# House Number Recognition

# Using Deep Learning

## I. Definition

### Overview

One important way for an AI agent to understand its surrounding is by seeing. In fields like detailed mapping and self driving car, errorless house number recognition is a basic need. A model was built in this project to do this task using convolutional neural network. The model was trained and tested on the *Street View House Number* dataset.

### Problem Statement

This project will focus on the single digit recognition. Given a cropped part of a door plate, the model will be able to recognize what number it is(from 0-9). \*\*\*\*formula\*\*\*\* For example, for a cropped image like fig 1, the model should be able to tell it's 3 because 3 is center aligned in the image. \*\*\*\*fig 1\*\*\*\*

The problem is similar to hand written digit recognition and notMNIST letter recognition. So the strategy is first to apply a model with similar architecture to the notMNIST model, then train it on the SVHN dataset. However, there're many differences between the two dataset, the most obvious of which is that most images in SVHN is mixed by other digits. Therefore, the model should be scaled and refined to handle the issue.

### Metrics

1) Accuracy

To evaluate the model, accuracy is the most important metrics. It's simply the ratio of correct predictions out of all predictions

2) Precision and Recall

Precision and Recall are commonly used in classification problems. For each class, precision is \*\*\*\* formula and picture \*\*\*\*

Recall is \*\*\*\* formula and picture \*\*\*\*

Specifically, the performance can be illustrate as a confusion matrix \*\*\*\* pic \*\*\*\*

3) Loss

The loss function I choose in this problem is the softmax cross entropy. \*\*\*\* formula \*\*\*\*

\*\*\*\*advantages\*\*\*\*

## II. Analysis

### Data Exploration and Visualization

1) Format:

The dataset used in this project was stored in .mat files, they are "train.mat", "test.mat" and "extra.mat". Two ndarrays will be returned by reading each file using Scipy.io.loadmat() function.

The first one (name it X) stands for the images, while the second (name it y) contains the class labels correspondingly with 0 marked as 10 (MATLAB style).

X is of shape<code>(pixel width, pixel height, number of channels, number of items), y is of shape(number of items, )</code>

\*\*\*\* table \*\*\*\*

The images in this dataset is in same size 32x32x3

2)Distribution

In the Jupiter notebook preprocess\_mat.ipynb, the first few cell will explore the data and plot the distribution of the all three datasets.

### Algorithms and Techniques

1) Frameworks

* Script is presented in Jupyter notebook using python.
* Scientific Computation & Visualization : Numpy, Matplotlib.
* Scikit-Learn: A machine learning library that implements classic machine learning algorithms.
* Tensorflow: This is a framework that help users to build computational graph with various kinds of nodes that have functionality of both forward and backward propagating computation, which make it easier to construct deep learning architecture.

2) Mathematics

* Convolution Neural Network

- Feature Extractor

Convolution Neural Networks(Fukushima, 1980; LeCun et al., 1988) are neural networks that contain convolution layer. Intuitively, convolution layers are layers that divides a image into small pieces by a windows which moves both horizontally and vertically by a certain stride of pixels. Each piece is seen as a feature, for example, the sharp head of the letter ‘A’. Those feature, as group of pixel values, will then be dot product with a group of weight and become a value for the next layer. \*\*\*\* picture \*\*\*\*

- Activation Function

Activation is an important step for neural network. It should stop the propagation when a neuron is not activated, which is simply whether the numbers is positive. There are lots of activation functions, like step, sigmoid and tanh, but most of them have various problems. In practice, people use rectified linear unit, ReLU\*\*\*\* cite \*\*\*\* right behind convolution. This unit or its improved version

are popular because it’s easy to both forward-propagate and back-propagate.

- Pooling and Connecting to Fully Connected Layers

Convolution layers gives output in unfriendly image like shape. Before taking it as the input to create one hot encoding vector, they should be flattened into stripe like vectors. But the output could be very large when images with many depths are flattened. Usually, people apply a method called pooling. Like feature extraction, a small window was moving horizontally and vertically on the image and simply take an average or maximum inside the window. Pooling can help reduce the number of weights to train in the architecture because pooling doesn’t need any parameter.

* Training

Training a deep neural network with convolution layers take more techniques than traditional fully connected neural networks. For me there are two basic ideas should be kept in mind: first is to make the information flow efficiently, second is when you are trying to go to the global minimum, adjust your pace according to the terrain.

- Initialization of Weights

The rule “make it random small” no more applies to deep neural networks. This is very intuitive: if you make all the weights random small, when the information, or numbers are back-propagating, the minus power is accumulating, therefore the numbers could become zero when it arrive some deep neurons and stop the back-propagation, which means the upstream layers will never be trained. In this project, I simply increase the standard deviation of the distribution from upstream to downstream to avoid the issue.

- Dropout

Dropout is to block the numbers flowing forward and backword in some neuron randomly. It can be seen as an ensemble technique, because in every step, only a part of the network is trained. However, dropout is not implemented in validation and test. It’s the equivalence to combine those partial model. And most of the cases, this technique will give more accuracy.

- Stochastic Gradient Descent

When it comes to large dataset, it would be too heavy for computer to use all the data in one iteration. One solution is to use Stochastic Gradient Descent \*\*\*\* cite \*\*\*\* Basically, we take a mini batch of the dataset and feed it to the network. It may take more iteration for the model to converge, but in works in practice.

- Update and Learning rate

Basically, the update space is a bowl shape with steep and smooth directions. In this project, I applied Adam update(Kingma and Ba, 2014), which make the update more on the steep direction.

In this project, exponential learning rate is applied.

***III. Methodology***

**Data Preprocessing**

I took roughly four steps in this project:

1) Reshape the Data

the data is not friendly for numpy, so I just made the index of items the first dimension.

The shape after this step: (number of items, pixel width, pixel height, number of channels) i.e., (number of items, 32, 32, 3)

2) Make the Images in Grayscale

Since the task is to recognize the shape, grayscale is enough. I applied mean grayscale in this project.

3) Collect a Normal Distributed Training Set and Validation Set

As the data exploration shown, real world data has more 1s and 2s than other numbers.

In training, the model should have equal chance to meet all ten labels. Therefore, I collect data from both “train.mat” and “extra.mat” to build my own training dataset of normal distribution. I also collect a normal distributed dataset, specifically, I used the method shown in this paper \*\*\*\* cite \*\*\*\*

4) Shuffle the Data

**Implementation**

(Note: conv means conv – relu, fc means fully connected)

The first step is to design a benchmark model. I chose a model that has a architecture of

[conv – conv – pool – fc – drop – fc – softmax], this is a model similar to that used in notMNIST problem.

Then is to introduce a dictionary to store the training log including validation accuracy and loss.

Finally is to run the Tensorflow session. In this project I use Stochastic Gradient Descent to do the update. I equally slice the shuffled dataset into mini batches and use them sequentially.

Use other architecture and repeat steps above and tune those model. I have committed every working model I had tried.

### Refinement

Here are some architecture I’ve tried:

## IV. Result

## Model Evaluation and Validation

## Justification

## V. Conclusion

### Free-Form Visualization

### Reflection

### Improvement