Capstone Proposal — Joel S

**Machine Learning Engineer Nanodegree** — January 21st, 2017

# Handwritten Digit Recognizer using Deep Neural Network

## Handwriting Recognition

Handwriting recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. The image of the written text may be sensed "off line" from a piece of paper by optical scanning (optical character recognition) or intelligent word recognition. Alternatively, the movements of the pen tip may be sensed "on line", for example by a pen-based computer screen surface, a generally easier task as there are more clues available [1].

Handwriting Recognition plays an important role in storage and retrieval of typed and handwritten information. A wealth of historical papers exists in a physical format. This includes genealogical information, old family records, written manuscripts, personal diaries and many other pieces of a shared past. Consistent review of this information damages the original paper and can lead to physical data corruption. Handwriting recognition allows people to translate this information into easily readable electronic formats [2].

Since 2009, the recurrent neural networks and deep feedforward neural networks developed in the research group of Jürgen Schmidhuber at the Swiss AI Lab IDSIA have won several international handwriting competitions. In particular, the bi-directional and multi-dimensional Long short-term memory (LSTM) of Alex Graves et al. won three competitions in connected handwriting recognition at the 2009 International Conference on Document Analysis and Recognition (ICDAR), without any prior knowledge about the three different languages (French, Arabic, Persian) to be learned. Recent GPU-based deep learning methods for feedforward networks by Dan Ciresan and colleagues at IDSIA won the ICDAR 2011 offline Chinese handwriting recognition contest; their neural networks also were the first artificial pattern recognizers to achieve human-competitive performance on the famous MNIST handwritten digits problem of Yann LeCun and colleagues at NYU [1].

Most modern tablet pc and hybrid laptops includes an active stylus and display which allows the user to write and draw on the screen. It would be very useful for the users if they can recognize, derive meaning and store the handwritten text into digital text which can be used within computer and text-processing applications. This motivated me to take the first step of implementing a Handwritten Digit Recognizer.

## Problem Statement

The goal is to predict the digit handwritten in an image. The dataset contains images of handwritten digits and their respective numbers which can be used to train and test the model. This is supervised learning problem – more specifically a classification problem. The model should be able to classify which number (0 - 9) is handwritten in the image. The model can be scored for its ability to predict the number correctly over large different test data and real data.

## Datasets and Inputs

The MNIST database [4] of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

The original black and white (bi-level) images from NIST were size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

The dataset can be downloaded and loaded using Keras, a high-level neural networks python library using the following code

```
from keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

where

- **X_train, X_test**: uint8 array of grayscale image split as train and test data
- **y_train, y_test**: uint8 array of digit labels (integers in range 0-9) split as train and test data

Thus, we can use the above variables to train the model with pixel-values of the handwritten image as features(X_train) and its respective label(y_train) as target. We can also test the model by evaluating it against the test variables X_test and y_test.

## Solution Statement

Given that the problem is a supervised learning problem and more specifically a classification problem, there are lot of algorithms available for training a classifier to learn from the data. The algorithm chosen for this project is Deep Neural Network(DNN) using Convolutional Neural Network(ConvNet).

**Convolutional neural network** (CNN, or ConvNet) is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex [5]. Since it models animal visual perception, it can be applied to visual recognition tasks such handwritten digit recognition from images. There are four main types of layers to build ConvNet architectures:

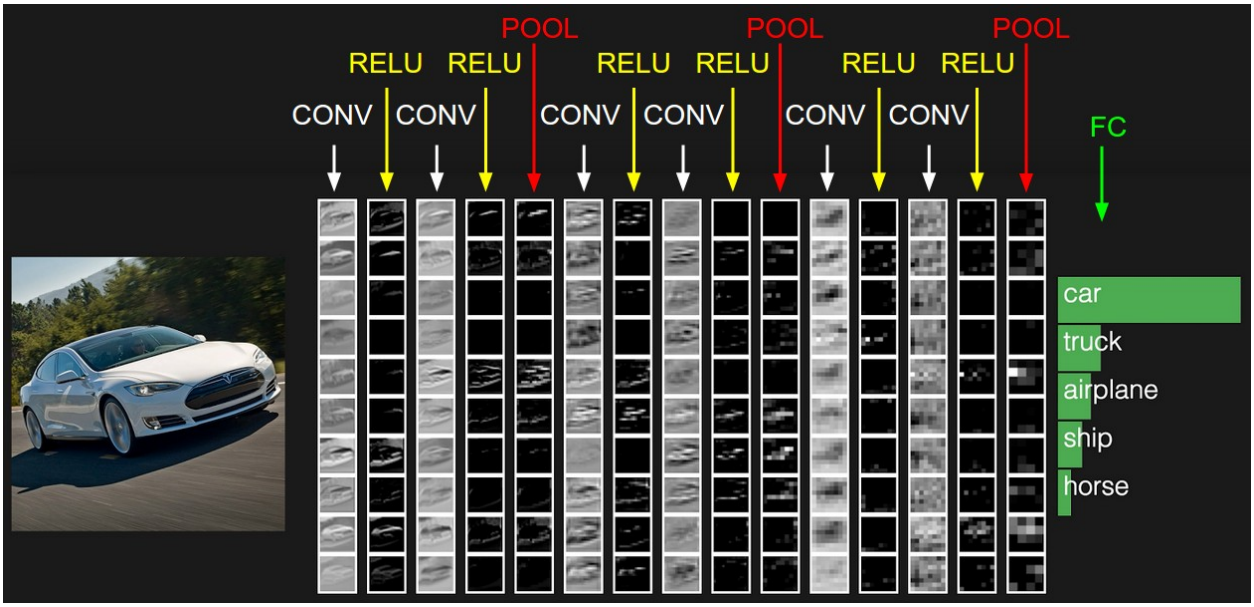
1. **Convolutional Layer** - The input to a convolutional layer is a m x m x r image where m is the height and width of the image and r is the number of channels, e.g. an RGB image has r=3. The convolutional layer will have k filters (or kernels) of size n x n x q where n is smaller than the dimension of the image and q can either be the same as the number of channels r or smaller and may vary for each kernel. The size of the filters gives rise to the locally connected structure which are each convolved with the image to produce k feature maps of size m−n+1. [6]

2. **ReLU Layer** - The Rectified Linear Unit apply an elementwise activation function, such as the max (0, x) thresholding at zero.

3. **Pooling Layer** - perform a down sampling operation along the spatial dimensions (width, height)

4. **Fully-Connected Layer** - compute the class scores (between 0-9 digits). As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume. [7]

An example of a Convolutional Neural Network is given below [7].



The model is trained with DNN on the training data (X_train, y_train) and then evaluated against test data (X_test, y_test).

## Benchmark Model

The benchmark model is chosen from one of the kernels of the Kaggle Digit Recognizer competition [8]. The benchmark model solves the same handwritten digit recognition problem mentioned in this project by using a Random Forest classifier algorithm. It uses the same dataset used in this project. Thus, making it a perfect benchmark model for this project. The result of the benchmark model is the accuracy score of the model. The same result can be measured for our model by calculating the accuracy score. Accuracy score of the benchmark model is 0.872.

## Evaluation Metrics

Accuracy is measured as ratio between the sum of true positives and true negatives and dataset size. True Positive and True Negative here are the same because both are the sum of all truly classified handwritten digit image samples. In general, it's the measure of correctness of the model. It's just the ratio between correctly classified samples to the overall samples. It ranges from 0 – 1, where 0 means poor performance and 1 means good performance.

The cross-entropy (loss function) measure has been used as an alternative to squared error. Cross-entropy can be used as an error measure when a network's outputs can be thought of as representing independent hypotheses (e.g. each node stands for a different concept), and the node activations can be understood as representing the probability (or confidence) that each hypothesis might be true. In that case, the output vector represents a probability distribution, and our error measure - cross-entropy - indicates the distance between what the network believes this distribution should be, and what the teacher says it should be. There is a practical reason to use cross-entropy as well. It may be more useful in problems in which the targets are 0 and 1 (thought the outputs obviously may assume values in between.) Cross-entropy tends to allow errors to change weights even when nodes saturate (which means that their derivatives are asymptotically close to 0.) [3]

## Project Design

The dataset is loaded into python using keras library as discussed in the Dataset and Inputs section. The loaded data will be first explored and visualized using numpy and matplotlib library to understand the nature of the data. Exploring the data will help us in deciding how to approach and whether any preprocessing of the data is needed. Preprocessing of the data is done as required. Once the data is ready, the Deep neural network(DNN) will be built based on the architecture(ConvNet) as discussed in the Solution Statement. Once the model is built, it will be compiled to check if the architecture has any error. Then the compiled model will be

trained on the training data (X_train, y_train) and evaluated using accuracy score against the testing data (X_test, y_test). Then the results can be analyzed and compared with respect to the benchmark model to know the overall performance of the model. Now we have a trained model, a test is conducted against the model by loading images of digits which are not from MNIST dataset to evaluate the performance of the model. The images are loaded and then preprocessed to match the MNIST dataset format so that we can test it. The model is then made to predict the digit in the preprocessed image. Thus, we can evaluate our model's performance over real time data.

## References


1. https://en.wikipedia.org/wiki/Handwriting_recognition
2. http://www.ehow.com/list_7353703_benefits-advantages-handwriting-recognition.html
3. http://www.cse.unsw.edu.au/~billw/cs9444/crossentropy.html
4. http://yann.lecun.com/exdb/mnist/
5. https://en.wikipedia.org/wiki/Convolutional_neural_network
6. http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/
7. http://cs231n.github.io/convolutional-networks/
8. https://www.kaggle.com/romanroibu/digit-recognizer/random-forest-digit-classifier