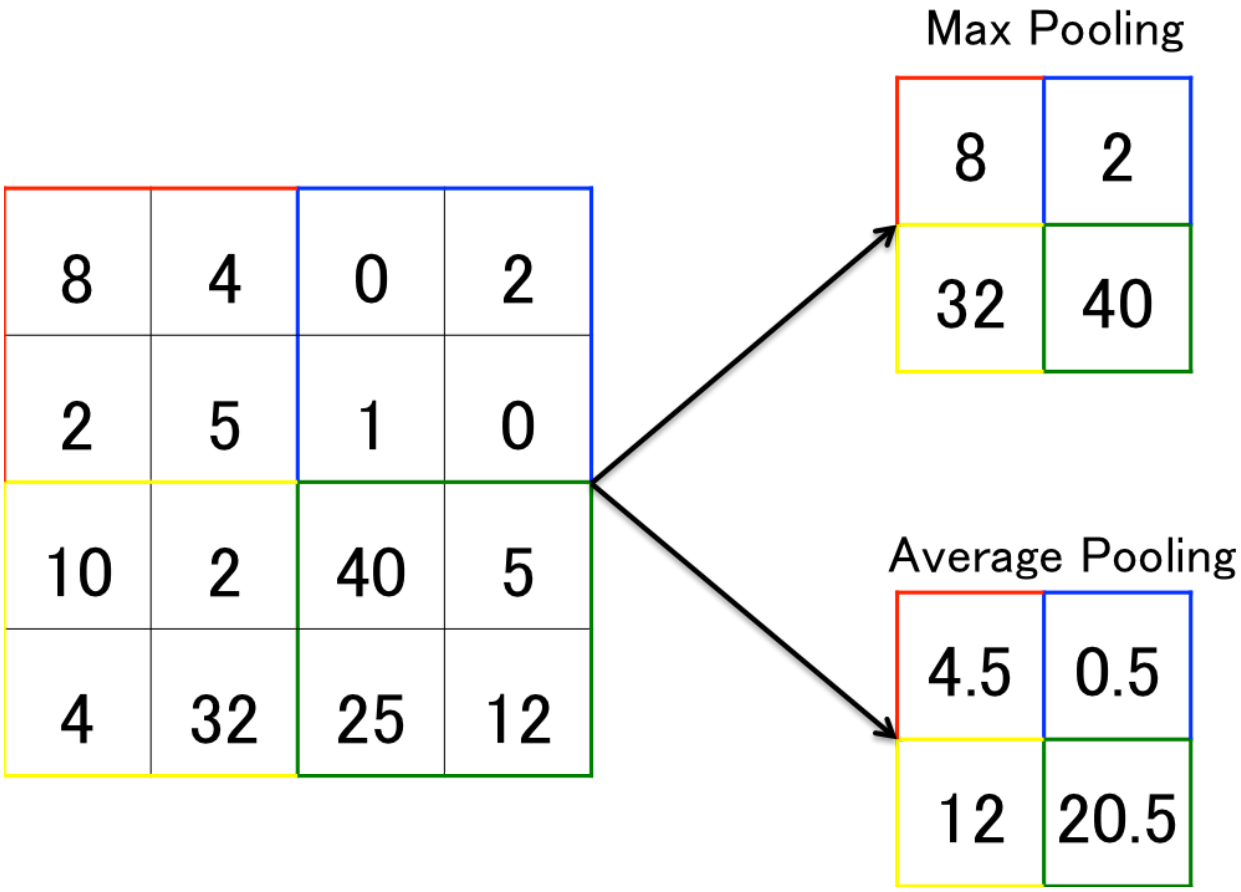

Figure 7: Example of Max Pooling and Average Pooling

Convolutional neural networks are biologically inspired variants of multilayer perceptron.They are emulated by the behavior of a visual cortex.[9] In the convolutional layer, the input images are convolved by filters. This process is basically the same as the convolution of the general image processing which convolves small size image into the input image so that the image gets blur or emphasizes edge.

To be more specific, the input image has S×S for each channel and 2D filter has L×L. The input image are convolved by the 2D filter for each channel and then each result are added. Suppose input image can be written as x_{ijk} ((i, j, k) ∈ [0, S − 1] × [0, S − 1] × [1, N]) and the calculation result is u_{ij} and as for the filter, I define w_{ijk} ((i, j, k) ∈ [0, L − 1] × [0, L − 1] × [1, N]). The output will be as follows.

$$u_{ij} = \sum_{k=1}^{N}\left[\sum_{(p,q)\in P_{ij}} x_{pqk} w_{p-i,q-j,k}\right] + b_k \tag{3}$$

P_{ij} is the L × L square area whose center is (i, j) and b_k is the bias.

$$P_{ij} = \{(i+i', j+j') | i' = 0, ..., L-1, j' = 0, ..., L-1\} \tag{4}$$

When the input image size is large, the stride of filter will be larger than 1.However, in this case, some features won't be captured. Therefore, the performance will decline in general. After this convolutional process, u_{ij} pass through the activation function, then the output will be produced.

$$y_{ij} = a(u_{ij}) \tag{5}$$

If the number of the fil ter is N' , the output dimension will be y_{ijk} ((i, j, k) ∈ [0, S − 1] × [0, S − 1] × [1, N'])

Pooling layer is put after the convolution layer. The function of the pooling layer is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. By introducing pooling layer, not only the architecture will be more robust but also the dimensionality will be reduced. Max pooling and Average pooling are the typical pooling which are generally utilized. Figure.7 is the example of the pooling.The original map is the size of 4×4. The stride for the pooling is 2 and the pooling size is 2×2. "Max pooling" is to extract the maximum pixel from each region and "Average pooling" is to calculate the average value for each region.In this task, I utilized max pooling at the pooling layers.

2.4 Benchmark

For the CNNs architecture, it is quite important to decide the number of layers.Therefore, I first choose several convolutional layers and decide one of them as a benchmark.

When deciding the architecture, I set the mutual parameters as follows. The number of batch size is 32.The number of filters of each convolutional layer is 32. and finally the number of epochs is 20. The metrics in this task is the accuracy score as I introduced in Section 1.3 Metrics.
I tried 4 architecture of CNNs.

1. 1 Convolutional layer & 2 Fully connected layers
   The simplest version in these model.The input and output dimension on each layer are below. The accuracy rate of training and validation data are also below.

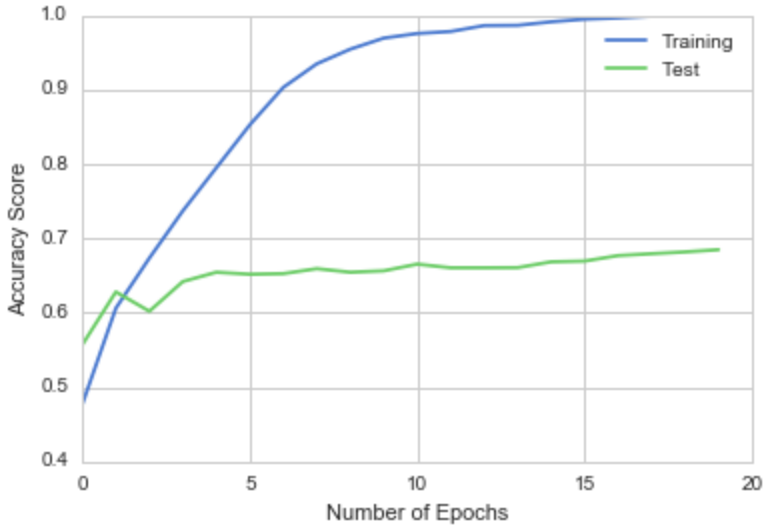

Figure 8: Accuracy rate of training and validation data

The accuracy rate on the training data is quite high,however the accuracy rate on the validation data is low. This means that this architecture falls into over-fitting.

2. 2 Convolutional layers & 2 Fully connected layers
The input and output dimension on each layer are below. The accuracy rate of training and validation data are also below.

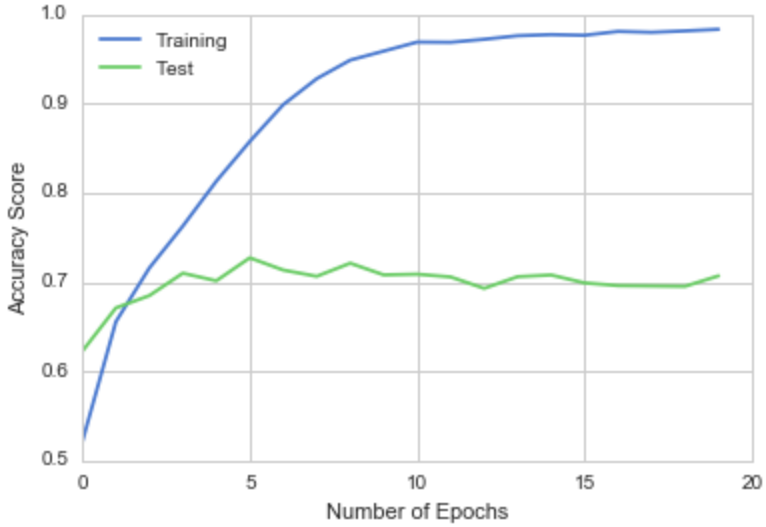

Figure 9: Accuracy rate of training and validation data

This architecture also caused over-fitting as mentioned above.

3. 3 Convolutional layers & 2 Fully connected layers
The accuracy rate of training and validation data are also below.

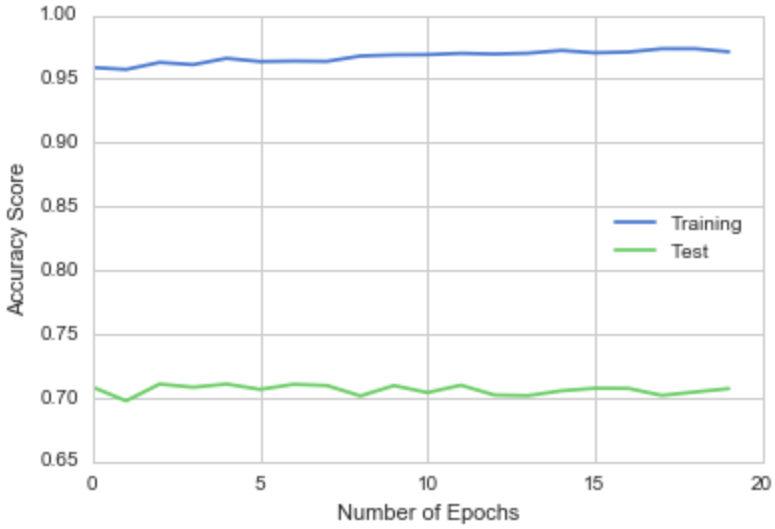

Figure 10: Accuracy rate of training and validation data

This architecture also falls into over-fitting.

4. 4 Convolutional layers & 2 Fully connected layers
The accuracy rate of training and validation data are also below.

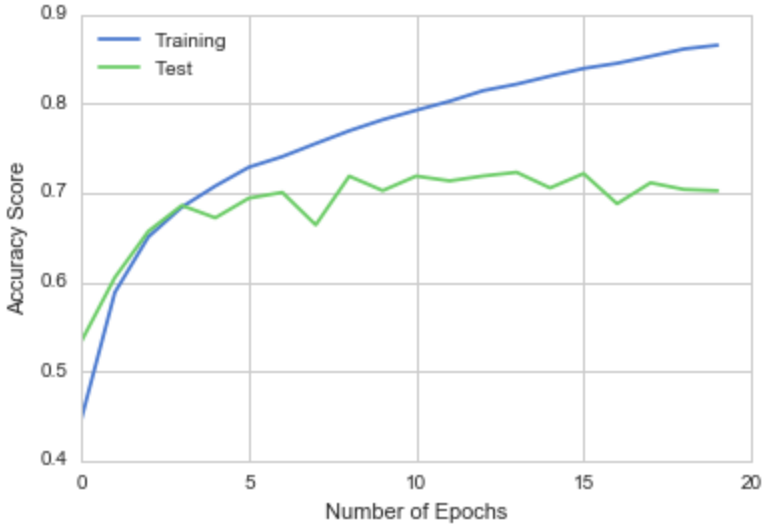

Figure 11: Accuracy rate of training and validation data

This architecture is far better than the others. The accuracy rate of both training and validation data is high as the number of epochs increases. The accuracy rate of this architecture is 70.2%. Actually the highest accuracy rate in CIFAR-10 is more than 95.0% and the benchmark is quite lower than the highest one. My target in this task is to find the optimal parameters for the architecture. Therefore, I'll utilize this 4 layers convolution & 2 fully connected layers architecture as a benchmark From these results, I choose the fourth architecture(4 convolution layers & 2 fully connected layers) as the benchmark for this task.