# Handwritten digit recognition

(Machine Learning)

**Term paper by:** Hamza Hussain(P15-6141) & Ali Amjad(P15-6121)

**Submitted to:** Dr. Muhammad Nauman

## Abstract

This paper presents the results of handwritten digit recognition on well-known MNIST dataset using deep learning. Perceiving the digit has turned into a fundamental part as far as certifiable application. Since, digits are composed in various styles in this way to distinguish the digit it is important to perceive and arrange it with the assistance of machine learning methods. The pictures of digits are perceived, prepared and tried. This paper describes the technique for pre-processing the handwritten digits, as well as a number of ways in which convolutional neural networks (CNN) were used for the recognition task.

- **Problem statement:** Handwriting number recognition is a challenging problem researcher had been research into this area for so long especially in the recent years. In our study there are many fields concern with numbers, for example, checks in banks or recognizing numbers in car plates, the subject of digit recognition appears. A system for recognizing isolated digits may be as an approach for dealing with such application. In other words, to let the computer understand the Arabic numbers that is written manually by users and views them according to the computer process. Scientists and engineers with interests in image processing and pattern recognition have developed various approaches to deal with handwriting number recognition problems such as, minimum distance, decision tree and statistics.
- **Approach:** The main objective for our system was to recognize isolated handwritten digits. For example, different users had their own handwriting styles where here the main challenge falls to let computer system understand these different handwriting styles and recognize them as standard writing.

# Introduction

Handwritten digit recognition is an active topic in pattern classification/learning research. Recognition is dealt with in postal mail sorting, bank check processing, postal form data entry, etc. For these applications, the performance (accuracy and speed) of digit recognition is crucial to the overall performance. While in pattern classification and machine learning communities, the problem of handwritten digit recognition is a good example to test the classification performance.

The performance of digit recognition largely depends on the feature extraction approach and the classification/learning scheme. For feature extraction of digit recognition, various approaches have been proposed. Many experiments have shown that the Convolutional Neural network feature is one of the most efficient features for hand-written digit recognition. This project is basically implemented on MNIST data set using Convolutional Neural Networks.

## - MNIST dataset:

The dataset was constructed from a number of scanned document dataset available from the National Institute of Standards and Technology (NIST). This is where the name for the dataset comes from, as the Modified NIST or MNIST dataset.

Images of digits were taken from a variety of scanned documents, normalized in size and centred. This makes it an excellent dataset for evaluating models, allowing the developer to focus on the machine learning with very little data cleaning or preparation required.

The sample binary images were normalized into 20 ×20 grey-scale images with aspect ratio preserved, and the normalized image is located in a 28 ×28 plane. A standard spit of the dataset is used to evaluate and compare models, where 60,000 images are used to train a model and a separate set of 10,000 images are used to test it. It is a digit recognition task. As such there are 10 digits (0 to 9) or 10 classes to predict. Results are reported using prediction error, which is nothing more than the inverted classification accuracy.

Fig. 3. Sample images of MNIST data.

- ## Convolutional Neural Networks (CNN):

Multilayer Neural Networks trained with the back-propagation algorithm constitute the best example of a successful Gradient-Based Learning technique. Convolutional Neural Networks are a special kind of multi-layer neural networks used to recognize visual patterns with extreme variability (such as handwritten digits), and with robustness to distortions and simple geometric transformations.

**Simple Architecture Neural Network**

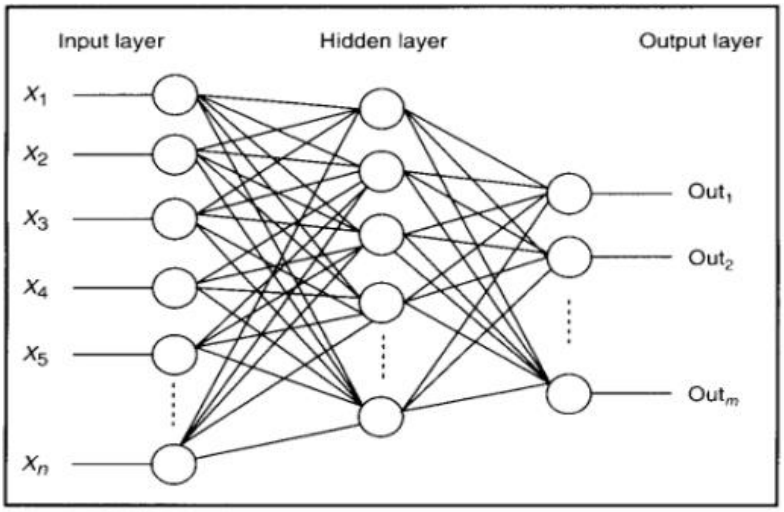
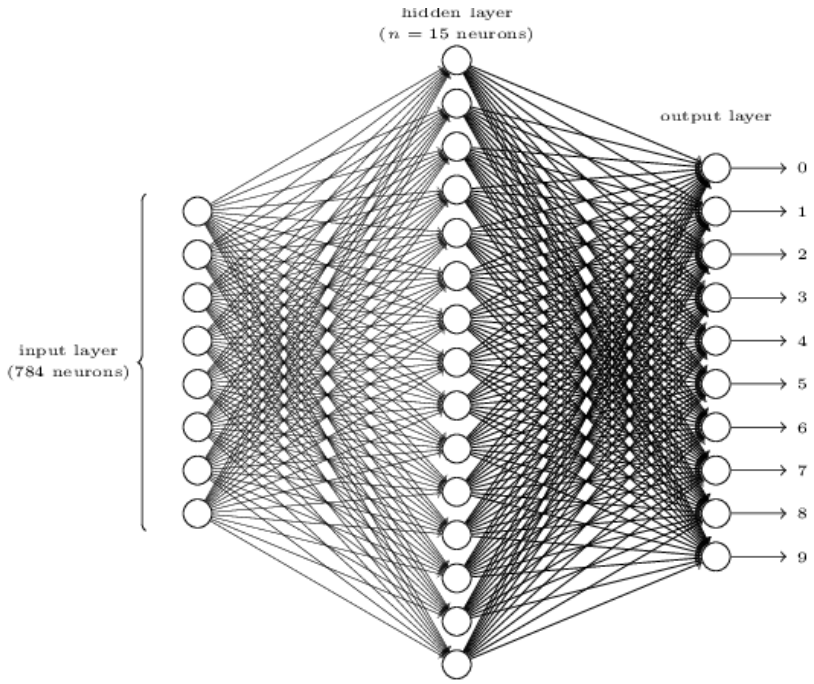



**Figure. 1 Neural Network Architecture**

We humans solve this segmentation problem with ease, but it's challenging for a computer program to correctly break up the image. Once the image has been segmented, the program then needs to classify each individual digit. There are many approaches to solving the segmentation problem. One approach is to trial many different ways of segmenting the image, using the individual digit classifier to score each trial segmentation. To recognize individual digits, we will use a three-layer neural network:




The input layer of the network contains neurons encoding the values of the input pixels. As discussed in the next section, our training data for the network will consist of many 28 by 28-pixel images of scanned handwritten digits, and so the input layer contains 784=28×28 neurons. For simplicity I've omitted most of the 784 input neurons in the diagram above. The input pixels are greyscale, with a value of 0.0 representing white, a value of 1.0 representing black, and in between values representing gradually darkening shades of grey.

The second layer of the network is a hidden layer. We denote the number of neurons in this hidden layer by n, and we'll experiment with different values for n. The example shown illustrates a small hidden layer, containing just n=15 neurons.

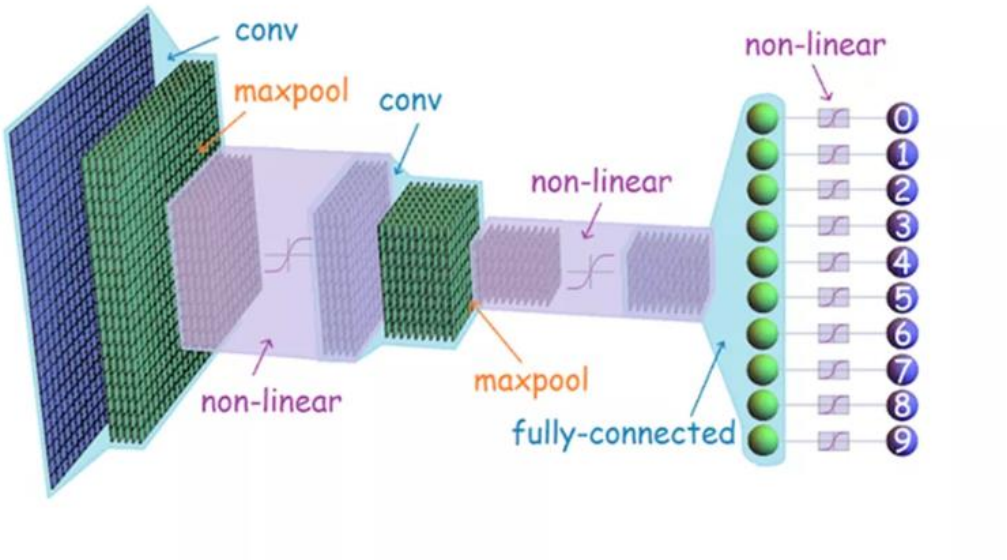
The output layer of the network contains 10 neurons. If the first neuron fires, i.e., has an output ≈1, then that will indicate that the network thinks the digit is a 0. If the second neuron fires then that will indicate that the network thinks the digit is a 1. And so on. A little more precisely, we number the output neurons from 0 through 9, and figure out which neuron has the highest activation value. If that neuron is, say, neuron number 6, then our network will guess that the input digit was a 6. And so on for the other output neurons.

## Methodology:

Loading the MNIST dataset in Keras. We convert the image in a vector of 784 pixels using reshape function. We performed scaling on input values and normalize the pixel values to the range 0 and 1 by dividing each value by 255.

We use one hot encoding of the class values and transform the vector of class integers into binary matrix by sing built-in function np_utils.to_categorical () in Keras. The Convolutional Neural Network architecture we used is as following:

- Convolutional layer with 30 feature maps of size 5x5
- Pooling layer taking the max over 2x2 patches
- Convolutional layer with 15 feature maps of size 3x3
- Pooling layer taking the max over 2x2 patches
- Dropout layer with probability of 20%
- Flatten layer
- Fully connected layer with 128 neurons and rectifier activation
- Fully connected layer with 50 neurons and rectifier activation
- Output layer




- **Convolutional Neural Network architecture**

## Results:

The input data is an image format file containing numbers. The data feed into the system and then passed through several pre-processing steps before it can be recognized.

We do training on 60,000 samples and validate on thousand samples. With epoch 10 we get the accuracy of 99.08%. The results are as following

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 45s - loss: 0.3912 - acc: 0.8798
val_loss: 0.0874 - val_acc: 0.9726
Epoch 2/10
60000/60000 [==============================] - 43s - loss: 0.0944 - acc: 0.9713
val_loss: 0.0603 - val_acc: 0.9800
Epoch 3/10
60000/60000 [==============================] - 43s - loss: 0.0697 - acc: 0.9781
val_loss: 0.0377 - val_acc: 0.9880
Epoch 4/10
60000/60000 [==============================] - 44s - loss: 0.0558 - acc: 0.9819
val_loss: 0.0331 - val_acc: 0.9885
Epoch 5/10
60000/60000 [==============================] - 44s - loss: 0.0480 - acc: 0.9852
val_loss: 0.0300 - val_acc: 0.9900
Epoch 6/10
60000/60000 [==============================] - 44s - loss: 0.0430 - acc: 0.9862
val_loss: 0.0293 - val_acc: 0.9897
Epoch 7/10
60000/60000 [==============================] - 44s - loss: 0.0385 - acc: 0.9877
val_loss: 0.0260 - val_acc: 0.9911
Epoch 8/10
60000/60000 [==============================] - 44s - loss: 0.0349 - acc: 0.9895
val_loss: 0.0264 - val_acc: 0.9910
Epoch 9/10
60000/60000 [==============================] - 44s - loss: 0.0332 - acc: 0.9898
val_loss: 0.0222 - val_acc: 0.9931
Epoch 10/10
60000/60000 [==============================] - 44s - loss: 0.0289 - acc: 0.9908
val_loss: 0.0226 - val_acc: 0.9918
```

## Conclusion

We can conclude that we reached the computer to the human's brain by the importance use of isolated digits recognition for different applications. This recognition starts with acquiring the image to be pre-processed throw a number of steps. As an important point, classification and recognition have to be done to gain a numeral text. We get accuracy of 99.08 % with training in 10 Epochs. In a final conclusion, neural network seems to be better than other techniques used for recognition.