VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECNOLOGY AND ENGINEERING

# AN EFFICIENT METHOD FOR HANDWRITTEN DIGIT RECOGNITION USING CNN

**Done by:**

**MEHUL GUPTA – 16BIT0117**

**Semester: FALL SEM’18-19**

## TABLE OF CONTENTS:

## CERTIFICATE

This is to certify that the project work entitled "**Handwritten digit recognition**" that is being submitted by "**Mehul Gupta**" is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

**Place: VIT, Vellore**
**Date: 30th Oct, 2018**

## ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my faculty Prof. Valarmathi B who gave me the golden opportunity to do this wonderful project on the topic Handwritten digit recognition, and whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project.

Furthermore, I would also like to acknowledge with much appreciation the crucial role of the library staff of VIT University, who gave me the permission to access the journals and research papers and provide me with all the required equipment to complete the task of "HANDWRITTEN DIGIT RECOGNITION". Last but not the least, many thanks go to the head of the department, of School SITE whose has invested his full effort in guiding me in achieving the goal.

I would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

**Mehul Gupta**
**(16BIT0117)**

## ABSTRACT

The handwritten digit recognition problem becomes one of the most famous problems in machine learning and computer vision applications. Many machine learning techniques have been employed to solve the handwritten digit recognition problem. The project describes one such approach to solve this traditional problem using CNN (Convolutional Neural Networks). The proposed CNN is presented using a layering approach. This approach provides a concise and accessible understanding of the main mathematical operations of a CNN. The handwritten digit recognition model is trained using the MNIST handwritten digit dataset and is used to experiment the performance of the proposed CNNs.

Further, a well-designed GUI is made for real-time handwritten digit recognition which accepts a drawn digit from the user and predicts the pattern.

However, there are deficient works accomplished on Arabic pattern digits because Arabic digits are more challenging than English patterns. Hence, the lacking research of using Arabic digits endeavours me to dig deeper by expanding my research to Arabic handwritten digits on the MADBase database which consists of more than 60,000 samples. As a challenging dataset is used for evaluation, a robust deep convolutional neural network is used for classification and superior results are achieved.

**OBJECTIVE:** To design an efficient pre-processing and feature extraction system to produce adequate data structures suitable for the classification task of handwritten digit recognition.

## EXISTING SYSTEM:

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset.
In 2004, a best-case error rate of 0.42 percent was achieved on the database by researchers using a new classifier called the LIRA, which is a neural classifier with three neuron layers based on Rosenblatt's perceptron principles.
Recently, the single convolutional neural network best performance was 0.31 percent error rate.

For the Arabic handwritten digits dataset, the paper "CNN for Handwritten Arabic Digits Recognition Based on LeNet-5" trained and tested the MADBase database (Arabic handwritten digits images) that contain 60000 training and 10000 testing images and managed to get an average recognition accuracy of around 99.15%.

## PROPOSED SYSTEM:

The main contribution of this project is that a multi-layered architecture of CNN is been proposed, and also introduce an innovative coding scheme of the most relevant parameters of CNN, covering for the first time the evolution of all aspects of design, including the architecture, the activation functions, the learning hyperparameters, etc. A comparison is held amongst the results, and it is shown by the end that the use of CNN was leaded to significant improvements across different machine-learning classification algorithms, and is able to outperform the existing accuracy achieved in Arabic handwritten digits dataset.
A Compute Unified Device Architecture (CUDA) implementation of the Convolutional Neural Network (CNN) for a digit recognition system is proposed to reduce the computation time and achieve high accuracy, thus making it an efficient model.

## DATASET DESCRIPTION:

### I) Dataset-1 (Recognition of European numerals):

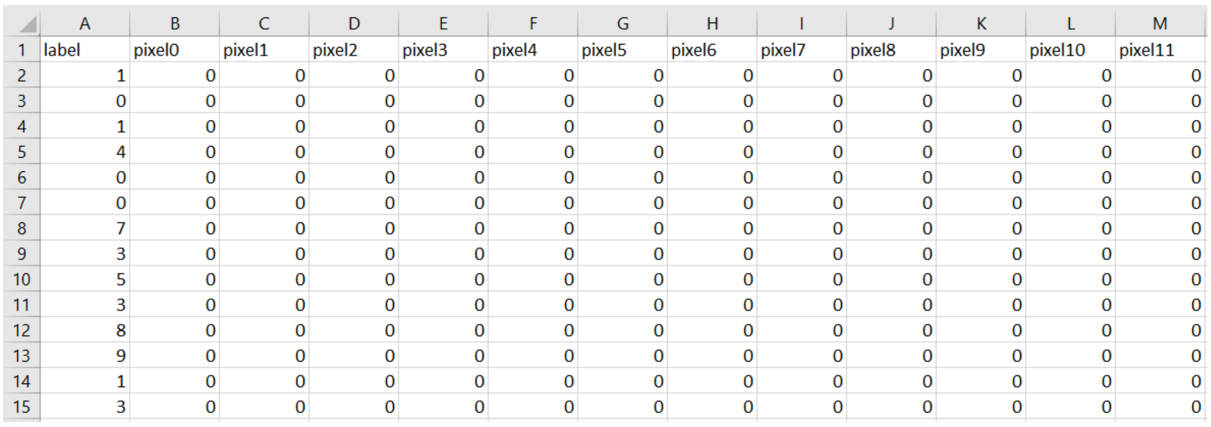
**a) Source link:** The dataset is the de facto “Hello world” of computer vision. It has been downloaded from the official link http://yann.lecun.com/exdb/mnist/
For further testing and visualizing the classifier produced, MNIST dataset is downloaded from https://www.kaggle.com/c/digit-recognizer/data

**b) Dataset description:**

The dataset chosen is the famous MNIST dataset of handwritten digits. The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. It was created by "re-mixing" the samples from NIST's original datasets.

The MNIST database contains **60,000 training images and 10,000 testing images.** The original black and white (bilevel) 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.

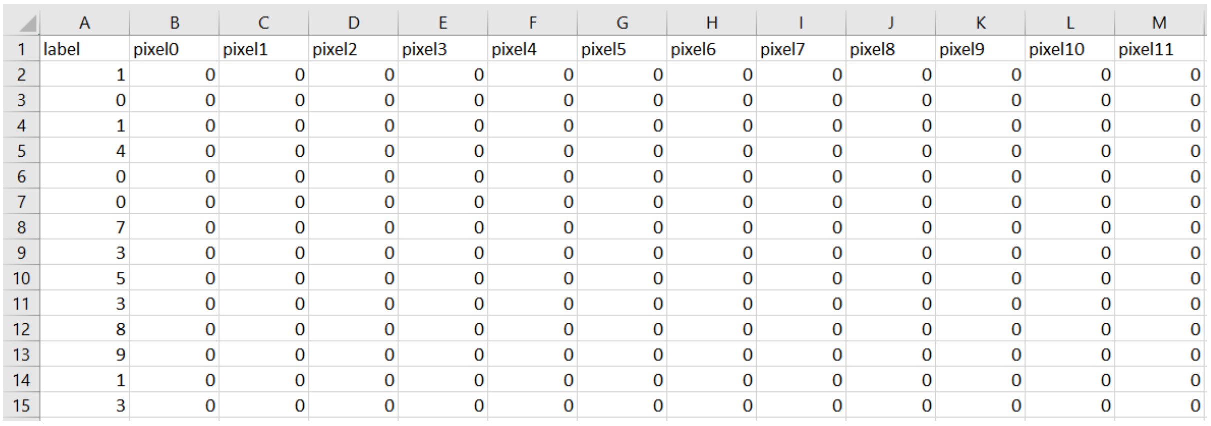
**c) Sample dataset:**



| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | label | pixel0 | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | pixel9 | pixel10 | pixel11 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

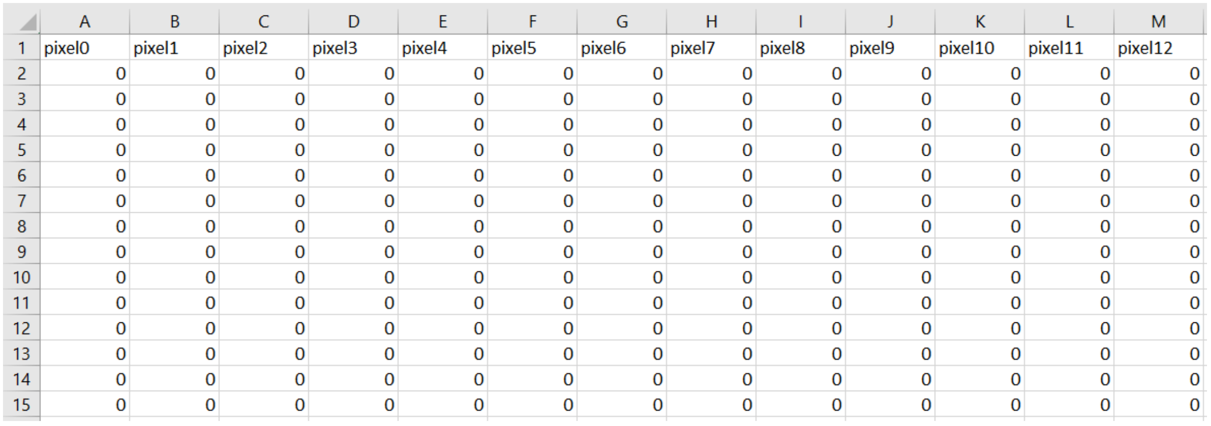
**d) Training and testing dataset:**

Training dataset:



| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | label | pixel0 | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | pixel9 | pixel10 | pixel11 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Testing dataset:



| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | pixel0 | pixel1 | pixel2 | pixel3 | pixel4 | pixel5 | pixel6 | pixel7 | pixel8 | pixel9 | pixel10 | pixel11 | pixel12 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## II) Dataset-2 (Arabic handwritten digits)

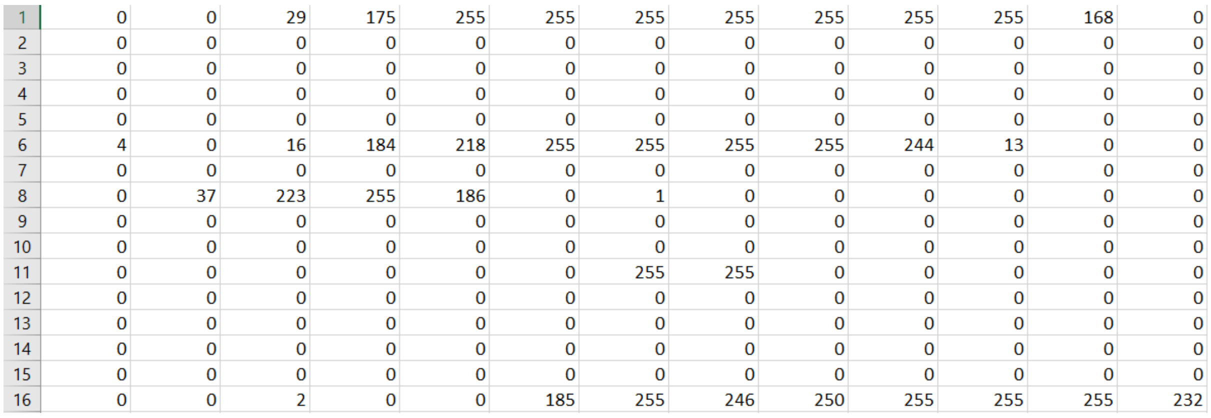
**a) Source link:** The dataset of Arabic handwritten digits has been downloaded from the link https://www.kaggle.com/mloey1/ahdd1/home

**b) Dataset description:**

The MADBase is modified Arabic handwritten digits database contains 60,000 training images, and 10,000 test images. MADBase were written by 700 writers. Each writer wrote each digit (from 0-9) ten times. To ensure including different writing styles, the database was gathered from different institutions: Colleges of Engineering and Law, School of Medicine, the Open University (whose students span a wide range of ages), a high school, and a governmental institution.

MADBase is available for free and can be downloaded from (http://datacenter.aucegypt.edu/shazeem/).

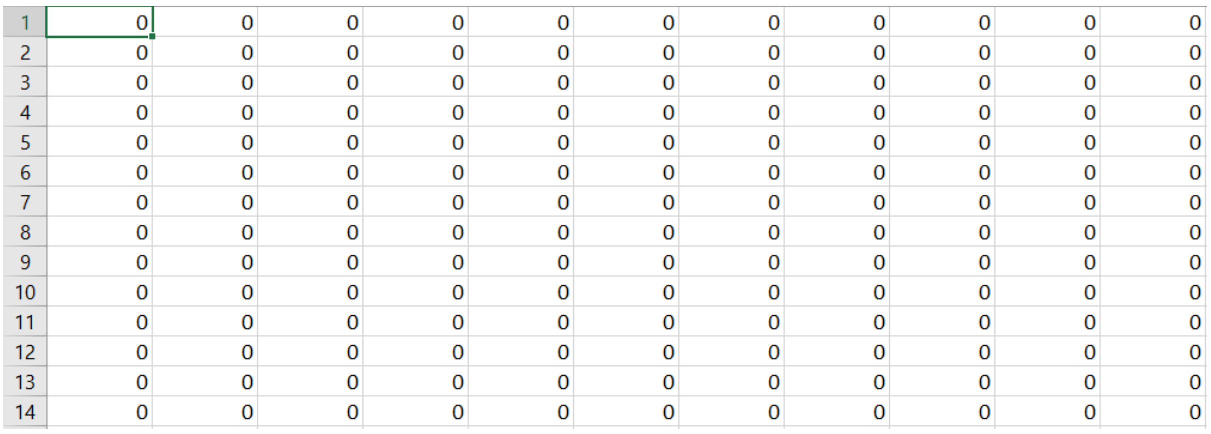
## c) Sample dataset:



| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 29 | 175 | 255 | 255 | 255 | 255 | 255 | 255 | 255 | 168 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 4 | 0 | 16 | 184 | 218 | 255 | 255 | 255 | 255 | 244 | 13 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 37 | 223 | 255 | 186 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 255 | 255 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 16 | 0 | 0 | 2 | 0 | 0 | 185 | 255 | 246 | 250 | 255 | 255 | 255 | 232 |

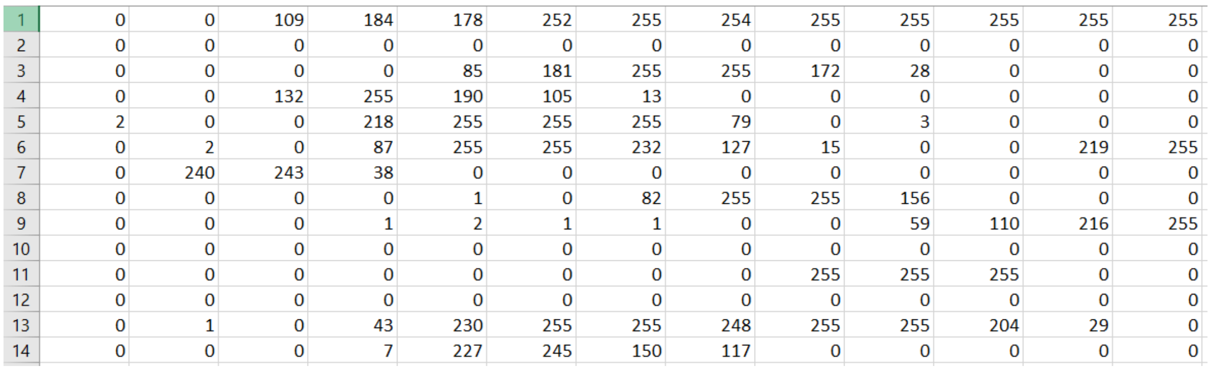
## d) Training and testing dataset:

Training dataset:



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Testing dataset:



| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 109 | 184 | 178 | 252 | 255 | 254 | 255 | 255 | 255 | 255 | 255 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 85 | 181 | 255 | 255 | 172 | 28 | 0 | 0 | 0 |
| 4 | 0 | 0 | 132 | 255 | 190 | 105 | 13 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 2 | 0 | 0 | 218 | 255 | 255 | 255 | 79 | 0 | 3 | 0 | 0 | 0 |
| 6 | 0 | 2 | 0 | 87 | 255 | 255 | 232 | 127 | 15 | 0 | 0 | 219 | 255 |
| 7 | 0 | 240 | 243 | 38 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 82 | 255 | 255 | 156 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 1 | 2 | 1 | 1 | 0 | 0 | 59 | 110 | 216 | 255 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 255 | 255 | 255 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 13 | 0 | 1 | 0 | 43 | 230 | 255 | 255 | 248 | 255 | 255 | 204 | 29 | 0 |
| 14 | 0 | 0 | 0 | 7 | 227 | 245 | 150 | 117 | 0 | 0 | 0 | 0 | 0 |

## LITERATURE SURVEY:

The following research papers are surveyed for the given problem statement:

| S.no | Paper Description | Algorithm used | Dataset used | Performance measures | Problems in Existing methods | Proposed methods |
|---|---|---|---|---|---|---|
| 1. | **BASE PAPER:**<br>**A Minimal Convolutional Neural Network for Handwritten Digit Recognition (2017)**<br><br>- Matthew Y.W. Teow<br><br>The contribution of this paper is to bridge the gap on understanding the mathematical structure and the computational implementation of a convolutional neural network using a minimal model. | LeNET5 | MNIST | 99.5% accuracy rate with LeNET5 | May not be able to classify digits with precision due to its minimal nature | May be applied to visual object recognition where small errors don't affect the prediction value. |
| 2. | **Handwritten digit segmentation: Is it still necessary? (2018)**<br><br>-A.G. Hochuli, L.S. Oliveira, A.S. Britto Jr, R. Sabourin<br><br>This paper presents that handwritten digit segmentation can be successfully replaced by a set of classifiers trained to predict the size of the string and classify them without any segmentation. | CNN | Touching Pairs Dataset and NIST SD19 | TP database: 97% accuracy, | The digit classification comprises 3 classifiers that are responsible for classifying isolated digits, and 2-digit, and 3-digit strings. This may lead to overfitting. | To avoid a hard decision and overcome some of the confusion caused by the length classifier, the fusion method may use the outputs of two-digit classifiers to produce the final decision. |
| 3. | **Review of Handwritten Pattern Recognition of Digits and Special characters using Feed Forward Neural Network and Izhikevich Neural Model (2014)**<br><br>- Mrs. Soni Chaturvedi; Ms. Rutika Titre; Ms. Neha Sondhiya<br><br>In this paper a Feed Forward Neural Network and an Izhikevich neuron model is applied for pattern recognition of Digits and Special characters. | ANN and SNN (Spiking Neural networks) | 10 random small samples | Izhikevich model has good computational efficient as compared to feed-forward neural networks | Feed forward Neural Network has limited computational power over Izhikevich neuron | Training the feed-forward neural network on parameters like Accuracy, Efficiency and simulation time. |
| 4. | **Hand written Digit Recognition System for South Indian Languages using Artificial Neural Networks**<br><br>- Leo Pauly, Rahul D Raj, Dr.Binu paul | ANN | Performed on a scanned RGB image of resolution 1229 x 1888 | Accuracy of 83.4% was obtained by the use of HOG features. | Work only when the digits are isolated and samples are collected in constrained conditions. | HOG is a feature descriptor that generalizes an image so that the same image when viewed in discrete |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  | In this paper a novel approach for recognition of handwritten digits for South Indian languages using artificial neural networks (ANN) and Histogram of Oriented Gradients (HOG) features is presented. |  |  |  |  | conditions produces the same feature descriptor as close as possible. |
| 5. | **Handwritten Digit Feature Extraction and Classification Using NN (1993)**<br><br>- B.K. Dolenko and H.C. Card<br><br>Presents results obtained on a handwritten numeral classification problem using NN | NN trained using back-propagation | Developed at University of Windsor | 1) Cascade correlation networks: 41.6% and rejects for 1% error<br>2) For two-layered perceptron was: 35.5% rejects for 1% error. | A more elaborate conjugate gradient algorithm used on this network did not yield significantly better classification Results as it required a longer training time | Hierarchical nature of the network can be crucial in solving the problem. The training algorithm can be parallelised using CUDA to reduce training time. |
| 6. | **Persian Handwritten Digit Recognition by Random Forest and Convolutional Neural Networks (2015)**<br><br>- Yasin Zamani, Yaser Souri, Hossein Rashidi and Shohreh Kasaei<br><br>Comparative study between random forest and CNN algorithms for Persian handwritten digit recognition. | Random forest (RF) and convolution al neural network (CNN) | Hoda Dataset | The efficiency of the model was evaluated using the confusion matrix generated. | RF and the state of art methods devised a gap in the learning phase | Investigating other pre-processing stages that will close the gap on RFs and the state-of-the-art methods on this dataset. Probably using CNN can help achieve better performance. |
| 7. | **Classification of Persian Handwritten digits using Spiking neural networks (2015)**<br><br>- Kourosh Kiani; Elmira Mohsenzadeh Korayem<br><br>In this paper SNN Model is used, in order to have robust learning and classification of handwritten digits | Spiking Neural Networks and Deep beliefs Networks. | Hoda Persian handwritten digits dataset | The accuracy achieved is 95%. | Due to the similarities among handwritten digits, the classifications have been erratic. | Deep Belief Networks solves this problem to a great extent by adding additional layer. |
| 8. | **Ncfm: Accurate Handwritten Digits Recognition using Convolutional Neural Networks (2016)**<br><br>- Yan Yin, JunMin Wu, HuanXin Zheng<br><br>In this paper, Ncfm (No combination of feature maps) has been introduced, a novel technique | Ncfm (No combination of feature maps) to improve CNN | MNIST | The accuracy achieved is 99. 81 % on MNIST datasets. | Ncfm is a technique that applied to convolutional layers to avoid the information loss, hence may lead to overfitting. | Using dropout function to reduce the chances of overfitting |

| | | | | | | |
|---|---|---|---|---|---|---|
| | to improve the performance of CNNs. | | | | | |
| 9. | **Parallelization of Digit Recognition System Using Deep Convolutional Neural Network on CUDA (2017)**<br><br>Srishti Singh; Amrit Paul; Dr. Arun M<br><br>A Compute Unified Device Architecture (CUDA) implementation of Deep Convolutional Neural Network (DCNN) for a digit recognition system is proposed to reduce the computation time of ANN and achieve high accuracy. | DCNN | MNIST (Modified National Institute of Standards and Technology) and EMNIST (Extended MNIST) database | 99.49% for MNIST dataset and 99.62% for the EMNIST dataset | Currently tensorflow does not allocate all available memory like it says in the documentation | Manually allocating a GPU for the batch job |
| 10. | **A Comparative Study on Handwriting Digit Recognition Using Neural Networks (2017)**<br><br>- Mahmoud M. Abu Ghosh; Ashraf Y. Maghari<br><br>This paper focuses on deep neural network (DNN), deep belief network (DBN) and convolutional neural network (CNN). | DNN, DBN and CNN | Random and standard dataset of handwritten digit have been used for conducting the experiments. | 98.08% accuracy rate | Margins of errors may occur with similarities between the digits | Backpropagating the errors inn such a way that it won't overfit the model. |
| 11. | **Handwritten Digits Recognition with Artificial Neural Network (2017)**<br>- Kh Tohidul Islam, Ghulam Mujtaba, Dr. Ram Gopal Raj, Henry Friday Nweke<br><br>In this study, a multi-layer fully connected neural network with one hidden layer for handwritten digits recognition is implemented. | ANN | MNIST database | 99.60% accuracy rate | Low optimised parameters for ANN which leaves a scope for higher accuracy. | Using a combination hybrid method of feature extraction with ensemble classifier. |
| 12. | **Handwritten digits recognition using ensemble neural networks and ensemble decision tree** (2017)<br><br>- Retno Larasati, Dr. Hak KeungLam<br><br>In this project, neural networks ensembles combined with another classifier are train and test in solving handwritten digit recognition problems. | Ensemble neural networks combined with ensemble decision tree (ENNEDT) | USPS and MNIST database. | ENNEDT reached 84% accuracy from classifying USPS dataset. | The new proposed (ENNEDT), performed better than single NN and ensemble neural network. | The accuracy of the model can be further increased by changing the mapping function, feature extraction, or many features that were used here. |

| | | | | | | |
|---|---|---|---|---|---|---|
| 13. | **Handwritten Digit Recognition Based on Depth Neural Network (2017)**<br><br>- Yawei Hou; Huailin Zhao;<br><br>In order to study the network model with high recognition rate, this paper firstly studies the double hidden layer BP neural network with different feature extraction. Secondly, the CNN is studied. | BP neural network and CNN | MNIST | Accuracy of 99.55% is achieved by the combined network. | Classification results of hybrid networks is better than the single network. E.g. time in computation is generally lower. | A more accurate recognition result can be achieved by the combined network (CNN + BP). |
| 14. | **Learning and Real-time Classification of Hand-written Digits with Spiking Neural Networks (2017)**<br><br>- Shruti R. Kulkarni, John M. Alexiades, Bipin Rajendran<br><br>It describes a novel spiking neural network (SNN) for automated, real-time handwritten digit classification and its implementation on a GP-GPU platform. | SNN (Spiking neural network) | MNIST database | Achieves an accuracy of 98.06% on the MNIST test dataset | Sometimes, GPU fails to allocate a virtual memory | Can be implemented by manually allocating a memory. Can be used for touch-screen based platform for real-time classification of user-generated images. |
| 15. | **Devanagari Digit Recognition by using Artificial Neural Network (2017)**<br><br>- Reena Dhakad; Dinesh Soni;<br><br>This paper describes a holistic system of offline written Devanagari digit recognition. | ANN | The algorithm was tested solely on 500 samples, which was pre-processed to scale the images having 32 x 32 (1024) feature coefficients | 93.21% accuracy on a syllabic script written digits set | Digit recognizers square measure advanced if they are general purpose however are easily if it is supported specific lexicon. | The accuracy of projected theme could also be increased by increasing the quantity of coaching samples and or applying the projected theme at completely different resolution theme. |
| 16. | **Handwritten Indian numerals recognition system using probabilistic neural networks (2004)**<br><br>- Faruq A. Al-Omari, Omar Al-Jarrah<br><br>This paper presents a system for the recognition of the handwritten Indian numerals one to nine (1–9) using a probabilistic neural network (PNN) approach. | Probabilistic neural network (PNN) | The dataset was collected from 120 persons who provided them with handwritten digit strings containing all nine digits 1–9. | A 99.72% recognition rate was achieved using 12 entries feature vector. | Recognition of handwritten numeral digits is a more tedious task due mainly to the handwriting style of the writer subject to inter, and intra-writer variations. | The system utilizes a PNN approach with feature vectors representing lengths of vectors taken from the COG of the digit object to its boundary. |
| 17. | **Classification of Handwritten Digits Using Evolving Fuzzy Neural Network (2004)** | The neuro-fuzzy model of evolving fuzzy neural | The data set consists of an N by F matrix whereby N | Accuracy is judged based on the table which | A reduction in the number of features may lead to a loss | A system can be developed using greater number of features |

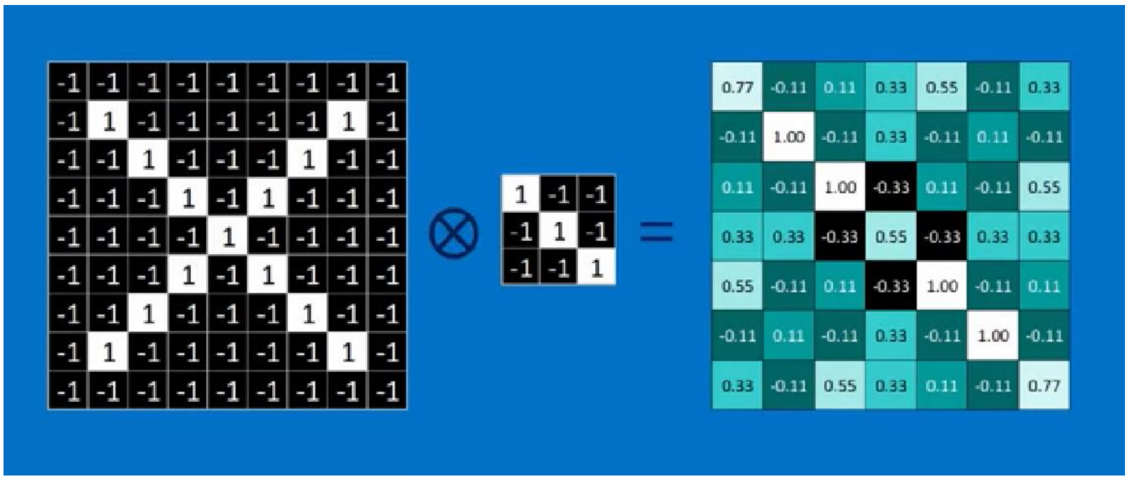
|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
|  | G. S. Ng, T. Murali, A. Wahab, N. Sriskanthan<br><br>This paper aims to analyse and obtain the optimal number of features that produces the most effective classification using EFuNN. | network (EFuNN) | ranges from 1 to 5000 and F varies from 1 to 16. | ranks system in descending order of recall and c1assiticatio n accuracy. | in the discrimination power and thereby the accuracy of the resulting recognition system. | ensuring greater classification accuracy. |
| 18. | **Comparison of Different Neural Network Architectures for Digit Image Recognition (2011)**<br><br>-Hao Yu, Tiantian Xie, Michael Hamilton and Bogdan Wilamowski<br><br>The paper presents the design of three types of neural networks with different features, including traditional backpropagation networks, radial basis function networks and counter-propagation networks. | Backpropaga tion networks, radial basis function networks and counter-propagation networks. | Sample consisting of image data in the 1st column and noised image data, from the 2nd column to the 8th column | Traditional backpropaga tion networks have best network efficiency to solve the problem – only 10 units are required; while both networks need 20 units. | Comparing with second order algorithms, both of the training speed and network efficiency are far from optimal when EBP algorithms are applied for training. | Algorithms for noise reductions can further improve its efficiency. |
| 19. | **Digit Recognition Using Single Layer Neural Network with Principal Component Analysis (2014)**<br>-Vineet Singh, and Sunil Pranit Lal<br><br>This paper presents an approach to digit recognition using single layer neural network classifier with Principal Component Analysis (PCA). | Backward propagation (BP) neural network | MNIST dataset | The proposed system was able to obtain 98.39% accuracy on the MNIST 10,000 test dataset. | Some misclassified digits are ambiguous either by unclear writing or segmentation problem. | This can be improved by increasing the feature and to use multiply feature extraction techniques. A combination of classifiers can also be used but this will increase computation. |
| 20. | **Handwritten digit recognition using multilayer feedforward neural networks with periodic and monotonic activation functions (2002)**<br><br>- Kwok-wo Wong Chi-sing Leung Sheng-jiang Chang<br><br>The problem of handwritten digit recognition is dealt with by multilayer feedforward neural networks with different types of neuronal activation functions. | Multilayer feedforward neural networks | The handwritten digit database previously used by Wallis was chosen. | 90% of the unpruned recognition rate in the test set is recorded | Activation functions such as sigmoid, hyperbolic and sigmoid packet functions suffer from the drawbacks of slow convergence and poor nonlinearity mapping. | More weights can be pruned by using the sinusoidal activation function. |

## ALGORITHM DESCRIPTION:

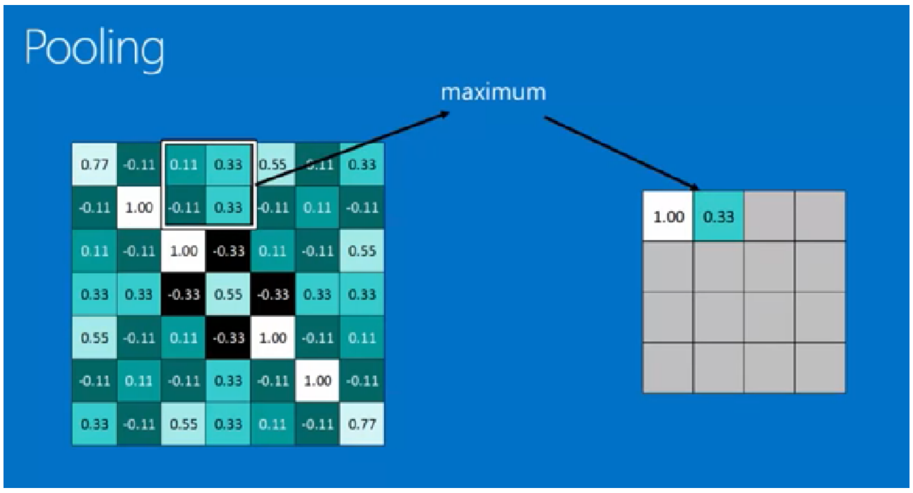
CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.

**Step1 (Input):** This CNN takes as input tensors of shape (image_height, image_width, image_channels). In this case, I configured the CNN to process inputs of size (28, 28, 1).

**Step2 (Conv2D Layer):** The Conv2D layers are used for the convolution operation that extracts features from the input images by sliding a convolution filter over the input to produce a feature map. Here I choose feature map with size 5x5 for the first group of the model and a feature map of 3x3 for the second and the third group.



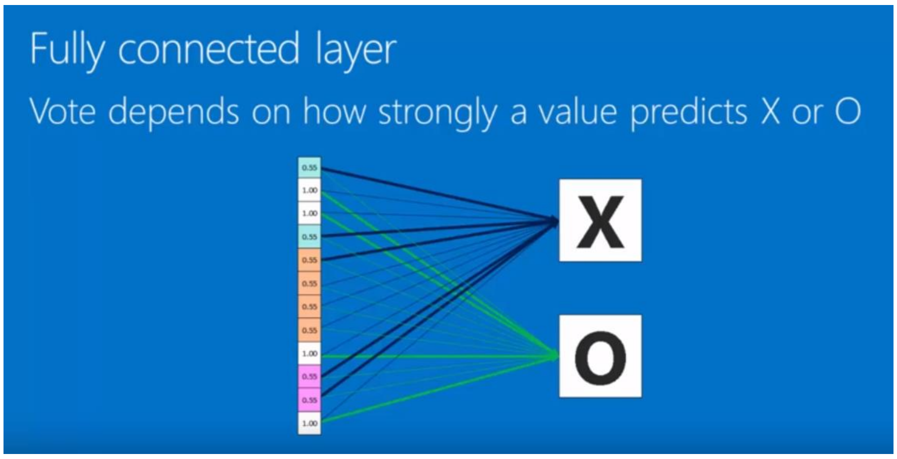
**Step3 (MaxPooling Layer):** The MaxPooling2D layers are used for the max-pooling operation that reduces the dimensionality of each feature, which helps shorten training time and reduce number of parameters. Here I choose the pooling window with size 2 x 2 for all the groups.



**Step4 (Normalization):** To normalize the input layers, I used the BatchNormalization layers to adjust and scale the activations. Batch Normalization reduces the amount by what the hidden unit values shift around (covariance shift). Also, it allows each layer of a network to learn by itself a little bit more independently of other layers.

**Step5 (Dropout):** To combat overfitting, I use the Dropout layers, a powerful regularization technique. Dropout is the method used to reduce overfitting. It forces the model to learn multiple independent representations of the same data by randomly disabling neurons in the learning phase. For example, the layers will randomly disable 40% of the outputs in all the groups.

**Step6 (Flatten):** In the end, we use Flatten() to get the image dimensionality down to 1D and add 2 Dense fully-connected layers on top. The Dense layers process 1D image vectors for our output.



## PROPOSED ARCHITECTURE:

The algorithm uses a 5 Layer Deep Convolutional Neural Network to classify handwritten digits with an accuracy of 99.47%.

The first and second layers of the model are the Conv2D layers which are used for the convolution operation that extracts features from the input images by sliding a convolution filter over the input to produce a feature map.

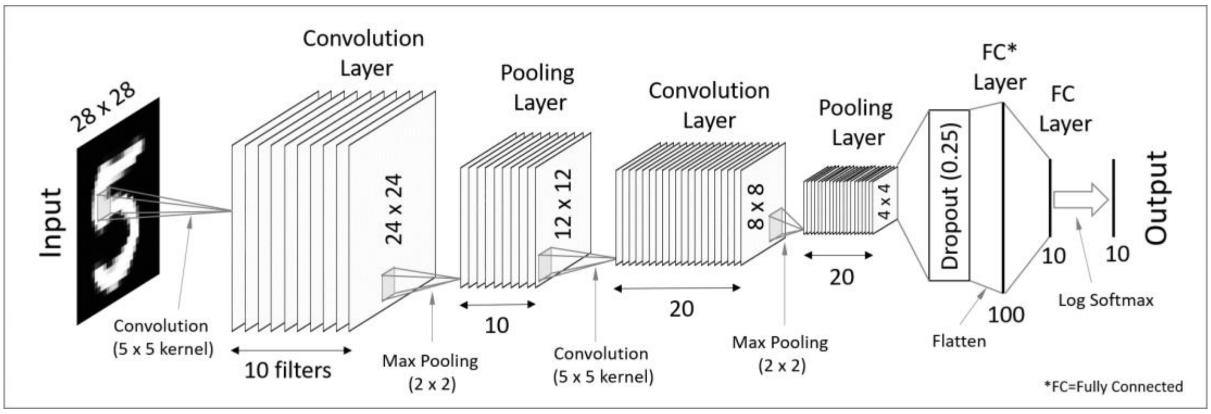
The next layer in the sequence is the MaxPooling2D layer used for the max-pooling operation that reduces the dimensionality of each feature, which helps shorten training time and reduce number of parameters.

The next layer in the architecture is the BatchNormalization layer to adjust and scale the activations, followed by a Dropout layer to reduce overfitting.

The layers are repeated in a similar fashion once again followed by again a Con2D layer and a BatchNormalization layer. Apart from these, model uses **softmax** activation which enables me to calculate the output based on the probabilities. Each class is assigned a probability and the class with the maximum probability is the model's output for the input.
The model uses "**relu**" activation function because "relu" improves neural network by speeding up the training process.

The model uses 5 Conv2D layers , 2 MaxPool2D, 3 layers of BatchNormalization and 4 layers of Dropout.

### Architecture of CNN:



## CODE:

## Sample Code to generate model:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd

from keras.utils.np_utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, Ba
tchNormalization
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import LearningRateScheduler
import tensorflow as tf

config = tf.ConfigProto()
config.gpu_options.allow_growth = True
session = tf.Session(config=config)

train_data = pd.read_csv("train.csv")
test_data = pd.read_csv("test.csv")

Y_train = train_data["label"]
X_train = train_data.drop(labels = ["label"],axis = 1)
X_train = X_train / 255.0
X_test = test_data / 255.0
X_train = X_train.values.reshape(-1,28,28,1)
X_test = X_test.values.reshape(-1,28,28,1)
Y_train = to_categorical(Y_train, num_classes = 10)

datagen = ImageDataGenerator(
        rotation_range=10,
        zoom_range = 0.1,
        width_shift_range=0.1,
        height_shift_range=0.1)

datagen.fit(X_train)

X_train, X_val, Y_train, Y_val = train_test_split(X_train, Y_train, tes
t_size = 0.1)

model = Sequential()

model.add(Conv2D(32, kernel_size=5,input_shape=(28, 28, 1), activation
= 'relu'))
model.add(Conv2D(32, kernel_size=5, activation = 'relu'))
model.add(MaxPool2D(2,2))
model.add(BatchNormalization())
model.add(Dropout(0.4))

model.add(Conv2D(64, kernel_size=3,activation = 'relu'))
model.add(Conv2D(64, kernel_size=3,activation = 'relu'))
model.add(MaxPool2D(2,2))
model.add(BatchNormalization())
model.add(Dropout(0.4))

model.add(Conv2D(128, kernel_size=3, activation = 'relu'))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dense(256, activation = "relu"))
model.add(Dropout(0.4))
model.add(Dense(128, activation = "relu"))
model.add(Dropout(0.4))
model.add(Dense(10, activation = "softmax"))

optimizer=Adam(lr=0.001)
model.compile(optimizer = optimizer , loss = "categorical_crossentropy"
, metrics=["accuracy"])

model.summary()

annealer = LearningRateScheduler(lambda x: 1e-3 * 0.95 ** x)
model_try = model.fit_generator(datagen.flow(X_train,Y_train, batch_siz
e=32), epochs = 30, validation_data = (X_val,Y_val), verbose = 1, steps
_per_epoch=300, callbacks=[annealer])

predictions = model.predict(X_test)
predictions = np.argmax(predictions,axis = 1)
predictions = pd.Series(predictions, name="Label")
submit = pd.concat([pd.Series(range(1,28001),name = "ImageId"),predicti
ons],axis = 1)
submit.to_csv("result.csv",index=False)
```

## RESULTS AND DISCUSSION:

Using the same epoch rate and batch size, the model is trained with a test accuracy of about 99.47% fluctuating in every iteration. The model accurately predicts the hand drawn digits on the canvas even with distortions, cursive writing, extra line segments, and even manages to predict correct digits with slight rotations (**~45 degrees**). In 2004, a best-case error rate of 0.42 percent was achieved on the database by researchers using a new classifier called the LIRA, which is a neural classifier with three neuron layers based on Rosenblatt's perceptron principles

The model took nearly 7 min to train on 60,000 training images in MNIST dataset on an Intel(R) Core (TM) i5-8250U CPU with NVIDIA GeForce-940MX.

For the Arabic handwritten digit recognition, an accuracy of whooping 99.38% is achieved.

## Data preparation:

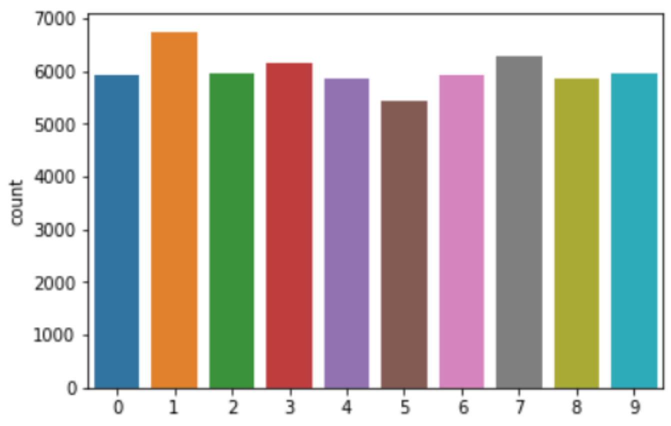
1) Number of instances of each digit in the dataset:

```python
import pandas as pd
import numpy as np
df=pd.DataFrame({'Labels':pd.Series(y_train)})
num=pd.DataFrame(dtype=float)
for j in range(10):
    ser=pd.Series(df[df['Labels']==j].index.values)
    num=pd.concat([num,ser], ignore_index=True, axis=1).astype(float)
print(num.notnull().sum())
```

```
0    5923
1    6742
2    5958
3    6131
4    5842
5    5421
6    5918
7    6265
8    5851
9    5949
dtype: int64
```

Plotting the number of instances:

```python
import seaborn as sns
g = sns.countplot(y_train)
```



## 2) Trained model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 24, 24, 32)        832
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 20, 20, 32)        25632
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
batch_normalization_10 (Batc (None, 10, 10, 32)        128
_________________________________________________________________
dropout_13 (Dropout)         (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 8, 8, 64)          18496
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 6, 6, 64)          36928
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
batch_normalization_11 (Batc (None, 3, 3, 64)          256
_________________________________________________________________
dropout_14 (Dropout)         (None, 3, 3, 64)          0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 1, 1, 128)         73856
_________________________________________________________________
batch_normalization_12 (Batc (None, 1, 1, 128)         512
_________________________________________________________________
flatten_4 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               33024
_________________________________________________________________
dropout_15 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_16 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_12 (Dense)             (None, 10)                1290
=================================================================
Total params: 223,850
Trainable params: 223,402
Non-trainable params: 448
_________________________________________________________________
```

```python
scores = model.evaluate(X_val, Y_val, verbose=0)
print("Training Accuracy: %.2f%%" % (scores[1]*100))
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

```
Training Accuracy: 99.52%
CNN Error: 0.48%
```

## 3) Confusion matrix:

```python
import sklearn.metrics as metrics
from keras.datasets import mnist
from keras.utils import np_utils

(X_train, y_train), (X_test, y_test) = mnist.load_data()
num_pixels = X_train.shape[1] * X_train.shape[2]
X_test = X_test/255.0

X_test = X_test.reshape(-1,28,28,1)

y_pred_ohe = model.predict(X_test)
y_pred_labels = np.argmax(y_pred_ohe, axis=1)  # only necessary if output has one-hot-encoding, shape=(n_samples)

confusion_matrix = metrics.confusion_matrix(y_true=y_test, y_pred=y_pred_labels)
confusion_matrix
```

```
array([[ 976,    0,    0,    0,    0,    0,    4,    0,    0,    0],
       [   0, 1129,    1,    0,    0,    0,    1,    4,    0,    0],
       [   1,    0, 1029,    2,    0,    0,    0,    0,    0,    0],
       [   0,    0,    0, 1009,    0,    1,    0,    0,    0,    0],
       [   0,    0,    0,    0,  978,    0,    0,    0,    1,    3],
       [   0,    0,    0,    3,    0,  888,    1,    0,    0,    0],
       [   0,    1,    0,    0,    0,    1,  954,    0,    2,    0],
       [   0,    1,    4,    1,    1,    0,    0, 1020,    0,    1],
       [   0,    0,    1,    0,    0,    1,    0,    0,  972,    0],
       [   0,    0,    0,    1,    4,    1,    0,    0,    0, 1003]],
      dtype=int64)
```

## 4) Predicted images:

```
Predicted:  [4]
```

```
Predicted:  [6]
```

```
Predicted:  [2]
```

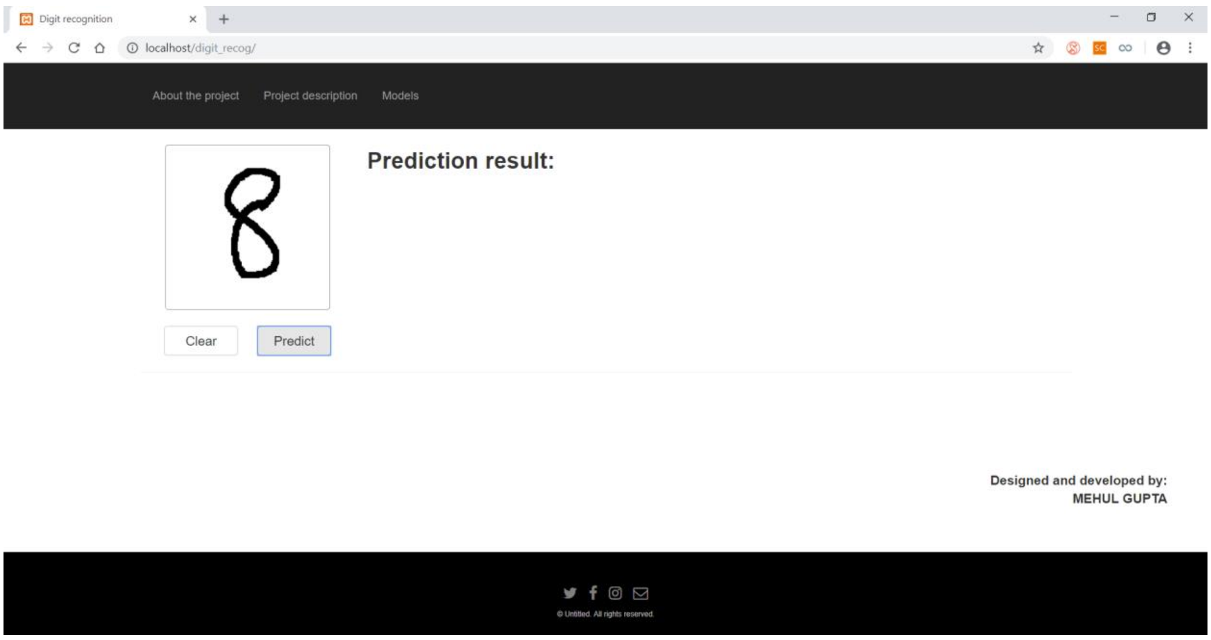
```
Predicted:  [5]
```

```
Predicted:  [3]
```

```
Predicted:  [8]
```

## Screenshots:

### Front-end for drawing digits:



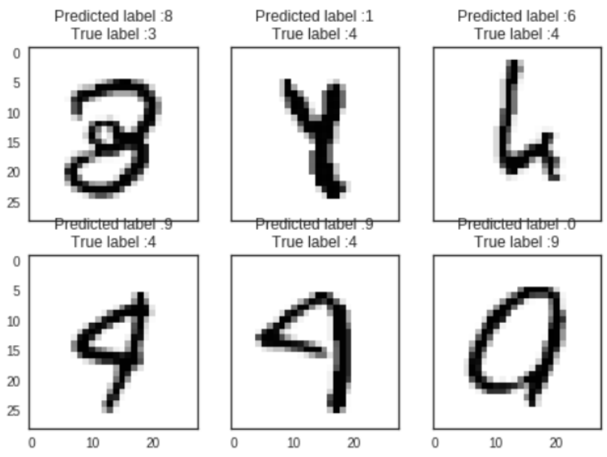
Prediction result:

Designed and developed by:
MEHUL GUPTA

## CONCLUSION:

While the model works excellent in predicting handwritten digits, it leaves very less scope in the 'test' dataset digit prediction as well. The digits misclassified in the dataset can be confused by a human as well. Hence, those digits are difficult to differentiate based on their features. For e.g., there is very less difference in the shapes of '4' and '9' when the curve is smooth.

Some of the misclassified digits are plotted below:



As we can see, these digits can easily be confused for another digits, hence the accuracy leaves very little scope to classify digits further.

Hence, given its less margin to expand and improve, the model performs outstanding in its defined boundaries and predicts handwritten digits accurately. Next, the CNN model has provided a concise and accessible understanding to the key computational structure of the latest CNNs. Lastly, a detail analysis and discussion on the layers of the model also had been presented here to help understand how CNNs learn image visual features.

## FUTURE WORK:

The algorithm can be re-designed such that if the digits are misclassified by the classifier, then the correct target label of the pattern digit can be re fed into the classifier to increase its performance.
In modern world, all the transactions carried out by means of cheque for example, need not be written along a straight line, or any running note need not be carried out in straight line fashion. Hence, the classifier can be trained to predict digits drawn at an angle to the horizontal.

## REFERENCES:

[1] A Minimal Convolutional Neural Network for Handwritten Digit Recognition (2017)- Matthew Y.W. Teow
[2] Handwritten digit segmentation: Is it still necessary? (2018) -A.G. Hochuli, L.S. Oliveira, A.S. Britto Jr, R. Sabourin
[3] Review of Handwritten Pattern Recognition of Digits and Special characters using Feed Forward Neural Network and Izhikevich Neural Model (2014) - Mrs. Soni Chaturvedi; Ms. Rutika Titre; Ms. Neha Sondhiya
[4] Hand written Digit Recognition System for South Indian Languages using Artificial Neural Networks - Leo Pauly, Rahul D Raj, Dr.Binu paul
[5] Handwritten Digit Feature Extraction and Classification Using NN (1993) - B.K. Dolenko and H.C. Card
[6] Persian Handwritten Digit Recognition by Random Forest and Convolutional Neural Networks (2015)- Yasin Zamani, Yaser Souri, Hossein Rashidi and Shohreh Kasaei
[7] Classification of Persian Handwritten digits using Spiking neural networks (2015) - Kourosh Kiani; Elmira Mohsenzadeh Korayem

[8] Ncfm: Accurate Handwritten Digits Recognition using Convolutional Neural Networks (2016)- Yan Yin, JunMin Wu, HuanXin Zheng
[9] Parallelization of Digit Recognition System Using Deep Convolutional Neural Network on CUDA (2017)- Srishti Singh; Amrit Paul; Dr. Arun M
[10] A Comparative Study on Handwriting Digit Recognition Using Neural Networks (2017)- Mahmoud M. Abu Ghosh; Ashraf Y. Maghari
[11] Handwritten Digits Recognition with Artificial Neural Network
(2017)- Kh Tohidul Islam, Ghulam Mujtaba, Dr. Ram Gopal Raj, Henry Friday Nweke
[12] Handwritten digits recognition using ensemble neural networks and ensemble decision tree (2017)- Retno Larasati, Dr. Hak KeungLam
[13] Handwritten Digit Recognition Based on Depth Neural Network (2017)- Yawei Hou; Huailin Zhao
[14] Learning and Real-time Classification of Hand-written Digits with Spiking Neural Networks (2017)- Shruti R. Kulkarni, John M. Alexiades, Bipin Rajendran
[15] Devanagari Digit Recognition by using Artificial Neural Network (2017)- Reena Dhakad; Dinesh Soni
[16] Handwritten Indian numerals recognition system using probabilistic neural networks (2004)- Faruq A. Al-Omari, Omar Al-Jarrah
[17] Classification of Handwritten Digits Using Evolving Fuzzy Neural
Network (2004)- G. S. Ng, T. Murali, A. Wahab, N. Sriskanthan
[18] Comparison of Different Neural Network Architectures for Digit Image Recognition (2011)- Hao Yu, Tiantian Xie, Michael Hamilton and Bogdan Wilamowski
[19] Digit Recognition Using Single Layer Neural Network with Principal Component Analysis(2014)- Vineet Singh, and Sunil Pranit Lal
[20] Handwritten digit recognition using multilayer feedforward neural networks with periodic and monotonic activation functions (2002)- Kwok-wo Wong Chi-sing Leung Sheng-jiang Chang