Mitchell Challenge Report

Chenlei Zhang ztimemakercl@gmail.com

I System Overview

In order to build up a model to predict the class of the object image, I used a TensorFlow based Convolutional Neural Network to train the model for the data of 10 image classes. There are three major parts in one CNN network, first is training the model, second is validating the model, the thrid is testing and evaluating the model. In the following sections, I present the details about how to prepare the data, the mathmetical methods behind the CNN.

1. System Components

There are 5 major layers in this network:

- conv1: the first convolution layer with one max pooling layer and one normalization layer.
- conv2: the second convolution layer with one max pooling layer and one normalization layer.
- local3: fully connected layer.
- local4: fully connected layer.
- softmax_linear: linear classifier.

2. Mathematic Details

Activation Function

(Rectified Linear Unit) ReLU was used as the activation function in this network. The function shows as follow:

$$ReLU = max(0, x)$$

ReLU is a commanly used activation function, it solves the gradient vanishing problem of Sigmoid function, it is more efficient than Sigmoid and tanh function to converge.

Classification Layer

After convolutions and fully connected, the softmax classifier can project the output of each neron into the [0, 1] range of each class. For example, if we have one matrix V. V_i is the i element in this matrix, then the softmax value of this element is:

$$S_i = \frac{e^i}{\sum_j e^j}$$

Softmax can predict the class of the object by using the combination of the probablities of falling into each class, so it is a great method to do multi-class classification.

Loss Function

Cross entropy was used as the loss function, which can be calculated as:

$$H_y(p) = -\sum_i y_i \log(p_i)$$

where y is the binary label (0 or 1) of the object, p is the predicted probablity observation. Cross entropy is a classical loss function for the multi-class classification problem. It can precisely describe the difference of each hypothesis model, hence it can help optimize the model better than other loss functions in classification problem.

Evaluation Metrics

The caculation of the precision of the model is,

$$precision = \frac{true\ count}{total\ count}$$

For each image in the testing dataset, use the trained model to predict the class of it. If the result is same as the label of the image, counting this prediction as $true\ count$. The number of the image is the $total\ count$. The precision of this model is the quotient of the $true\ count$ and the $total\ count$.

II Experience and Improvements

1. Change max_steps

The model may not converge if the epoch of the learning process is not big enough. I tested two situations,

max_steps	precision
10000	0.815
100000	0.863

2. Data Augmentation

A small change or normalization on the input data may increase the model precision. I used the Whitening method to normalize the image as a Gaussian distribution image. This method can reduce the correlation between features of the images.

$$x_{rot}^i = U^T x^i, \quad x_{white,i} = \frac{x_{rot,i}}{\sqrt{\lambda_i}}$$

where λ is the diagonal value.

3. Modify Learning Rate

Decreasing the learning rate slightly can increase the fitting capacity of the model, so that increase the precision of the model.

Learning Rate	Epoch	Precision
0.01	10000	0.785
0.0005	10000	0.824

4. Enhance the Network Structure

Add Local Response Normalization (LRN) layer after the max pooling layer.

$$b_{x,y}^{i} = a_{x,y}^{i} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^{j})^2 \right)^{\beta}$$

where $a_{x,y}^{i}$ is the output of the previous layer, N is the number of the feature map. LNR can increase the output of the high influence neuron and decrease the output of the low influence neuron, hence it can enhance the generalization ability of the model.

III Conclusion

After several optimizations, the model can reach the precision of 0.863.

We still can increase the precision by using deeper CNN, which requires a better computing environment.

IV User Manual for Testing the Model

Environment Requirements

Python 2.7, and TensorFlow 1.1

It will be better if you can test it in a Linux system.

Present the result of the model

Please run the following command in the terminal, it will download the testing data firstly and then evaluate the trained model which was stored in *cifar10_train/* folder.

```
python cifar10_eval.py
```