

**HANDWRITTEN DIGIT RECOGNITION  
USING DEEP LEARNING**

**A Report for the Evaluation 3 of Project 2**

*Submitted by*

**SHUBHAM SACHDEVA**

**(18SCSE2030048/  
18032030067)**

**RAJA**

**(18SCSE2030030/  
18032030051)**

*in partial fulfilment for the award of the degree of*

**Masters of Computer Application**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

**Under the Supervision of  
Mr. AJAY KUMAR, B. Tech., M. Tech.,  
Professor**

**APRIL / MAY- 2020**



## **SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING**

### **BONAFIDE CERTIFICATE**

Certified that this project report **“HAND WRITTEN DIGIT RECOGNITION  
USING DEEP LEARNING”** is the bonafide work of **“SHUBHAM SACHDEVA  
(18032030067) and RAJA (18032030051)”** who carried out the project work under  
my supervision.

#### **SIGNATURE OF HEAD**

Dr. MUNISH SHABARWAL,  
**PhD (Management), PhD (CS)**  
Professor & Dean,  
**School of Computing Science & Engineering**

#### **SIGNATURE OF SUPERVISOR**

Dr. AJAY KUMAR, **B.Tech., M.Tech.,**  
Professor,  
**School of Computing Science & Engineering**

## TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>ABSTRACT</b>	<b>iii</b>
	<b>LIST OF FIGURES</b>	<b>iv</b>
	<b>LIST OF SYMBOLS/ ABBREVIATION</b>	<b>v</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>1.1</b>	PROJECT DEFINITION	1
1.2	RELEVANT THEORY	
1.2.1	BENEFITS OF DIGIT RECOGNITION	1
1.2.2	WHAT IS NEURAL NETWORK	1
1.3	FEASIBILITY STUDIES	2
1.4	METHOD AND MATERIALS	
1.4.1	MULTILAYER PERCEPTION	2
1.4.2	SUPPORT VECTOR MACHINE	3
1.4.3	NAÏVE BIAS	3
<b>2.</b>	<b>LITERATURE SURVEY</b>	
2.1	K- NEAREST CLASSIFIER	4
2.2	FULLY CONNECTED MULTI LAYER NEURAL NETWORK	5
2.3	HANDWRITTEN DIGIT RECOGNITION USING MACHINE LEARNING	5
2.4	HANDWRITTEN DIGIT RECOGNITION USING CNN IN PYTHON AND KERAS	6
2.5	ACTIVATIONS	<b>6</b>
<b>3.</b>	<b>REQUIREMENT SPECIFICATION</b>	
3.1	FUNCTIONAL REQUIREMENTS	8
3.2	NON- FUNCTIONAL REQUIREMENTS	8
<b>4.</b>	<b>PROPOSED MODEL</b>	
4.1	MNIST DATASET FOR TRAINING AND TESTING	9
4.2	EXPERIMENTAL DESIGN OF CNN ARCHITECTURE	9
4.3	OPENCV	10
<b>5.</b>	<b>IMPLEMENTATION AND ARCHITECTURAL DIAGRAM</b>	
5.1	CONVOLUTIONAL NEURAL NETWORK	12
5.2	DEEP NEURAL NETWORK	12
5.3	IMPLEMENTATION STEPS	
5.3.1	DATA PREPROCESSING- RESHAPE STUFF	13
5.3.2	ENCODING	13
5.3.3	BUILD THE MODEL	13
5.3.4	COMPILING THE MODEL	14
5.3.5	TRAIN THE MODEL	14
5.3.6	PREDICTING AND TESTING THE CURRENT MODEL	14

5.4	FORWARD PROPOGATION	15
5.5	BACKWARD PROPOGATION	
5.5.1	COMPUTING THE LOSS	15
5.5.2	UPDATING THE WEIGHTS AND BIAS	15
5.6	ARCHITECTURAL DIAGRAM	16
<b>6.</b>	<b>DATA MODELS</b>	
6.1	USE- CASE	17
6.2		
<b>7.</b>	<b>OUTPUT/ SCREENSHOTS</b>	<b>16</b>
<b>8.</b>	<b>CONCLUSION</b>	<b>22</b>
	<b>APPENDIX</b>	<b>23</b>
	<b>REFERENCES</b>	<b>26</b>

## **ABSTRACT**

Handwritten digit recognition is gaining a huge demand in the branch of computer vision. We are going to implement a better and accurate approach to perceive and foresee manually written digits from 0 to 9. A class of multilayer sustain forward system called Convolutional network is taken into consideration. A Convolutional network has a benefit over other Artificial Neural networks in extracting and utilizing the features data, enhancing the knowledge of 2D shapes with higher degree of accuracy and unvarying to translation, scaling and other distortions. The LeNet engineering was initially presented by LeCun et al in their paper. The creators execution of LeNet was primarily focused on digit and character recognition. LeNet engineering is clear and simple making it easy for implementation of CNN's. We are going to take the MNIST dataset for training and recognition. The primary aim of this dataset is to classify the handwritten digits 0-9 . We have a total of 70,000 images for training and testing. Each digit is represented as a 28 by 28 grey scale pixel intensities for better results. The digits are passed into input layers of LeNet and then into the hidden layers which contain two sets of convolutional, activation and pooling layers. Then finally it is mapped onto the fully connected layer and given a softmax classifier to classify the digits. We are going to implement this network using keras deep learning inbuilt python library.

## LIST OF FIGURES

<b>2.1</b>	K- NEAREST NEIGHBOUR	5
<b>2.2</b>	FULLY CONNECTED NEURAL NETWORK	6
<b>2.3</b>	SIGMOID ACTIVATION FUNCTION	7
<b>2.4</b>	RELU ACTIVATION FUNCTION	7
<b>4.1</b>	DETECTED HANDWRITTEN DIGIT	11
<b>5.1</b>	CONVOLUTIONAL NEURAL NETWORK	12
<b>5.2</b>	DEEP NEURAL NETWORK	13
<b>5.3</b>	ARCHITECTURE of CNN	16
<b>6.1</b>	OUTPUT 1	17
<b>6.2</b>	OUTPUT 2	18
<b>6.3</b>	OUTPUT 3	19
<b>6.4</b>	OUTPUT 4	20
<b>6.5</b>	OUTPUT 5	21

## **LIST OF SYMBOLS/ ABBREVIATIONS**

<b>ANN</b>	Artificial Neural Network
<b>CNN</b>	Convolutional Neural Network
<b>KNN</b>	K- Nearest Neighbour
$\phi$	Activation Function
$\phi(z)=1/(1 + e^{-z})$	Sigmoid Activation Function
<b>RELU</b>	Rectifier Linear Unit
$y=\max(0, x)$	Activation Function
<b>SGD</b>	Stochastic Gradient Descent
<b>MNSIT</b>	Modified National Institute of Standards and Technology database
<b>CONV2D</b>	Convolutional 2- dimension

# CHAPTER 1

## INTRODUCTION

Handwriting recognition is the ability of a machine to receive and interpret handwritten input from multiple sources like paper documents, photographs, touch screen devices etc. Recognition of handwritten and machine characters is an emerging area of research and finds extensive applications in banks, offices and industries. The main aim of this project is to design expert system for , **“HANDWRITTEN DIGIT RECOGNITION using DEEP LEARNING”** that can effectively recognize a particular character of type format using the Artificial Neural Network approach.

Neural computing Is comparatively new field, and design components are therefore less well specified than those of other architectures. Neural computers implement data parallelism. Neural computer are operated in way which is completely different from the operation of normal computers.

### 1.1 Project Definition

This application is useful for recognizing all character (digits) given as in input image. Once input image of character is given to proposed system, then it will recognize input character which is given in image. Recognition and classification of characters are done by Neural Network. The main aim of this project is to effectively recognize a particular character of type format using the Artificial Neural Network approach.

### 1.2 Relevant Theory

#### 1.2.1 Benefits of Digit Recognition :

1. In forensic application Handwritten digit recognition will be an effective method for evidence collection.
2. It will also help to reduce noise from the original character.
3. Our method develop accuracy in recognizing character in divert font and size.
4. More set of sample invites more accuracy rate because of heavy training and testing session.

#### 1.2.2 What is Neural Network

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of large no. of highly interconnected processing element (neurons) working in union to solve specific problems. Learning in a Biological system involves adjustments to the synaptic connections that exist between the neuron.

Handwritten digit dataset are vague in nature because there may not always be sharp and perfectly



straight lines. The main goal in digit recognition is feature extraction is to remove the redundancy from the data and gain a more effective embodiment of the word image through a set of numerical attributes. It deals with extracting most of the essential information from image raw data [6]. In addition the curves are not necessarily smooth like the printed characters. Furthermore, characters dataset can be drawn in different sizes and the orientation which are always supposed to be written on a guideline in an upright or downright point. Accordingly, an efficient handwritten recognition system can be developed by considering these limitations. It is quite exhausting that sometimes to identify hand written characters as it can be seen that most of the human beings can't even recognize their own written scripts. Hence, there exists constraint for a writer to write apparently for recognition of handwritten documents.

Before revealing the method used in conducting this research, software engineering module is first presented. Pattern recognition along with Image processing plays compelling role in the area of handwritten character recognition. The study [7], describes numerous types of classification of feature extraction techniques like structural feature based methods, statistical feature based methods and global transformation techniques. Statistical approaches are established on planning of how data are selected. It utilizes the information of the statistical distribution of pixels in the image. The paper [8], provided SVM based offline handwritten digit recognition system. Authors claim that SVM outperforms in the experiment. Experiment is carried out on NIST SD19 standard dataset. The study [9] provide the conversion of handwritten data into electronic data, nature of handwritten characters and the neural network approach to form machine competent of recognizing hand written characters. The study [10] addresses a comprehensive criterion of handwritten digit recognition with various state of the art approaches, feature representations, and datasets. However, the relationship of training set size versus accuracy/error and the dataset-independence of the trained models are analyzed. The presents convolution neural networks into the handwritten digit recognition research and describes a system which can still be considered state of the art.

### **1.3 FEASIBILITY STUDY**

The aim of our feasibility study is to select the system that fits the most our performance requirements. We would like to determine if it is possible to develop our product in terms of resources and technicalities. Thus, we will analyze the problem and collect information for the product, including the data we will input to our system, how we will carry our process on that data, and the output we wish to obtain following our process, as well as the constraints applied on how the system behaves. On the technical level, our two main concerns will be finding the right data and finding and coding the right algorithms. As for the input data, we will use a famous handwritten digits dataset assembled by the National Institute of Standards and Technology and arranged by Yann Lecun, professor at NYU, available at <http://yann.lecun.com/exdb/mnist/>. It contains a training set of 60,000 examples and a testing set of 10,000 examples. They have been centered and size-normalized in fixed-size labeled images, each number in one image. Then, to input the data, the images will be scanned pixel by pixel by our program. Each pixel will have a value saying how dark its color is, and the value for each pixel will be put in an array. Then, the value of each element from the array will be fed to the input layer of our neural network. Each pixel value will be given to a unit from the input layer. We will then use different types of neural network algorithms and select the one that gives us the most precise predictions. Part of our work will consist of adapting the different codes for neural networks available to our program.

### **1.4 Methods and Materials**

#### ***1.4.1 Multilayer Perceptions***

A neural network based classifier, called Multi- Layer perception (MLP), is used to classify the digits. Multilayer perceptron consists of three different layers, input layer, hidden layer and output layer. Each of the layers can have certain number of nodes also called neurons and each node in a layer is connected to all other nodes to the next layer [12]. For this reason it is also known as feed forward network. The number of nodes in the input layer depends upon the number of attributes present in the dataset. The number of nodes in the output layer relies on the number of apparent classes exist in the dataset. The convenient number of hidden layers or the convenient number of nodes in a hidden layer for a specific problem is hard to determine. But in general, these numbers are selected experimentally. In multilayer

perceptron, the connection between two nodes consists of a weight. During training process, it basically learns the accurate weight adjustment which corresponds to each connection. For the learning purpose, it uses a supervised learning technique named as Back propagation algorithm.

#### **1.4.2 Support Vector Machine**

SVM or Support Vector Machine is a specific type of supervised ML method that intends to classify the data points by maximizing the margin among classes in a high-dimensional space [14]. SVM is a representation of examples as points in space, mapped due to the examples of the separate classes are divided by a fair gap that is as extensive as possible. After that new examples are mapped into that same space and anticipated to reside to a category based on which side of the gap they fall on [15]. The optimum algorithm is developed through a “training” phase in which training data are adopted to develop an algorithm capable to discriminate between groups earlier defined by the operator (e.g. patients vs. controls), and the “testing” phase in which the algorithm is adopted to blind-predict the group to which a new perception belongs [16]. It also provides a very accurate classification performance over the training records and produces enough search space for the accurate classification of future data parameters. Hence it always ensures a series of parameter combinations no less than on a sensible subset of the data. In SVM it's better to scale the data always; because it will extremely improve the results. Therefore be cautious with big dataset, as it may leads to the increase in the training time.

#### **1.4.3 Naive Bayes**

The Naive Bayes classifier [19] contributes a simple method, representing and learning probabilistic knowledge with clear semantics. It is termed naive due to it relies on two important simplifying assumes that predictive attributes are conditionally self-reliant given the class, and it considers that no hidden attributes influence the prediction method. It is a probabilistic classifier which relies upon Bayes theorem with robust and naive independence assumptions. It is one of the best basic text classification approaches with numerous applications in personal email sorting, email spam detection, sexually explicit content detection, document categorization, sentiment detection, language detection. Although the naïve design and oversimplified assumptions that this approach uses, Naive Bayes accomplishes well in many complicated real-world problems. All though it is often out performed by other approaches such as boosted trees, Max Entropy, Support Vector Machines, random forests etc, Naïve Bayes classifier is very potent as it is less computationally intensive (in both memory and CPU) and it needs a small extent of training data. Moreover, the training time with Naive Bayes is considerably smaller as opposed to alternative approaches.

## CHAPTER 2

# LITERATURE SURVEY

CNN is playing an important role in many sectors like image processing. It has a powerful impact on many fields. Even, in nano-technologies like manufacturing semiconductors, CNN is used for fault detection and classification. Handwritten digit recognition has become an issue of interest among researchers. There are a large number of papers and articles are being published these days about this topic. In research, it is shown that Deep Learning algorithm like multilayer CNN using Keras with Theano and Tensorflow gives the highest accuracy in comparison with the most widely used machine learning algorithms like SVM, KNN & RFC. Because of its highest accuracy, Convolutional Neural Network (CNN) is being used on a large scale in image classification, video analysis, etc. Many researchers are trying to make sentiment recognition in a sentence. CNN is being used in natural language processing and sentiment recognition by varying different parameters. It is pretty challenging to get a good performance as more parameters are needed for the large-scale neural network. Many researchers are trying to increase the accuracy with less error in CNN. In another research, they have shown that deep nets perform better when they are trained by simple back-propagation. Their architecture results in the lowest error rate on MNIST compare to NORB and CIFAR10. Researchers are working on this issue to reduce the error rate as much as possible in handwriting recognition. In one research, an error rate of 1.19% is achieved using 3-NN trained and tested on MNIST. Deep CNN can be adjustable with the input image noise. Coherence recurrent convolutional network (CRCN) is a multimodal neural architecture. It is being used in recovering sentences in an image. Some researchers are trying to come up with new techniques to avoid drawbacks of traditional convolutional layer's. Ncfm (No combination of feature maps) is a technique which can be applied for better performance using MNIST datasets. Its accuracy is 99.81% and it can be applied for largescale data. New applications of CNN are developing day by day with many kinds of research. Researchers are trying hard to minimize error rates. Using MNIST datasets and CIFAR, error rates are being observed. To clean blur images CNN is being used. For this purpose, a new model was proposed using MNIST dataset. This approach reaches an accuracy of 98% and loss range 0.1% to 8.5%. In Germany, a traffic sign recognition model of CNN is suggested. It proposed a faster performance with 99.65% accuracy. Loss function was designed, which is applicable for light-weighted 1D and 2D CNN. In this case, the accuracies were 93% and 91% respectively.

### **2.1 K-nearest neighbour classifier:**

A KNN classifier with a distance measure like Euclidean distance between the data sets input images is also capable of classification of digits but at higher error rate than a fully connected ML neural network. The key features of this classifier is that it requires no training time and no input from the programmer in terms of knowledge for designing the system. The big over head of this classifier is memory requirement and the classification or recognition time. We take into consideration that this nearest-neighbor system works on raw pixels instead of feature vectors.

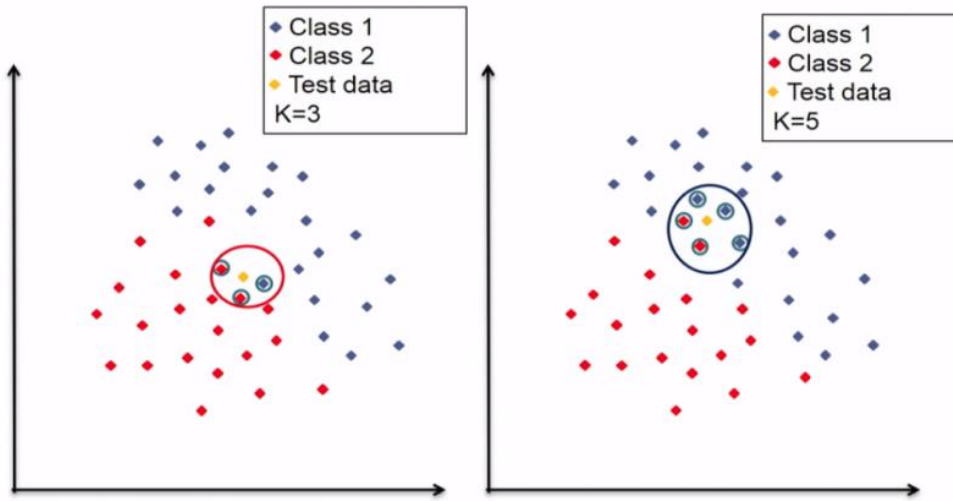


Fig 2.1 K- Nearest Neighbour

KNN is the non-parametric method or classifier used for classification as well as regression problems. This is the lazy or late learning classification algorithm where all of the computations are derived until the last stage of classification, as well as this, is the instance-based learning algorithms where the approximation takes place locally. Being simplest and easiest to implement there is no explicit training phase earlier and the algorithm does not perform any generalization of training data.

## 2.2 Fully Connected Multi-Layer Neural Network:

A Multi-Layer Neural Network with one or more number of hidden units is capable of classifying the digits in MNIST data set with a less than 2 % error rate on test set. This network extracts features based on the entire spatial domain of images hence the number of parameters required is very high. The problem with these networks is they tend to be over parameterized, in order of 100,000's which is unwanted when working with complex classification problems with complex data sets.

One way of thinking about fully connected networks is that each fully connected layer effects a transformation of the feature space in which the problem resides. The idea of transforming the representation of a problem to render it more malleable is a very old one in engineering and physics. It follows that deep learning methods are sometimes called "representation learning." (An interesting factoid is that one of the major conferences for deep learning is called the "International Conference on Learning Representations.")

## 2.3 Handwritten Digit Recognition using Machine Learning.

As using machine learning algorithms are used like KNN, SVM, Neural networks along with different parameters and feature scaling vectors, we also saw the different comparison among the classifiers in terms of the most important feature of accuracy and timing. Accuracy can alter as it depends on the splitting of training and testing data, and this can further be improved if the number of training and testing data is provided. There is always a chance to improve accuracy if the size of data increases. Every classifier has its own accuracy and time consumption. We can also include the fact that if the power of CPU changes to GPU, the classifier can perform with better accuracy and less time and better results can be observed.

The performance of the classifier can be measured in terms of ability to identify a condition properly (sensitivity), the proportion of true results (accuracy), number of positive results from the procedure of classification as false positives (positive predictions) and ability to exclude condition correctly (specificity). In this, we saw a brief comparison to the classifiers of Machine learning and deep learning.

Till now, the algorithms of Deep learning have performed better in the application of Handwritten Digit Recognition.

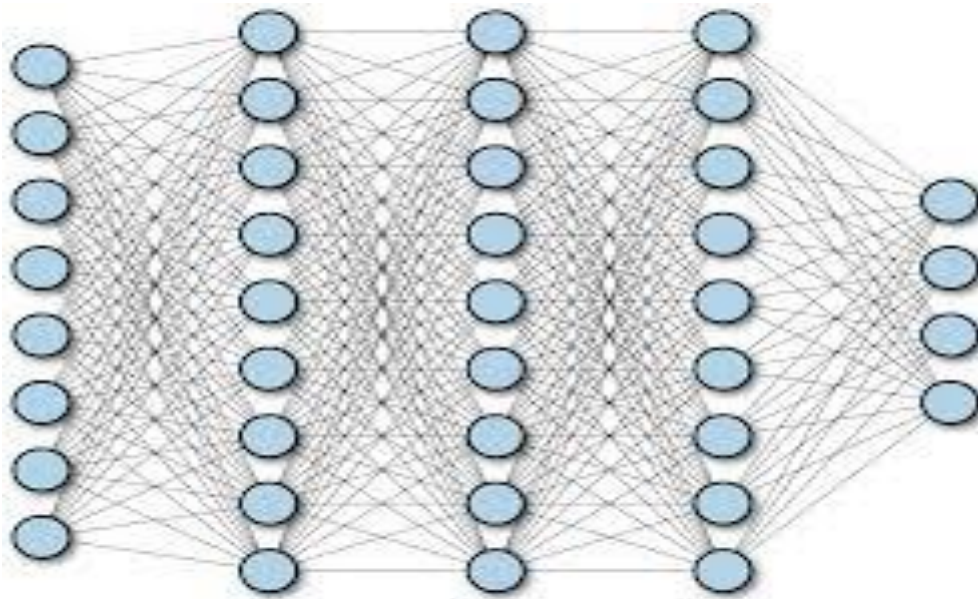


Fig 2.2 Fully connected Neural Network

## 2.4 Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras

The [MNIST problem](#) is a dataset developed by Yann LeCun, Corinna Cortes and Christopher Burges for evaluating machine learning models on the handwritten digit classification problem. 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 centered. 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. Each image is a 28 by 28 pixel square (784 pixels total). A standard split 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. Excellent results achieve a prediction error of less than 1%. State-of-the-art prediction error of approximately 0.2% can be achieved with large Convolutional Neural Networks.

## 2.5 ACTIVATIONS ( $\phi$ )

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated (“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction. Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1.

An additional aspect of activation functions is that they must be computationally efficient because they are calculated across thousands or even millions of neurons for each data sample. Modern neural networks use a technique called backpropagation to train the model, which places an increased computational strain on the activation function, and its derivative function.

### Sigmoid:

The sigmoid activation function, also called the logistic function, is traditionally a very popular activation function for neural networks. The input to the function is transformed into a value between 0.0 and 1.0

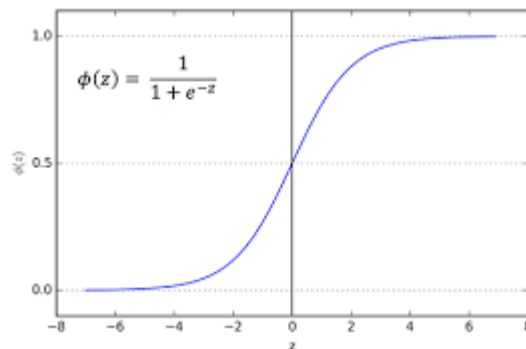


Fig 2.3 Sigmoid Activation Function

### Relu:

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as  $y = \max(0, x)$ . ... ReLU is the most commonly used activation function in neural networks, especially in CNNs.

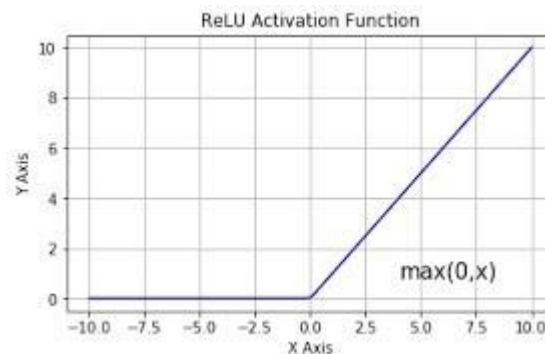


Fig 2.4 Relu Activation Function

The **sigmoidal** is the classical nonlinearity in fully connected networks, in recent years researchers have found that other activations, notably the rectified linear activation (commonly abbreviated **ReLU** or relu)  $\sigma(x) = \max(x, 0)$  work better than the sigmoidal unit. This empirical observation may be due to the *vanishing gradient* problem in deep networks. For the sigmoidal function, the slope is zero for almost all values of its input. As a result, for deeper networks, the gradient would tend to zero. For the ReLU function, the slope is nonzero for a much greater part of input space, allowing nonzero gradients to propagate.

## CHAPTER 3

### REQUIREMENTS SPECIFICATION

#### 3.1 Functional requirements –

Import the Modified National Institute of Science and Technology Dataset:

- o Import dataset file directly to the program from a command that will download the dataset from its website
- o Save the dataset file in the same directory as the program
  - Create the model
- o Take the value of the color is the pixels
- o Put the value of all the pixels in a one-dimensional array
- o Build a Neural Network with a number of nodes in the input layer equal to the number of pixels in the arrays
- o Activate the Neural Network
- o Test the precision of the model in a testing set
  - Recognize handwritten number input
- o Allow user to input a number using a touchscreen
- o Predict in real-time the value of the number written

#### 3.2 Non-functional requirements –

Windows Application

- Using Python
- Using Tensorflow

## CHAPTER 4

# PROPOSED MODEL

The purpose of this study is the development of system that takes handwritten English characters as input, process the input, extract the optimal features, train the neural network using either Resilient Back-propagation or Scaled conjugate gradient, recognize the class of input text, and finally generate the computerized form of input text.

Key rationale toward optical character recognition (OCR) from handwritten image includes features extraction technique supported by a classification algorithm for recognition of characters based on the features. Previously, several algorithms for feature classifications and extraction have been utilized for the purpose of character recognition. But, with the advent of CNN in deep learning, no separate algorithms are required for this purpose. However, in the area of computer vision, deep learning is one of the outstanding performers for both feature extraction and classification. However, DNN architecture consists of many nonlinear hidden layers with a enormous number of connections and parameters. Therefore, to train the network with very less

amount of samples is a very difficult task. In CNN, only few set of parameters are needed for training of the system. So,

CNN is the key solution capable to map correctly datasets for both input and output by varying the trainable parameters and number of hidden layers with high accuracy. Hence, in this work, CNN architecture with DeepLearning4j (DL4J) framework is considered as best for the character recognition from the handwritten digit images. For the experiments and verification of system's performance, the normalized standard MNIST dataset is utilized.

4.1 MNIST database used for training and testing

The subset of NIST database is MNIST dataset. Out of 70,000 images of handwritten digits, 60,000 images are used for training and 10,000 images are utilized for testing. Resolution of every image is  $28 \times 28$  with pixel values in the range of 0–255 (gray scale). 0 gray value (in black) is representing background of digit, while digit itself is appeared as 255 gray value (in white). The MNIST dataset comprised of labeled training and test. For training and test set files, the pixel values are arranged in row form. Therefore, training set (images) and test set file (images) consist of 60,000 rows and 784 columns and 10,000 rows and 784 columns, respectively. On contrary, in the training and test label files, the labels'

values are 0–9. Hence, 10,000 rows and ten columns for testing files followed by 60,000 rows and ten columns (0–9) for training label.

## 4.2 Experimental design of CNN architecture

The performance of a CNN for a particular application depends on the parameters used in the network. In general, CNN architecture comprised of two main units or parts:

(a) feature extractor and (b) feature classifier. In the feature extraction unit, every layer of network collects the output from the immediate previous layer (as input) and forward the current output to the immediate next layer as inputs; contrarily, classification unit generates the predicted outputs. So, the CNN architecture with convolutional layers is implemented for MNIST digit recognition. The overall architecture of suggested network is enlisted below. At first, convolutional layer having alter map with size 5 takes  $(28 \times 28)$  one image as input and provides output feature map with shape  $(24 \times 24)$ . After that, a pooling layer is employed. The pooling layers reduce the resolution of features. It is the technique of



moving window across the 2D window space, and the maximum value in the window is the output. This depends on the size of pooling layer taken by the user. The down-sampling operation is performed by using a pool size  $2 \times 2$  with stride by 2, and it reduces the output size from  $(24 \times 24)_{20}$  to  $(12 \times 12)_{20}$ . Afterward, ReLU activation was done. ReLU or rectified linear unit has been used as the activation function. There is a wide range of activation function available when training neural network models. The mainly used activations are sigmoid, tanh, ReLU and leaky ReLU. The ReLU nonlinearity is a popular activation function used in machine learning algorithm because: (a) With ReLU, it is easier to train larger neural networks; (b) it is a simple and efficient function, which helps to solve the problem of vanishing gradients in neural networks. It removes any negative values from the output and makes sure that input and output layer sizes are the same. It replaces all the negative entities in feature maps to zero, and (c) ReLU activation function is added in each layer so that network learns about nonlinear decision boundaries. Function for nonlinearity used has alter map with size 5. It takes  $(12 \times 12)_{20}$  image as input and provides output feature map with shape  $(8 \times 8)_{50}$ . Along with ReLU layer, a layer with max pooling function is used, which helps to make assumptions about features, thus reducing overfitting and also the training time. The outcome of the ReLU is directed to max pooling layer where it progressively reduces the spatial size of the feature map representations, thereby reducing the number of parameters and computations in the network. Max pooling performs the overfitting on the linearized convolved outcomes with the help of max filter and produces more abstract representation of the convolved outcomes. Down-sampling operation was performed using a pool size  $2 \times 2$  with stride by 2 to reduce the output size from  $(8 \times 8)_{50}$  to  $(4 \times 4)_{50}$ .

After that, a fully connected layer is used with 1024 output nodes. Finally, another fully connected layer with ten output nodes is used to get network results for ten digits (0–9).

### **Libraries required to install**

. The most important library is the NUMPY is a library that provides support for large, multi-dimensional arrays where we can store our input pixel matrix of size 28 by 28, using numpy we can express images as multi-dimensional arrays of pixel intensity values. The next library which is to be installed is the Python SCIPY library. It adds further help for scientific and technical computing of our functions. Then we come to OPENCV library and the main goal of this library is real-time image processing.

### **4.3 OpenCV:**

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.

Since OpenCV is an open source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also. So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it. And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCV in github, make necessary corrections and send a pull request to OpenCV. OpenCV developers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become an open source contributor. Similar is the case with other tutorials, documentation etc. As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who know about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV.

### **FEATURES:**

1. Reading an image
2. Extracting the RGB values of a pixel

3. Resizing the Image
4. Rotating the Image
5. Drawing a Rectangle
6. Displaying text

### **Find and Draw Contours using OpenCV**

Contours are defined as the line joining all the points along the boundary of an image that are having the same intensity. Contours come handy in shape analysis, finding the size of the object of interest, and object detection.

OpenCV has `findContour()` function that helps in extracting the contours from the image. It works best on binary images, so we should first apply thresholding techniques, Sobel edges, etc.

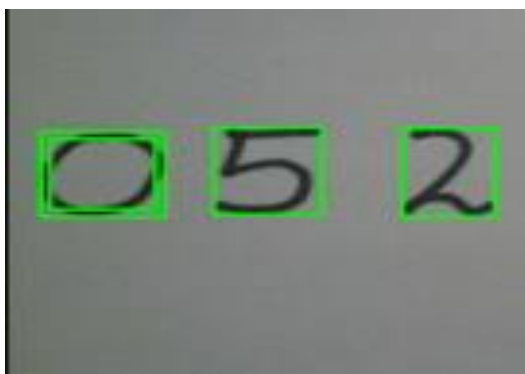


Fig 4.1 Detected Handwritten digits

## IMPLEMENTATION AND ARCHITECTURAL DIAGRAM

### 5.1 Convolutional Neural Network (CNN):

A simple CNN model can be seen in Fig. 1. The first layer is the input layer; the size of the input image is  $28 \times 28$ . The second layer is the convolution layer C2, it can obtain four different feature maps by convolution with the input image. The third layer is the pooling layer P3. It computes the local average or maximum of the input feature maps. The next convolution layer and pooling layer operate in the same way, except the number and size of convolution kernels. The output layer is full connection; the maximum value of output neurons is the result of the classifier in end. A simple structure of CNN.

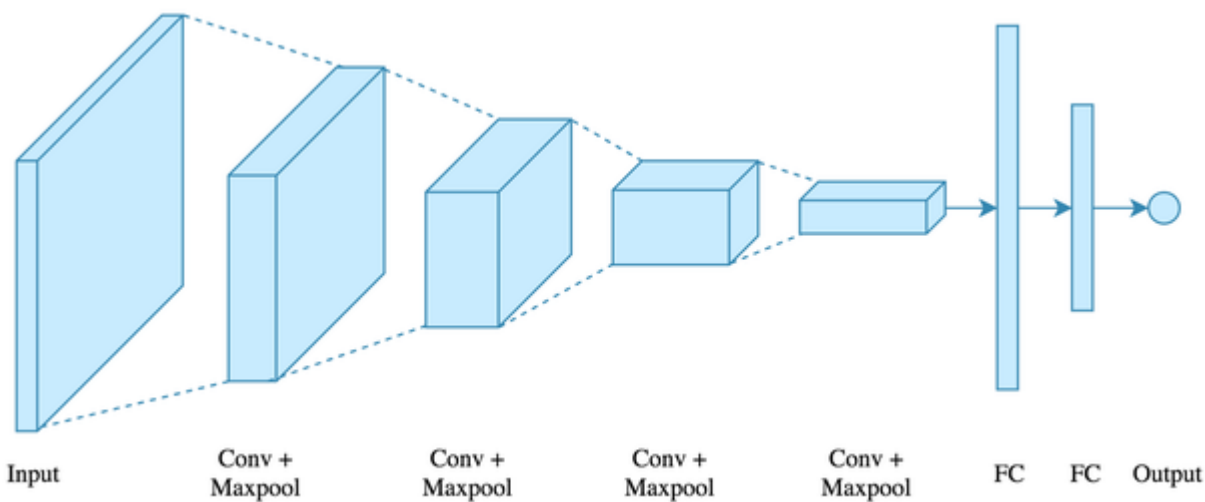


Fig 5.1 Convolutional Neural Network (CNN):

### 5.2 Deep Neural Network (DNN):

“The initially random weights of DNN are iteratively trained to minimize the classification error on a set of labeled training images; generalization performance is then tested on a separate set of test images. DNN has 2-dimensional layers of winner-take-all neurons with overlapping receptive fields whose weights are shared. Given some input pattern, a simple max pooling technique determines winning neurons by partitioning layers into quadratic regions of local inhibition, and selecting the most active neuron of each region. The winners of some layer represent a smaller, down-sampled layer with lower resolution, feeding the next layer in the hierarchy. The approach is inspired by Hubel and Wiesel’s seminal work on the cat’s primary visual cortex, which identified orientation selective simple cells with overlapping local receptive fields and complex cells performing downsampling-like operations is shown in.” The structure of DNN . D. Neural Network Toolbox in matlab (simulate): Neural Network Toolbox provides algorithms, functions, and apps to create, train, visualize, and simulate neural networks. The toolbox includes convolutional neural network and auto encoder deep learning algorithms for image classification and feature learning tasks by used MATLAB programming language.

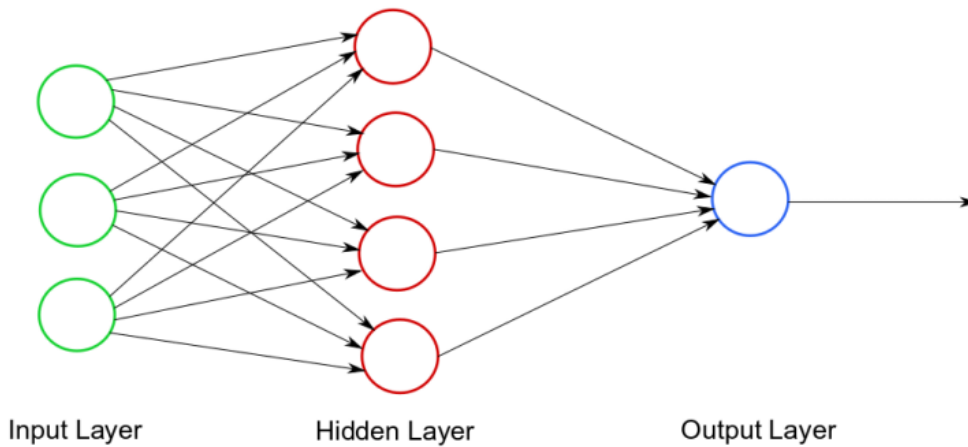


Fig 5.2 Deep Neural Network

## 5.3 Implementation steps:

### 5.3.1 Data Preprocessing — Reshaping Stuff:

This is the most important part of training a model.

We need to reshape our dataset inputs ( $X_{train}$  and  $X_{test}$ ) to the shape that our model expects when we train the model. The first number is the number of images ( $X_{train} \rightarrow 60000$ ,  $X_{test} \rightarrow 10000$ ). Then comes the shape of each image i.e. (28, 28). The last number 1 signifies that the image is grayscale.

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
print ("Shape of X_train: {}".format(X_train.shape))
print ("Shape of y_train: {}".format(y_train.shape))
print ("Shape of X_test: {}".format(X_test.shape))
print ("Shape of y_test: {}".format(y_test.shape))
```

### 5.3.2 Encoding:

The final layer of our CNN model will contain 10 nodes, each of them corresponding to the respective digit (first node  $\rightarrow$  0, second node  $\rightarrow$  1 and so on).

For example, if the image is of the number 6, then the label instead of being = 6, it will have a value 1 in column 7 and 0 in rest of the columns, like [0,0,0,0,0,0,1,0,0].

```
X_train = X_train.reshape(60000, 28, 28, 1)
X_test = X_test.reshape(10000, 28, 28, 1)
Shape of X_train: (60000, 28, 28, 1)
Shape of y_train: (60000,)
Shape of X_test: (10000, 28, 28, 1)
Shape of y_test: (10000,)
```

### 5.3.3 Build the model:

*add()* function is used for adding successive layers. The first 2 layers are **Conv2D** layers. These are convolution layers that will deal with our input images, which are seen as 2D matrices.

**Activation** is the activation function for the layer. The activation function here being used for the first 2

layers is the ReLU, or Rectified Linear Activation. This function outputs 0 if the input is a negative number and output the same input if the input is a positive number. In simple words, ReLU->max(0, input). This activation function is known for performing well in terms of speed and output in the neural nets.

### 5.3.4 Compiling the model

Compiling the model takes three parameters:

- **Optimizer** — It controls the learning rate. It is a very good optimizer as it utilises the Stochastic gradient optimizer.
- **Loss function** — We will be using ‘categorical\_crossentropy’ loss function. A lower score corresponds to better performance.
- **Metrics** — To make things easier to interpret, we will be using ‘accuracy’ metrix to see the accuracy score on the validation set while training the model.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

### 5.3.5 Training the model

Let's train the above the model with specified characteristics.

So, the model will train on (X\_train, y\_train) and it will get validated on (X\_test, y\_test).

*1 epoch -> One iteration/cycle of the dataset throughout the Neural Network*

```
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3)
```

#### Output:

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/3
60000/60000 [=====] - 23s 388us/step - loss: 0.2451 -
accuracy: 0.9568 - val_loss: 0.0692 - val_accuracy: 0.9793
Epoch 2/3
60000/60000 [=====] - 17s 290us/step - loss: 0.0568 -
accuracy: 0.9829 - val_loss: 0.0791 - val_accuracy: 0.9776
Epoch 3/3
60000/60000 [=====] - 17s 291us/step - loss: 0.0377 -
accuracy: 0.9884 - val_loss: 0.0871 - val_accuracy: 0.9768
<keras.callbacks.callbacks.History at 0x7f7093c0e978>
```

### 5.3.6 Predicting and testing on the current dataset

- **First output** — It prints the ‘softmaxed’ list output consisting of 10 probabilities of the digit fed as input.
- **Second output** — I converted that “softmaxed” list in form where I replaced all the elements with 0 except the highest probability, which I replaced with 1.
- **Third output** — It displays the test image and predicted digit corresponding to it.

```
example = X_train[364]
prediction = model.predict(example.reshape(1, 28, 28, 1))# First output
print ("Prediction (Softmax) from the neural network:\n\n {}".format(prediction))#
Second output
hard_maxed_prediction = np.zeros(prediction.shape)
hard_maxed_prediction[0][np.argmax(prediction)] = 1
print ("\n\nHard-
maxed form of the prediction: \n\n {}".format(hard_maxed_prediction))# Third output
print ("\n\n----- Prediction ----- \n\n")
plt.imshow(example.reshape(28, 28), cmap="gray")
```

```
plt.show()
print("\n\nFinal Output: {}".format(np.argmax(prediction)))
```

## **5.4 FORWARD PROPOGATION:**

Once the data is input, it will go from the input nodes to the output nodes through a process called Forward Propagation. The weights of the links are first initialized to random values. Then, starts a process called Logistic regression. We go layer by layer in that process. Then, inside each layer, we go neuron by neuron. The activation (or value) of a neuron is calculated through steps. We first multiply the weight of each link between the neuron and the neurons of the previous layer. Then, we sum those products and add a bias, which is a value that is proper to each neuron. The bias works as a threshold that measures whether the activation is significant enough. That step is called linear regression. However, the result we will get will not be between 0 and 1. Thus, we finally apply an activation function to the value obtained in order to obtain a value between 0 and 1. An example of activation function is the sigmoid function, which is the inverse of the sum of one and the exponential of minus the activation. The output value of that function will always be between 0 and 1. Another activation function is the Rectifier Linear Unit (ReLU) function. It equals the maximum between the input number and zero. Finally, another example is the Softmax function (i.e. Normalized Exponential function). It is a function that turns the numbers into probabilities so that their sum is equal to one.

## **5.5 BACKWARD PROPOGATION:**

### **5.5.1 Computing the Loss:**

Once we have done a forward propagation, we do the backward propagation. This means that, this time, we will go layer by layer backwards. That is to say we will go through the layers in the opposite order, starting by the output layer. Furthermore, we obtain certain values in the output layer corresponding to the guess of the neural network. That value can be close or far from the actual value. For example, if we input a 7 in the input layer, the right activation we should get for the neurons of the output layer is 1 for node 7 and zero for all the other nodes. The Loss Function measures how close our guesses were to the right answer. An appropriate function for logistic regression we can use is the Cross-Entropy function. It is the negative of the sum of the product of the right value by the logarithm of the guessed value, and the product of one minus the right value and the logarithm of one minus the guessed value.

### **5.5.2 Updating the weights and biases :**

Once we computed the loss function, we need to use it to update the weights and the biases. Thus, before going to the next neuron in the layer, we will update all the weights linking the neuron to the previous layer, and the bias of each neuron. The new weight will be the difference between the old weight and the product of alpha and the derivative of the loss function with respect to the weight. The new bias will be the difference between the old weight and the product of alpha and the derivative of the loss function with respect to the bias. Alpha is the learning rate. It is a constant we configure for the neural network.

## 5.6 ARCHITECTURAL DIAGRAM

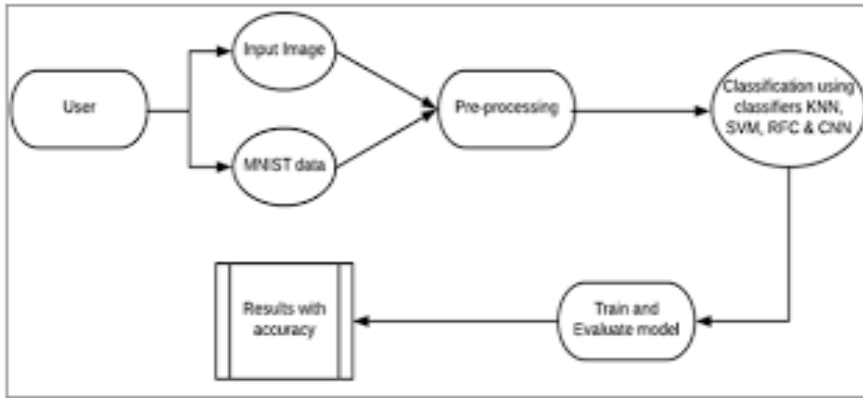
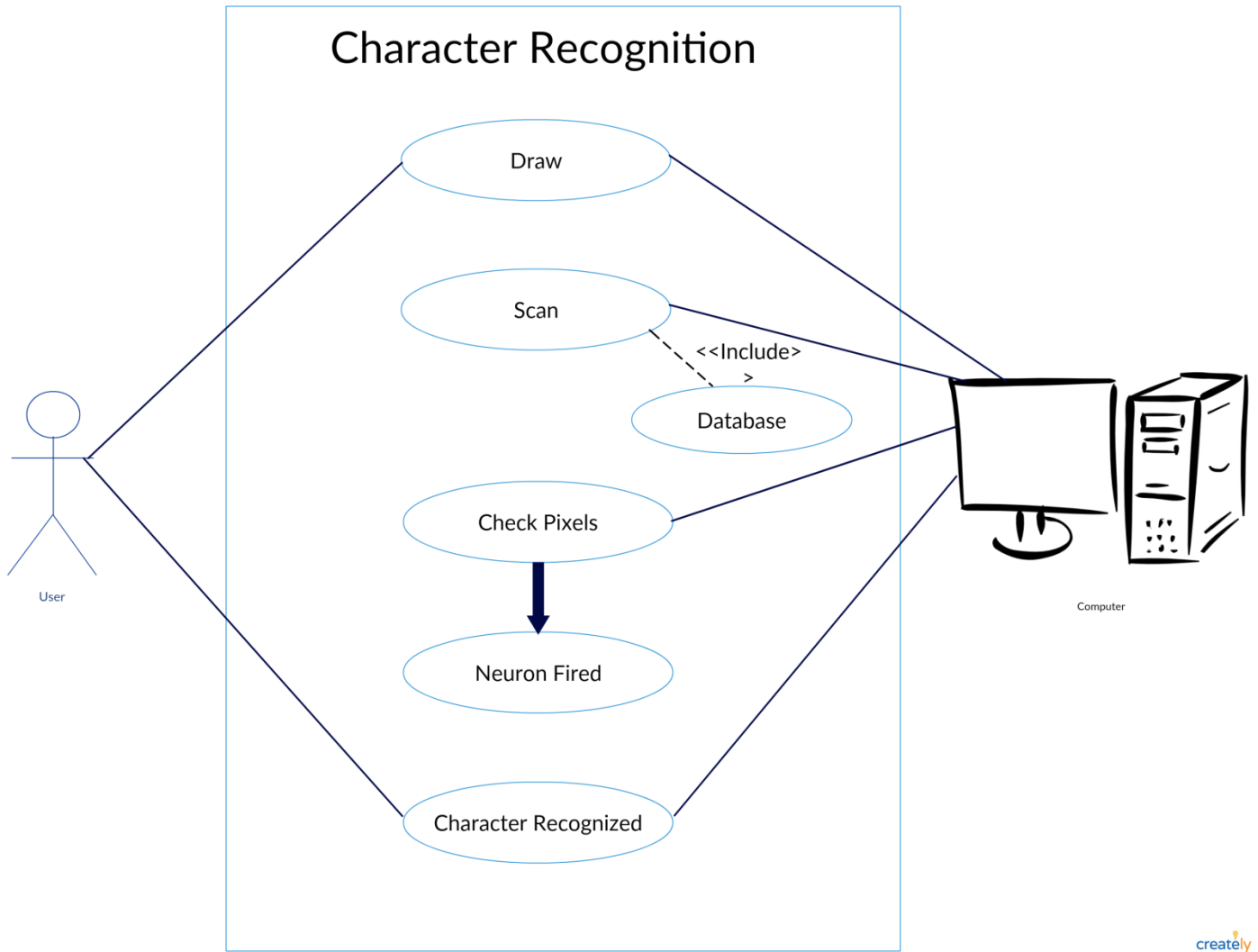


fig 5.3 Architecture of CNN

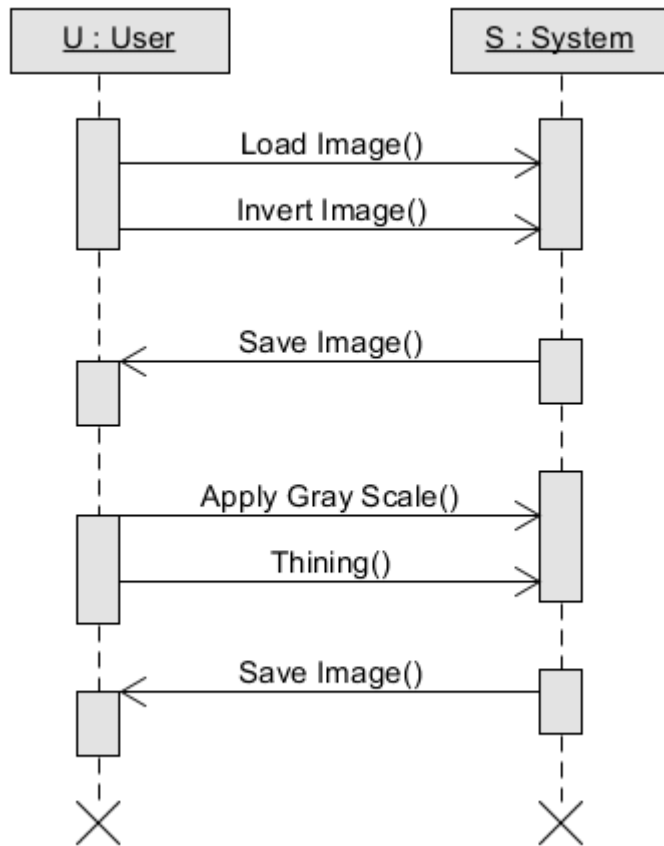
CHAPTER 6  
DATA MODELS

6.1 USE- CASE MODEL





## 6.2 SEQUENCE DIAGRAM



## 6.3 FLOW CHART

### 5.3 Flow Chart

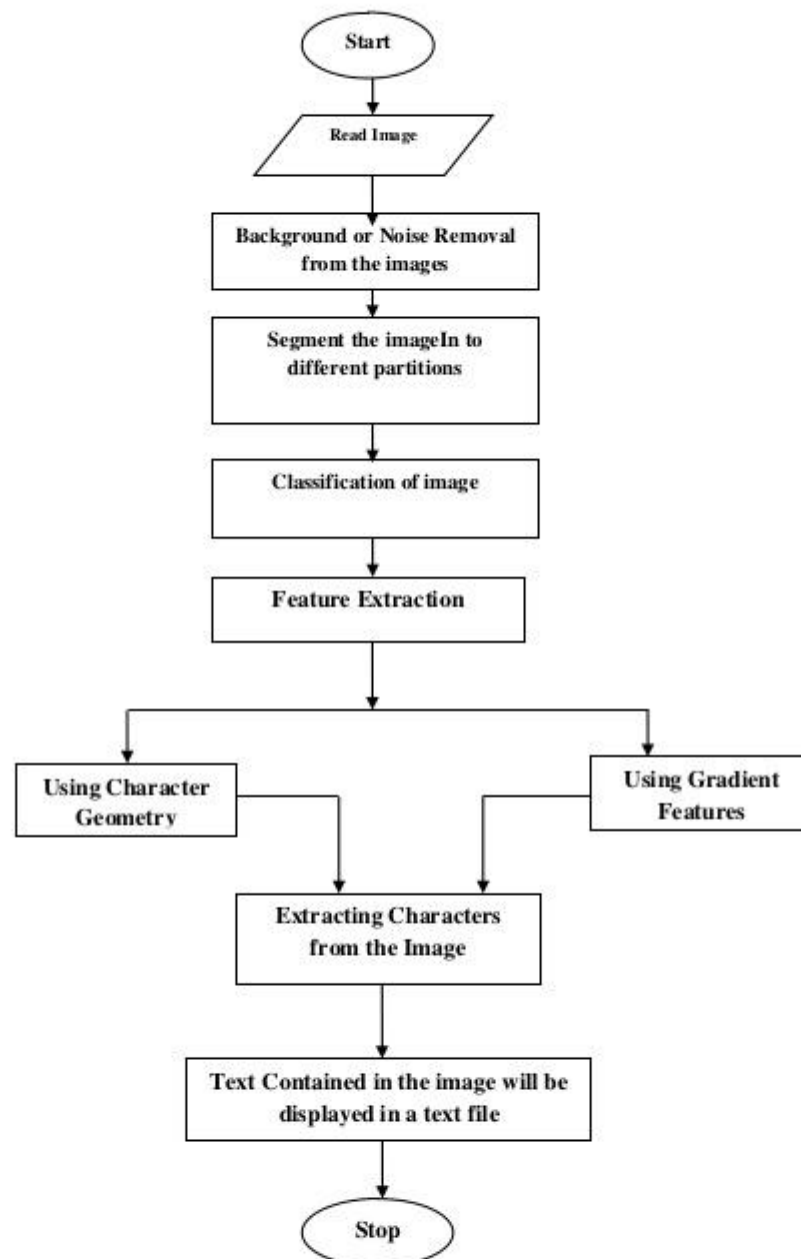


Fig: 5.3.1Flow chart

## CHAPTER 7

# OUTPUT/ SCREENSHOTS

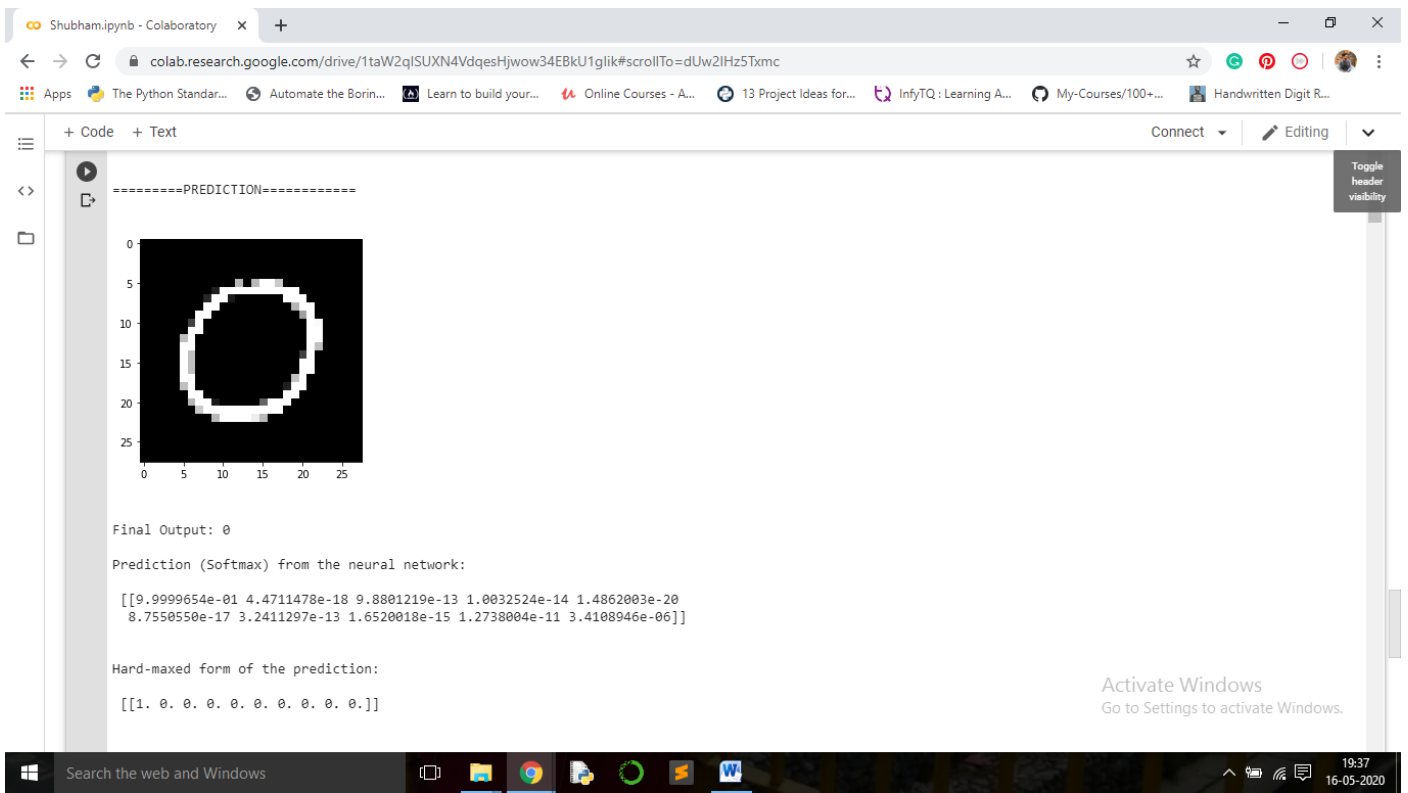
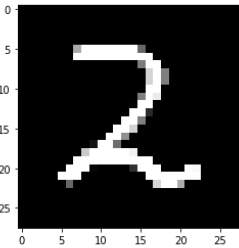


Fig 7.1 Output 1

=====PREDICTION=====



Final Output: 2

Prediction (Softmax) from the neural network:

```
[[3.8392318e-22 2.2702480e-15 1.0000000e+00 7.4774967e-15 5.4509693e-22  
8.2302498e-27 4.0824346e-28 2.4340786e-15 3.8691988e-13 3.8404222e-20]]
```

Hard-maxed form of the prediction:

```
[[0. 0. 1. 0. 0. 0. 0. 0. 0.]]
```

-----

Activate Windows  
Go to Settings to activate Windows.

Fig 7.2 Output 2

The screenshot shows a Google Colab notebook interface. At the top, the browser address bar displays the URL: `colab.research.google.com/drive/1taW2qISUXN4VdquesHjwov34EBkU1glik#scrollTo=dUw2IHZ5Txmc`. The notebook's toolbar includes options for '+ Code' and '+ Text', along with 'Connect' and 'Editing' menus. The main workspace contains a code cell with the following output:

```
-----PREDICTION-----  
0  
5  
10  
15  
20  
25  
0 5 10 15 20 25
```

Final Output: 5

Prediction (Softmax) from the neural network:

```
[[2.3271182e-31 7.2467877e-34 9.5686250e-34 2.6615110e-21 8.8444666e-27  
1.0000000e+00 1.9545209e-26 1.0351461e-29 2.6113719e-25 2.1368532e-23]]
```

Hard-maxed form of the prediction:

```
[[0. 0. 0. 0. 0. 1. 0. 0. 0.]]
```

-----

An 'Activate Windows' watermark is visible in the bottom right corner of the notebook area. The Windows taskbar at the bottom shows the search bar, taskbar icons, and system tray with the time 19:37 and date 16-05-2020.

Fig 7.3 Output 3

Shubham.ipynb - Colaboratory

colab.research.google.com/drive/1taW2qSUXN4VdqesHjwow34EBkU1glik#scrollTo=dUw2IHZ5Txmc

+ Code + Text

Connect Editing

Toggle header visibility

0  
5  
10  
15  
20  
25

0 5 10 15 20 25

Final Output: 8

Prediction (Softmax) from the neural network:

```
[[5.5801180e-19 2.5941081e-24 7.2330217e-06 2.2611815e-08 1.5796226e-19  
8.5842842e-15 4.4011745e-18 8.1459467e-20 9.9999273e-01 2.5408446e-13]]
```

Hard-maxed form of the prediction:

```
[[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]]
```

-----

Activate Windows  
Go to Settings to activate Windows.

Search the web and Windows

19:38  
16-05-2020

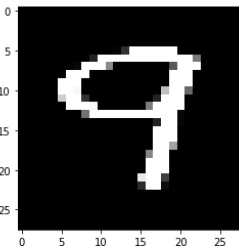
Fig 7.4 Output 4

Shubham.ipynb - Colaboratory

colab.research.google.com/drive/1taW2qISUXN4VdqesHjwow34EBkU1glik#scrollTo=dUw2IH5Txmc

+ Code + Text

-----PREDICTION-----



Final Output: 4

Prediction (Softmax) from the neural network:

```
[[[7.1364113e-08 1.5090452e-10 5.3099790e-03 3.1175206e-03 7.5487590e-01  
2.8482790e-04 6.3875234e-13 1.5568221e-01 6.3833930e-02 1.6895615e-02]]]
```

Hard-maxed form of the prediction:

```
[[[0. 0. 0. 0. 1. 0. 0. 0. 0.]]]
```

Activate Windows  
Go to Settings to activate Windows.

Search the web and Windows

19:39  
16-05-2020

Fig 7.5 Output 5

## CHAPTER 8

### CONCLUSION

In this study, interdisciplinary research has been carried out for Handwritten Digit Prediction through interaction of 60,000 sample dataset. A robust multilayer prediction model is generated in combination with the computation of maximum obtainable seismic features. 36 seismic features are considered for this problem. The features are employed for training of an earthquake prediction model for better result. The prediction model consists of Random forests classifier (RFC) method. RFC provides an initial estimation for Handwritten Digit prediction. Thus RFC based prediction model is trained and tested successfully with encouraging and improved result by selecting important features and training model. Performance of a network depends on many factors like low memory requirements, low run time and better accuracy, although in this paper it is primarily focused on getting better accuracy rate for classification. Before Artificial neurons had better accuracy but now the branch of computer vision mainly depends on deep learning features like convolutional neural networks. The branch of computer vision in artificial intelligence primary motive is to develop a network which is better to every performance measure and provide results for all kinds of datasets which can be trained and trained and recognized.



## APPENDIX

### Source Code:

#### 1.4.3.1 Importing Libraries:

```
import cv2

import numpy as np

from keras.datasets import mnist

from keras.layers import Dense, Flatten

from keras.layers.convolutional import Conv2D

from keras.models import Sequential

from keras.utils import to_categorical

import matplotlib.pyplot as plt
```

#### 1.4.3.2 Downloading the MNIST data:

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

#### 1.4.3.3 Checking the gray scale picture from MNIST dataset:

```
plt.imshow(X_train[0], cmap="gray")

plt.show()

print (y_train[0])
```

#### 1.4.3.4 Test and Train the data:

```
print ("Shape of X_train: {}".format(X_train.shape))

print ("Shape of y_train: {}".format(y_train.shape))

print ("Shape of X_test: {}".format(X_test.shape))

print ("Shape of y_test: {}".format(y_test.shape))
```

#### 1.4.3.5 Reshape the data:

```
X_train = X_train.reshape(60000, 28, 28, 1)

X_test = X_test.reshape(10000, 28, 28, 1)
```

#### 1.4.3.6 Updating the Test and Train values:

```
y_train = to_categorical(y_train)

y_test = to_categorical(y_test)
```

#### 1.4.3.7 Adding the Hidden layers to the dataset:

```
model = Sequential()
```

```

layer_1 = Conv2D(32, kernel_size=3, activation="relu" , input_shape=(28, 28,
1))
layer_2 = Conv2D(64, kernel_size=3, activation="relu")
layer_3 = Flatten()
layer_4 = Dense(10, activation="softmax")

## Add the layers to the model
model.add(layer_1)
model.add(layer_2)
model.add(layer_3)
model.add(layer_4)

```

#### 1.4.3.8 **Compiling the model:**

```

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[
'accuracy'])

```

#### 1.4.3.9 **Fit the model or Validate the model:**

```

model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3)

```

#### 1.4.3.10 **Predict the Training data:**

```

example = X_train[364]

prediction = model.predict(example.reshape(1, 28, 28, 1))## First output
print ("Prediction (Softmax) from the neural network:\n\n {}".format(predi
ction))## Second output
hard_maxed_prediction = np.zeros(prediction.shape)
hard_maxed_prediction[0][np.argmax(prediction)] = 1

print ("\n\nHard-
maxed form of the prediction: \n\n {}".format(hard_maxed_prediction))## Th
ird output

print ("\n\n----- Prediction ----- \n\n")

plt.imshow(example.reshape(28, 28), cmap="gray")
plt.show()

print("\n\nFinal Output: {}".format(np.argmax(prediction)))

```

#### 1.4.3.11 **Test the Real time image:**

```

image = cv2.imread('/content/test_image.jpg')
grey = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2GRAY)

ret, thresh = cv2.threshold(grey.copy(), 75, 255, cv2.THRESH_BINARY_INV)
contours, _ = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIM
PLE)

```

```

preprocessed_digits = []

for c in contours:
    x,y,w,h = cv2.boundingRect(c)

    # Creating a rectangle around the digit in the original image (for displaying the digits fetched via contours)
    cv2.rectangle(image, (x,y), (x+w, y+h), color=(0, 255, 0), thickness=2)

    # Cropping out the digit from the image corresponding to the current contours in the for loop
    digit = thresh[y:y+h, x:x+w]

    # Resizing that digit to (18, 18)
    resized_digit = cv2.resize(digit, (18,18))

    # Padding the digit with 5 pixels of black color (zeros) in each side to finally produce the image of (28, 28)
    padded_digit = np.pad(resized_digit, ((5,5), (5,5)), "constant", constant_values=0)

    # Adding the preprocessed digit to the list of preprocessed digits
    preprocessed_digits.append(padded_digit)

print("\n\n\n-----Contoured Image-----")
plt.imshow(image, cmap="gray")
plt.show()

inp = np.array(preprocessed_digits)

```

## REFERENCES

- [1] Isha Vats, Shamandeep Singh, “*Offline Handwritten English Numerals Recognition using Correlation Method*”, International Journal of Engineering Research and Technology (IJERT): ISSN: 2278-0181 Vol.
- [2] *Recognition of Handwritten Hindi Characters using Back propagation Neural Network*”, International Journal of Computer Science and Information Technologies ISSN 0975-9646, Vol. 3 (4)
- [3] S S Sayyad, Abhay Jadhav, Manoj Jadhav, Smita Miraje, Pradip Bele, Avinash Pandhare, ‘*Devnagiri Character Recognition Using Neural Networks*’ , International Journal of Engineering and Innovative Technology (IJEIT) Volume 3
- [4] Shabana Mehfuz, Gauri katiyar, ‘*Intelligent Systems for Off-Line Handwritten Character Recognition: A Review*’ , International Journal of Computer Science and Information Technologies Volume 2.
- [5] Prof. Swapna Borde, Ms. Ekta Shah, Ms. Priti Rawat, Ms. Vinaya Patil, “*Fuzzy Based Handwritten Character Recognition System*” , International Journal of Engineering Research and Technology (IJERT) ISSN: 2248-9622, VNCET
- [6] [Online]. Available: <https://www.udemy.com/course/deeplearning/>
- [7] [Online]. Available: <https://www.numpy.org/devdocs/about.html>.