Capstone Project

Yatin Gupta

Machine Learning Engineer Nanodegree

March 11, 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.

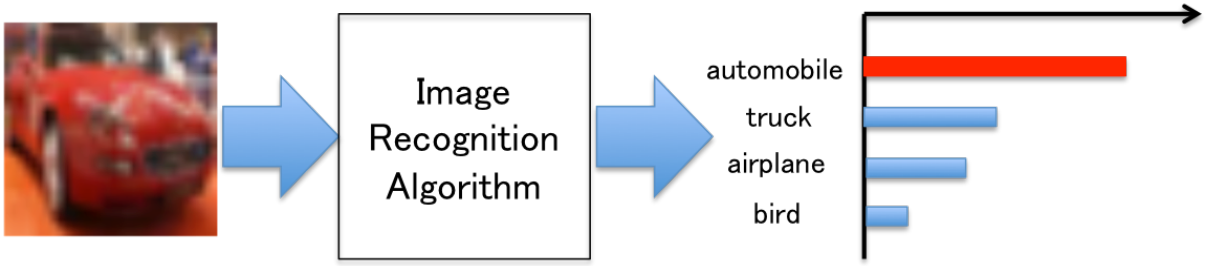

Figure 1: Image of the Algorithm

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.

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

I opted for accuracy score as it is multiclass classification which is based on probability(softmax). Softmax is kind of Multi Class Sigmoid, but if we see the function of Softmax, the sum of all softmax units are supposed to be 1. In sigmoid it's not really necessary. Digging deep, we can also use sigmoid for multi-class classification. When we use a softmax, basically we get a probability of each class, (join distribution and a multinomial likelihood) whose sum is bound to be one. In the case of softmax, increasing the output value of one class makes the the others go down (sigma=1). So, sigmoids can probably be preferred over softmax when our outputs are independent of one another. To put it more simple, if there are multiple classes and each input can belong to exactly one class, then it absolutely makes sense to use softmax, in the other cases, sigmoid seems better. One more thing is, I have used ReLu activations(in the hidden layers) and using sigmoid blows up ReLu apparently.

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'.

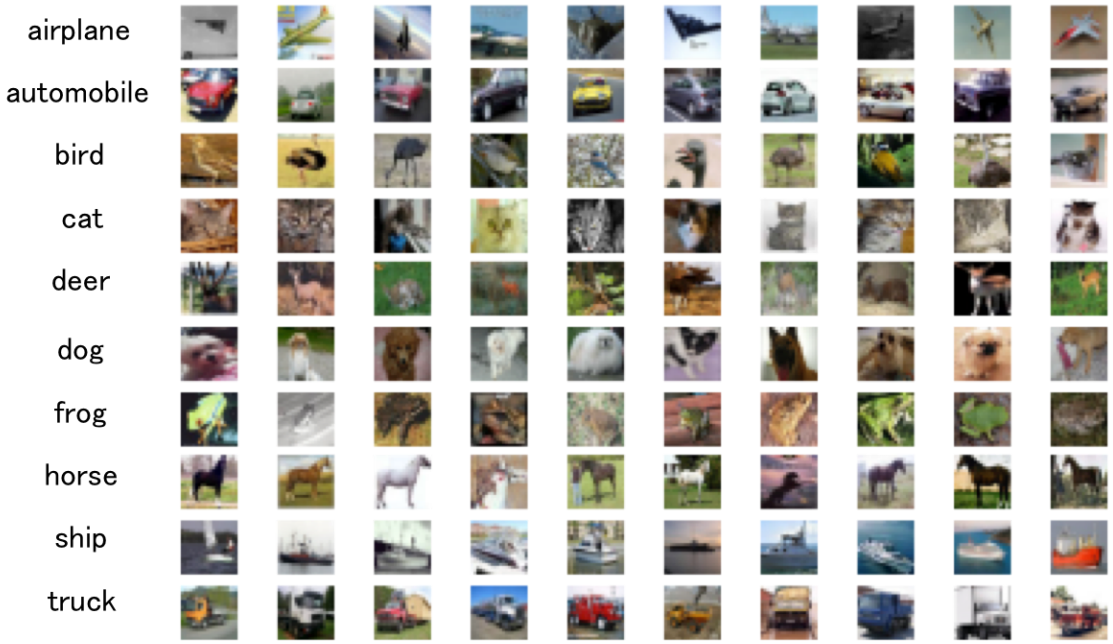

Figure 2: Sample of the Images

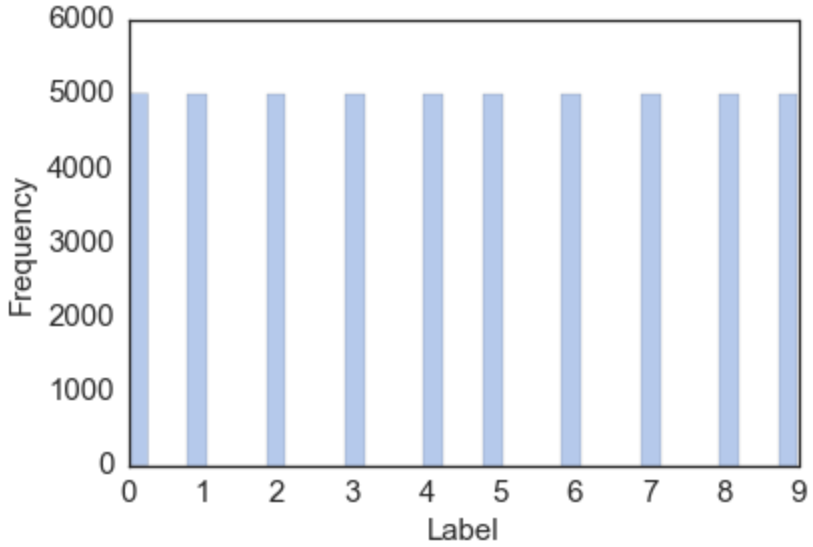

Figure 3: Distribution of Training Data

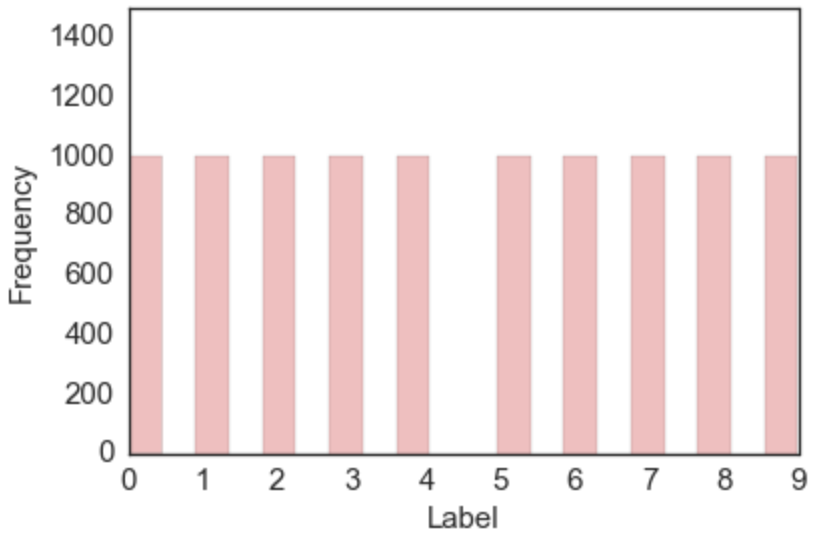

Figure 4: Distribution of Test Data

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]

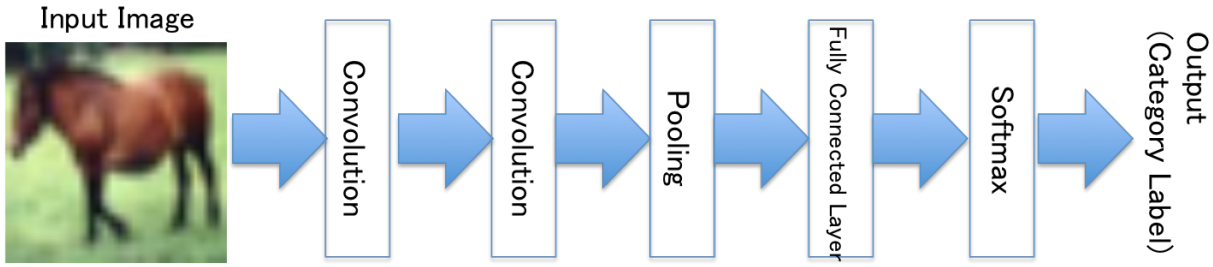

Figure 5: An example of CNN Architecture

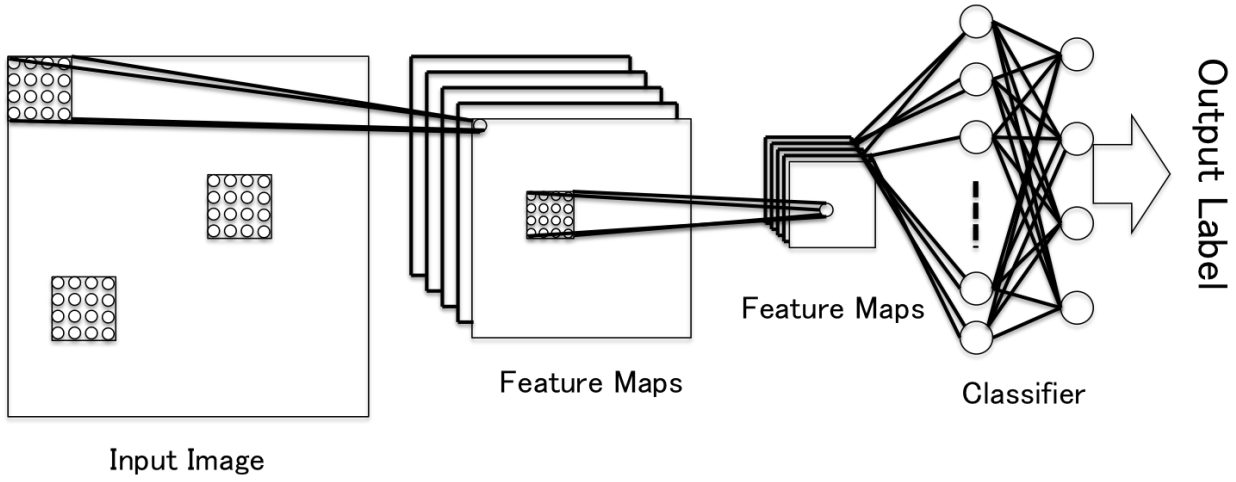

Figure 6: Overview of CNN

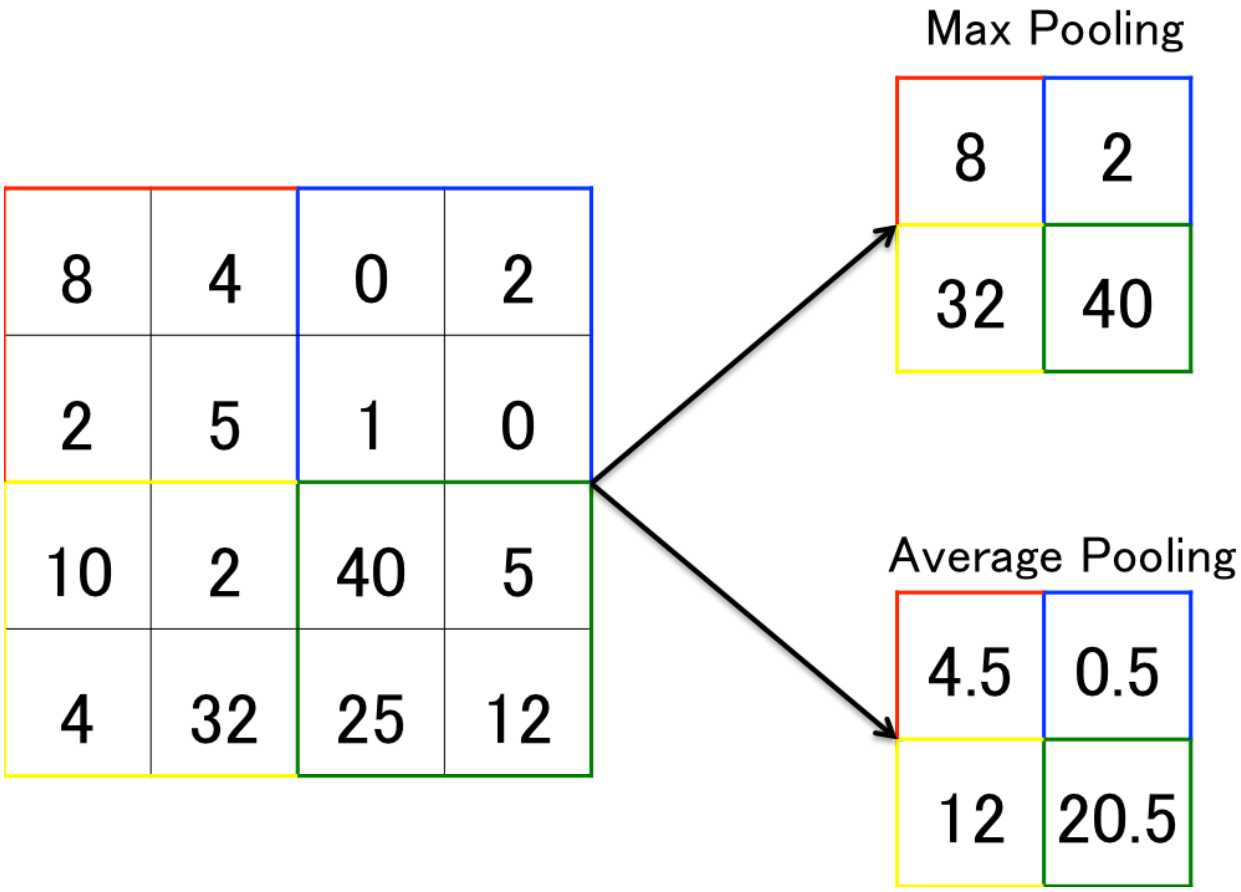

Figure 7: Example of Max Pooling and Average Pooling

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.

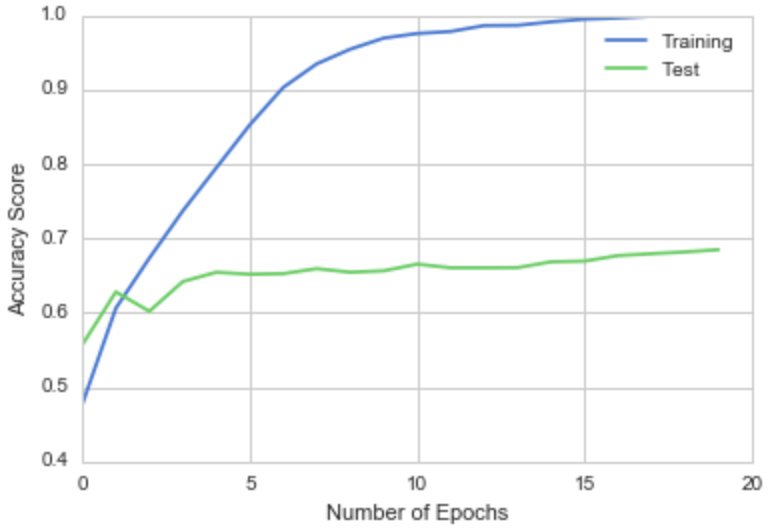

Figure 8: Accuracy rate of training and validation data

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.

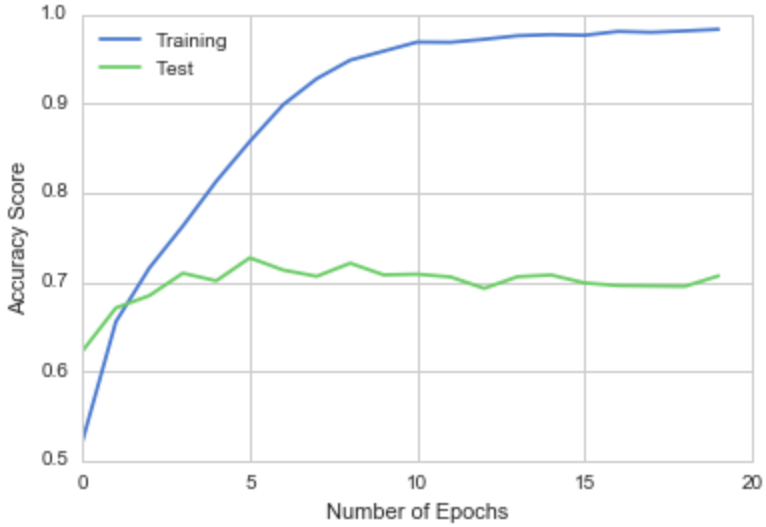

Figure 9: Accuracy rate of training and validation data

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.

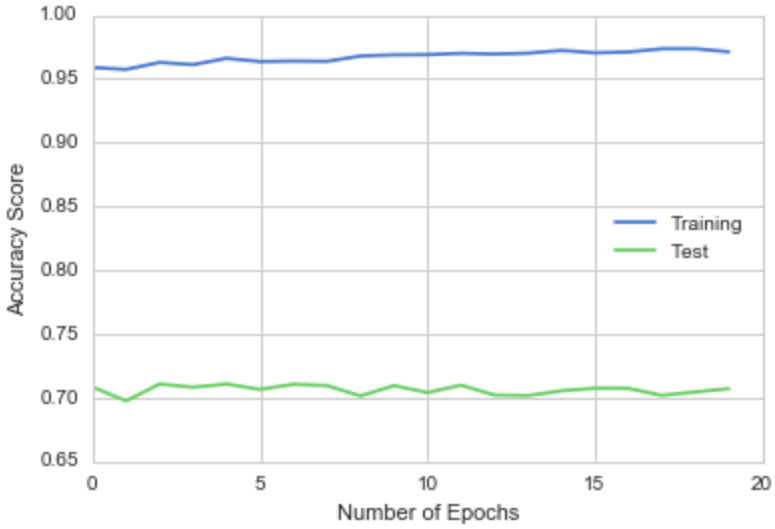

Figure 10: Accuracy rate of training and validation data

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.

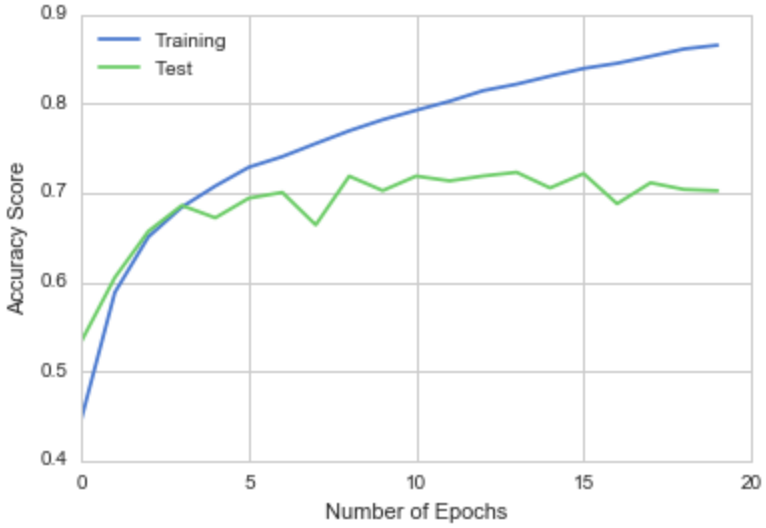

Figure 11: Accuracy rate of training and validation data

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.

3. Methodology

3.1 Data Preprocessing

Apart from the models of image processing or computer vision, CNN doesn't need complex preprocessing since the heavy lifting is done by the algorithm which is the one in charge of learning the features. It only makes sense to apply preprocessing if you have a reason to believe that different input features have different scales (or units), but they should be of approximately equal importance to the learning algorithm. In case of images, the relative scales of pixels are already approximately equal (and in range from 0 to 255), so it is not strictly necessary to perform this additional preprocessing step(normalization).

However,when analyzing the data, data preprocessing plays a crucial role. One of the example of data preprocessing is Data Augmentation and indeed helps CNN. Data Augmentation always improves performance though the amount depends on the dataset. If you want to augmented the data to artificially increase the size of the dataset you can do the following if the case applies (it wouldn't apply if for example were images of houses or people where if you rotate them 180 degrees they would lose all information but not if you flip them like a mirror does):

- rotation: random with angle between 0° and 360° (uniform)
- translation: random with shift between -10 and 10 pixels (uniform)
- rescaling: random with scale factor between 1/1.6 and 1.6 (log-uniform)
- flipping: yes or no (bernoulli)
- shearing: random with angle between -20° and 20° (uniform)
- stretching: random with stretch factor between 1/1.3 and 1.3 (log-uniform)

One of the first steps is the normalization of the data. This step is essential when dealing with parameters of different units and scales. In this dataset, pixel values range from 0 to 255. I processed normalization to this dataset. Normalization scales all numeric variables in the range of [0,1].The formula is given below.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{6}$$

The minimum pixel value is 0 and maximum pixel value is 255. Therefore, I normalized the data by following.

$$x_{new} = \frac{x}{255} \tag{7}$$

3.2 Implementation

As I introduced in the section 2.4(Benchmark), I utilized the 4 convolutional layers and 2 fully connected layers. In this section, I will explain the architecture of the model in more depth.

The objective of the learning is to minimize the cross entropy in the learning process. The architecture of the model is the following.

Layer	Batch	Stride	Map Size	Function
data	–	–	3 × 32 × 32	–
conv1	3 × 3	1	32 × 32 × 32	ReLu
pool1	2 × 2	2	32 × 16 × 16	–
conv2	3 × 3	1	32 × 16 × 16	ReLu
pool2	2 × 2	2	32 × 8 × 8	–
conv3	3 × 3	1	32 × 8 × 8	ReLu
pool3	2 × 2	2	32 × 4 × 4	–
conv4	3 × 3	1	32 × 4 × 4	ReLu
pool4	2 × 2	2	32 × 2 × 2	–
fc5	–	–	512	ReLu
fc6	–	–	10	Softmax

Figure 12: Architecture of the model

To train the CNN model, I used Keras which is one of the neural network libraries like Theano or Tensorflow.

As for the optimizer, I utilized 'Adam'. The important parameters training the model is as follows. Optimization parameters:Learning rate α=0.001, β_1=0.9, β_2 = 0.999, $\varepsilon = 1.0 \times 10^{-8}$ [6]

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 used uniform distribution. I used normalized uniform distribution which is proposed by Glorot[7] for the initialization of the weights in fully connected layers I used ReLU(Rectified Linear Unit) as the activation except for the final layer and I used softmax as the final layer of activation.

$$a_{ReLU} = \log(1 + \exp(x)) \simeq \max(0, x) \tag{13}$$

With this setting, I got the accuracy rate of 70.2%

Now am going to discuss the complications that I encountered in the coding process. I had to decide layers of CNN architecture(as I had to go through multiple architectures and finally decide upon the most efficient one) and then also tune parameters in a way to obtain optimal parameters. Apart from that I had no issues with the implementation.

3.3 Refinement

Finding the optimal parameters for deep learning is quite difficult though it is important. When it comes to typical machine learning algorithm(Decision tree, Support Vector Machine etc.), grid search is taken to search the optimal parameters. However, it's almost impossible to apply grid search in deep learning because of the computational time. It has reported that the grid search strategies are inferior to random search.[8] Therefore, other methods are indispensable. A good choice is Bayesian optimization, which has been shown to outperform other state of the art global optimization algorithms on a number of challenging optimization benchmark functions.

Bayesian Optimization[4]

For continuous functions,Bayesian optimization typically works by assuming the unknown function was sampled from a Gaussian process and maintains a posterior distribution for this function as observations are made or, in our case, as the results of running learning algorithm experiments with different hyper-parameters are observed.

There are three steps for the Bayesian Optimization.

1. Suppose the prior distribution that express assumptions about the function being optimized
2. Decide the posterior distribution of function by evaluating the randomly sampled sample data
3. Evaluate the posterior distribution of function and the candidates of the sample data.

When implementing Bayesian Optimization, we have to pay attention to two major factors.

First, we have to suppose the prior distribution of the blackbox function. The function is difficult to be optimized directly so that we have to approximate the function. For the flexibility and the tractability, Gaussian process has been chosen.Gaussian Process is a probabilistic model of regressor which can predict the new output $y^{(n+1)}$ for $x^{(x+1)}$ when given the data D = $\{(x^{(n)}), (y^{(n)})\}_{n=1}^{N}$ For more overview of Gaussian process, please refer to Rasmussen and Williams[5] Gaussian process needs mean and covariance as parameters.We have to use kernel to calculate covariance.As for the Covariance function, squared exponential kernel is often used as a default function for Gaussian Process regression.

$$K_{SE}(x, x') = \theta_0 \exp\left(-\frac{1}{2} r^2(x, x')\right) \tag{14}$$

$$r^2(x, x') = \sum_{d=1}^{D} (x_d - x'_d)^2 / \theta_d^2 \tag{15}$$

However, sample functions with this covariance function aren't smooth for practical use. Therefore, I use ARD Matern 5/2 kernel instead.

$$K_{M52}(x, x') = \theta_0 (1 + \sqrt{5r^2(x, x')} + \frac{5}{3} r^2(x, x')) \exp\left(-\sqrt{5r^2(x, x')}\right) \tag{16}$$

Second, we have to decide the sampling points to explore after the blackbox function is approximated by gaussian process. To get the good sample, acquisition function is used. As for the acquisition function for Bayesian Optimization, I used GP Upper Confidence Bound

$$a_{UCB}(x; \{x_n, y_n\}, \theta) = \mu(x; \{x_n, y_n\}, \theta) + \kappa\sigma(x; \{x_n, y_n\}, \theta) \tag{17}$$

Here, κ is tunable parameter to balance exploitation against exploration.I set κ as 2.576.
To implement the ARD Matern 5/2 kernel, I used BayesianOptimization library [10]. I optimize the number of the dimensions of the first fully-connected layer and the learning rate in the optimizer(Adam) in this task. The range of exploring values of these are between 10 to 1024 and between 0.0001 to 0.01,respectively. 25 of randomly sampled values are taken to find the optimal values by Bayesian optimization.

4. Result

4.1 Model Evaluation,Validation and Justification

By utilizing the bayesian optimization, I optimized the number of dimensions of first layer of fully connected layer and learning rate. I got the optimal number of first layer of fully connected layer and the learning rate of the optimizer.Those numbers are 204 and 0.000597,respectively. By using those numbers, I got the test accuracy which is 72.1%

From the result of Bayesian Optimization, I chose 204 of first fully connected layer and 0.000597 of learning rate as the final model parameters. The training and test accuracy in this model is as follows(Fig.13).

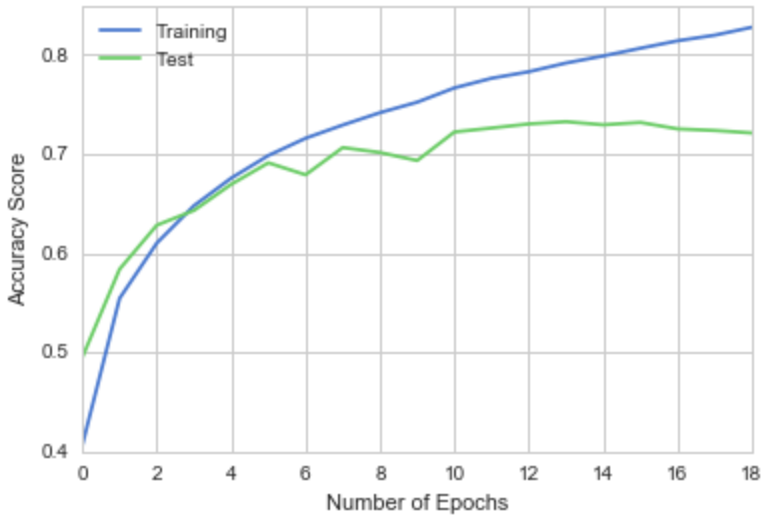

Figure 13: Accuracy rate of training and validation data

The loss of the model is shown in Fig.14

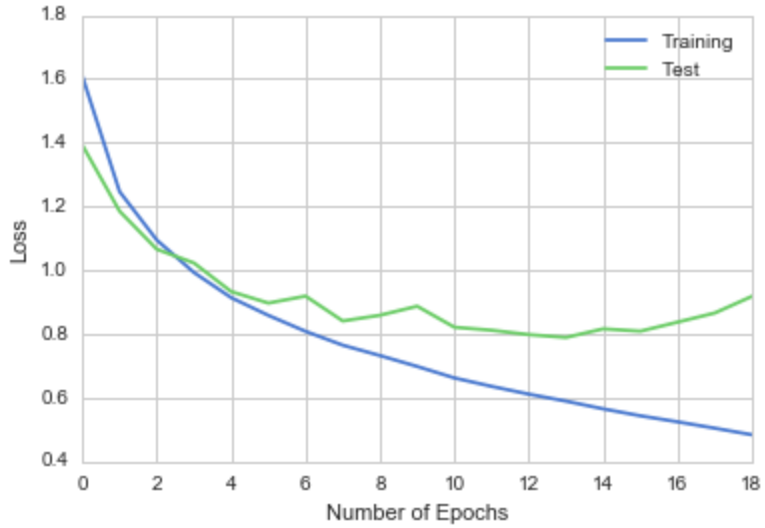

Figure 14: Loss of training and validation data

The model improves the accuracy rate by 2.0%. The final model doesn't seem to fall into the overfitting according to the Fig.13 and when compared to the Fig.11 which is the benchmark result, the difference between the training and test accuracy rate is much smaller. I made 6 new datasets of CIFAR-10 by combining the original training and the original test data and then splitting it. The new dataset has the same distribution of each class as the original dataset had. I run the benchmark model and the final model for the new 6 datasets. The result is the figure below.

Therefore, the model improved compared to the benchmark result.

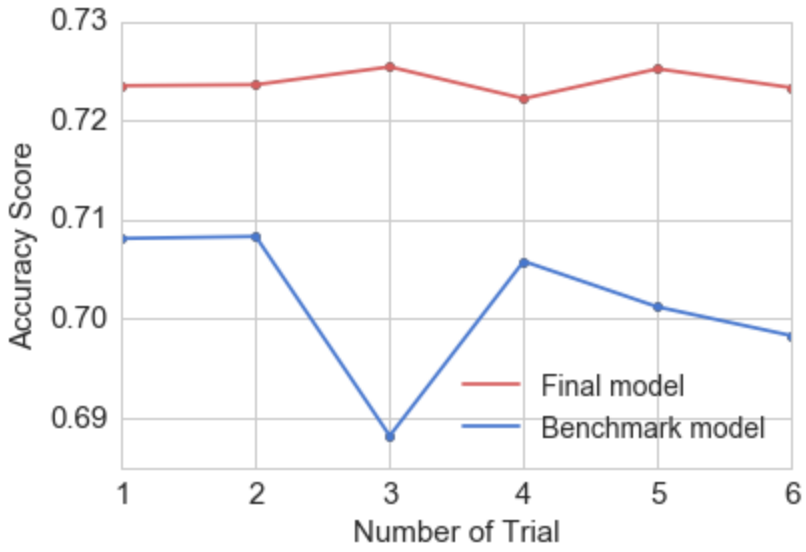

Figure 15: Accuracy Score of the benchmark model and the Final model

For the 6 new datasets, the accuracy score of the final model surpasses the accuracy score of the benchmark model. That means the final model is robuster and better than the benchmark model.

I only optimize two parameters, learning rate and the number of first fully connected layer's neurons. The number of random sampling points of finding the optimal score in bayesian network is 25. When using grid search, we can only search 5 values for each parameters(5×5=25). Apparently 5 different values for each parameter is not enough to search optimal parameters for grid search. Therefore, much more different values are necessary, resulting much computationally expensive. The difference will be much distinguishable when I try to optimize more parameters. In this case, grid search won't be a choice to find optimal parameter because of the scalability. As for bayesian optimization, it is highly possible that I can't find the best parameters within 25 sample points. However, the parameters found within 25 sample points are better result than the benchmark model parameters. When I choose more sample points, it will probably get better results.

5. Conclusion

5.1 Free-Form Visualization

Some data are miss-classified in this task. I'll look into which label is the poorest result and I'll put some of the images which are miss-classified. The distribution of the correctly classified images are Fig.15

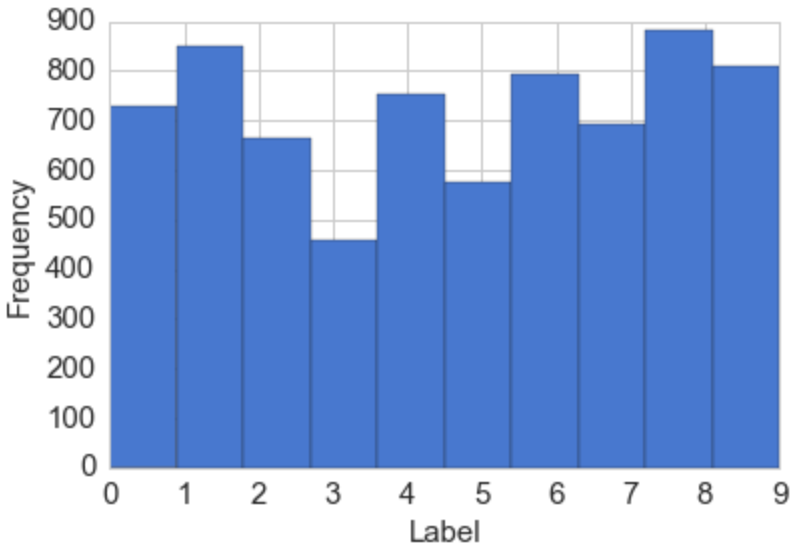

Figure 16: Distribution of the correct prediction

As you can see, the number of label 3 which is ”cat” is apparently small.That means ”cat” is overly miss-classified in this dataset.

The heatmap below clearly explains the miss-classified data. Label 3(”cat”) is miss-classified as label 5(”dog”). Label 2(”bird”) and label 7(”horse”) are miss-classified as label 4(”deer”). Label 1(”automobile”) is miss-classified as label 9(”truck”).Judging from the heatmap, the miss-classified data are somewhat understandable since some cats are similar to dogs and some automobiles may be similar to ship because of the color and shape(Low resolution may be another reason).

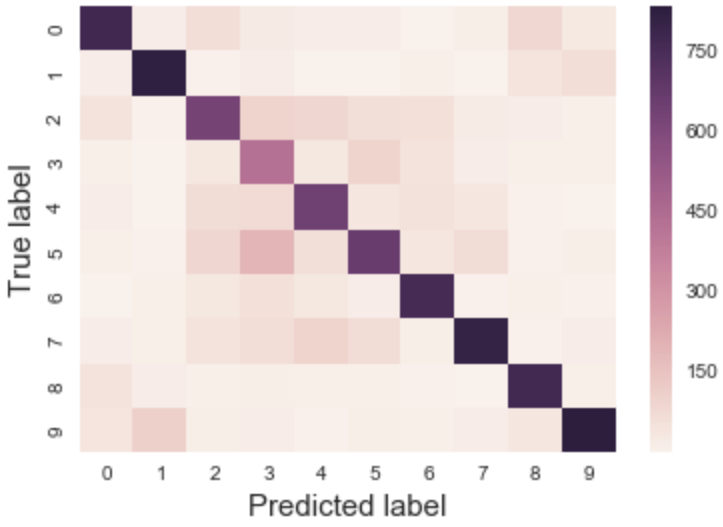

Figure 17: Heatmap of the data

The sample of miss classified pictures are below. 10 images are picked up for each class.

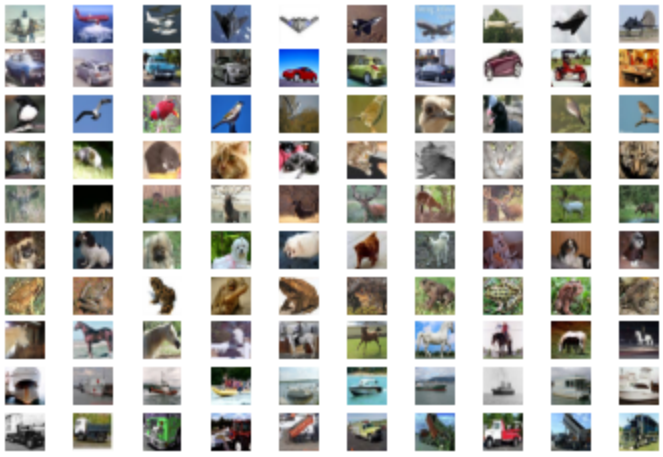

Figure 18: Miss classified pictures

5.2 Reflection

In this project, I solved the image recognition problem. These days, deep neural networks surpass the other typical image recognition algorithms and even surpass human recognition. However, one of the problems of deep learning is tuning hyper-parameters. As is often said, the architecture of the neural network is black art.Putting more layers may tend to grasp the feature of the input data. However, because of the back-propagation process, the gradient will vanish or explode while calculating. Therefore, the deep neural networks don't always reach the good result. What's more, the deep networks tend to take much more time than the shallower ones. Personally, I don't have any GPU environment so that trying deep network was quite tough.Therefore, I tried training the shallow networks with hyper-parameter tuning.

First, I decided the number of layers of convolutional layers and then I optimized the parameters with bayesian optimization which is one of the methods to tune hyper-parameters. I chose 4 layers of convolutional layers and 2 layers of fully connected layers for the benchmark, then I optimized the number of dimension of the first layer of the fully connected layer with bayesian optimization.

One of the difficulties of this task was parameter tuning. At first, I tried to tune by grid search which is quite popular in the machine learning field. However, I soon realized the method isn't reasonable because of the number of the parameters and the computational time. Thus I chose other hyper-parameter tuning method which is bayesian optimization. I chose two parameters to optimize in this task by bayesian optimization. Bayesian optimization is superior to the grid search for for the scalability reason. Grid search isn't good choice if the number of tuning parameters is high. If there are 5 parameters which take 5 different values, the number of iteration is $5 \times 5 \times 5 \times 5 \times 5 = 3125$. Some of the parameters will not be the optimal parameters. However, if we choose Bayesian optimization, the apparently incorrect parameters can be avoided by the acquisition function.

5.3 Improvement

In this task, I chose 4 layers of convolutions and 2 layers of fully connected layers. It is reported that the number of layers is the crucial factor in the high accuracy rate. For example, GoogLeNet,VGG,SPP are around 20 convolution layers. In 2015, ResNet which surpassed the human recognition had maximum 152 layers. In fact, deep networks can represent certain function classes far more efficiently than shallow ones. That means basically deeper networks can get the higher accuracy rate.

However, the stacking of several non-linear transformations in convolutional feed-forward network architectures typically result in poor propagation of activations and gradients. Therefore, training deep network is difficult. ResNet takes the strategy of skipping connections between layers. When the number of layers increases, finding optimal parameters becomes more and more important. Bayesian optimization can be one of the methods to find the optimal parameter.

References

[1] Rupesh Kumar Srivastava, Klaus Greff, Jurgen Schmidhuber, Training Very Deep Networks,arXiv:1507.06228v2[cs.LG]23 Nov 2015,2015

[2] Benjamin Graham.Fractional Max-Pooling,arXiv:1412.6071v4[cs.CV] 12 May2015,2015

[3] Sebastian Raschka,Python Machine Learning,Packt Publishing.2015

[4] Jasper Snoek,Hugo Larochelle,Ryan P.Adams,Practical Bayesian Optimization of Machine Learning Algorithms

[5] Carl E.Rasmussen and Christopher Williams. Gaussian Process for Machine Learning, MIT Press,2006.

[6] Diederik Kingma and Jimmy Ba. Adam: a method for stochastic optimization, In the 3rd International Conference for Learning Representations ICLR 2015,2015

[7] Xavier Glorot, Yoshua Bengio,Understanding the difficulty of training deep feedforward neural networks, JMLR W&CP 9:249-256, 2010

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. Journal of Machine Learning Research,13:281-305,2012

[9] https://en.wikipedia.org/wiki/Convolutional_neural_network

[10] https://github.com/fmfn/BayesianOptimization