Capstone Proposal

Yatin Gupta

Machine Learning Engineer Nanodegree

March 4, 2017

Domain Background

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.

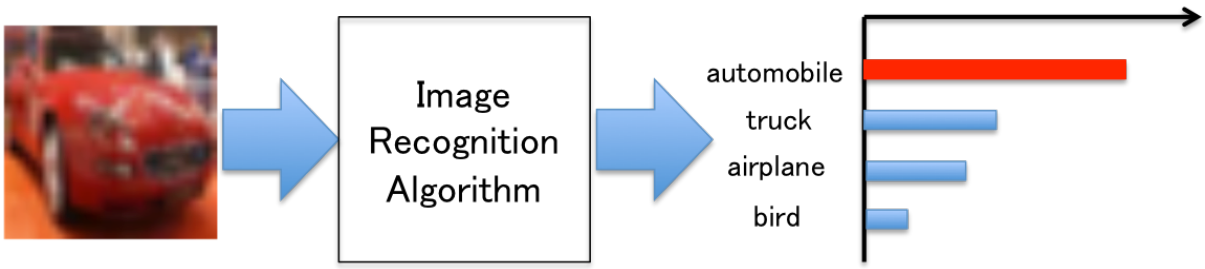

Figure 1: Image of the Algorithm

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 researches use very deep neural networks for training. 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.

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.

Datasets and Input

CIFAR-10 dataset(https://www.cs.toronto.edu/~kriz/cifar.html) 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 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.

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'.

Solution Statement

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).

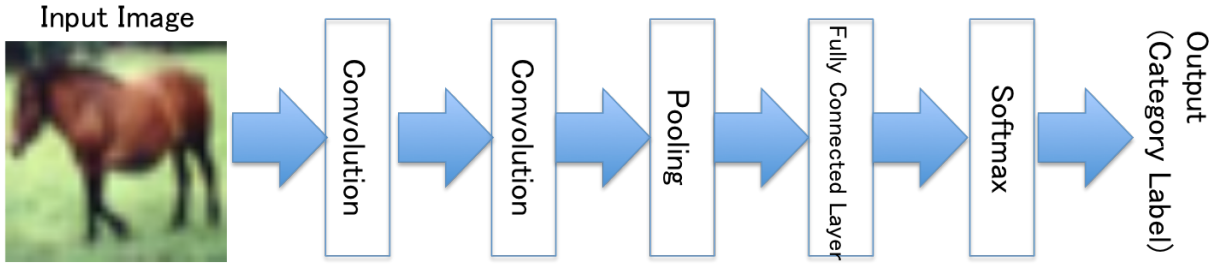

Figure 5: An example of CNN Architecture

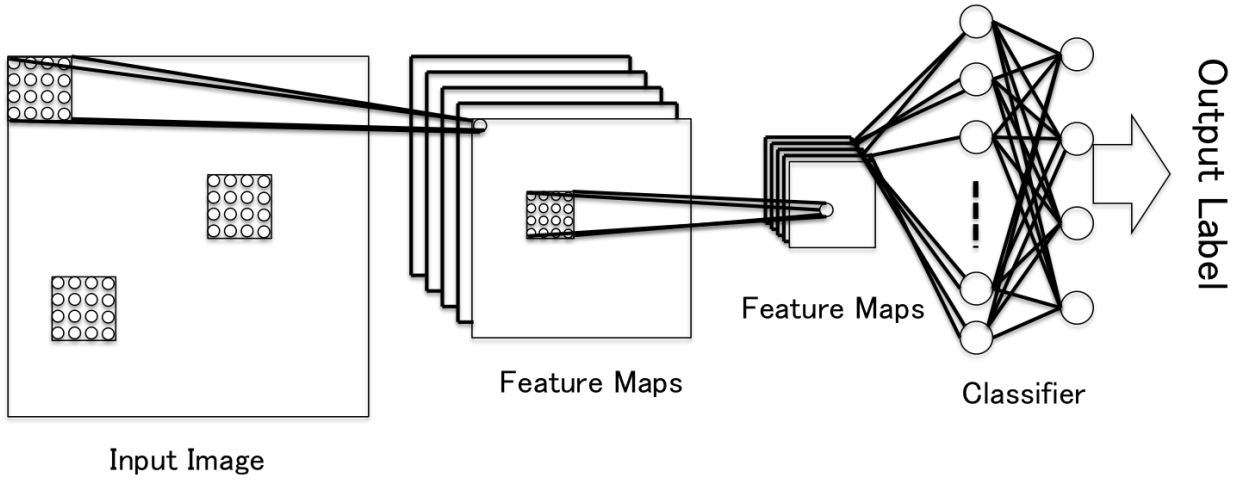

Figure 6: Overview of CNN

Benchmark Model

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

Evaluation 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.

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

Project Design

I will utilize multiple convolutional layers and multiple fully connected layers. The objective of the learning is to minimize the cross entropy in the learning process.

To train the CNN model, I will use Keras which is one of the neural network libraries like Theano or Tensorflow.As for the optimizer, I will use 'Adam', Adaptive Moment Estimation, is an online method to estimate the average and variance of the gradient. With these information, adam can update learning rate. Adam keeps not only an exponentially decaying average of the past squared gradient but also an exponentially decaying average of the past gradient.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{8}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{9}$$

m_t , v_t are estimates of the first moment and the second moment of the gradients. At first these values are set to be 0. Then, m_t , v_t are divided by$1-\beta_1$, $1-\beta_2$, respectively.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{10}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{11}$$

Finally, by using these to update parameters, we get the formula of Adam.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{12}$$

The number of stride or map size are shown in Fig.12. As for the initialization of the random weights in convolutional layers, I will use uniform distribution.