## 

**CSC 3305 – Data Science**

**Semester 1 2017/18**

**Section: [1]**

**Group Project**

**Predictive analysis of handwritten digit recognition**

**Group- 5**

**Group Members**

Tabassum Tahera – **1413986**

S M Raju – **1416463**

Nazmus Sakib – **1425433**

Md. Shariful Islam – **1412813**

Md. Asir Faisal – **1437059**

Ali Mohammad Tarif – **1418735**

Md. Nazmus Sajid – **1437059**

Mishkat Nur Rahman – **1437185**

Supervised by

**DR. RAINI BINTI HASSAN**

**1.0 Introduction**

The Handwritten digit recognition is the process to recognize any kind of English single digit that is written or in digital form (e.g. image). When we, human, see any handwritten object written in different style, we can easily recognize it just by having a vision. But our project is to copy the ability of human of understanding a digit just by processing it using Artificial Intelligence model. Our machine learning algorithm model is capable of recognizing any handwritten English digit without any prior programing logic. We have used the supervise learning methodology to train our model.

There's nothing more frustrating that waiting in front of an ATM during your lunch break to deposit a check. Many bank patrons today use a feature on their phones to deposit checks which uses their phone cameras. But just how accurate is it to be able to trust? Sadly, we don't have that information available, but we would consider anything under 1% error to be excellent. Luckily to us, we can make a neural network to mimic what these apps use. In this project, we will develop a deep learning model to achieve a near state-of-the-art performance on the MNIST handwritten dataset. We’re going to use Keras with TensorFlow.

## 2.0 Problem Statement

We tried to identify the best algorithm or method for handwritten digit recognition. Handwritten digit recognition is important in a world where tasks and processes are becoming more automated. Banking, for example, is significantly more automated than it used to be. In order to deposit checks, a person may insert a check into an ATM and the ATM will read the amount and account numbers. A person may also simply take a picture of their check with their phone and use a banking application to deposit the check. Both of these methods require precise handwritten digit recognition. Misclassification during these processes would be costly to either the recipient or the patron. Due to this we plan to compare several methods of both clustering and classification algorithms in order to see which method performs best.

## 3.0 Project Objectives

In this project we want to implement a simple Convolutional Neural Network to recognize handwritten English digits 0 to 9. Convolutional Networks work by moving small filters across the input image. This means the filters are re-used for recognizing patterns throughout the entire input image. This makes the Convolutional Networks much more powerful than Fully-Connected networks with the same number of variables. This in turn makes the Convolutional Networks faster to train.

**Data Product:** Our project is about predicting integer value (dependent variable) from input image of handwritten digit (independent variable). This data product can be used in bank systems where cheque book are used to withdraw cash from automated machines. Another possible usage is using in automated gate barriers or, traffic security cameras to detect individual cars from car’s number plate.

## 4.0 Methods

To teach our machine how to use neural networks to make predictions, we are going to use deep learning from TensorFlow. Deep learning is a field of machine learning that uses algorithms inspired by how neurons function in the human brain. TensorFlow is a machine learning framework that Google created to design, build, and train deep learning models. The name, "TensorFlow", is derived from how neural networks perform on multidimensional data arrays or tensors. It's a flow or tensors, just like how the human brain has a flow of neurons!

We will also use Keras, which runs on top of TensorFlow. Keras was developed with a focus on enabling fast experimentation.

## 5.0 Literature Review

### 5.1 Related works

**Liu, Cheng-Lin, et al. [1]** has compared the performances of ten normalization functions and eight feature vectors in handwritten digit recognition on large databases of different sources. The normalization functions implement dimension-based linear/nonlinear normalization and moment-based normalization with varying aspect ratio mapping. This paper proposed some enhanced normalization and feature extraction strategies and estimates their performance with contrast to existing techniques.

**Subhashini, P. P. S., and V. V. K. D. V. Prasad et al. [2**] has proposed, a method based on Radial basis function (RBF). Neural network plays an important role in pattern classification problems. Training neural network was a challenging nonlinear optimization problem. Multiple algorithms have been premeditated for choosing the RBF neural network prototypes and used to train the network. The efficiency of the proposed methodology was tested on the handwritten digits of different fonts and found to be successful in recognizing the digits. This method is tested on handwritten digits of 0 to 9 of 25 different fonts. The success rate of this method for recognizing handwritten digit is viable.

**Bathani, Ms Rinku, and Ms Honey Patel. [3]** has discussed the Container Code Recognition (CCR) can be deployed simply using currently available state of art Optical Character Recognition solution. CCR system consists of four steps: capture the container image from digital camera, pre-processing, character segmentation and character recognition. Container markings follow an International Organization for Standardization standard. There may be additional characters beside eleven ISO characters on the container. Each container ode recognition system uses different grouping of algorithms.

**Jeong, Cha-Sup, and Dong-Seok Jeong. [4]** proposed a method for the recognition of handwritten digits. This method of recognition was based on contour information and Fourier descriptors. First was the preprocessing in which contours of the input digit image was extracted and separate the outer and inner contour from the contour image. Second, the outer contour feature output data was extracted and use them to build standard models. In the last step, digit was recognized by comparing the features of input digits with those of models. 500 data for each digits was used in this paper. So total of 5000 data were used in this paper. The overall recognition rate was 99.04% and to reduce the errors by eliminating the broken contour problems during preprocessing.

**Sahlol, Ahmed T., et al.[5]** described a proficient approach for the recognition of off-line Arabic handwritten characters. It was based on novel preprocessing operations (including different kinds of noise removal and dilation), structural, statistical and topological features from the main body of the character and also from the secondary components. The popular Feed Forward Neural Network was used for classification of methodology which enhances the performance. The proposed algorithm obtained has promising results in terms of accuracy (success rate of 100% for some letters with an average rate of 88%).In comparisons to others work this technique performs well in terms of performance. We hope also that we will complete a system for recognizing handwritten Arabic texts passing through segmentation techniques for segmenting the words to characters.

**Wang, Jeen-Shing, and Fang-Chen Chuang[6]** presents an accelerometer-based digital pen for handwritten digit and motion trajectory recognition applications. The digital pen consists of a triaxial accelerometer, a microcontroller, and an RF wireless transmission module for sensing and collecting accelerations of handwriting and gesture trajectories. Pen was used by the user to write digits or make hand gestures, and the accelerations of hand motions measured by the accelerometer were wirelessly transmitted to a computer for online trajectory recognition. The recognition rate for overall handwritten digit was 98%, and for gesture recognition rates it was 98.75%. This result encourages us to further investigate the possibility of using our digital pen as an effective tool for HCI applications

### 5.2 Different from existing works

The difference of our work from the existing works are, we have used deep learning convolutional neural networks to train our model. Deep learning is a field of machine learning that uses algorithms inspired by how neurons function in the human brain. TensorFlow is a machine learning framework that Google created to design, build, and train deep learning models. We will also use Keras, which runs on top of TensorFlow. Keras was developed with a focus on enabling fast experimentation. This made it possible to achieve up to 99.4% accuracy in recognizing digits. We have implemented our solution in python which is a more popular and general-purpose language so that open source contributor can contribute to our project to make it better as well as integrate it into different systems.

## 6.0 Project Expected Result

We have reviewed some previously implemented Handwritten digit recognition projects which is implemented using Neural network and got 91% accuracy level. It is only capable of recognizing centralized image. If the written digit is not centralized properly may cause into wrong result.

Example of centralized image and result:



Example of decentralized image and result:

To overcome with this issue, we plan to use Convolutional Neural Networks. As a result, we are expecting to have accuracy level of more than 99%.

## 7.0 Data Description

We will be using the MNIST dataset. This dataset was constructed from a number of scanned document datasets available from the National Institute of Standards and Technology. These images were normalized in size and centered. Each image is in a 28x28 square (784 pixels). 60,000 images were used to train a model and 10,000 were used to test it. Excellent results achieve a prediction error of 1%. State-of-the-art results are approximately 0.2% which could be achieved with a large convolutional neural network.

## 8.0 Data Processing

### 8.1 Pre-processing

During our initial review of the various classifier methods, we undertook the approach of processing the training set images to reduce the data by thresholding the given image to a binary image. Fig.1 represents the sample images taken from the MNIST database.



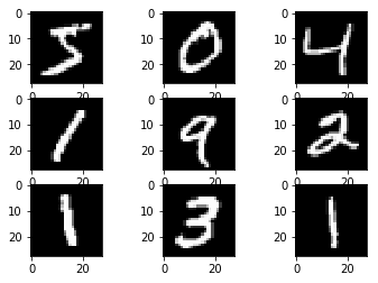
*Figure-1: Sample digits used for training the classifier*

We also looked at various processing methods such as edge‐detection, and thinning of the digit image to get a skeleton of the digit. This approach of acquiring the skeleton of the digit is widely used in the classifiers which mainly reply upon a well‐defined input image for their accuracy. PCA is the holistic approach that extracts eigendigits based on the overall information contained in the image. So, the information such as grayscale values and thickness of the digits actually assist in providing more information. Therefore, extensive pre‐processing was not required in the implementation of the system. The training set was segmented into 10 groups – one for each digit, and then each group was individually passed into the PCA algorithm for the extraction of eigendigits.

### 8.2 Feature Extraction using Principal Component Analysis

Principal component analysis (PCA) is a fundamental multivariate data analysis method which is encountered into a variety of areas in neural networks, signal processing, and machine learning. It is an unsupervised method for reducing the dimensionality of the existing data set and extracting important information. PCA does not use any output information; the criterion to be maximized is the variance.

PCA can be applied to economically represent the input digit images by projecting them onto a low‐dimensional space constituted by a small number of basis images. These basis images or the “eigendigits” are derived by finding the most significant eigenvectors of the pixel wise covariance matrix, after mean centering the data for each attribute



**9.0 Machine Learning Algorithms**

**Baseline Model with Multiplayer Perceptrons**

We start with a baseline model, so we can compare our convolutional neural network that we will use later. To do a multilayer perceptron model, we flatten our 28 by 28-pixel images into a single 784 length vector for each image. We then change the grayscale values from 0-255 to 0-1 to make things easier on our neural network. (Normalization) Finally, we change the categories 1-9 into a binary matrix. Our current neural network structure is as follows:

**Visible Layer (784 Inputs) >> Hidden Layer (784 Neurons) >> Output Layer (10 Outputs)**

**Performance**

With training on 60,000 samples and validation on 10000 samples, we get a baseline error of: **1.76%**.

**Simple Convolutional Neural Network**

As expected, we achieved around 1-2% error which is great. However, we can do better. Here, we take advantage of Keras capability of creating convolutional neural networks. We will use all aspects of a modern CNN implementation, including convolutional layers, pooling layers, and dropout layers.

Here are our changes for the baseline model:

1. We add a convolutional layer with 32 feature maps, with a size of 5 x 5. This is also our input layer which expects images to be added.
2. We then define a pool size of 2 x 2.
3. We randomly dropout 20% of our neurons to reduce the amount of overfitting.
4. We then flatten our data.
5. We add 128 neurons with a rectifier activation function like above.
6. Finally, we use 10 neurons for the 10 prediction classes with a softmax activation function to output probability-like prediction for each class.

Our new neural network structure is as follows:

**Visible Layer (1x28x28 Inputs) >> Convolutional Layer (32 maps, 5x5) >> Max Pooling Layer (2x2) >> Dropout Layer (20%) >> Flatten Layer >> Hidden Layer (128 Neurons) >> Output Layer (10 Outputs)**

**Performance**

With training on 60,000 samples and validation on 10000 samples, we get a CNN error of: **1.07%**.

**Larger Convolutional Neural Network**

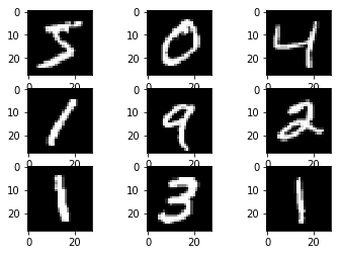
Here we achieved around 1% error which is excellent. However, we can hit state-of-the-art results. To do this, we deepen then widen our neural network.

Our new neural network structure is as follows:

**Visible Layer (1x28x28 Inputs) >> Convolutional Layer (30 maps, 5x5) >> Max Pooling Layer (2x2) >> Convolutional Layer (15 maps, 3x3) >> Max Pooling Layer (2x2) >> Dropout Layer (20%) >> Hidden Layer (128 Neurons) >> Hidden Layer (50 Neurons) >> Output Layer (10 Outputs)**

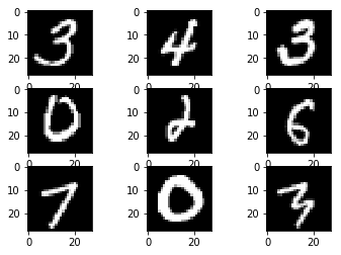
**Baseline**

This is our original image.



**Feature Standardization**

Similar to different scalar values, we can standardize different images. The result that standardizing images brings is slightly darkening and lightening different images.



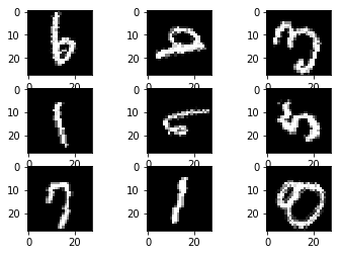
**ZCA Whitening**

Here, we reduce the redundancy of certain pixels in order to highlight certain features of images. Similar to principal component analysis, we use ZCA for images.



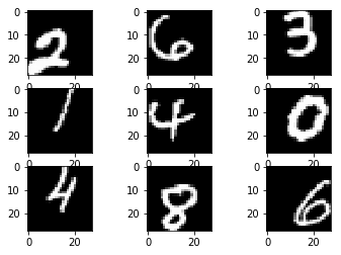
**Random Rotations**

Different people write in different angles. Here, we randomly rotate images up to 90 degrees.



**Random Shifts**

Sometimes numbers won't be exactly centered. Here we randomly shift numbers to be slightly off-centered.



**Performance**

With training on 60,000 samples and validation on 10000 samples, we get a CNN error of: **0.74%**.

## 10.0 Conclusion & Enhancement

With our ability to take advantage of larger convolutional neural network with Keras, we were able to go from 1-2% prediction error to less than 1%, near-state-of-the-art results! However, even with this model there are still further improvements which we can do with image augmentation and a much more powerful GPU. First, we start with the baseline image again, then we do image augmentation to the dataset.

**References**

[1]. Tuan Trung Nguyen, ―Adaptive Classifier Construction: An Approach to Handwritten Digit Recognition‖, J.J. Alpigini et al. (Eds.): RSCTC, LNAI 2475, Springer-Verlag Berlin Heidelberg, pp. 578–585, 2002.

[2]. T. Vasudev, G Hemanth kumar, P. Nagabhushan, ―Transformation of Arc-Form-Text to Linear Form Text Suitable for OCR‖, Pattern Recognition Letters, vol. 28, issue 16, pp.2343-2351, 2007.

[3]. Sonka M., Hlavac V., Boyle R., ―Image Processing Analysis and Machine Vision‖, 2nd Edition, 1999.

[4]. Chen M. Y., Kundu A., Zhou J., ―Offline Handwritten Word Recognition Using HMM Type Stochastic Network‖, IEEE Transaction of Pattern Analysis and Machine Intelligence, 16, 481-496, 1994.

[5]. Saula J., Pietikainen M., ―Adaptive Document Image Binarization‖, Pattern Recognition, vol. 33, issue 2, 2000.

[6]. Dayashankar Singh, Sanjay Kr. Singh, Dr. (Mrs.) Maitreyee Dutta, ―Handwritten Character Recognition Using Twelve Directional Feature Input and Neural Network‖, International Journal of Computer Applications, Volume 1 – No. 3, 2010.