# Machine Learning Nanodegree Capstone Project – Deep Learning and Digits Recognition

Zhenghe Jin *October 21, 2016*

## 1. Definition

1.1 Project Overview:

In this project, I would like to work on visual recognition via convolutional neural networks. This classical problem in computer vision, image processing, and machine vision is that of determining whether or not the image data contains some specific object, feature, or activity. Different varieties of the recognition problem are described as object recognition, identification and detection：

• Object recognition (also called object classification) – one or several pre-specified or learned objects or object classes can be recognized, usually together with their 2D positions in the image or 3D poses in the scene. Blippar, Google Goggles and LikeThat provide stand-alone programs that illustrate this functionality.

• Identification – an individual instance of an object is recognized. Examples include identification of a specific person's face or fingerprint, identification of handwritten digits, or identification of a specific vehicle.

• Detection – the image data are scanned for a specific condition. Examples include detection of possible abnormal cells or tissues in medical images or detection of a vehicle in an automatic road toll system. Detection based on relatively simple and fast computations is sometimes used for finding smaller regions of interesting image data which can be further analyzed by more computationally demanding techniques to produce a correct interpretation.

Currently, the best algorithms for such tasks are based on convolutional neural networks. An illustration of their capabilities is given by the ImageNet Large Scale Visual Recognition Challenge; this is a benchmark in object classification and detection, with millions of images and hundreds of object classes. Performance of convolutional neural networks, on the ImageNet tests, is now close to that of humans. The best algorithms still struggle with objects that are small or thin, such as a small ant on a stem of a flower or a person holding a quill in their hand. They also have trouble with images that have been distorted with filters (an increasingly common phenomenon with modern digital cameras). By contrast, the conventional treatment by multilayer perceptrons or SVM rely on and-crafted features instead of learned features through convolutional networks, will be limited by the size of features of the images with raw input pixel information of more than a million. Specifically, our problem will be training a learning model to automatically recognize one or multiple digits within an image. This can be widely applied to large scale problem from identifying hard written digits to recognizing house numbers for unmanned delivery such as Amazon Prime Air.

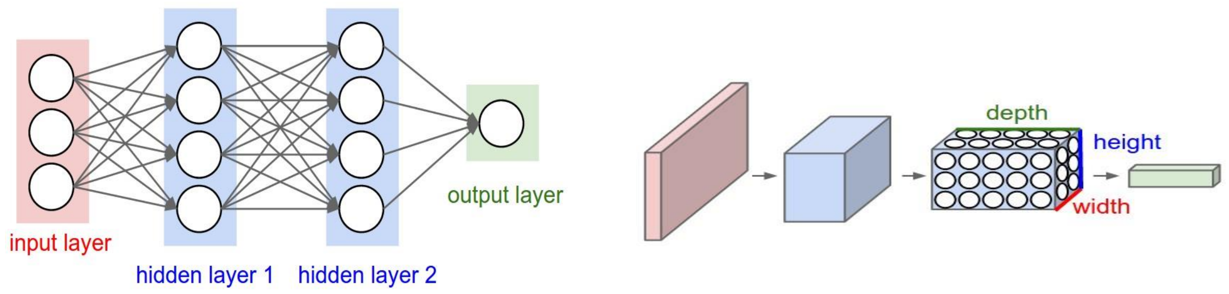

**Figure 1:** Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels). [Ref 1]

1.2 Problem Statement:

Recognizing numbers on images might be accomplished by human eyes with little problem, but it can be of great difficulty for computers, and this is the place when machine learning and convolution neural networks come into play.

The convolution neural network (ConvNet) is a special artificial neural network. Unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. A simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. Three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer These layers are then stacked to form a full ConvNet architecture. The details of ConvNet are discussed through the Stanford CS231n lecture notes [Ref 1].

In this project, a 7-layer ConvNet will be implemented, this architecture resembles the classical LeNet [Ref 2], which is the first successful application of ConvNet developed by Yann LeCun in 1990's. The LeNet architecture was used to read zip codes, digits, etc. Actual parameters will be changed, as described in details in the Methodology part.

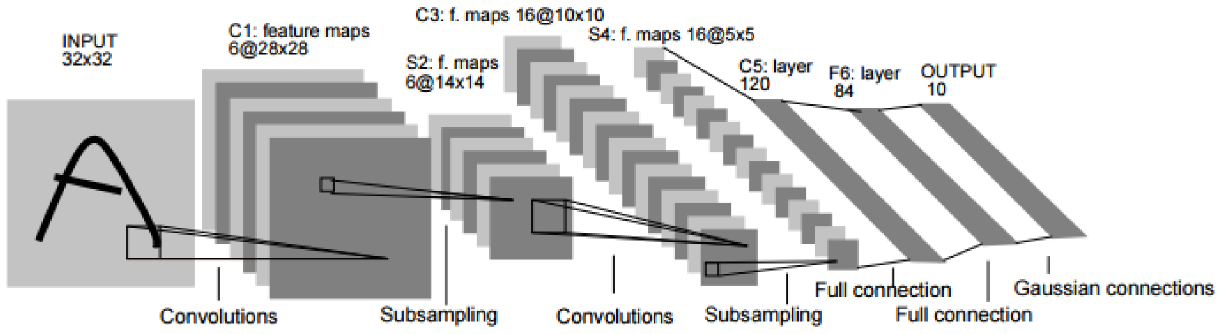

**Figure 2:** Architecture of LeNet-5, a ConvNet, here for digits' recognition. [Ref 2]

1.3 Metrics:

Since this is the image recognition problem, the accuracy will be our main criteria. We will be evaluating the training accuracy as well as test accuracy.

In Seciton 2, we describe the image dataset, in Section 3 we provide a general flow of CNN and our architecture. In Section 4, we briefly explain the preprocessing steps of the input data. In Section 5, we provide the results of our ConvNet.

## 2. Analysis

### 2.1 Data Exploration

The dataset we will be using to train and test our model are from the SVHN dataset[1], which is a good large scale dataset collected from house numbers in Google Street View. SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be seen as similar in flavor to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). SVHN is obtained from house numbers in Google Street View images.

For each image in the dataset, there are one or multiple numerical digits and each digit below to one class (10 classes for “0” to “9”). The dataset comes in two formats, one is the original images with character level bounding boxes, the other is MNIST-like 32-by-32 images centered around a single character. The dataset is splitted into three sets, train, test and extra. For the project, I will be mainly use train and test for training and testing, while include data from extra set to improve the accuracy. Additional shuffering will be applied for randomizing the inputs.

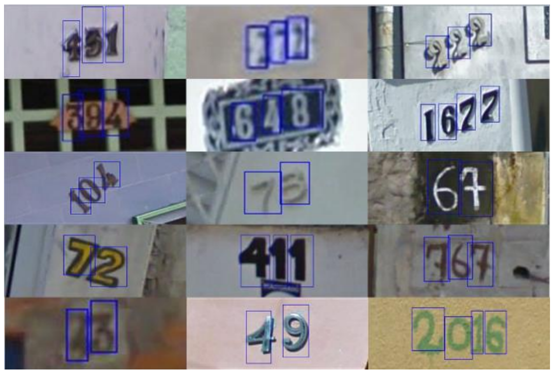

**Figure 3:** sample image from the SVHN datasets.

[1] Details of the SVHN dataset are described at http://ufldl.stanford.edu/housenumbers/

It is important that we know the features of our data and it will help us build the recognition network. For our problem with SVHN dataset, we downloaded and extracted the dataset, then performed the feature analysis before the actual learning.

First we try to plot the number of digits for the training, test and extra dataset. The results are shown as follows:

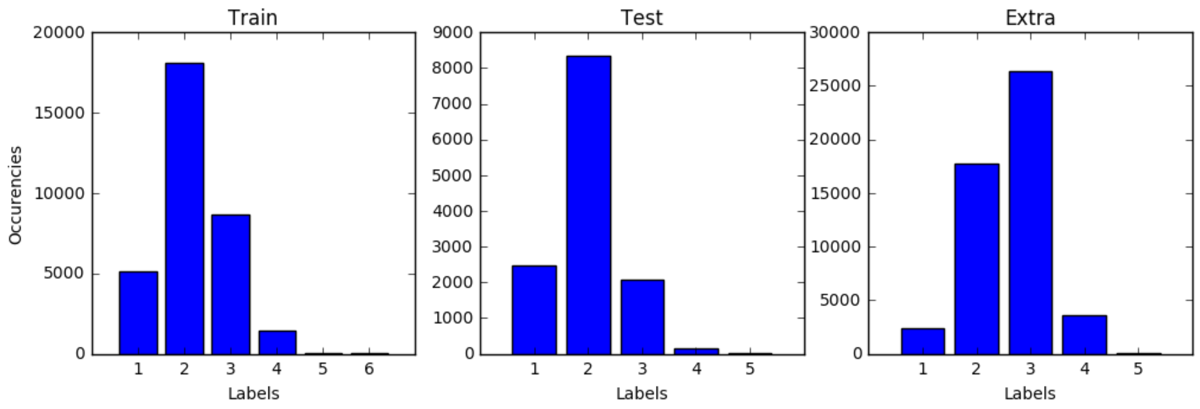

**Figure 4:** Distribution of number of digits in the dataset.

It turns out that the cases that there are more than 4 digits are very rare. So we consider them as noise in the input data and exclude them from the learning task.

The second step we did is to include some of the extra dataset into the training set. Then we shuffle the training and test dataset and use some other part of the extra dataset as the model validation dataset. There is no overlap between the training and validation dataset.

After the dataset merging and shuffling, the occurrence of number of digits in the three dataset are as follows:

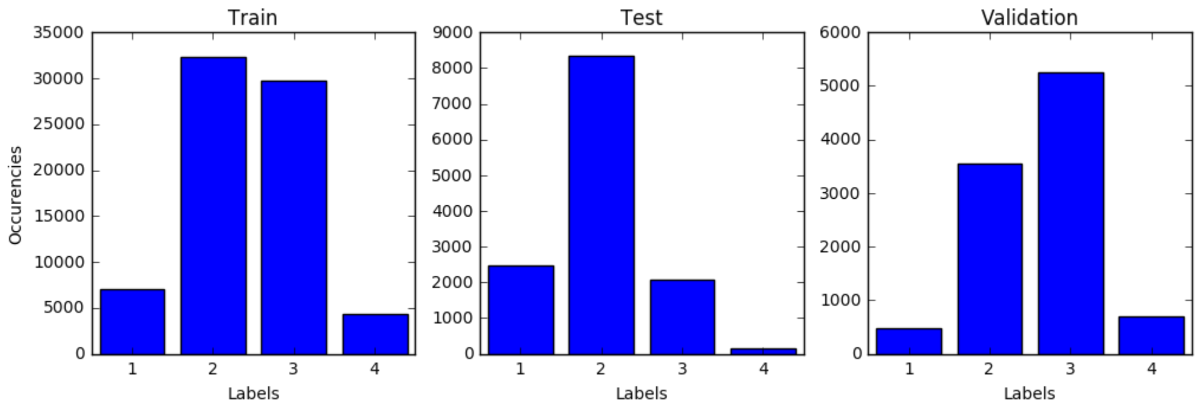

**Figure 5:** Distribution of number of digits in the training, validation and test dataset.

Also the occurrence of each digits in the three dataset is plotted as follows (numbers from 0-9):

The distribution shares some common features, such as there are more 1-3 and less 4-9 and 0 and the occurrence of digit 4-9 are almost the same. The occurrence of digits is not uniform, which is the character of this dataset.

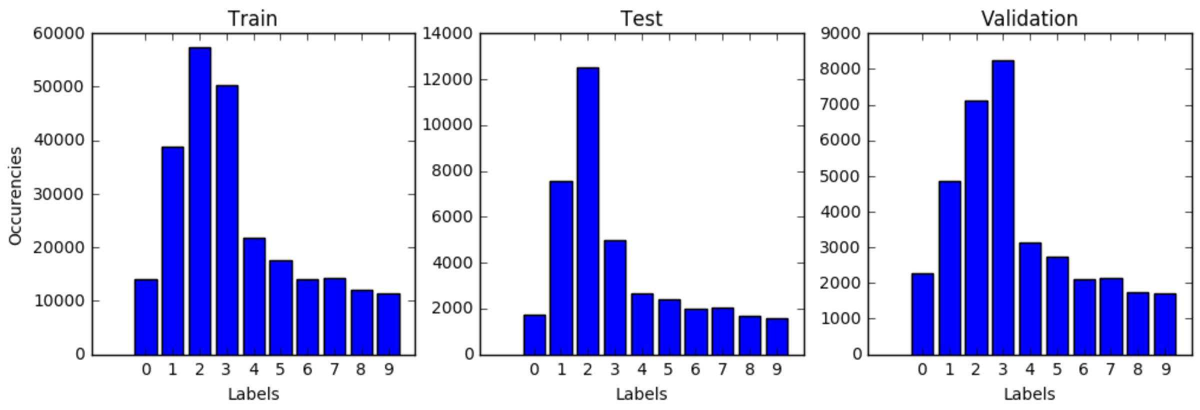

**Figure 6:** Distribution of number 0-9 in the training, validation and test dataset.

## 2.2 Algorithms and Techniques

The structure of the CNN is shown below:

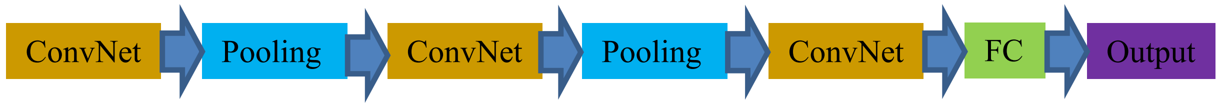

**Figure 7:** Structure of the Convolution Neural Network.

The detailed function of each layer is as follows:

- Convolution Layer
  Convnets are neural networks that share their parameters across space. It works by sliding over the vector with depth, height and width and producing another matrix(called a convolution) that has a different weight, depth height. Essentially, rather than having multiple matrix multipliers, we have a set of convolutions. For example, if we have an image of size 1024x1024px in 3 channels, we could progressively create convolutions till our final convolution has a size of 32x32px and a much larger depth.
- Pooling (Max Pooling)
  Pooling is a technique that can be used to reduce the spatial extent of a convnet. Pooling finds a way to combine all information from a previous convolution into new convolution. For our example, we'll be using maxpooling, takes the maximum of all the responses in a given neighborhood. We choose it because it does not add to our feature space and yet gives accurate responses.
- Fully Connected Layer
  This layer connects every neuron from the previous convolutions to every neuron that it has. It converts a spatiallike network to a 1d network, so we can then use this network to produce our outputs
- The output layer
  The output from this layer are logits which represents matrix showing the probabilities that of having a character in a particular position

### 2.3 metrics

We use the accuracy of prediction to evaluate the performance of the ConvNet, specifically the training accuracy, validation accuracy and test accuracy are three different metrics, which will be used in our estimation. The details and optimization methods will be discussed in the methodology part.

In comparison with the known solution to this problem, the accuracy using different algorithms is shown below [Ref 3]:

| ALGORITHM | SVHN-TEST (ACCURACY) |
|---|---|
| HOG | 85.0% |
| BINARY FEATURES (WDCH) | 63.3% |
| K-MEANS | 90.6% |
| STACKED SPARSE AUTO-ENCODERS | 89.7% |

I would say the potential of over 90% percent of accuracy could be considered as a successful model, which is almost better than any other recognition algorithm other than convolution neural networks.

## 3. Methodology

In the project, there are 4 ipython notebook available. The 1-data_analysis.ipynb covers the data download and extraction for the further training. 2-ConvNet.ipynb covers the training and optimization for small training set. 3-large_data_analysis.ipynb is almost the same as the 1-data_analysis.ipynb, the only difference is that it creates a larger training dataset for further optimize the performace. 4-large-ConvNet.ipynb covers the optimization for large training dataset.

### 3.1 data cleaning

The data are downloaded from the http://ufldl.stanford.edu/housenumbers/ and extracted with labels. The images are scaled to 32 by 32 pixels and then greyscaled by the following formula:

$$grey = [red, green, blue] \cdot [0.2989, 0.5870, 0.1140]$$

The labels are mapped to a 5 digit vector with the first digit ranging from 1-4, the rest digits ranging from 0 to 10. The first digit means the number of digit in the picture while the i$^{th}$ position (i = 2,3,4,5) represents the (i-1)$^{th}$ digit and label 10 means no digit in that position.

### 3.2 Convolution Neural Network and training

For the training, the initial configuration of the 7-layer CNN are as follows:

- ConvNet C1: 28*28*16, convolution size: 5*5*1*16
- Pooling P1: 14*14*16

- ConvNet C2: 10*10*32, convolution size: 5*5*16*32
- Pooling P2: 5*5*32
- ConvNet C3: 1*1*64, convolution size: 5*5*32*64
- FC: 64*11
- Output layer: 16*11

The input greyscaled figure is first processed with the Local Contrast Normalization (LCN) layer [Ref 4], enforcing a sort of local competition between adjacent features in a feature map, and between features at the same spatial location in different feature maps. Then the feature map is processed with (conv-relu-pooling) cycle which learns the feature from the image. The operation of each operation for the first cycle is as follows:

- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [28x28x16] if we decided to use 16 filters.
- RELU layer will apply an elementwise activation function, such as the max(0,x)max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([28x28x16]).
- POOL layer will perform a down sampling operation along the spatial dimensions (width, height), resulting in volume such as [14x14x16].

After two cycles of conv-relu-pooling (same operation, different parameter, as shown above), the data will go through (conv-relu-dropout) cycle. Dropout is a technique for addressing overfitting problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. The dropout rate is set to 0.6 in our training.
A fully connected layer takes all neurons in the previous dropout layer and connects it to every single neuron it has. Lastly, the output layer will compute the class scores, resulting in volume of size [1x1x11], where each of the 10 numbers plus the default void correspond to a class score.
During the training phase, stochastic gradient descent(SGD) optimization algorithms is applied. SGD is a simple but efficient approach to discriminative learning. The learning rate is set to 0.01 with exponential decay rate of 0.95, which proves to be effective parameter during the training. The training steps are set to 25,001 initially.

## 4. Results

### 4.1 initial training accuracy

The neural network is trained with selected dataset, without changing the parameter, this model achieve accuracy as follows:

| Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|
| 86.3% | 89.2% | 89.9% |

### 4.2 Increase the size of neural network

The neural network can be enlarged by one hidden parameter, which is the size of the fully connected layer. Initial value is set to 64, which is the same as the output of the last convolution layer. We can enlarge to 128, 256 and 512. The size change of neural network is aimed at improving the training accuracy.

| Num_hidden | Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| 128 | 91.4% | 91.8% | 91.1% |
| 256 | 94.5% | 93.1% | 91.9% |
| 512 | 96.5% | 93.8% | 92.2% |

We can conclude that increasing the size of hidden layer will improve the training accuracy with the cost of extra memory and computing time. We would use 512 for the later network.

4.3 improving the validation accuracy

The validation accuracy can be improved by introducing the regulation. We use the l2 regulation of the last layer and compare the effect with different regulation coefficient.

| regulation coefficient | Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| 1e-5 | 96.5% | 94.1% | 92.4% |
| 1e-4 | 94.9% | 93.8% | 92.3% |
| 1e-3 | 96.1% | 93.8% | 92.7% |
| 1e-2 | 90.2% | 92.7% | 91.6% |

We can conclude that introducing the regulation will improve the validation accuracy and test accuracy and avoid the overfitting. We choose 1e-4 for further optimization.

4.4 improving the test accuracy by more complex model

To further improve the test accuracy, we need to have a deeper model, larger dataset and more training time. We would explore these and check the improvement individually.

To build a deeper model, we can introduce another FC layer before output. The effect of such FC layer functions as the normal neural network with more layer which would add more non-linearity that cannot be included in single layer. The size of the second FC layer with its potential improvement is shown as follows:

| Num_hidden2 | Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| 32 | 98.0% | 93.6% | 92.4% |

|  |  |  |  |
|---|---|---|---|
| 64 | 97.3% | 94.1% | 92.5% |
| 128 | 98.4% | 94.7% | 93.0% |

We see that the accuracy can be improved further with deeper and wider neural network model. The cons are more memory and computing time. We use 128 as the width of the second FC layer later.

### 4.5 Improve the test accuracy by larger data set and longer training step

Further we could use larger dataset by including the rest of the extra dataset into the training set. The size of training set increased from 73392 to 225630. With the same configuration, the accuracy of neural network is improved as follows:

| Num_hidden2 | Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| before | 98.4% | 94.7% | 93.0% |
| After | 95.7% | 96.2% | 94.0% |

In the last step, we can try improving the accuracy by increase the training step from 25001 to 300001. The improvement is shown as follows

| Num_hidden2 | Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| before | 95.7% | 96.2% | 94.0% |
| After | 100% | 96.7% | 94.3% |

In doing so, our final test accuracy is improved up to 94.3%.

### 4.6 Model validation and robustness check

To validate the model, I mixed and shuffled the training, test and validation set and retrain the model using the new datasets and the accuracy remains the same, which proves that the model is robust.

|  | Training accuracy | Validation accuracy | Test accuracy |
|---|---|---|---|
| Test1 | 100% | 96.7% | 94.3% |
| Test2 | 98.8% | 96.4% | 96.3% |
| Test3 | 98.0% | 96.3% | 96.3% |

So the model is robust and the average accuracy is set to (94.3%+96.3%+96.3%)/3 = 95.6%

### 4.7 Benchmark comparison

In comparison with the best accuracy of 90.6% using K-means, our ConvNet could improve the accuracy by 5%, which can be treated as great improvement and satisficing result as an adequate solution. The test accuracy of 95.6% outperforms many other CNN models because of the design such as the LCN layer and wide hidden FC layer and appropriate selection to avoid overfitting such as relu layer and dropout layer.

## 5. Conclusion

In this project we trained the state of art convolution neural network for digits recognition. The SHVN dataset are downloaded and preprocessed with scaling and greyscaling. The training process are convolutional neural network which consists of LCN layer, conv-relu-pooling layers, conv-relu-pooling layers, conv-relu-dropout layers, FC and output layer. Parameters of the model are optimized or set to better solving the problem. The highest accuracy we achieved is 95.6%. Some of the testing result are shown as follows. Based on the optimization, we learn that highest accuracy is achieved through deeper and wider neural network model, moderate regulation and appropriate dataset that is large enough for training.

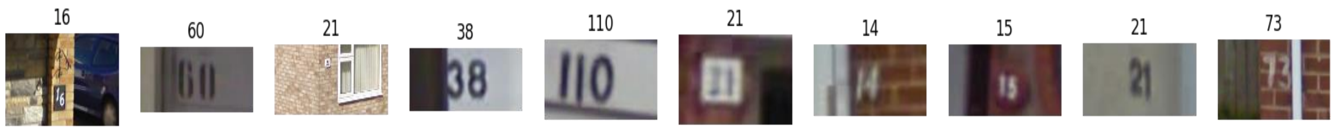

**Figure 8:** sample output image from the test dataset.

The optimization of ConvNet is as follows 1) increase the width of neural network to improve the training accuracy 2) increase the validation accuracy by introducing regulation. 3) increase test accuracy by deeper and wider network, more training data and more training step. The step by step improvement is shown in the result part. The most difficult part is to extract the potential of neural network model and improve the test accuracy. Usually a little bit improvement would mean a lot more calculation.

To further improve the accuracy, it would be difficult by simply adding the width of the neural network or extend the computing steps. Adding another FC layer might help improve the accuracy. It could be more helpful to use more recent neural network model such as Recurrent Neural Network (RNN) for such task.

## References

1. http://cs231n.github.io/convolutional-networks/

2. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

3. Netzer, Yuval, et al. "Reading digits in natural images with unsupervised feature learning." (2011).

4. Jarrett, Kevin, Koray Kavukcuoglu, and Yann Lecun. "What is the best multi-stage architecture for object recognition?." *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009.