Capstone Project
Machine Learning Engineer Nanodegree

Haoran Zhang
October 15st, 2016

Definition

Project Overview

This project belongs to computer vision, and more detailed, belongs to an image classification problem. This project is for detecting and reading digits from real world images. Comparing to the well-studied Optical Character Recognition(OCR) technic, reading digits from a more complex background is a far more difficult goal. The origin of this project is from Google Maps. When Google Street View Cars range on road, they take images of every house number. And by what we doing in this project, images of house numbers can be recognized and stored in digits.

The data set of the project is The Street View House Numbers (SVHN) Dataset, which including around 60,000 labeled images of house numbers. By using partial of the Full Name format data, the project can produce a good model on recognize digits.

Problem Statement

Given images with labels, this problem is a supervised problem. Also, the problem is a classification problem. Because in the end, the model need classifiers to determine which type of total 11 types does the image belong to.

To deal with this problem, I choose to use convolution neural network. Because computer vision is always related to a huge number of features, for example a common 720p image contains 921,600 pixels, which is a disaster if we take every pixel for a feature and use regular machine learning classification method. Instead, dealing with images is a strength of convolution neural network.

By using convolution and pooling method, the picture can be compressed without losing too much features to classifying. After convolution process, five classifiers are used for determining on every word, which because we assume the length of house numbers is no longer than five digits. After thousands time of training on a batch of data set, the neural network can gradually learn what features are useful for the recognition, as well as the five classifier can learning to classify. According to the result, this method

can really work and performs good on recognition purpose.

Metrics

The metrics I use for measuring the model is simple, only the accuracy of the prediction. By seeing five positions of digits separately, if a prediction of a poison is correct, the correct number will add one. In training process, actually, you can see the model can easily get 50% accuracy, because it's easy to recognize whether there is any digit on the fourth or fifth digit position. After that, more training process will be needed for more accurate prediction on what the number is of the first or the second position.

The reason I use accuracy as my only metric, which is every classifier may choose one digits from 0-9 ten digits plus one empty signal, while other metrics like recall and F1 score are usually used for binary classification.

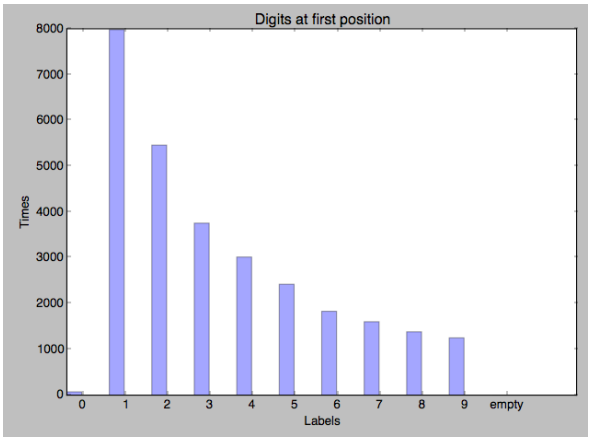
Analysis

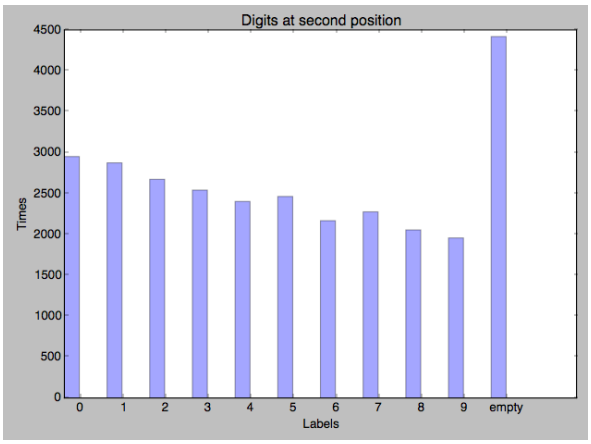
Data Exploration

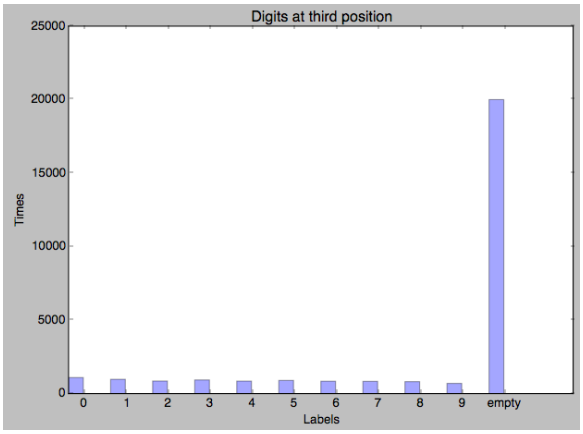
The dataset the project used is The Street View House Numbers (SVHN) Dataset, which contains around 60,000 labeled images. Actually, the data set consists of three parts, train.tar.gz, test.tar.gz and extra.tar.gz. And as the document of the dataset says, the extra.tar.gz contains around 30,000 and more easy images, for extra training need. After I trained my first model without extra.tar.gz, the performance is over 90%, so I think the extra easy data is useless in my model.

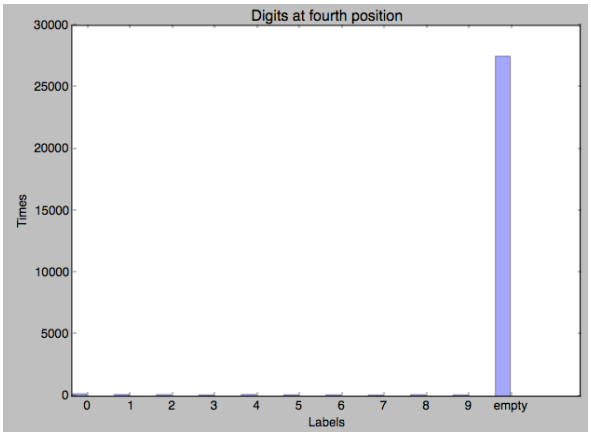
I choose to use the Full Name format data, which is provided raw image of house numbers with positions of every digit, as well as labels. The train.tar.gz and test.tar.gz totally contains around 30,000 images.

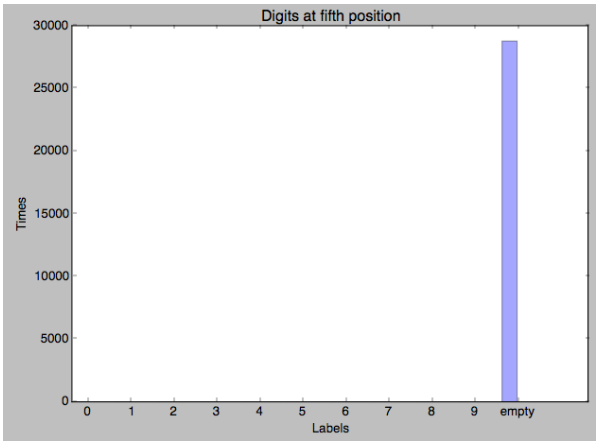
The data set is skewed. The method I use is to set 5 classifiers, however, most lengths are less than 3 digits. So that the data of 3^{rd}, 4^{th} and 5^{th} digits is very skewed. As we can see from the following plots, labels at every position is skewed.


Digits at first position
Times
Labels
0
1
2
3
4
5
6
7
8
9
empty
1000
2000
3000
4000
5000
6000
7000
8000


Digits at second position
Times
Labels
0
1
2
3
4
5
6
7
8
9
empty
500
1000
1500
2000
2500
3000
3500
4000
4500

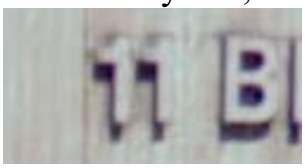

Digits at third position
Times
Labels
0
1
2
3
4
5
6
7
8
9
empty
5000
10000
15000
20000
25000


Digits at fourth position
Times
Labels
0
1
2
3
4
5
6
7
8
9
empty
5000
10000
15000
20000
25000
30000



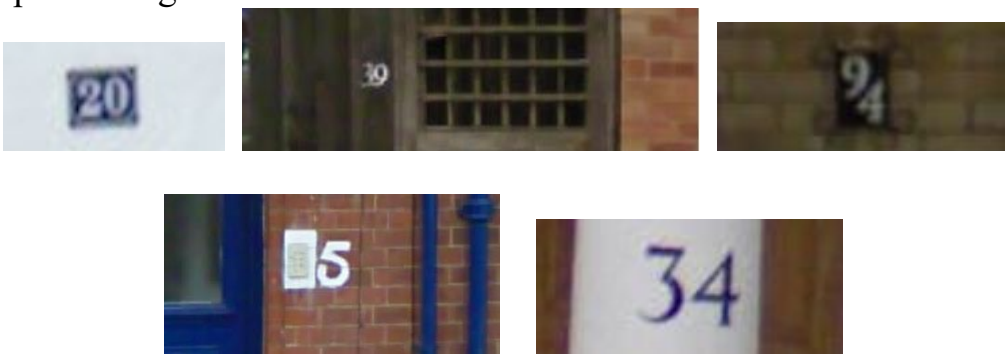
Histogram of labels at every position

Every length of the house numbers is less than 5 digits. However, some house numbers include characters like “11B”, these cannot be recognized. Fortunately, this kind of errors is every few, and can be ignored.



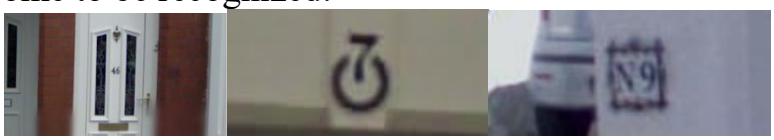
Exploratory Visualization

1. Here are some examples of raw image.
The images are clear, however, containing different fonts, colors and complex background.



2. After being cropped and resized into gray scale images (32*32 pixels). As we can see, the gray scale images, which is actually input into model, are also clear and keep most details.

3. Troublesome to be recognized.

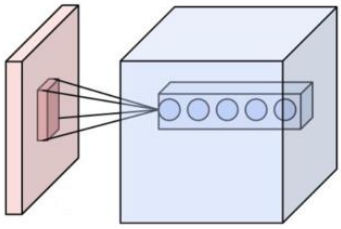


Take this tree images for example. The first images, after being resize, the digits will be too small for the model to recognize. The second image, I think this is too much puzzling for the model. And the third image, combine same font character and digit, maybe impossible for model to distinguish.

Algorithms and Techniques

The algorithm I use for dealing with this problem is CNN algorithm. The reason I choose CNN algorithm because it's more suitable for computer vision problem. Before the prevailing of CNN, other methods of computer vision like sliding window algorithm with SVM classifier once was successful. The pros of CNN is high accuracy in image recognition problem. And cons of CNN, is that it's high computational cost, however, by using GPU and dramatically increase the compute speed, so this con seems less malign. Another con is CNN use a lot of training data. As the situation we chase for higher accuracy, I think CNN is suitable for this problem.

The core of mechanism in CNN is the application of convolution layer: "The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when they see some specific type of feature at some spatial position in the input. "[1]



Neurons of a convolutional layer, connected to their receptive field

In this model, the convolution neural network is the primary techniques.

1. I crop and resize the raw images into 32*32 gray scale images, and pixel divided by 256. So the size of input of every image is 32*32*1.

Here, I resize of images. According to the paper[2] , the author chose the

[1] Wikipedia. Convolutional neural network, https://en.wikipedia.org/wiki/Convolutional_neural_network
2 Goodfellow I J, Bulatov Y, Ibarz J, et al. Multi-digit number recognition from street view imagery using deep

input of size 64*64, which means the length-width ratio doesn't infer the result. Considering the performance of my computer, I choose 32*32 as input size.

2. At the first convolution layer, which has shape of 5*5, and turns feature maps from 1 to 16, with valid padding. The images become 28*28*16.

3. Then a max pooling layer turns images into 14*14*16.

4. At the second convolution layer, also has the shape of 5*5, transfers the image into 10*10*64.

5. Then the second max pooling layer, turns the images into 5*5*64.

6.By the experiment, the deeper inception model can easily go overfitting, so one more dropout layer is set here with ratio of 0.5

7. A inception model was implemented here. According to the article, the inception contains three parts, 1*1 conv net, 3*3 conv net after 1*1 conv net, and 3*3 max pooling layer after 1*1 conv net. The input goes through these three parts then be concatenated as an output with size of 5*5*160. By using inception model, the feature maps increase a lot, which means the neural network is much deeper.

8. In order to decrease the parameters, the third convolution layer is implemented, which turns 5*5*160 into 1*1*256.

9. Then, a fully connected layer with input size of 1*1*256 and output size of 1024 is set.

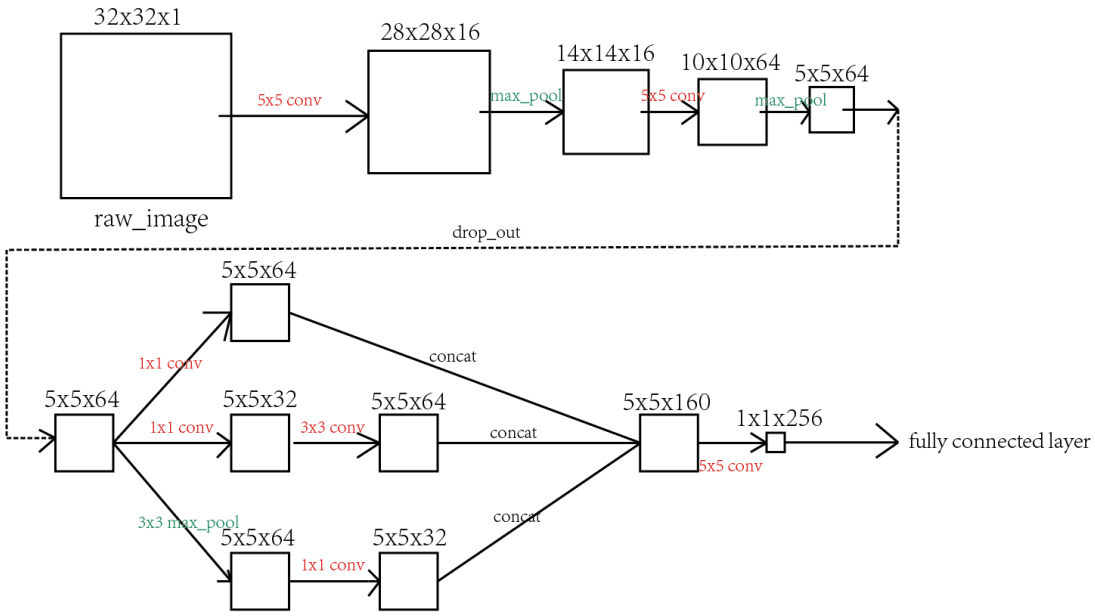
10. The second dropout layer with ratio of 0.5 was implemented after a fully connected layer.

11. Five separate classifiers was implemented after the dropout layer. Each turns the list of 1024 nodes into a list of 11 nodes. Then, softmax and cross entropy are used for calculate total loss on five classifiers.

12. By using adagrad optimizer with decreasing learning rate, the model can be trained and corrected recognized the numbers in images.

---

convolutional neural networks[J]. arXiv preprint arXiv:1312.6082, 2013.



Core part of project model

Besides the model, I'd like to introduce the method I used for evaluate. The SVHN Dataset only contains training dataset and test dataset, and I choose 4617 items from the training dataset to form a validation set. After being shuffled, at every 1000 steps, the model should print out the accuracy on this batch, as well as random select 100 items from validation set, with dropout ratio with 1.0, to evaluated prediction accuracy. After all the training, the whole test set will be used for a total accuracy.

The model is accuracy in recognizing digits from validation set. To avoid overfitting, I choose to use two dropout layers in the model. And as we can see, the accuracy on batch is similar to the accuracy on random validation set, which means while the loss decrease slowly, it's very robust and without worries of overfitting.

Benchmark

According to the paper cited before, "Multi-digit number recognition from street view imagery using deep convolutional neural networks", the google team get 97.84% accuracy, as well as human operator performance is around 98%. This model choose 97.84% as a benchmark, however, with far less computation ability, my model can hardly reach the benchmark.

Methodology

Data Preprocessing

As I mention before, the SVHN Dataset only contains training dataset and test dataset. I choose 4617 items from the training dataset to form a validation set. I randomly rearrange the images, turn RGBs into gray scale, and then resize every image into 32*32 size.

As for the labels, I use one-hot encoding, because every digit here is only a symbol. A list of 11 binaries represents a digit, and if the 11th binary equal to 1, means this digit is empty, while if other binary is equal to 1, means this digit is the index of the binary. So, the label of an image can be represented as a 11*5 matrix. The transferred images and labels was packed into a pickle file: train_data.p, test_data.p, va_data.p.

Example of one-hot encoding:

```
[[0,1,0,0,0,0,0,0,0,0,0],
 [0,0,1,0,0,0,0,0,0,0,0],
 [1,0,0,0,0,0,0,0,0,0,0],
 [0,0,0,0,0,0,0,0,0,0,1],
 [0,0,0,0,0,0,0,0,0,0,1]]
```

All this preprocessing is to produce a uniform and easy to process data, which to be used at training process. The outliers, which is very few but hard to be drop out, would stay in dataset, which doesn't affect the final performance of the model.

Implementation

I'd like to introduce more about inception model. As I mentioned before, the inception layer in this model contains three parts, 1*1 conv net, 3*3 conv net after 1*1 conv net, and 3*3 max pooling layer after 1*1 conv net. The input goes through these three parts then be concatenated as an output, which turns 5*5 size with 64 feature maps into 5*5 with 160 feature maps.

As far as I know, the use of three different filter, can extract different features. In another word, the filter of 1*1 conv net will extract absolutely different features comparing to 3*3 conv net, simply because the different size of patch. By using three different filters, the information can be utilized as much as possible. Also, by applying 1*1 conv net, in a way, it deeps the network, in another way, it can reduce the feature maps. As I used in my model, a 1*1 conv net transfer feature maps from 64 to 32, then followed by a 3*3 conv net. By this, the total feature maps are less than only using 3*3 conv net. In a word, inception method is a good way to deeps the network without introducing too much parameters to train.

Refinement

At first, I tried not to crop image and directly resize the image into 32*32 and grey scale. The result is very poor. After many times of tuning, while accuracy on batch can reach to 97%, the accuracy on test set is only 72%. The generalization is quite poor.

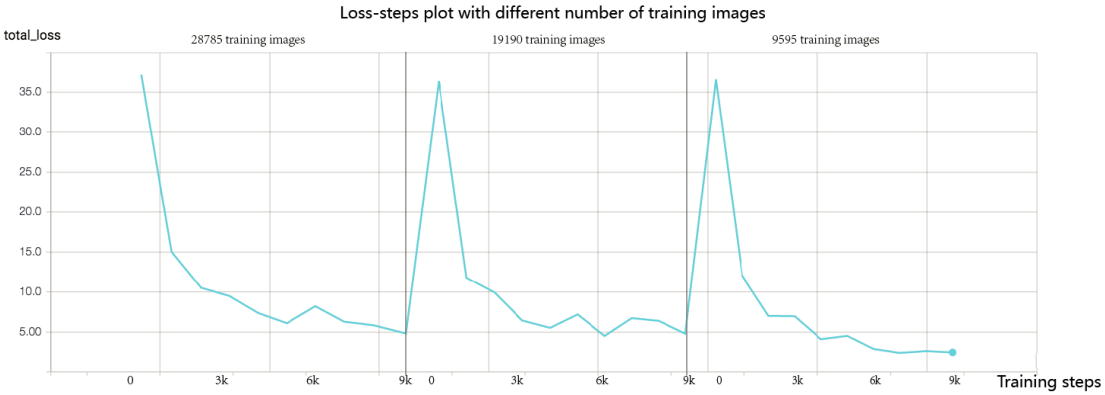
After observation on the result, I find out the images after being directly resized can even hard for human to recognize. Because the different size of house number, some digits are quite blur. To improve this problem, I choose to crop the image before resizing. After this, the accuracy on test set can reach over 80%, with only simplest CNN model. Then I tried out other structure like inception method, which is effective and finally reach to 92.18%, which is good enough to me. The generalization is quite good, too. The accuracy on batch and on validation set is close to which on test set.

Results

Model Evaluation and Validation

The final accuracy on test dataset is 92.18%, which I think is ideal.

To validate the qualities of final model, first I tried different size of training set:



This image shows with different number of training images (full training set, 2/3 of the training set, 1/3 of the training set), how the total loss decline. And this three models are tested on test set, with separately 91. 56%, 90. 34% and 89. 28% accuracy. There are around 13071 images in test set, so the around 1% difference in accuracy is significant. The model with least images, although get least total loss while training, the accuracy on test is the worst, because there is overfitting problems. So the

model is robust with different size of training set. More the training images, more accuracy on the result.

Also, I tried different dropout rate, with dropout rate equal to 0.5, 0.3, 1.0 and 0.7, the comparisons are showed:

The image shows how total loss decline with different dropout rate. The accuracy of four models on test set are: 91.76%, 90.20%, 90.19%, and 91.51%. The dropout rate of 0.5 is the most accuracy on test set. The difference between 91.76% and 91.51% is 0.25%, referring to the size of test set is 13071 images having 65355 digits, the 0.25% difference is around 163 different digits. So the difference is also significant. As we can see, when dropout rate equal to 1.0(which means no dropout), the model gets the least total loss, however the accuracy on test is the most. So, the dropout method can effectively prevent overfitting problem.

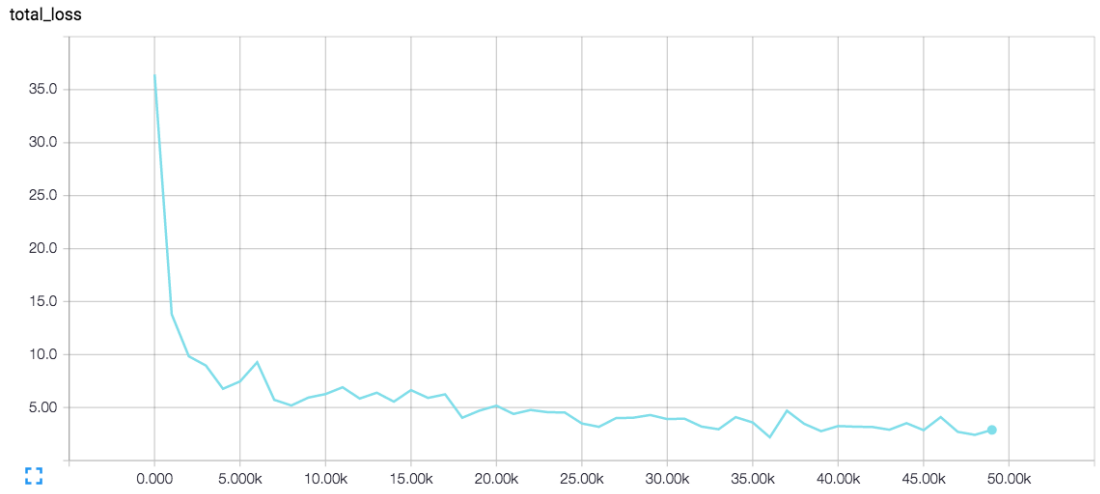
Justification

Although my model is far from benchmark, the performance of my model is good enough for realistic problem. With more data and much deeper and wider neural network, the performance can be improved.

Conclusion

Free-Form Visualization

Loss decline while training steps increase



training steps

I think this picture can tell everything I did. The x-axe is training steps, and y-axe is total loss. While during the training process, the total loss continually declines, means the errors of prediction made by the model is far less. By feeding in batches of training set and around 50k rounds of training, the model can finally learn its way to recognize the digits in images. The accuracy on test set is also good, and the model can be used on practice.

Reflection

I think I have thoroughly summarized the entire process. Feed in the images, tune hyper parameters, and the model will learn by itself to optimize all other parameters, then turn out a number it predicts. All I need to do is finding the most accuracy and as well as an efficient model to recognize images.

The interesting aspects of the project is the usage of Tensorflow, this deep learning tool is quite impressive at its profound method and high efficiency. Using Tensorflow likes using another program language, and the graph-session structure is quite novel. I think I will have more chances to use this powerful tool.

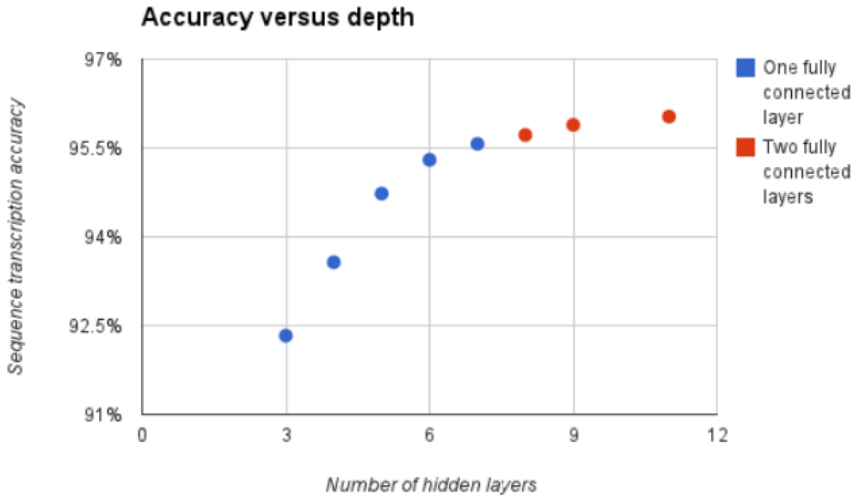
The difficult aspect comes from the natural drawbacks of neural network. The training process cost a lot of time, because the parameters in a model are far more than other classification method like SVM. So tuning hyper-parameters cost a lot of time. Also, neural network has hidden layers, which means what features the model pick to classify are unknown to us. All I

can do is see the final accuracy on batch, on validation set and on test set, and this lead to difficulty in tuning the model.

Improvement

I think the improvement can be implied in two aspects. First, the input image can be bigger. I used 32*32 pixels' images due to the computing capacity. If I use the recommended 64*64 pixels' images from the paper cited before, the cropped images can keep more details and sure to be more accurate.

Second, I think the model can be more complexed, which can make the performance improve. As mention in the cited paper, the team from google tried two fully connected layers with one fully connected layer, the difference is significant. [3] Also, to make improvement, add more layers and units for the neural networks to be deeper and wider are also useful.



The paper mentioned, "Our best architecture …… we use only five convolutional layers ……. The locally connected layers have 128 units per spatial location, while the fully connected layers have 4096 units per layer." These are a huge number of nodes. I think with help of GPU I can reach near to the benchmark.

[3] Goodfellow I J, Bulatov Y, Ibarz J, et al. Multi-digit number recognition from street view imagery using deep convolutional neural networks[J]. arXiv preprint arXiv:1312.6082, 2013.