A

Project Report

on

Handwritten Digit Recognition System Using Machine Learning

submitted as partial fulfillment for the award of (font size 14)

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2022-23

in

COMPUTER SCIENCE AND ENGINEERING

By

Ujjwal Jindal (1900290100177)

Ujjwal Gupta (1900290100176)

Vikas Panwar (1900290100192)

Under the supervision of

Asst. Prof. Gaurav Parashar

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2023

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Ujjwal

Name: Ujjwal Jindal
Roll No.: 1900290100177
Date: 26-05-2023

[illegible]

Name: Vikas Panwar
Roll No.: 1900290100192
Date: 26-05-2023

Ujjwal Gupta

Name: Ujjwal Gupta
Roll No.: 1900290100176
Date: 26-05-2023

CERTIFICATE

This is to certify that Project Report entitled “Handwritten Digit Recognition System Using Machine Learning” which is submitted by Ujjwal Jindal (1900290100177) , Vikas Panwar (1900290100192) , Ujjwal Gupta (1900290100176) in partial fulfillment of the requirement for the award of degree B. Tech. in Department of Computer Science & Engineering of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

.

[illegible]

Date: 26-05-2023

Supervisor Name

Asst. Prof. Gaurav Parashar

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to our supervisor Prof. Gaurav Parashar, Department of Computer Science & Engineering, KIET, Ghaziabad, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. Vineet Sharma, Head of the Department of Computer Science & Engineering, KIET, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members, especially, Prof. Vipin Deval of the department, for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Ujjwal

Name: Ujjwal Jindal
Roll No.: 1900290100177
Date: 26-05-2023

Vikas Panwar

Name: Vikas Panwar
Roll No.: 1900290100192
Date: 26-05-2023

Ujjwal Gupta

Name: Ujjwal Gupta
Roll No.: 1900290100176
Date: 26-05-2023

ABSTRACT

One of the most significant and practical challenges in pattern recognition applications is handwritten digit recognition. There are several applications of handwritten digit recognition such as cheque processing in banks, sorting mails, data entry in forms, etc. The goal to be achieved through this project is the creation of an efficient handwritten digit recognition system using machine learning algorithms that will recognize user-made digits on a canvas provided within a Graphical User Interface. This report presents an approach to recognize user-defined handwritten digits and convert them to other number systems by using machine/deep learning concepts and various other programming techniques.

KEYWORDS: *handwritten digit recognition, pattern recognition system, machine learning algorithm, digit recognition system, classification, convolutional neural network.*

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1. CNN - Convolutional Neural Network
2. SVM - Support Vector Machine
3. HDR - Handwritten Digit Recognition
4. ML - Machine Learning
5. MLP - Multilayer Perceptrons
6. NumPy - Numerical Python

CHAPTER 1. INTRODUCTION

Human beings can process images by allowing their brains to segment these images and recognize the different elements present within it. The brain undergoes this process by itself, which includes analyzing these images and comparing their many qualities to what it already knows to detect these elements. The discipline of computer science known as image processing attempts to do the same thing for machines.

Image processing takes images as input and converts them into a readable digital format as output. It accomplishes so by applying particular algorithms to these photos, resulting in higher-quality pictures or using some of their properties to extract useful information.

1.1 WHAT IS MACHINE LEARNING?

A computer science concept which basically enables a machine to imitate the way in which a human brain learns is known as machine learning. It does so by using vast datasets and advanced programming algorithms.

Machine learning is an AI subset that's also presently one of the most profoundly used technologies.

A machine learning model gradually improves its accuracy through various iterations that it undergoes during the learning process.

Machine learning is a crucial part of the exponentially evolving field of data analysis. Machine learning algorithms are used to classify data into different class labels and also make predictions.

These algorithms are trained by using various statistical methods. As a result, data mining systems employ machine learning algorithms to identify key characteristics and insights. These details further steers decision making within business applications mainly enhancing metrics related to system growth.

Here is the basic difference between Machine Learning and Traditional Programming which existed way before the machine learning and these are as follows:

- Traditional programming is when data (input) and logic are fed into a machine, and output is obtained.

- Machine Learning is a sort of current programming in which we take in data (input) and output (output), run it on a computer during training, and the machine builds its own programme (logic) that can be tested.

Now we have this idea of what exactly machine learning is and what is the difference between traditional and modern programming. Now let's discuss the fact that what exactly learning means to the computer and this idea can be put forward by explaining what the three part process includes in the learning part of Machine Learning.

So, If a computer's performance (P) in a given task increases with experience, it is said to learn from experiences (E) in relation to certain classes of tasks (T).

For example, if we take "playing chess”

Then,

E = the knowledge gained from several chess games

T = the task of playing a chess game.

P = the likelihood of the programme winning the following game

And in broad classification, machine learning problem can be classified into two parts:

1.Supervised Learning

2.Unsupervised Learning

Data in Machine Learning

Data:

Any unprocessed fact, value, text, sound, or image that has not been understood and analyzed qualifies. All data analytics, machine learning, and artificial intelligence rely heavily on data.

We cannot train any model without data, and all current research and automation will be useless. Large corporations are investing a significant amount of money in collecting as much reliable data as possible.

Example: A company called Ultrahuman collects a whole lot of data from the users who are using their device called “CGM Cyborg M1 Device”. And this device collects data from the device that is implanted on the skin of the user, and this collected data is then further used in many processed settings in order to provide the insight to all their users.

Facebook, for example, paid $19 million to acquire WhatsApp. So, why are these companies acquiring other companies for data ?

The explanation is straightforward and logical: it is to gain access to information about users that Facebook may not have, but WhatsApp will. Facebook values this information from its users since it will help them improve their services.

Information:

Data has been evaluated and altered to provide consumers with some useful predictions.

Knowledge:

Inferred facts, experiences, learning, and insights are combined. An individual or organization gains awareness or develops a notion.

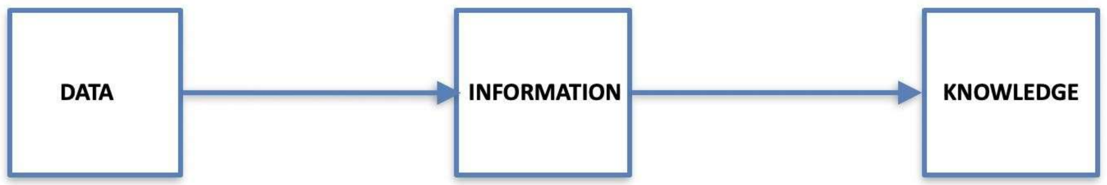


Fig 1.1 - Flow of Data Forms

1.2 WHAT IS HANDWRITTEN DIGIT RECOGNITION?

Handwritten digit recognition is the process of recognising and classifying user-defined handwritten digits into pre-defined digit class labels, such as (0-9).

Handwritten digit recognition has a wide range of applications, including check processing in [9] banks, mail sorting, data entry in forms, and so on.

1.3 NUMBER SYSTEMS

A number system represents numbers. It defines a collection of values to represent a value and is also known as a numerical system. The most commonly used digits are 0 and 1, which represent binary numbers.

1.3.1 Number System Types

Different sorts of number systems exist. Among the most crucial are:

- System of binary numbers i.e., base ‘2’.
- System of octal numbers i.e., base ‘8’.

- System of decimal numbers i.e., base ‘10’.
- System of hexadecimal numbers i.e., base ‘16’.

1.4 VARIOUS ALGORITHMS

A. Multilayer Perceptrons

Multilayer Perceptrons (MLP) is a machine learning approach for classifying handwritten digits [4][5]. It's a classifier based on neural networks. The input, hidden, and output layers are the three layers of multilayer perceptrons. Any of the levels can have a certain number of nodes or neurons, and all of the nodes in one layer are connected to the nodes in the next layer. The number of nodes is determined by trial and error. Two nodes are connected in multilayer perceptrons via a weight. The machine learning model learns the precise weight modification corresponding to each link during the training process. It employs a backpropagation algorithm during the learning phase, a supervised learning technique. The neural network of multilayer perceptrons, perhaps the most practical artificial neural network, is commonly referred to as the neural network or multilayer perceptron. The perceptron is a type of neuron that predates big neural networks. It's a field that looks into how simple biological brain models may be used to accomplish challenging computer tasks like predictive modeling in machine learning. The goal is to construct robust algorithms and data structures that may be used to model complex issues rather than to produce real brain models.

The capacity of neural networks to learn representations in your training data and how to best relate them to the output variables you wish to predict is what gives them their power. In this way, neural networks learn how to map. They can learn any mapping function mathematically and have shown to be a universal estimating algorithm.

The hierarchical or multi-layered structure of neural networks gives them their approximate capacity. Data structures can mix features from various scales or resolutions (learn representation) to create higher-order features, for instance, for the transformation of lines into forms.

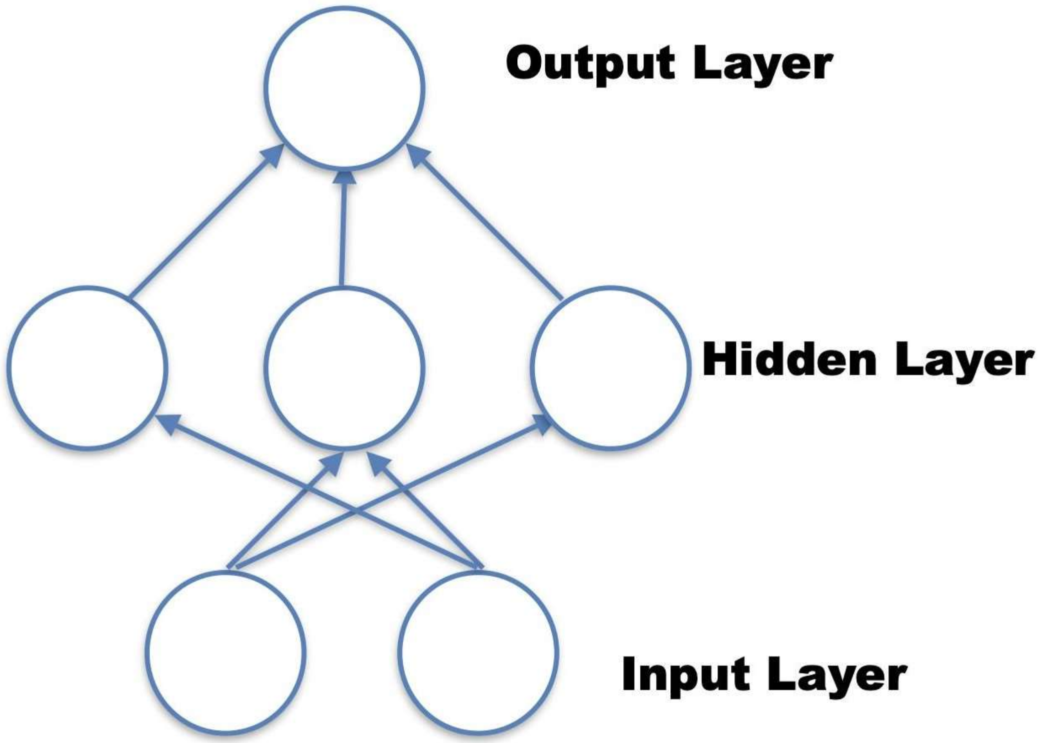


Fig 1.2 - Model of Simple Network

Each of the three layers in the aforementioned Simple Neural Network, namely the input layer, hidden layer, and output layer, is addressed below:

Input or Visible Layers

Because it is the exposed section of the network, the bottom layer that takes input from the data set is called the visible layer. A neural network is frequently depicted as a visible layer with one neuron for each column or input value of a data collection. It's simply forwarding an input value to the next layer, as stated above.

Hidden Layers

Because it is not directly affected by the input, the layer after the input layer is called a hidden layer. In the buried layer, the simplest network structure now includes one neuron that outputs a value. You can develop incredibly deep neural networks because of the increased computer power and efficient library. In neural networks, deep learning can have a lot of hidden layers. They are significant because they would have taken an inordinate amount of time to train if not for modern methodologies and gear.

Output Layer

Being the final layer of the neural network, the output layer is responsible for the resulting vector of values or a vector that corresponds to the format necessary for the problem outlined in the model.

Following the input and hidden layers, the output layer computes and displays the output based on the input.

B. Support Vector Machine(SVM)

The Support Vector Machine [10] is a supervised machine learning algorithm that seeks to classify data points by optimizing class margins in a high-dimensional space. SVM depicts data points in a high-dimensional space, mapped using examples of different classes that are separated by a fair gap. New models are then drawn in the same space, with the expectation that they will stay in the category regardless of which side of the gap they fall into. [1] SVM classifiers have the advantage of being extremely accurate; however, training the model takes a long time. To reduce this time SVM can be replaced by its proximal SVM model i.e., PSVM.

The main purpose of the SVM algorithm is to generate the best line or decision boundary that can divide n-dimensional space into classes so that we can simply place fresh data points in the correct category, and this best decision boundary is referred to as a hyperplane.SVM classifiers can be extremely accurate; however, training the model takes a long time.

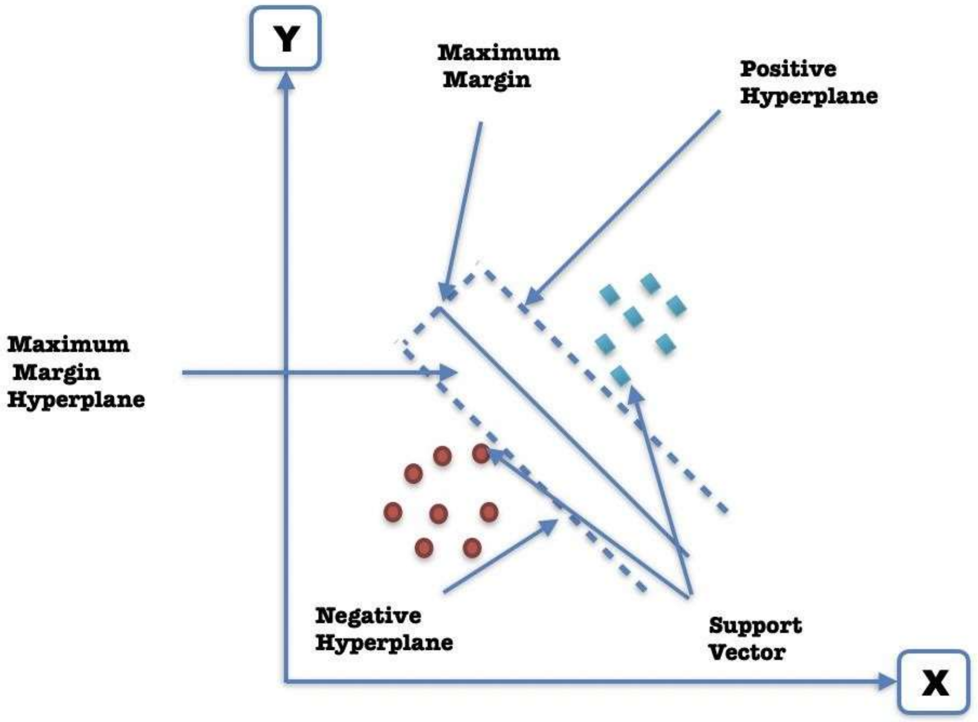


Fig 1.3 - Graphical Representation of Support Vector Machine

The Support Vector Machine's approach is to find a hyperplane (or separation line) between two data classes. The data is fed into the SVM algorithm, which produces a line that separates the two classes if possible and according to the various criteria like support vector, negative hyperplane which lies near to the origin and the positive hyperplane which lies opposite to the maximum margin the whole support vector machine outputs the desired output for the machine learning model.

The algorithm behind the Support Vector Machine can be followed by following the below discussed point:

Find the point in both classes that are closest to the line using the SVM method.Support vectors are the names given to these points. Calculate the distance between the support vector and the line. The margin is the term for this distance. Our goal is to increase our profit margins. The best hyperplane is the one with the most significant margin.

As a result, SVM seeks to create a decision boundary as broad as feasible between the two classes (those street looking classifications).

C. Naive Bayes

The classifier Naive Bayes [4] uses an easy method that learns probabilistic knowledge and represents it with comprehensible semantics. It is known as ‘naive’ because it is based on two vital clarifying assumptions that speculative attributes are independent of the terms when a category is assigned, and assumes that there are no hidden variables that affect the prediction process. It's a classification system that is based on the Bayes probabilistic theorem with robust suppositions. It is one of the most effective and efficient digit classification systems, with several applications, including bank check processing, mail sorting, data entry in forms, and so on. Naive Bayes is a probabilistic algorithm commonly used for classification problems. Naive Bayes is simple and intuitive, but in many cases it works surprisingly well. For example, the spam filter used by email applications is based on Naive Bayes.

$$X = (X_1, X_2, \dots, X_k)\ represent\ k\ features. Y\ is\ the\ label\ with\ K\ possible\ values\ (class)$$
$$From\ a\ probabilistic\ perspective, X_i \in X\ and\ Y\ are\ random\ variables$$
$$The\ value\ of\ X_i\ is\ x, Y\ is\ y.$$

The above notation is used to make a classification where we need X to predict Y. When put simply it means that for given X data point, what is the odd of Y being y

So, rewriting the given equation results into the following:

$$P(Y = y | X = (x_1, x_2, \dots x_k))$$
$$Find\ the\ most\ likely\ y \rightarrow prediction = argmax_y P(Y = y | X = (x_1, x_2, \dots x_k))$$

So, this is the actual foundation of Naive Bayes, the remainder of the algorithm is more focused on how to calculate the conditional probability.

D. K-Means Clustering

KMeans clustering is an unsupervised learning technique that groups unlabeled data sets into various clusters, with k determining the number of predefined clusters formed in the process. It is used to tackle clustering problems in machine learning or data science. For example, there are 4 clusters if K=4, 5 clusters if K=5, and so on. If we only summarize this k in a broader sense, the k-mean clustering technique is an iterative algorithm that divides an unlabeled dataset into k separate clusters, with each dataset belonging to only one group with identical attributes.

Each cluster is paired with a centroid in this centroid-based technique. This algorithm's main purpose is to reduce the sum of distances between data points and clusters.

The algorithm starts with an untagged data set, divides it by the number of clusters, and repeats the procedure until a better cluster is found. The value of k must be provided in advance in this algorithm.

This allows you to categorize your data and is a convenient technique to discover group categories in untagged datasets without having to train.

The k-means clustering technique is used to fulfill the following two tasks:

- Iteratively determines the optimal value for centroids or K center points.
- Every single point of data is allocated to the closest k-center. A cluster is formed by the data points that are closest to a certain k-center.

Following that, each cluster is distinguished from the others by data points that have certain commonalities.

The following diagram displays the functioning of the K-means Clustering Algorithm:

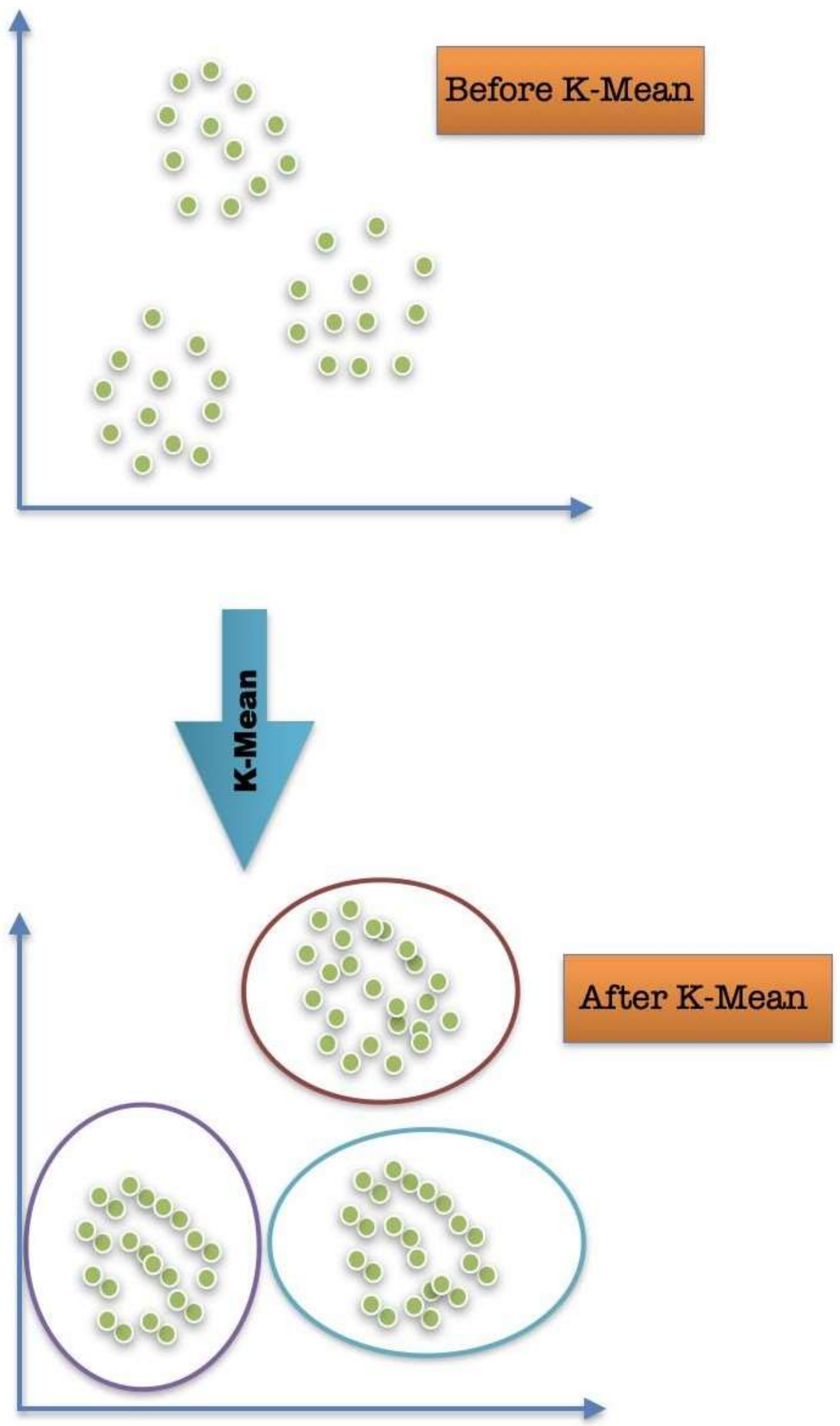

Before K-Mean
K-Mean
After K-Mean

Fig 1.4 - Graphical Representation of K-Mean

E. K-Nearest Neighbor(KNN)

K-Nearest Neighbors (or kNN for short) is a simple machine learning technique that classifies input data using the k nearest neighbors approach.

For example, suppose weight and height data points for specific males and females as shown below given to the kNN algorithm. To determine the gender of the unknown input (green dots), kNN can look at the k nearest neighbors (assuming k=3) and determine that the gender of the input is male. This is a very simple and logical way to label unknown inputs with a high probability of success.

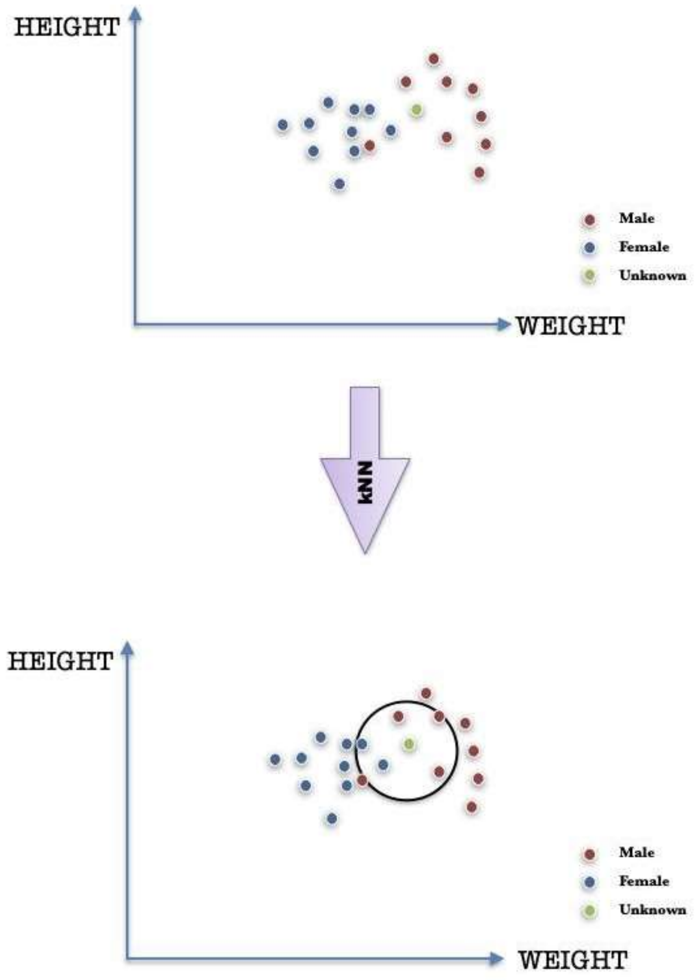


Fig 1.5 - Graphical Representation of kNN

kNNs are used in various machine learning problems. For example, in computer vision, kNN canhelp identify handwritten letters like in our project, while in gene expression analysis, algorithms are used to determine which genes contribute to a particular trait. In general, nearest-neighbor algorithms combine simplicity and efficiency, making them effective algorithms for many machine learning problems nowadays.

Pros of using KNN Algorithm are as following:

- It is simple to implement.
- It can handle noisy training data.
- This approach works well and is effective when the training data is large.

Cons of using KNN Algorithm are as following:

- The need of always computing K may be complicated in many complicated problems
- Calculating the distance between data points across all training samples would consume very high computational power, hence high computational cost which would hinder the flow in more complex problems.

CHAPTER 2. LITERATURE REVIEW

2.1 INTRODUCTION

With robots being able to replicate human intelligence, a significant amount of research and development has gone into it, resulting in deep learning and machine learning, as well as artificial intelligence. Machines are becoming increasingly smart as time passes; from fundamental math to retina recognition, they have made our lives more secure and manageable. Similarly, handwritten digit identification is an essential aspect of deep learning and machine learning that aids in detecting forgeries, and a lot of research has already been done that includes a thorough examination and implementation of a variety of standard methods.

Handwritten digit recognition (HDR) is a significant problem that allows us to understand machine learning programming concepts better. Many researchers have used this problem as an experiment to understand machine learning concepts over the years. During recent times, CNNs are widely and popularly used to provide the best solutions with the most accuracy in handwritten digit recognition systems.

Methods based on machine learning can be utilized to construct handwritten digit recognition systems. When different algorithms are employed to solve this problem, the time it takes to train the modeland the absolute accuracy can differ. There is typically a time-accuracy tradeoff when constructing such systems using various algorithms. Compared to other algorithms, specific techniques take less time to train the model completely, but they yield less accuracy. Whereas many other algorithms require a large amount of time to completely train the ML model which ultimately produces better results in terms of accuracy. As this technology has various applications, some of them require more accuracy to run effectively and have no time boundation whereas other applications may require the model to be trained faster but may allow a greater margin of error.

A Convolutional Neural Network was used in our suggested project to construct a handwritten digit recognition system. We used numerous convolution and pooling layers to train our model to achieve high accuracy. Our proposed project necessitates the precise recognition of digit strings to transform the recognised decimal number into the binary/octal/hexadecimal number system. According to our research, CNN proved to be the most significant potential method for reaching maximum accuracy.

2.2 ALGORITHM COMPARISON

Table 2.1 - Survey Table According to Research

YEAR	PAPER TITLE	ABSTRACT	ANALYSIS
2014	Recognition of Handwritten Digits using Proximal Support Vector Machine	Implementation of HDR System using feature extraction and classification of feature vectors by using Proximal Support Vector Machine (PSVM).	Proximal SVM requires lesser computation time than SVM for classification. Accuracy : 98.65% Training Time : 59 milliseconds
2014	Fast Efficient Artificial Neural Network for Handwritten Digit Recognition	HDR system implemented on GPU for parallel training to reduce training time using standard back propagation and multilayer perceptron classification.	Computing Unified Device Architecture (CUDA) on a GPU reduces training time and improves efficiency. Accuracy : 97.7% Performance factor = (T_{CPUs} / T_{GPUs}) = 2617/ 1070 = 2.445794392523364
2015	Intelligent Handwritten Digit Recognition using Artificial Neural Network	HDR system implemented by training through gradient descent back propagation algorithm and testing through feed forward algorithm.	Neural networks with 25 hidden neurons and 250 iterations were used to test 64 samples as optimal solution parameters. Training Accuracy : 99.32% Testing Accuracy : 100%
2018	Handwritten Digit Recognition Using Machine Learning	Waikato Environment for Knowledge Analysis(WEKA) used	Algorithm - Accuracy :

	Algorithms	with various machine learning algorithms to recognize handwritten digits effectively.	Multilayer Perceptron - 90.37% Support Vector Machine - 87.97% Random Forest - 85.75% Bayes Net - 84.35% Naive Bayes - 81.85% J48 - 79.51% Random Tree - 75.06%
2019	Recognition of Handwritten Digit using Convolutional Neural Network (CNN)	MatConvNet used to implement the model through Convolutional Neural Network and MNIST dataset	MatConvNet implements CNN as MATLAB functions for computing linear convolutions with feature pooling. Accuracy : 99.15% Training Time : 30 minutes
2019	An efficient and improved scheme for handwritten digit recognition based on convolutional neural network	Handwritten digit recognition based on DL4J framework for faster computation and higher accuracy on MNIST dataset using CNN.	DeepLearning4J framework with Rectified Linear Units (ReLU) activation is implemented with suitable parameters to improve accuracy of the proposed CNN. Accuracy : 99.21%
2020	Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)	Improving accuracy of HDR system using a pure and improved CNN architecture without ensemble architecture to reduce computational complexities and cost.	Extensive evaluation of MNIST dataset alongside fine tuning of CNN hyper parameters to improve performance. Accuracy : 99.87%
2020	Handwritten Digit Recognition using Machine and Deep	Comparing accuracy and runtime of SVM, CNN and MLP models	Highest accuracy on training dataset : SVM. Highest accuracy on

	Learning Algorithms	to find the best possible digit recognition model while using the MNIST dataset.	testing dataset : CNN. Minimum runtime : SVM. Maximum runtime : CNN.
2021	Recognition of Automated Hand- written Digits on Document Images Making Use of Machine Learning Techniques	CNN, ANN, SVM approaches used to implement an automatic handwritten digit recognition framework through attribute extraction.	CNN approach is significantly better than ANN and SVM approaches when vertical projection histograms are used for segmentation. CNN ranks 71% higher than ANN and SVM approaches.
2021	Handwritten Digit Recognition for Banking System	Recognizing handwritten characters for cash related transactions such as deposit, withdrawal, etc. and also recognizing handwritten account number and amounts to automate banking processes.	Python toolbox named “Neural Network Toolbox” used to implement the model using state-of-the-art classification based CNN architecture proposed by Hayder M. Albeahdili et al to achieve accuracy up to 99.61% on MNIST dataset.

The above table displays the machine learning based research work of many researchers and compares the accuracies of various machine learning algorithms that can be used to create handwritten digit recognition systems or other systems based on the pattern recognition technique.

Even though these ML algorithms may prove to be effective in some fields, numerous other fields, such the banking industry, require superior outcomes, which can be achieved using algorithms that provide accuracy higher than the ones proposed in the table above.

CNN is one such algorithm and hence it was chosen for the creation of this particular user-drawn digit recognition system proposed in this specific project report.

CHAPTER 3. PROBLEM FORMULATION

3.1 DESCRIPTION OF PROBLEM DOMAIN

Machine learning is an ever growing technology that is very widely used and humongously popular these days throughout the technology industry. Due to the amount of problems that have to be solved in the technology industry currently, machine learning is a very important tool that should be learned by upcoming software developers to enhance their overall development skills and also become avid problem solvers. This will not only make them very highly skilled individuals but also enable them to help the society overall.

The amount of requirement of machine learning these days and its related problems that need to be solved, this becomes a topic which should be addressed as quickly as possible. Hence, a project related to machine learning became the problem formulation domain for this specific project report.

One of the most essential and practical difficulties in pattern recognition applications is handwritten digit recognition [2]. It allows a developer to learn machine learning from within and hence the creation of which ultimately became the problem that was decided to be addressed in this specific project report.

When dealing with natural difficulties like check processing in [9] banks, sorting mails, data entry in forms, verifying signatures, deciphering addresses, analyzing and verifying documents, and so on, a handwritten digit recognition system comes in handy.

Because of all of these important uses, we developed a handwritten digit recognition system using the Convolutional Neural Network machine learning technique and numerous Python packages and the MNIST dataset. It is a hard task for a machine to recognize handwritten digits as they are not perfect and can be drawn with various different sizes and shapes. This inspired us even more to tackle the problem at our hands i.e., the creation of a handwritten digit recognition system which provides good accuracy as well as efficiency.

Also, neural networks comprises elements that work parallelly that take their inspiration from biological nervous systems which will enable us to get in depth knowledge of how a human body's nervous system functions alongside learning the booming machine learning technologies and algorithms.

The need of converting very long decimal numbers to other number systems that may be hard to type(due to a typing mistake being made) as compared to writing them on a paper instead helped us formulate the problem statement for this particular project. This will also help us to understand how various number systems work and are converted to one another. Implementing the small details for number system conversion will also help us become detail-oriented and not just focus on implementing the required machine learning algorithm to complete the project.

Because many handwritten digit recognition systems are intended to work in real-time, they must be fast, efficient, and resilient. As a result, in this project, we constructed an efficient handwritten digit recognition system combined with a real-time GUI built using the standard Tkinter python library.

3.2 PROBLEM STATEMENT

The problem statement for this project states the creation of a handwritten digit recognition system using machine learning for the conversion of handwritten decimal numbers to binary numbers(and/or other number systems) using the Convolutional Neural Network (CNN) and the dataset used should be the MNIST dataset.

The neural net should consist of various pooling and convolutional layers for maximizing efficiency and not waste any resources alongside taking a 3x3 kernel for image recognition.

The machine learning model should then be incorporated within a GUI which will consist of a canvas widget created using the standard GUI library of python named Tkinter.

The user will be able to draw handwritten digits on the canvas widget by keeping the left mouse button clicked while dragging the cursor across the screen accordingly. Alongside this, several buttons which will be used to implement all the other required functionalities will be created.

3.3 DEPICTION OF PROBLEM STATEMENT

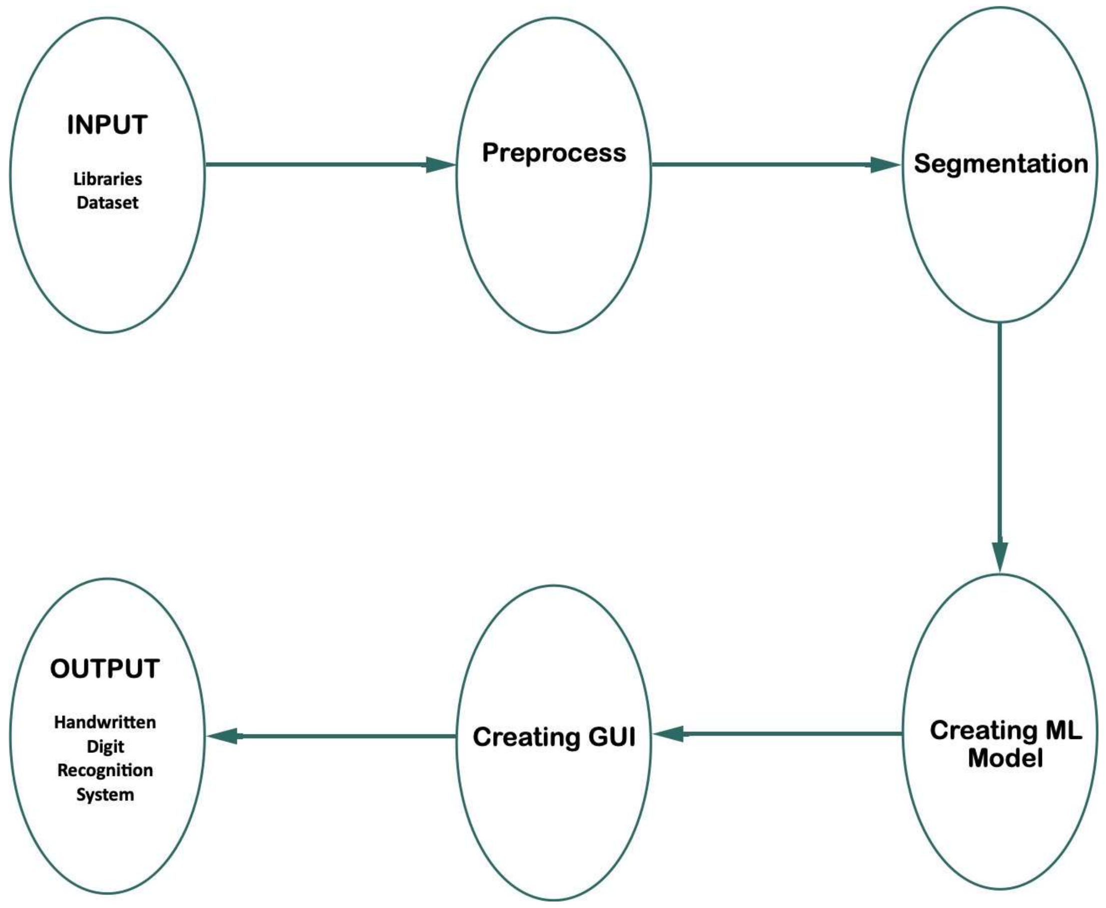


Fig 3.1 - Block Diagram of Problem Statement

The block diagram describing the problem statement proposed in this project can be explained as follows:

1. Input - This is the beginning of the problem statement and will include the import of libraries and dataset required to create the project.

2. Preprocess - This is the step that will include the preprocessing and cleaning of the dataset for making it appropriate for the machine learning model to utilize and provide good results.

This will in turn improve the accuracy as well as the efficiency and make the overall process faster.

3. Segmentation - This is the 3rd step and will be done alongside feature extraction. This will be done to understand the image data at pixel level and extract useful information out of it to recognize handwritten digits drawn by the user properly.

4. Creating ML Model - This will be the 4th and main step which will include the creation i.e., training, testing and evaluating the machine learning model so that it can be used accordingly to fulfill the requirements of the proposed project.

5. Creating GUI - This is the 5th step which will include the creation of the GUI using the Tkinter python GUI library. This GUI will incorporate the machine learning model created in the previous step and use it suitably.

6. Output - This will be the final step which will mark the end i.e., the finalization of the handwritten digit recognition system creation process result of which will be the final software that will recognize handwritten digits drawn by the user on a canvas widget and provide functionalities of converting the recognized handwritten number (decimal number system by default) to another number system format.

CHAPTER 4. PROPOSED WORK

The main goal of the proposed initiative is to recognise user-defined handwritten digits drawn on a canvas and identify the whole number (in decimal number system by default) that the user inputs, which is subsequently translated to binary number system (and/or other number systems). For the same, a graphical user interface (GUI) will be developed. The user can draw handwritten digits on a canvas widget, which will subsequently be used to generate a specific result based on the functions implemented in this project. A Convolutional Neural Network (CNN)-based machine learning model will be developed and tested using a specific dataset. The MNIST dataset will be utilized for this research.

Weights/Bias will be set while training the model and the model will then be evaluated based on specific conditions to provide the desired output. Each neuron in a neural network computes the output value by applying a specified function to the input value obtained from the previous layer receptive field. This function applied is determined by the weight vector of and the offset(usually real numbers). Training consists of continuously modifying these respective biases and weights.

For implementing this project as per the requirements, several libraries will first be imported alongside loading the MNIST dataset for training our machine learning model. After that, the data will be preprocessed and normalized as a 28x28 matrix comprising grayscale pixel values. Scaling input inputs to normalize pixel values to the range 0 and 1 by dividing each value by the maximum value is a good idea. This process is known as binarization.

Then our machine learning model will process the given gray-scale input image and derive a pattern using the relative 1's and 0's matrix to recognize the input shape and produced the desired output in the form of a number label. After training and evaluating our machine learning model, a GUI will be created by using the standard python GUI library Tkinter. Several buttons will be included to implement the functional requirements within the proposed software. The software will consist of 5 buttons - Recognize Number, Clear Canvas, Convert to Binary, Convert to Hexadecimal, Convert to Octal.

The proposed software will also consist of an additional copy functionality which will also be implemented in a button format that can be used to copy the output which is produced according to the user’s choices.

Our suggested method combines CNN with numerous pooling and convolutional layers with a 3x3 kernel size. The kernel is nothing more than a filter that extracts features from a picture. A kernel traverses a matrix input, conducts a dot product on a subregion of the input, and provides the result as a dot product matrix. The proposed machine learning model will be trained through 5 epochs as a standard to achieve great accuracy.

Ultimately the project will be divided into 3 files:

1. train_digit_recognizer - This file will consist of the python code required to create the machine learning model using the CNN algorithm. It will result in the creation of a machine learning model that will then be used by the final_gui file.

2. final_gui - This file will consist of the python code required to create the final GUI for the proposed project. It will do so by importing the machine learning model created by the train_digit_recognizer file and incorporating it within the GUI which will in turn be created with the help of the standard Tkinter python GUI library.

3. trained_model - This file will be the trained machine learning model saved with the extension ".h5" which will be used within the GUI file. It will act as the brain of the system as it will be the major part that will be required to recognize handwritten digits accurately and efficiently.

CHAPTER 5. SYSTEM DESIGN

One of the most important aspects of a handwritten digit recognition project to get right is the system design. We'll do this by leveraging the MNIST dataset and pre-processing the images it contains before classifying them. Many numbers are handwritten as training samples in MNIST (Modified National Institute of Standards and Technology). [2] [14] It contains 70,000 photos, with 60,000 being used in the training dataset and 10,000 in the testing dataset. Both feature suitably labeled representations of the ten numbers ranging from 0 to 9.

Algorithm used for classification in our machine learning model will be CNN i.e., [3] Convolutional Neural Network which works with the help of multiple neuron layers.

CNNs' essential building pieces are convolutional layers. A set of trainable filters (or kernels) with a narrow receptive field but spanning the entire depth of the input volume constitutes a hierarchical parameter.

Each filter is convolved over the width and height of the input volume during the forward pass to compute the dot product of the filter element and the input, resulting in a two-dimensional map of the corresponding filter activations. As a result, the network learns to activate a filter when a specific type of object is recognised at a specific spatial point in the input.

A CNN is a network used for image processing in the deep learning sector. Convolution is a linear function that, like a conventional neural network, sets weights according to input in a convolutional neural network. These weights are eventually utilized for training the neural network in order to improve its accuracy and reliability.

In comparison to other algorithms that classify images, CNNs use relatively less preprocessing. In other words, the network learns how to optimize filters (or kernels) through automatic learning, but in traditional algorithms these filters are designed manually. This freedom from past feature extraction information and human interference is a big plus.

Multiple convolutional layers and pooling layers will be used in the proposed project's CNN. The input is convolved by a convolutional layer, which then sends the result to the next layer. This is similar to how neurons in the visual brain react to specific stimuli. Only the receptive region's data is processed by each convolutional neuron.

The output of a cluster of neurons from one level is merged into a single neuron from the next level via a pooling layer, which decreases the data size. Small clusters, usually two × two tiles, are combined in local pools. All neurons in the functional map are affected by the global pool. Maximum and average pooling layers are two common types of pooling layers. The maximum value of each local neuron cluster in the functional map is used in the max pool, whereas the average pool utilizes the average value.

Pooling layers typically operate independently at each depth or slice of input and scale in space. A common version of maximum pooling is a 2x2 filter layer that increments by 2, discarding 75% of the activations by subsampling each input depth slice by 2 in width and height according to the following equation:

$$f_{X,Y}(S) = \max_{a,b=0}^{1} S_{2X+a,2Y+b}\,.$$

CNN has emerged as one of the best machine learning algorithms as the demand for machinelearning grows. In computer vision, end-to-end learning and prediction is a prevalent approach. Critical systems, such as autonomous automobiles, require explanations that are understandable by humans. By showing the most relevant spatial domains/temporal events, recent breakthroughs in visual salience, spatial, and temporal attention help support CNN predictions.

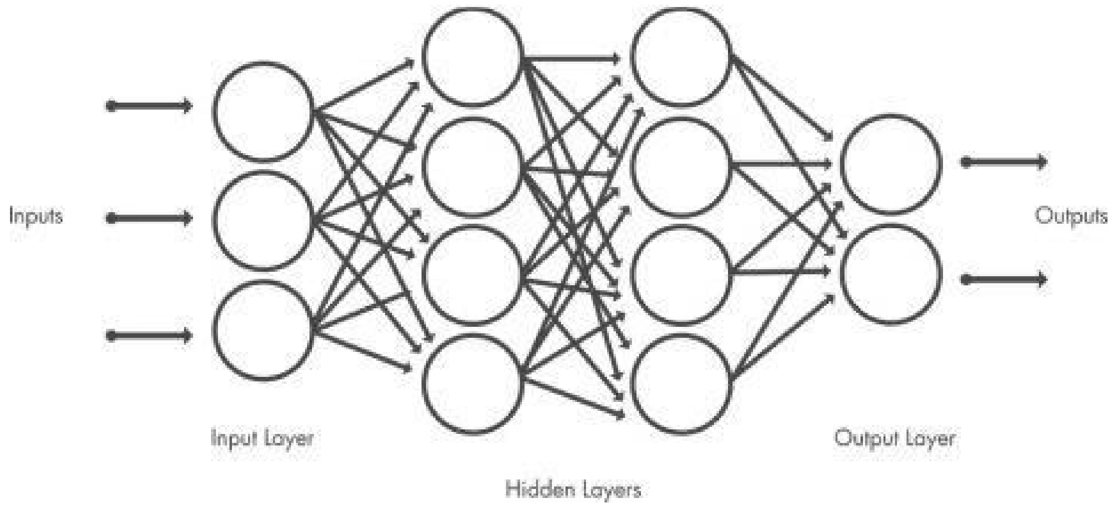


Fig 5.1 - Convolutional Neural Network

5.1 ARCHITECTURE

The proposed model contains the following stages in order to [15] classify and detect the digits:

5.1.1 Preprocessing

In machine learning, data preprocessing is an important step that helps improve data quality to help extract insights from the data that are very meaningful.

Data preprocessing is the process of cleaning and organizing raw data in order to make it suitable for building and training machine learning models. Simply put, data preprocessing in machine learning is a data mining technique that recognises raw data and converts it into a readable format.

Preprocessing is generally done through various steps:

1. Obtaining the data needed to train and evaluate the machine learning model.
2. Importing the required libraries for implementing the machine learning model using the acquired dataset.
3. Importing the gathered dataset alongside the libraries for ultimately implementing the functional requirements of the proposed machine learning model.
4. Finding the data that is missing from the acquired dataset.
5. Using the acquired dataset and encoding categorical data within it.
6. Dividing the gathered dataset into training and testing parts/datasets.
7. Feature scaling.

The main goal of the pre-processing step is to execute different operations on images that are provided to the network as input. The quality of the image is upgraded by preprocessing and hence it makes it very reasonable for the segmentation process. The first step is to train a set of images that are to be processed in order to filter out the data, by thresholding the input images into their binary formats. Preprocessing portrays and characterizes this process at a smaller extent. Binarization is the conversion of a grayscale image to a binary image.

5.1.2 Segmentation

Image segmentation is a computer vision approach that is used to understand the information of a given image at the pixel level. This is not the same as object detection, which assigns several labels to the entire image. Object detection works by constructing a bounding box around things in a picture. Image segmentation offers more precise information about the content of the image it is working on at the end.

Image segmentation allows a user to decompose an image into parts which are meaningful and also understand the image at a more detailed level. So it's different from other computer vision tasks that people are generally familiar with. Image segmentation is useful for seeing the contents of an image as a whole. While object detection allows a user to track by finding the contents of an image, segmentation allows a user to identify and understand the boundaries and shapes of objects in an image.

When you consider the variety of photo editing tools available today, it's simple to see why image segmentation is so important. Image segmentation offers a wide range of possibilities for these creative tools, from automatically separating background and foreground to cropping segmented items and creating adaptive portrait modes.

Formation of sub images from the given sequence of images is done once the preprocessing step is finished. These digit images are categorized into a sub-image of independent digits. After that, each of these individual digits is resized. During this segmentation process, an edge detection approach is applied to segment the collection of digit pictures.

5.1.3 Feature Extraction

The method for converting raw data into numerical characteristics is known as feature extraction. These features can be processed while retaining the information which is present in the original data set. This gives better results than directly applying machine learning on the given data set. With the advent of deep learning, the first layer of deep networks has mostly been replaced by feature extraction, but mostly for image data.

Feature extraction can be achieved manually or automatically:

- Manual feature extraction involves the identification and description of features that are relevant to a given problem and accordingly implementing the methods that are required to extract those features.

- Automatic feature achieves feature extraction automatically by using special algorithms or deep networks to extract features from an image without any manual human input.

After the segmentation and preprocessing processes are completed, the pre-processed images are represented as a matrix consisting of pixels from extremely large images. It is very vital to represent the digits in these images that contain the required information. This process of extracting the digits from whole images is known as feature extraction. The feature extraction phase undergoes the data redundancy removal process.

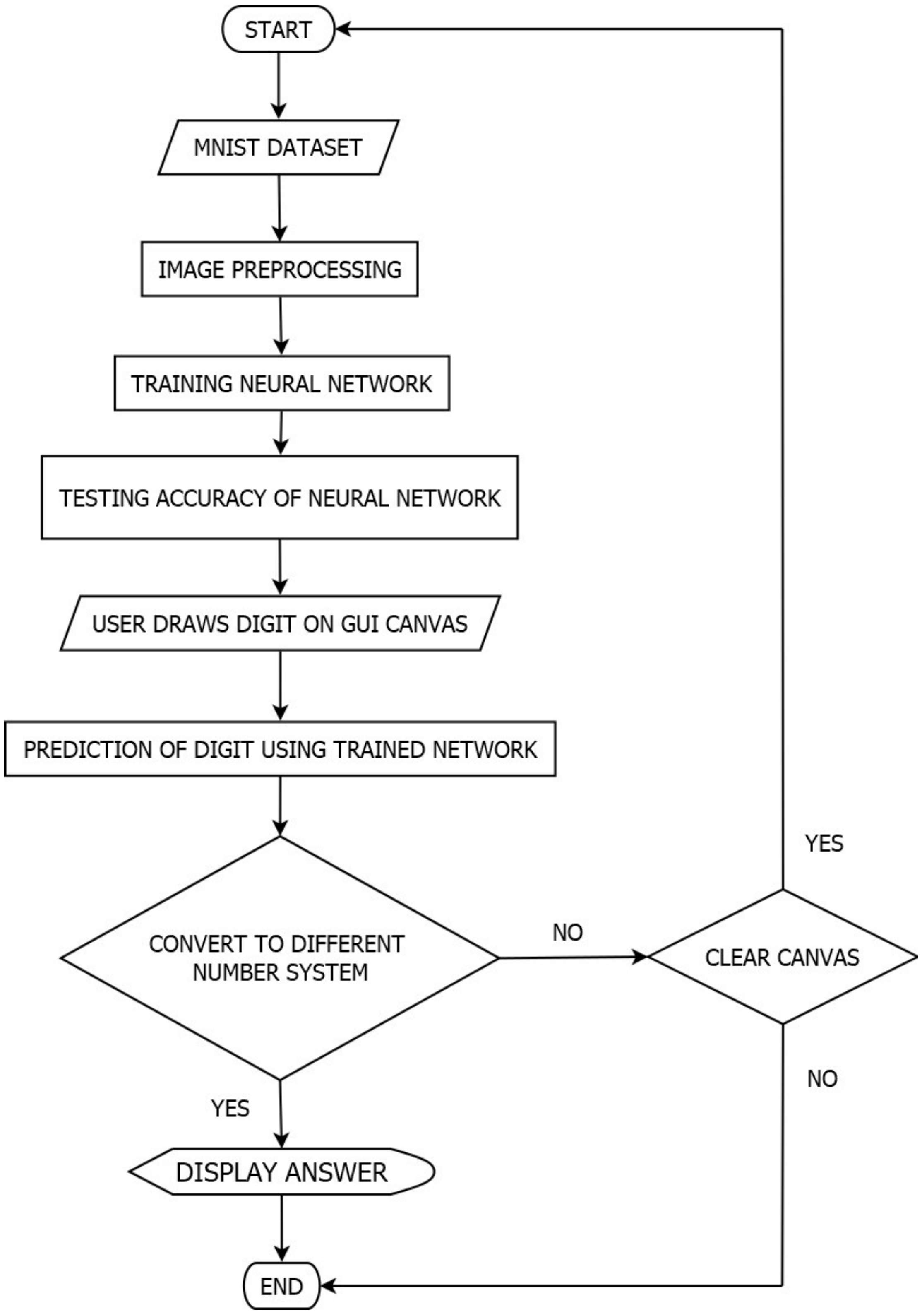

START
MNIST DATASET
IMAGE PREPROCESSING
TRAINING NEURAL NETWORK
TESTING ACCURACY OF NEURAL NETWORK
USER DRAWS DIGIT ON GUI CANVAS
PREDICTION OF DIGIT USING TRAINED NETWORK
CONVERT TO DIFFERENT NUMBER SYSTEM
NO
CLEAR CANVAS
YES
NO
YES
DISPLAY ANSWER
END

Fig 5.2 - Activity Diagram of Proposed HDR System

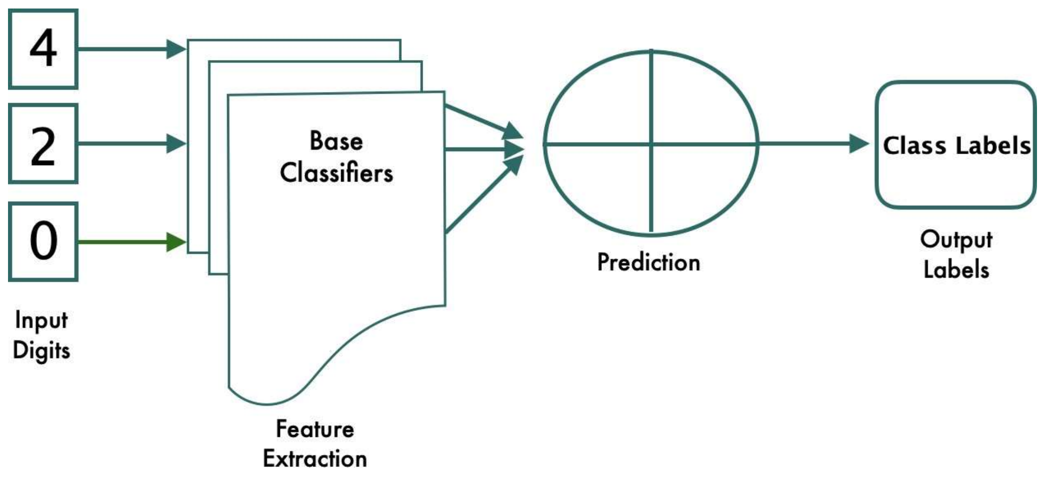


Fig 5.3 - Component Diagram of Proposed HDR System

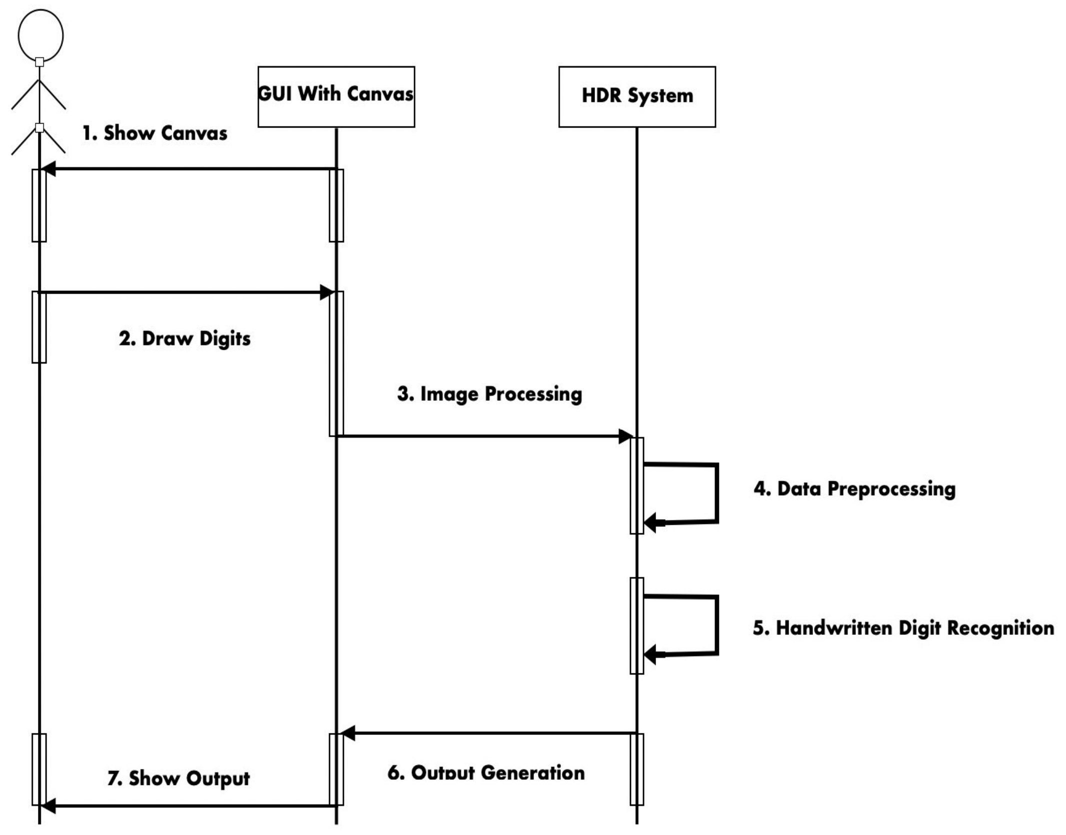


Fig 5.4 - Sequence Diagram of Proposed HDR System

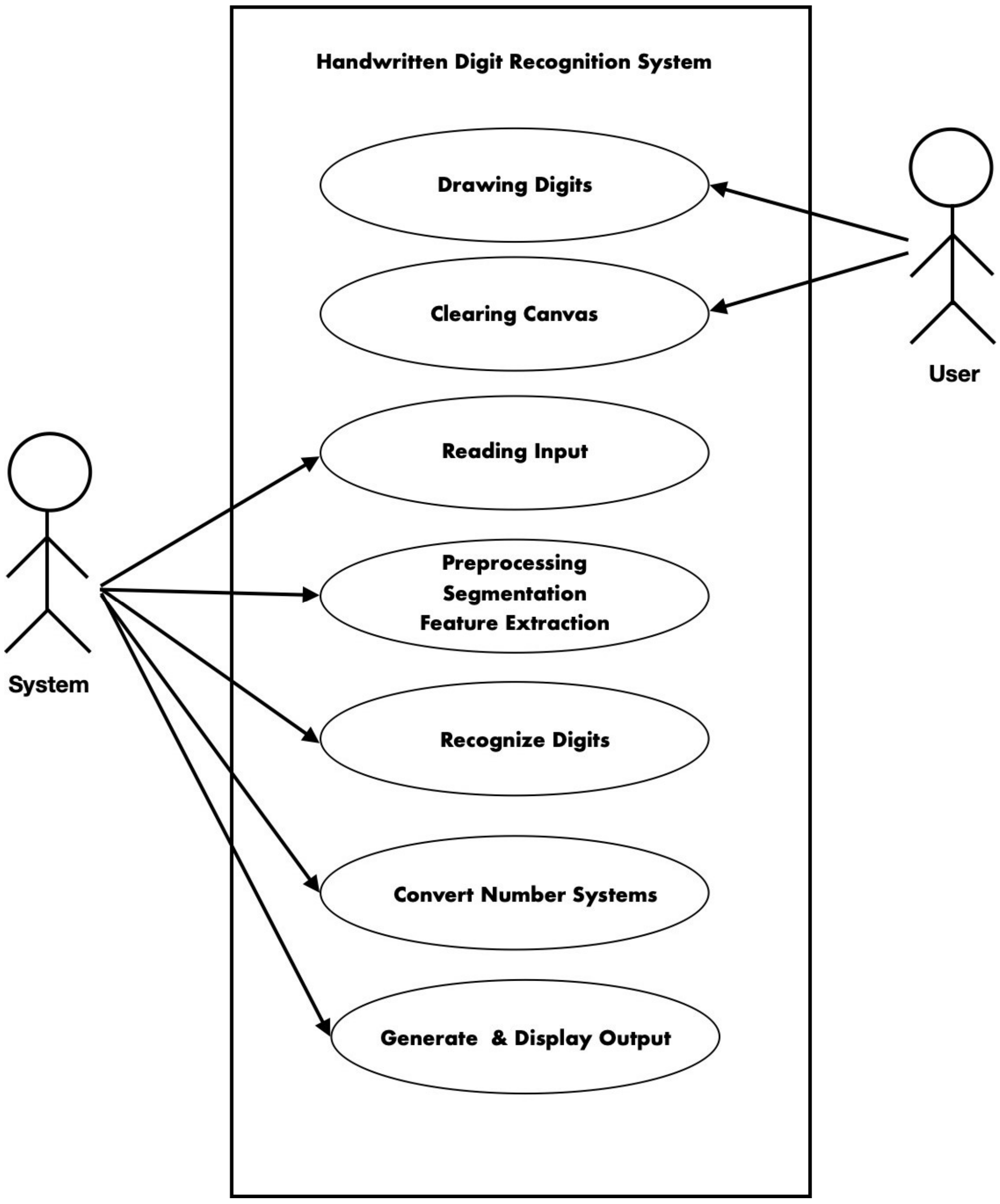


Fig 5.5 - Use-Case Diagram of Proposed HDR System

CHAPTER 6.
IMPLEMENTATION

6.1 EXPERIMENTAL SETUP

6.1.1 Algorithms/Techniques Used

A. Existing System

Proximal Support Vector Machine (PSVM), Multilayer Perceptron, Support Vector Machine (SVM), Random Forest, Bayes Net, Naive Bayes, J48, and Random Tree are some of the methods that can be utilized to create machine learning-based handwritten digit recognition systems.

These algorithms alongside their accuracies can be defined as follows:

- Proximal SVM -

  In contrast to the standard SVM, which classifies points by placing them in one of two separated half-spaces, proximal SVMs classify points by placing them in the closest of two distant parallel planes (in input or feature space). This formula can as well be interpreted as a method of normalized least squares and understood in a general context of networks which are normalized.

  Standard SVMs require the solution of a linear or quadratic programme, which takes a long time to compute, whereas proximal SVMs just require the resolution of a single system of linear equations.

  Proximal SVM provides an accuracy of the order of 98%.

- Multilayer Perceptron -

  Because it is a neural network-based classifier, the Multilayer Perceptron (MLP) can be used to classify handwritten digits. The input, hidden, and output layers make up multilayer perceptrons.

  These levels have a set number of nodes, and all of the nodes in one layer are connected to all of the nodes in the following layer. These nodes are also known as neurons. The number of neurons required for a machine learning model is decided experimentally. The nodes within a multilayer perceptron system are connected to each other through weights/bias.

Multilayer Perceptron provides an accuracy of the order of 90%.

- SVM -

  The Support Vector Machine is a supervised machine learning technique that offers data point classification by maximizing class fields in a multidimensional space.

  SVMs represent data points in multidimensional space rendered due to instances of individual classes separated by fair spacing. A new model is then drawn into the same space and is expected to remain in categories according to the side of the gap to which that model belongs.

  Standard SVM provides an accuracy of the order of 87%.

- Random Forest -

  Random Forest is a popular supervised machine learning algorithm. It may be used for both classification and regression in machine learning.

  Random forest is a strategy that uses ensemble learning to mix several classifiers to solve complicated issues and enhance model performance. It is a classifier that includes multiple decision trees for different subsets of a given data set and uses the mean to improve the prediction accuracy of that data set.

  Random Forest provides an accuracy of the order of 85%.

- Bayes Net -

  A Bayesian Network (BN) is a probabilistic graphical model for communicating knowledge of a region of uncertainty, where each node is referred to by a random variable and each edge is represented by a conditional probability for that random variable.

  Belief Network is another name for Bayesian Network. Because of their dependencies and conditional probabilities, BNs correspond to directed acyclic graphs (DAGs) that do not allow loops or self-joins.

  Bayes Net provides an accuracy of the order of 84%.

- Naive Bayes -

  Naive Bayes is an algorithm related to supervised machine learning techniques. It is commonly used to solve classification problems and is based on Bayes' theorem.

Although it is one of the most basic machine learning algorithms, it's also a very efficient categorization strategy for developing rapid machine learning models that can make accurate predictions.A probabilistic classifier, the Naive Bayes classifier predicts the likelihood of an item.

Naive Bayes provides an accuracy of the order of 81%.

- J48 -

  J48 is a decision tree classification-based machine learning algorithm that correlates to the Iterative Dichotomiser 3.. It is a very useful algorithm for data exploration continuously and categorically.

  However, using it for instance purpose takes up more memory space and reduces the performance and accuracy of classifying data.

  J48 provides an accuracy of the order of 79%.

- Decision Tree -

  Decision trees are a type of supervised learning technique-based machine learning algorithm that may be used to solve both regression and classification issues but are most commonly used to solve classification problems. The data set features are represented by the inner nodes, the decision-making rules are represented by the branches, and the conclusion is represented by each leaf node.

  There are two nodes in it: a leaf node and a decision node. Leaf nodes are the consequence of these decisions and have no more branches, whereas decision nodes are used to create decisions and have numerous branches.

  Decision Tree provides an accuracy of the order of 75%.

Even though these algorithms may be effective in some applications based on this technology, many others, for as [9] banking industry applications, require superior outcomes that can be attained utilizing alternative algorithms than the ones stated above.

B. Proposed System

[3] Convolutional Neural networks (CNN) can be used to construct handwritten digit recognition systems to reduce error and increase overall efficiency.

Our proposed method achieves this by combining a 3x3 kernel with a CNN with various pooling and convolutional layers. 60,000 28*28 grayscale photos are used in the training of our model. Our model is trained over a normal five epochs to reach an accuracy of 99.16 per cent, which is significantly higher than traditional handwritten digits recognition algorithms such as SVM, Multilayer Perceptron, Bayes Net, Random Forest, and others.

6.1.2 Tools Used

A. Tensorflow

TensorFlow is a Google-developed software library or framework for quickly implementing machine learning and deep learning principles. For development, it uses algebraic calculation algorithms to quickly calculate many mathematical expressions. TensorFlow is an open-source library developed by the Google Brain Trust to handle large calculations in machine learning and deep learning challenges. TensorFlow is extremely fast because it was written in C and C++, but it can also be used with Python, Java, and Go APIs, depending on the type of task.

For each machine learning model, TensorFlow creates a data flow graph, which is made up of two units: a tensor and a node. A tensor is a multi-dimensional collection of numbers. A node is a unit of mathematical computation that is currently being worked on to produce the desired outcome.

TensorFlow is a set of procedures that allow both novices and experts to build machine learning models in a range of languages using intuitive, high-level APIs. Developers can use a variety of platforms to deploy models, including servers, clouds, mobile and edge devices, browsers, and many other JavaScript platforms. This makes the transition from model creation and training to deployment easier for developers.

TensorFlow can compute gradients for a model's parameters automatically, which is important for algorithms that require gradients to enhance performance, such as backpropagation. To accomplish so, the platform must keep track of the actions conducted on the input tensor of the model and then compute the gradient using the necessary parameters.

A "hard run" mode is available in TensorFlow. That is, the job is evaluated instantly rather than being added to a later-run computational graph. Because the data grows with each line of code rather than later in the computation tree, the debugger can step through the code that is being performed eagerly. Because of its step-by-step clarity, this execution paradigm is considered simple to debug.

TensorFlow is the most popular machine learning platform and library. Keras is used by the TensorFlow API to allow users to build their own machine learning models. TensorFlow can help you load model training data and deploy it via TensorFlow Serving, in addition to constructing and training models.

It can be used to speed up and improve the accuracy of MRIs when detecting specific body areas. TensorFlow was utilized by the virtual learning platform InSpace to filter out potentially dangerous classroom chat posts.

B. Keras

Keras is a very widely used high-level Application Programming Interface(API) based on neural networks. It is easily extensible, flexible and easy to use. It is built in Python and provides a base for various back-end neural network engines. Keras gives the advantages of vast adoption, providing a basis for a huge variety of production deployment alternatives, and integration with back-endengines.

Keras contains several implementations of common neural network building elements like layers, targets, activation functions, and optimizers, allowing you to work with picture and text data while reducing the amount of coding required to construct deep neural network code. There are numerous tools offered. The source is hosted on GitHub, and there's a GitHub problems page and a Slack channel in the community support forum.

Keras enables users to build deep models using their cellphones, the web, or a Java Virtual Machine.

[3] It also allows for distributed deep learning model training on the graphics processing unit (GPU) and tensor processing unit (TPU) clusters.

Keras is based on a simple framework that makes it simple to create deep learning models using TensorFlow or Theano. Keras is a framework for quickly defining deep learning models. Keras is the best deep learning framework available.

Keras is a versatile and powerful framework with numerous benefits. The general public's support. Keras neural networks are easy to work with and are built-in Python. Keras supports both convolutional and recursive networks. Deep learning models can be merged in a variety of ways because they are distinct components.

C. Tkinter

Tkinter is Python's standard GUI library. Tkinter is a toolkit for creating graphical user interfaces that are quick, efficient, and easy to use. A dynamic object-oriented interface is supported by Tkinter. The first step in creating a GUI application with the Tkinter Python library is to import the Tkinter module into our GUI python programme. Various widgets relating to certain actions caused by user interactions are then added to the GUI main window.

Tkinter adds a strong and dynamic system for dealing with events and, ultimately, binding handlers to these events. Tkinter is a Python wrapper for the entire Tcl interpreter that comes with the Python interpreter. Tkinter calls are converted to Tcl commands and passed to this built-in interpreter, allowing you to use Python and Tcl together in the same application.

Benefits of Tkinter are Layered Approach: The layered approach used in Tkinter development gives Tkinter all the benefits of the TK library. So Tkinter has inherited the strengths of GUI toolkits that initially had time to mature. This makes the initial version of Tkinter much more stable and reliable than if it had been rewritten from scratch. Also, converting Tcl/Tk to Tkinter is very easy, making it very easy for Tk programmers to learn how to use Tkinter.

Accessibility:Learning Tkinter is very intuitive, so it's fast and easy. The Tkinter implementation hides detailed and complex calls in a simple and intuitive way. This is an extension of the Python mindset, as the language is well suited for rapid prototyping. Therefore, his preferred GUI library is expected to be implemented in the same way.

Portability using Tkinter Python scripts requires no modifications to be portable from one platform to another. Tkinter can be used on all platforms where Python is implemented: Microsoft Windows, X

Windows, and the Macintosh. This provides a huge advantage over most competing libraries, which are often limited to one or two platforms. Tkinter also provides a default look and feel for the specific platform it is running on.

AvailabilityTkinter is now included in all Python distributions. So no additional modules are required to run scripts with Tkinter.

D. Numpy

Numerical Python is referred to as NumPy. NumPy is a prepackaged Python library that contains a collection of routines for manipulating multidimensional array objects. Using NumPy, various mathematical operations as well as logical operations can be performed on the previously included multidimensional array objects.

NumPy is a bytecode interpreter that targets CPython, the Python reference implementation. Due to a lack of compiler optimizations, mathematical algorithms developed for this version of Python are substantially slower than compiled algorithms. NumPy addresses the slowness issue in part by providing multidimensional arrays as well as array-friendly functions and operators. Using Numpy necessitates the rewriting of certain code (mainly inner loops).

Because both are interpreted, NumPy in Python provides features equivalent to MATLAB, allowing users to construct fast programmes when most operations are performed on arrays or matrices rather than scalars. NumPy interacts smoothly with Python, a more current and full programming language, while MATLAB has many more toolkits, such as Simulink.

NumPy arrays are used to store and modify data in a Python interface to the popular OpenCV computer vision toolkit. Indexing, slicing, or masking with other arrays is a particularly efficient approach to accessing specific pixels in a multi-channel image because they are easily represented as 3D arrays. NumPy arrays, OpenCV's general image data structures, extracted feature points, filter kernels, and other tools make programming and debugging much easier.

The "ndarray" data structure for multidimensional arrays is a crucial element of NumPy. These arrays are incremental memory representations. These arrays are uniformly typed, unlike Python's built-in list data structures. An array's items must all be of the same type.

Adding or removing elements from an array is more difficult than adding or removing things from a Python list. The np.pad() technique expands an array by creating a new one with the necessary form and padding values, copying the supplied array into it, and returning it. The NumPy np.concatenate([a1,a2]) action returns a new array filled sequentially with items from the two given arrays, rather than concatenating them. As long as the number of members in the array does not change, np.reshape() can be used to reshape it. NumPy arrays must be representations of contiguous memory buffers, which leads to this problem. Many large-scale scientific computing applications today have requirements that transcend the capability of NumPy arrays. NumPy jobs also only use one processor.

E. Matplotlib

Matplotlib is a cross-platform python library whose numerical extension is the Numerical Python i.e., NumPy library. Matplotlib is generally used for data visualization and plotting graphs to analyze the results of our machine learning model.

It is used widely as it is an open source library. It provides various key features, one of which is the integration of plots in our GUI applications using the matplotlib API. A Python matplotlib script is so concise that it requires the use of only a few lines of code to display a visual plot in most of the cases.

Matplotlib is an object-oriented library for embedding graphs in programmes written in general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. State machine-based "pylab" procedural interfaces (such as OpenGL) that closely mimic the MATLAB interface are also available, but their use is discouraged. Matplotlib is used by SciPy.

In most circumstances, the Python matplotlib script is organized enough to generate graphic data plots with only a few lines of code. The two APIs are overlaid by the matplotlib scripting layer. Matplotlib.pyplot leads the pyplot API hierarchy of Python code objects. A set of object-oriented (OO) API objects that are easier to put together than pyplot. This API allows you to access Matplotlib's core layers directly.

The architecture of the matplotlib is divided into three layers:

The backend layer is the drawing's bottom layer, and it contains the implementation of the various plotting functions. FigureCanvas (the drawing surface), Renderer (the class that handles drawing on the surface), and Event are the three main classes in the inner layer (handles mouse and keyboard events).

The second architectural layer is the artist's layer. Axes that coordinate the use of the renderer on the shape canvas are part of the plotting functions.

The scripting layer is where the majority of your code is executed. The remaining layers are virtually completely handled by script level methods; all we need to worry about is the current state (drawings and subplots).

F. OpenCV-Python

It is one of the most widely used python libraries built to solve vision problems in the computer science field. The module name to be imported for using OpenCV-Python is “cv2”. It may be used to process photos and videos in order to distinguish various objects, faces, and human handwriting, leading to its eventual usage in a handwritten digit recognition system. To execute specialized machine learning tasks, NumPy operations can be combined with OpenCV.

OpenCV is written in C++, and while the main interface is written in C++, it is still incomplete, with a large but obsolete C interface. The C++ interface displays all the latest advances and algorithms. Python, Java, and MATLAB/OCTAVE all have bindings. The web documentation contains APIs for these interfaces. Shells in a variety of programming languages have been created to improve user adoption. JavaScript bindings for a subset of OpenCV functionalities were released as OpenCV.js for use on the web platform in version 3.4.

G. Pandas

Pandas is a very widely used python libraries that is used to perform several machine learning operations alongside data analysis. Pandas is generally included in most of the python data science projects as it works well with various other data science packages. Some operations that can be performed using the Pandas package are cleaning of data, filling data, normalization of data, merging as well as joining data and most importantly visualization of data.

Pandas is generally used for data analysis and manipulation of related tabular data on data frames.

Pandas can read data from comma-separated values, JSON, Parquet, table or SQL database queries, and Microsoft Excel. Pandas allows you to execute data operations like merging, reshaping, selecting, and cleaning, as well as data manipulation. Many of the Data Frame features present in the R programming language were transferred to Python with the introduction of pandas. Pandas is a NumPy module that focuses on efficient array manipulation rather than Data Frame properties.

Pandas software benefits include quick and efficient data processing and analysis. Data can be loaded from a variety of file objects. Missing data (expressed as NaNs) can be easily handled with both floating-point and non-floating point data. Resizable: DataFrames and multidimensional objects can have columns added and removed. Merging and combining datasets. Flexible data set reconstruction and rotation. The time series function is available.

Pandas supports two data structures for data management:

Any sort of data can be stored in a Pandas series, which is a one-dimensional array of labels (integers, strings, floating point numbers, Python objects, etc.). Axis labels are collectively referred to as indexes. A Pandas series is nothing more than a column in an Excel spreadsheet. Labels do not have to be one-of-a-kind, but they must be hashable. Objects enable both integer and label-based indexing, as well as a number of indexing strategies.

A Pandas DataFrame is a resizable 2D tabular data format with labeled axes that might be potentially diverse (rows and columns). A data frame is a type of two-dimensional data structure. In the table view, data is organized in rows and columns. Data, rows, and columns are the three primary components of a Pandas Dataframe.

H. Sklearn

Sklearn stands for Scikit-learn which is a well built python package that offers a wide range of robust machine learning algorithms predefined within it. It is one of the most widely used python libraries for scientific computing. It can be used to perform many operations such as classification, regression, clustering, preprocessing, selection of model, etc.

Support vector machines, random forests, gradient boosting, k means, and DBSCAN are among the classification, regression, and clustering algorithms included in Sklearn, which is designed to work with Python NumPy and SciPy numerical and scientific libraries.

Scikit Learn is mostly built-in Python, with NumPy being used extensively for high-performance linear algebra and array operations. Some basic algorithms are also developed in Cython to increase performance. A Cython wrapper for LIBSVM implements the support vector machine. LIBLINEAR wrappers for logistic regression and linear support vector machines are similar. In such circumstances, extending these methods with Python may be impossible.

It provides a complete set of supervised and unsupervised learning algorithms covering the following areas:

Classification: Determining which category an object belongs to a regression prediction of a continuous property associated with an entity. Automatically group similar objects into sets using models such as clustering k means. Dimension Reduction uses models like Principal Component Analysis to minimize the number of attributes in data for summary, visualization, and feature selection (PCA). Compare, view and select model selection parameters and models. Extraction and normalization of features, including attribute definitions of preprocessed image and text data.

6.2 DATASET DESCRIPTION AND FUNCTIONALITY IMPLEMENTATION

The MNIST dataset will be used to develop the handwritten digit recognition system required for this project. The Convolutional Neural Network algorithm will be utilized to create the HDR system (CNN).

As training examples, MNIST (Modified National Institute of Standards and Technology) contains a large number of handwritten digits. [5][14] It has 70,000 photos, with 60,000 in the training set and 10,000 in the testing set, all with suitably labeled images of the ten numbers spanning from 0-9. The photos of handwritten numbers are 28*28 grayscale images. MNIST is a computer science and vision database that contains handwritten numbers with labels that identify them. Every MNIST data point is made up of two parts: a handwritten digit image and a target label.

The data that is provided directly within the MNIST dataset is not cleaned and hence it will be preprocessed and cleaned before being fed into the machine learning model. This is done by reshaping the dimensions of the training data to fit our requirements. [3] After completing data preprocessing, the CNN model with convolutional layers will be generated. Because picture data may be represented as grid structures, CNN works well for image classification tasks. The model will then be trained using multiple python modules that were preloaded to accomplish these specific tasks on the basis of training and validation data.

We use the testing dataset to evaluate how effectively the model functions after it has been trained using the training dataset. Because the testing dataset was not included in the training dataset, it is completely separate and serves as new data for our model. The MNIST dataset is very efficient and well-balanced, which enhances accuracy and allows us to produce useful findings. After training and testing our model, a finalized trained model is created that is imported and used within the Graphical User Interface(GUI) Python file which is a completely new and independent file.

To execute our application, we implement a mainloop() method in the GUI Python file on the main window.Until the user exits the programme, this function will continue indefinitely, waiting for user actions.

The GUI Python file uses the standard Tkinter Python library to create a canvas. Canvas is a graphic-drawing widget. This canvas widget allows us to draw whatever we want on it. The users can create handwritten digits within this canvas in real-time that will be recognized using the trained model that was generated before.

After that, for performing handwritten digit recognition we import and load the saved model by providing its path.

After the user has created a handwritten digit using the canvas provided within the GUI, the user will have various options displayed in the form of buttons that will provide many functionalities. When the left mouse button is clicked by the user, it usually maps directly to a user action, and the functions assigned to it are triggered.

Button-1 and B1-Motion events will be used in our application. The user's actions, such as key presses and mouse movements, are used to generate events. When the mouse pointer is over a certain widget, the Button-1 event informs the system that the left mouse button has been pressed. B1-Motion indicates that the mouse was moved while holding down the left button.

One of the important features will be the conversion capability, which will allow users to convert recognised decimal numbers to binary, octal, and hexadecimal number systems.

The GUI will also provide “clear” functionality which will enable the user to clear the canvas for further recognition.

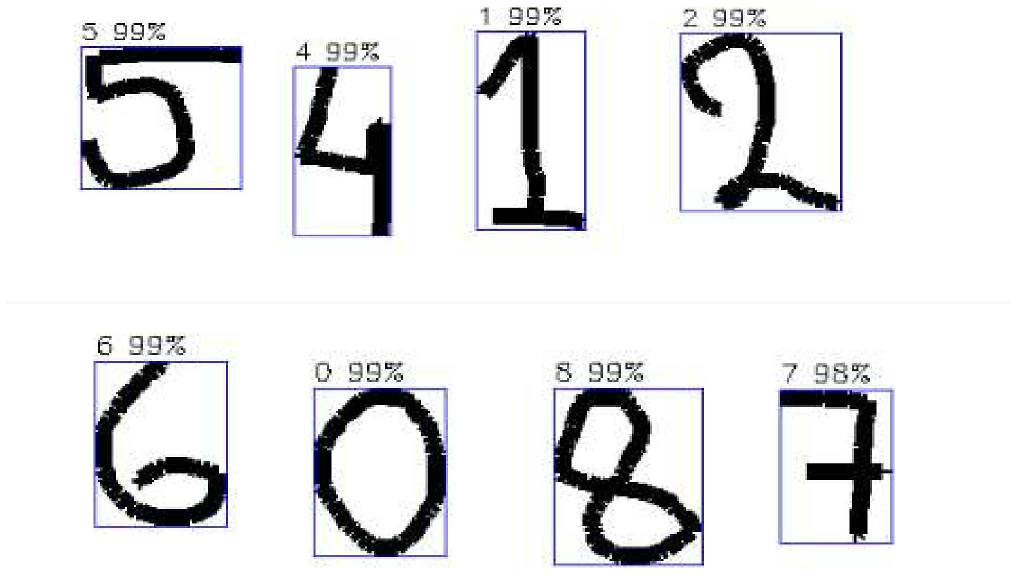


Fig 6.1 - Predicted Labels of Handwritten Digits with Accuracy

Functionalities and modules implemented for our project:

1. Dataset

The suggested model is trained using the MNIST dataset. There are 70,000 digital photos in total that can be used to train and evaluate the model. A certain ratio is used to create these training and testing datasets. The image data is then cleaned and preprocessed in preparation for the next step.

2. Image Preprocessing

Image preprocessing uses numerous approaches such as resizing images, converting them to grayscale format, and augmenting images to properly utilize digital image data within a machine learning model.

3. Training Neural Network

The CNN model, which comprises several convolutional and pooling layers alongside a 3x3 sized kernel, will be built after completing the data preprocessing.

The model will then be trained using multiple python libraries preloaded to perform these specific tasks, such as TensorFlow, Pillow, OpenCV, Tkinter, and Numpy, based on training and validation data.

4. Testing Accuracy of Neural Network

We use the testing dataset to evaluate how effectively the model functions after being trained using the training dataset. The proposed model's accuracy is computed using a subset of the whole MNIST dataset as the testing dataset.

5. User Draws Digit on GUI Canvas

The trained model is then utilized via a Graphical User Interface (GUI) canvas. A user creates digits using the mouse cursor by clicking and dragging the mouse suitably once the model has been evaluated, i.e. trained and tested using the MNIST dataset.

6. Recognize Number/Clear Canvas

Following the user's selection of digits on the GUI canvas, the user is presented with two possibilities-

- Recognize Number: This option uses the CNN model to predict the user's digit string.
- Clear Canvas: This option allows the user to clear the canvas and continue drawing new digits.

7. Convert to Different Number System

Following the digit prediction, the user is presented with three possibilities-

- Convert to Binary: This option transforms a decimal integer into its binary counterpart.
- Convert to Hexadecimal: This option transforms a decimal integer to its hexadecimal representation.
- Convert to Octal: This option transforms a decimal number into its octal counterpart.

CHAPTER 7. RESULT ANALYSIS

The front end design of this project will represent a basic GUI created using the standard interface library of python i.e., Tkinter.

The GUI will consist of 5 buttons alongside a canvas to draw handwritten digits in real-time. These buttons will be bound to particular functions which will be called upon/triggered by clicking on these buttons.

The 5 buttons included within the GUI are as follows:

- Recognize Number: The "Recognize Number" button will be used after the user has drawn handwritten digits on the canvas to use the trained machine learning model and predict the digits. These predicted digits will then be used to implement further functionalities.
- Clear Canvas: The "Clear Canvas" button will be used to clear the canvas for further handwritten digit recognitions.
- Convert to Binary: The "Convert to Binary" button will be used after the currently drawn digits are recognized as a decimal number i.e., base '10'. It will convert the decimal number into a binary number i.e., base '2'.
- Convert to Octal: The "Convert to Octal" button will be used after the currently drawn digits are recognized as a decimal number i.e., base '10'. It will convert the decimal number into an octal number i.e., base '8'.
- Convert to Hexadecimal: The "Convert to Hexadecimal" button will be used after the currently drawn digits are recognized as a decimal number i.e., base '10'. It will convert the decimal number into a hexadecimal number i.e., base '16'.

In the backend, a trained machine learning model is created using the Convolutional Neural Network (CNN) algorithm which uses additional neuron layers for feature extraction and in turn provides improved accuracy.

The MNIST dataset being a very vast dataset for image processing provides very good accuracy for the proposed model as well.

7.1 FIRST SECTION

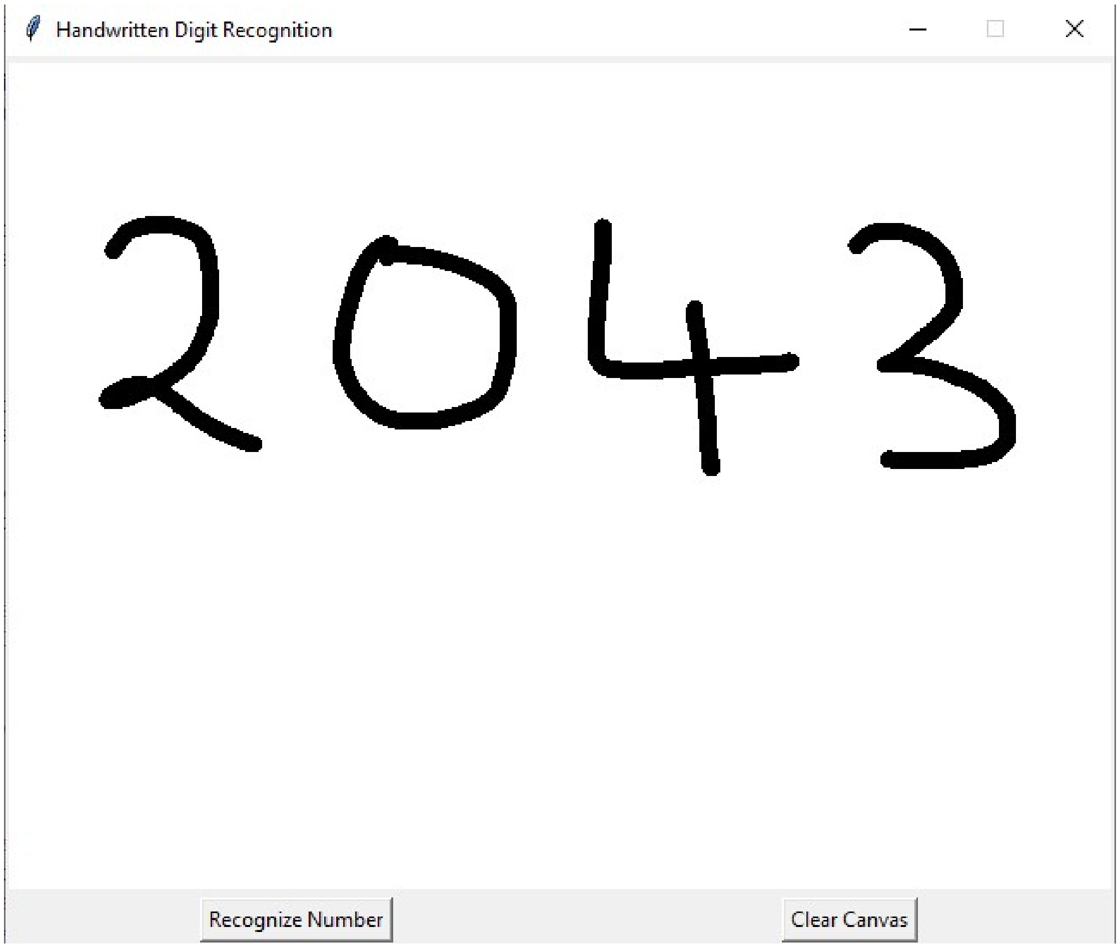


Fig 7.1 - First Section of the GUI

The above image shows the first window/section of our GUI which consists of three parts:

- The canvas where the user draws handwritten digits for recognition.
- The "Recognize Number" button which uses the machine learning model to predict the number, output of which will be displayed on the next section/window.
- The "Clear Canvas" button which clears everything drawn on the canvas as soon as it is clicked by the user.

7.2 SECOND SECTION

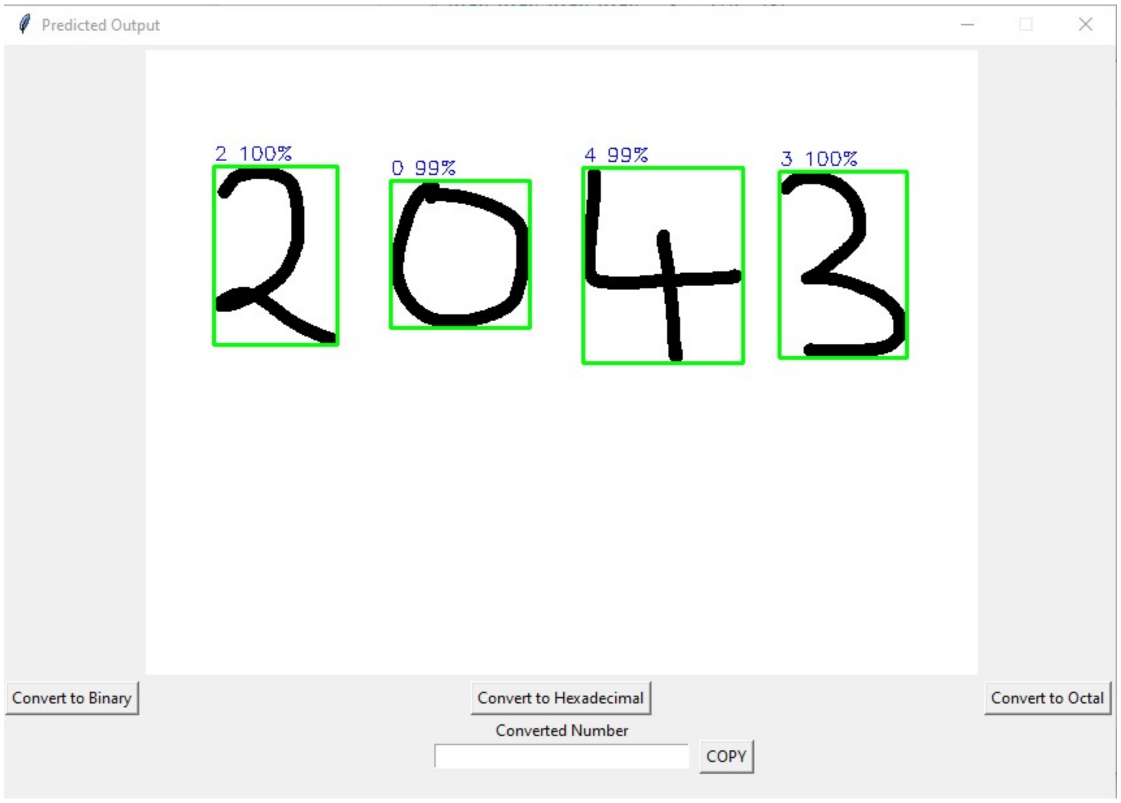
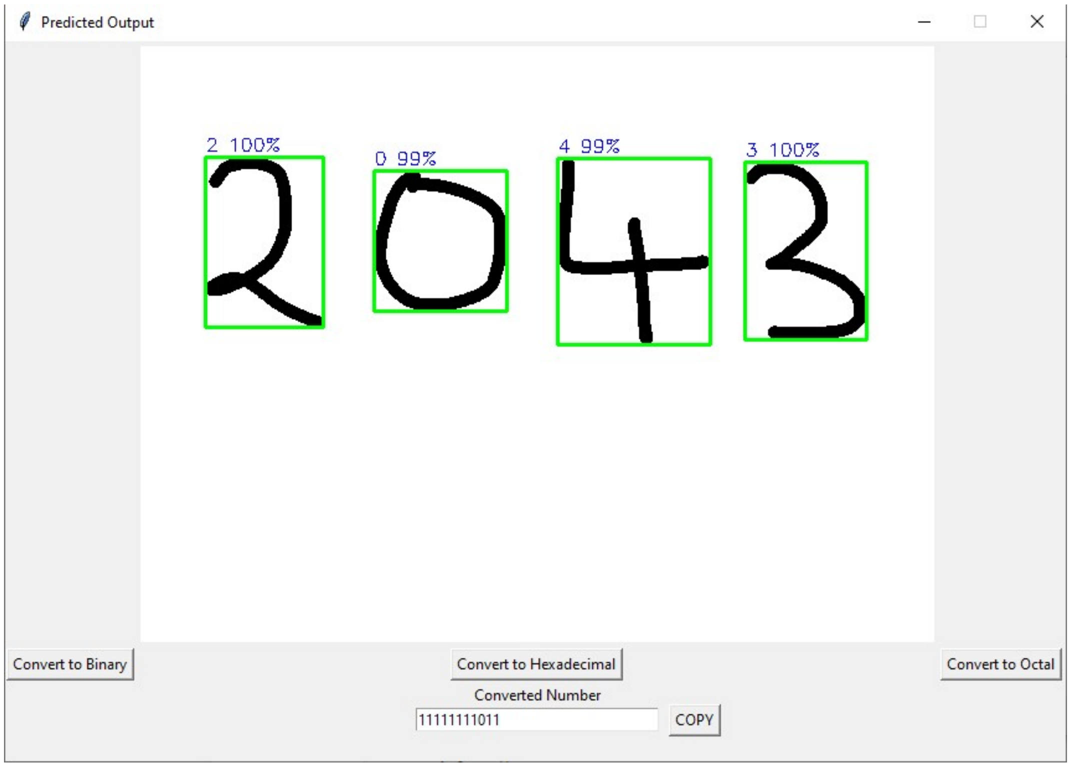


Fig 7.2 - Second Section of the GUI

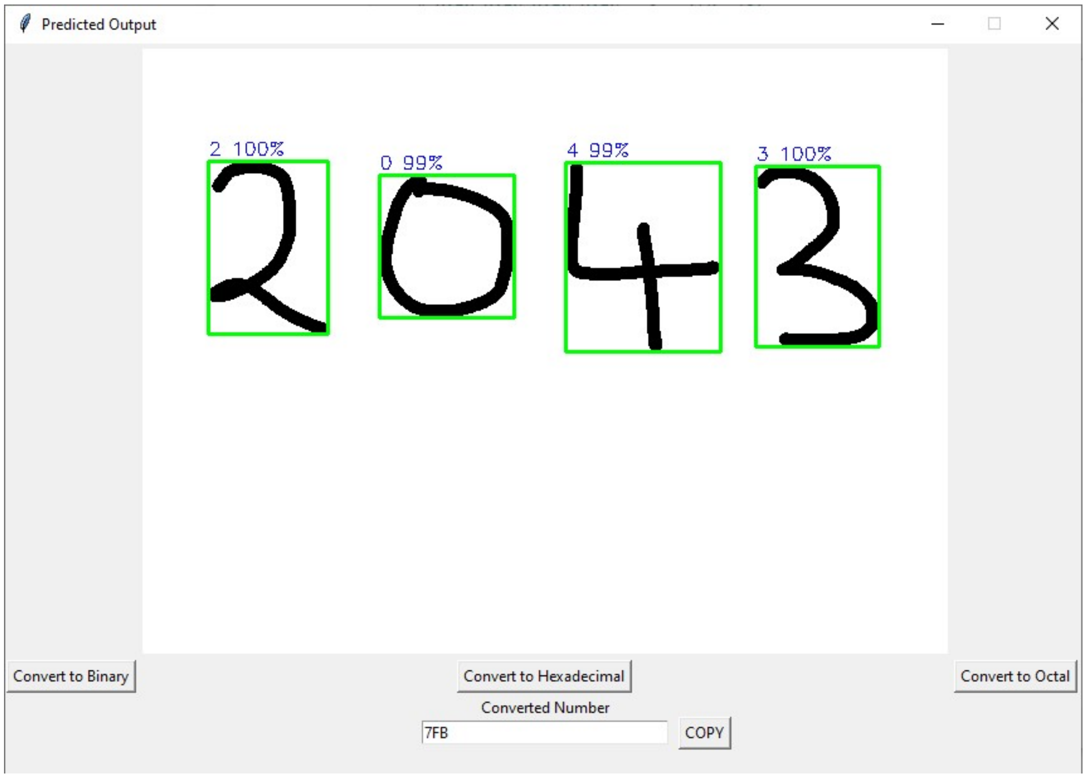
The above image shows the second window/section of our GUI which consists of three parts:

- The predicted number that was drawn on the canvas displaying the digit label alongside the accuracy separately which is then combined to obtain the complete number.
- The three conversion buttons provide the functionality of converting a recognised decimal number to a binary, hexadecimal, or octal number system based on the user's preference.
- The converted number text box which displays the converted number as the output when the user decides to implement the conversion functionality by clicking on one of the provided buttons.



11111111011

Fig 7.3 - Second Section of the GUI with Binary Form Result



7FB

Fig 7.4 - Second Section of the GUI with Hexadecimal Form Result

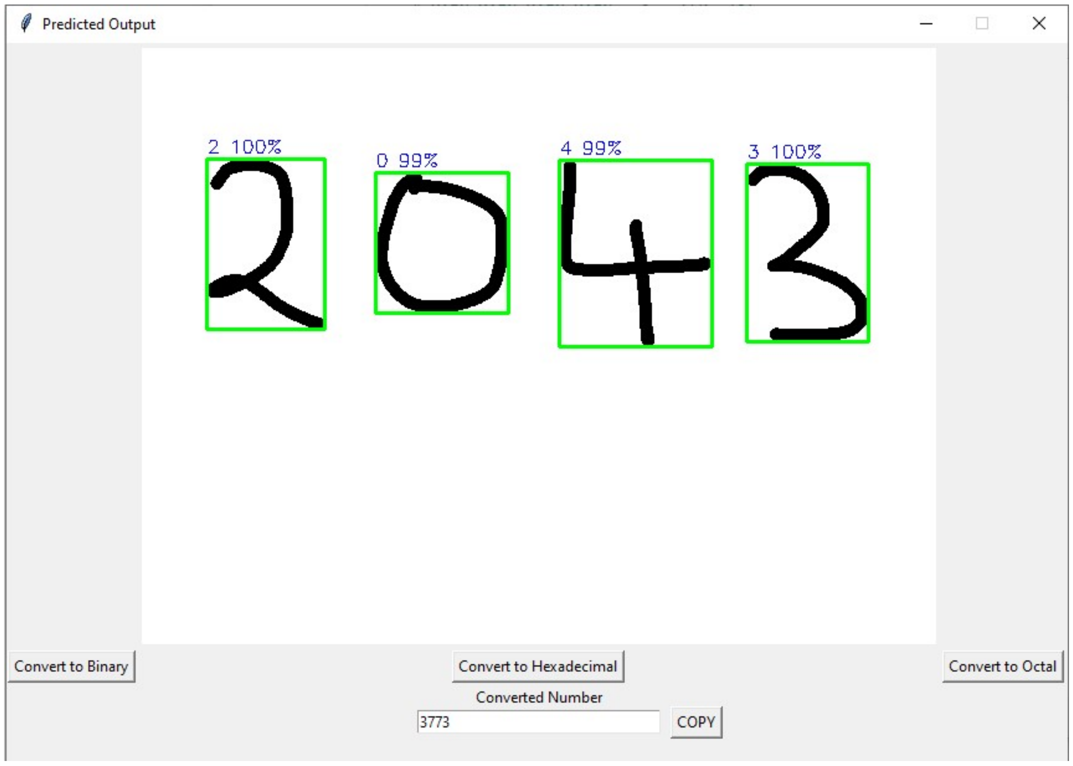
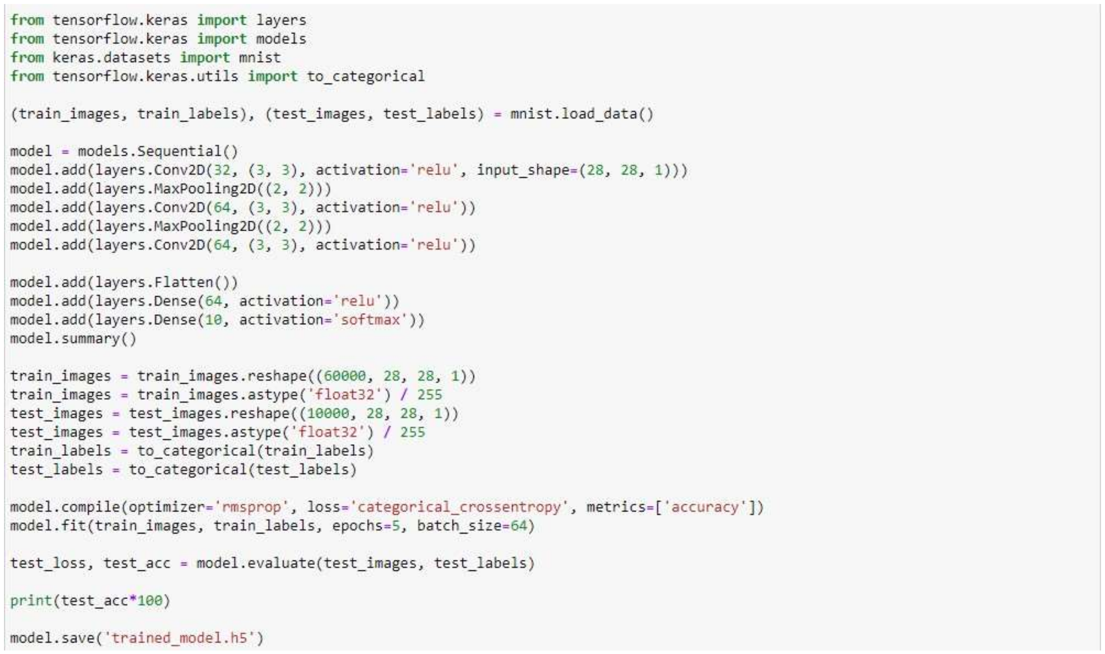


Fig 7.5 - Second Section of the GUI with Octal Form Result

Figures 7.3, 7.4, and 7.5 are the three photos shown above in which the predicted number converted to its binary equivalent, hexadecimal equivalent, and octal equivalent when the "Convert to Binary", "Convert to Hexadecimal" and the "Convert to Octal" buttons are clicked by the user respectively.

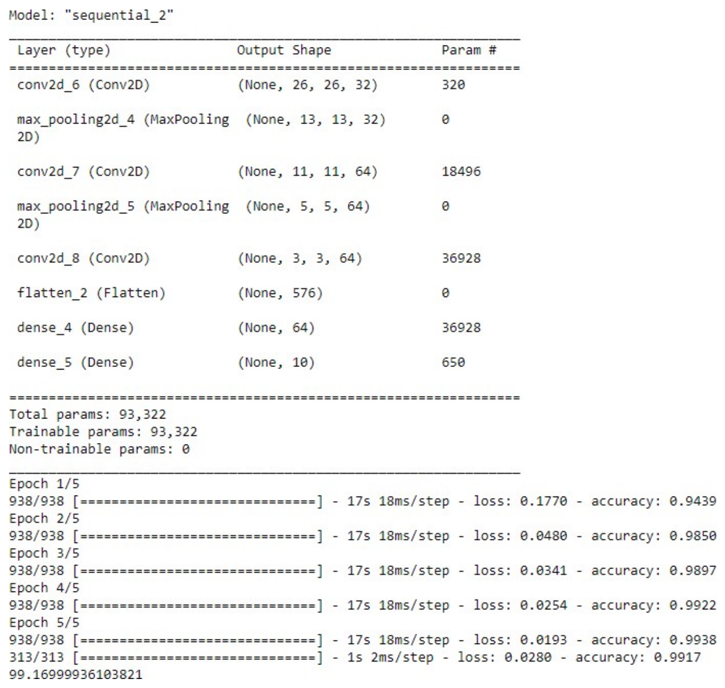
The second window/section also consists of an additional "Copy" button functionality which allows the user to copy the converted result for further use.

7.3 SOURCE CODE



```python
from tensorflow.keras import layers
from tensorflow.keras import models
from keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()

train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)

test_loss, test_acc = model.evaluate(test_images, test_labels)

print(test_acc*100)

model.save('trained_model.h5')
```

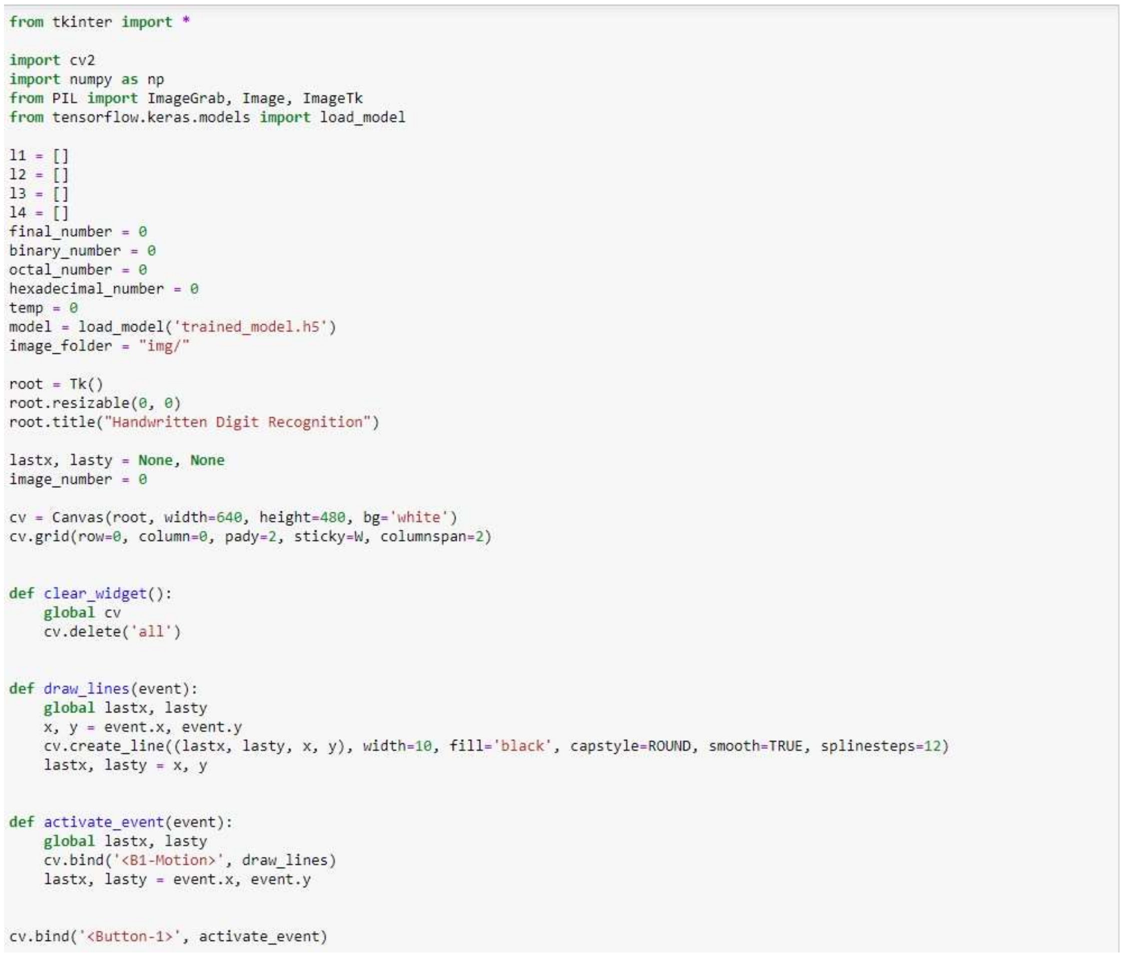
Fig 7.6 - Python Code for the “train_digit_recognizer” File



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_4 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 conv2d_8 (Conv2D)           (None, 3, 3, 64)          36928

 flatten_2 (Flatten)         (None, 576)               0

 dense_4 (Dense)             (None, 64)                36928

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 93,322
Trainable params: 93,322
Non-trainable params: 0
_________________________________________________________________
Epoch 1/5
938/938 [==============================] - 17s 18ms/step - loss: 0.1770 - accuracy: 0.9439
Epoch 2/5
938/938 [==============================] - 17s 18ms/step - loss: 0.0480 - accuracy: 0.9850
Epoch 3/5
938/938 [==============================] - 17s 18ms/step - loss: 0.0341 - accuracy: 0.9897
Epoch 4/5
938/938 [==============================] - 17s 18ms/step - loss: 0.0254 - accuracy: 0.9922
Epoch 5/5
938/938 [==============================] - 17s 18ms/step - loss: 0.0193 - accuracy: 0.9938
313/313 [==============================] - 1s 2ms/step - loss: 0.0280 - accuracy: 0.9917
99.16999936103821
```

Fig 7.7 - Resulting Output on Execution of “train_digit_recognizer” File

The two images above display the source code and the output of the “train_digit_recognizer” file which will be executed to create the required CNN based machine learning model which will be used by the GUI python file as shown in the following images.



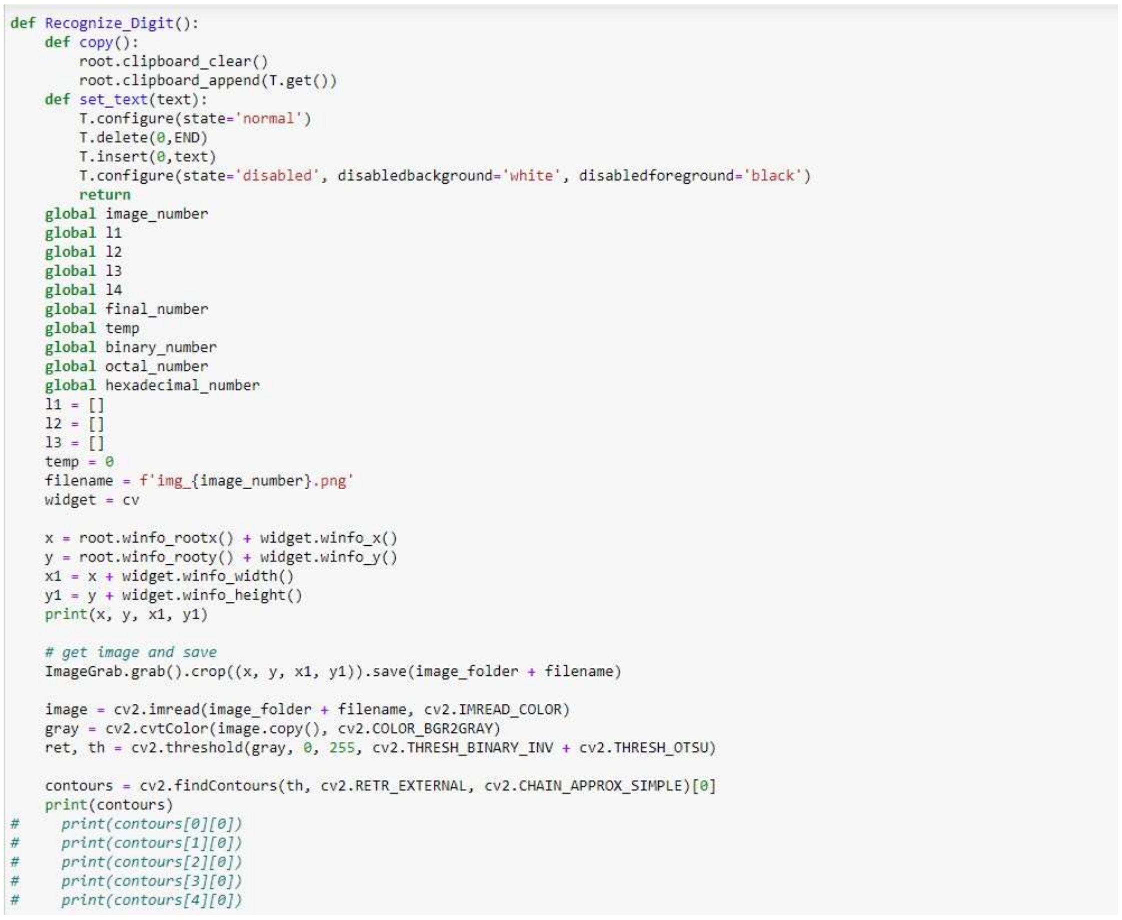
```
from tkinter import *

import cv2
import numpy as np
from PIL import ImageGrab, Image, ImageTk
from tensorflow.keras.models import load_model

l1 = []
l2 = []
l3 = []
l4 = []
final_number = 0
binary_number = 0
octal_number = 0
hexadecimal_number = 0
temp = 0
model = load_model('trained_model.h5')
image_folder = "img/"

root = Tk()
root.resizable(0, 0)
root.title("Handwritten Digit Recognition")

lastx, lasty = None, None
image_number = 0

cv = Canvas(root, width=640, height=480, bg='white')
cv.grid(row=0, column=0, pady=2, sticky=W, columnspan=2)


def clear_widget():
    global cv
    cv.delete('all')


def draw_lines(event):
    global lastx, lasty
    x, y = event.x, event.y
    cv.create_line((lastx, lasty, x, y), width=10, fill='black', capstyle=ROUND, smooth=TRUE, splinesteps=12)
    lastx, lasty = x, y


def activate_event(event):
    global lastx, lasty
    cv.bind('<B1-Motion>', draw_lines)
    lastx, lasty = event.x, event.y


cv.bind('<Button-1>', activate_event)
```

Fig 7.8 - Python Code of GUI file (Part 1)

The above image displays the libraries that are imported in the GUI file alongside the source code of the clear button functionality by the name of “clear_widget”.

The draw_lines function is the source code which describes the formatting of the lines such as the width, color, smoothness, etc. that will be used to draw the digits on the canvas by the user.

The activate_event function describes how the mouse cursor will be activated to draw the digits on the editable canvas as the user clicks the left mouse button i.e., B-1 and also how it is bound to the draw_lines function.



```
def Recognize_Digit():
    def copy():
        root.clipboard_clear()
        root.clipboard_append(T.get())
    def set_text(text):
        T.configure(state='normal')
        T.delete(0,END)
        T.insert(0,text)
        T.configure(state='disabled', disabledbackground='white', disabledforeground='black')
        return
    global image_number
    global l1
    global l2
    global l3
    global l4
    global final_number
    global temp
    global binary_number
    global octal_number
    global hexadecimal_number
    l1 = []
    l2 = []
    l3 = []
    temp = 0
    filename = f'img_{image_number}.png'
    widget = cv

    x = root.winfo_rootx() + widget.winfo_x()
    y = root.winfo_rooty() + widget.winfo_y()
    x1 = x + widget.winfo_width()
    y1 = y + widget.winfo_height()
    print(x, y, x1, y1)

    # get image and save
    ImageGrab.grab().crop((x, y, x1, y1)).save(image_folder + filename)

    image = cv2.imread(image_folder + filename, cv2.IMREAD_COLOR)
    gray = cv2.cvtColor(image.copy(), cv2.COLOR_BGR2GRAY)
    ret, th = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

    contours = cv2.findContours(th, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[0]
    print(contours)
#     print(contours[0][0])
#     print(contours[1][0])
#     print(contours[2][0])
#     print(contours[3][0])
#     print(contours[4][0])
```

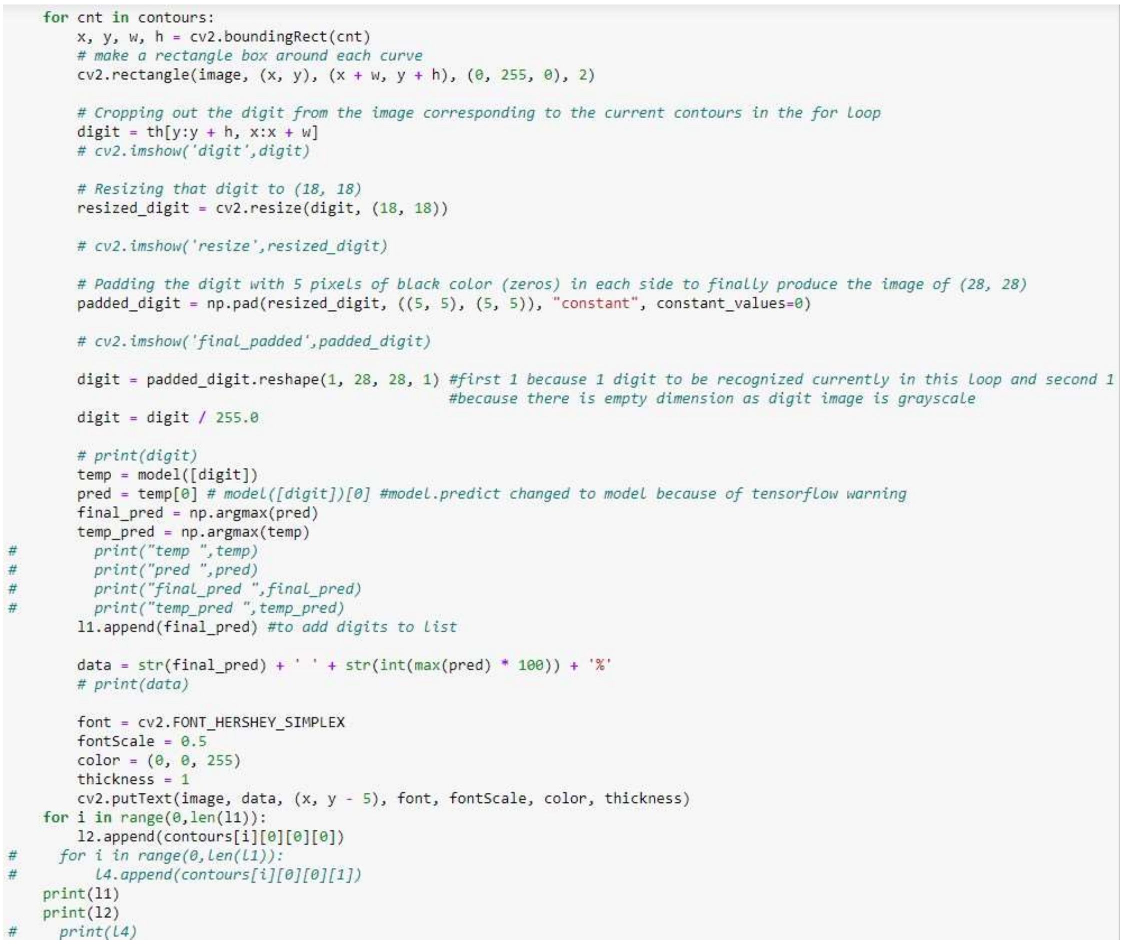
Fig 7.9 - Python Code of GUI File (Part 2)

The above image i.e., Fig 7.9 shows the beginning of the Recognize_Digit function which is the backbone of this project. This function constructs the canvas(based on the dimensions specified within the function code) on which the user draws handwritten digits for recognition using the proposed machine learning model.

This function also describes how a screenshot of the canvas on which the digits are drawn is taken for applying the recognition process on it accordingly.

The screenshot taken is then saved in a specific folder and the image data is processed after it is converted to its grayscale equivalent format.

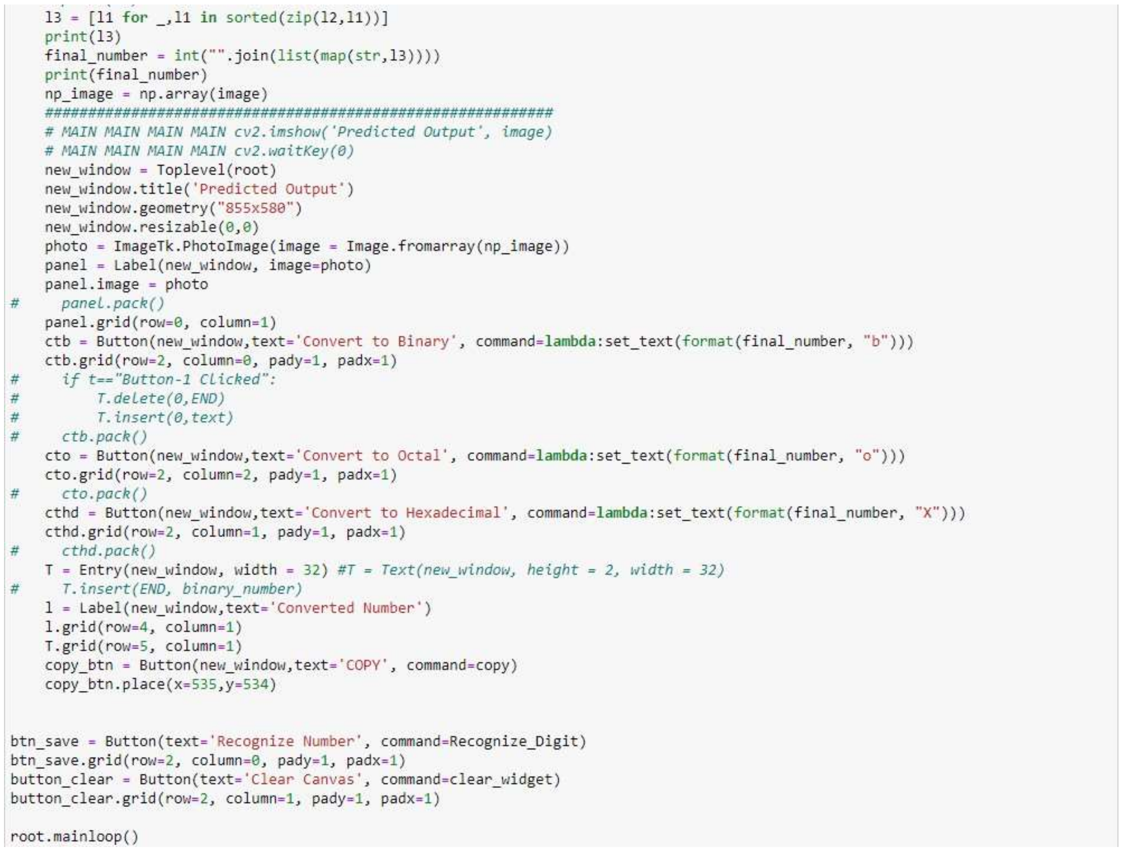
After that contours or color coded regions are found out from the image data to help us find only the relevant image data that will be required to recognize the shape of the digit properly and exclude the blank space that is present in the canvas which will be of no use in the process of handwritten digit recognition.



```
    for cnt in contours:
        x, y, w, h = cv2.boundingRect(cnt)
        # make a rectangle box around each curve
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Cropping out the digit from the image corresponding to the current contours in the for loop
        digit = th[y:y + h, x:x + w]
        # cv2.imshow('digit',digit)

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

        # cv2.imshow('resize',resized_digit)

        # 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)

        # cv2.imshow('final_padded',padded_digit)

        digit = padded_digit.reshape(1, 28, 28, 1) #first 1 because 1 digit to be recognized currently in this loop and second 1
                                                   #because there is empty dimension as digit image is grayscale
        digit = digit / 255.0

        # print(digit)
        temp = model([digit])
        pred = temp[0] # model([digit])[0] #model.predict changed to model because of tensorflow warning
        final_pred = np.argmax(pred)
        temp_pred = np.argmax(temp)
#         print("temp ",temp)
#         print("pred ",pred)
#         print("final_pred ",final_pred)
#         print("temp_pred ",temp_pred)
        l1.append(final_pred) #to add digits to list

        data = str(final_pred) + ' ' + str(int(max(pred) * 100)) + '%'
        # print(data)

        font = cv2.FONT_HERSHEY_SIMPLEX
        fontScale = 0.5
        color = (0, 0, 255)
        thickness = 1
        cv2.putText(image, data, (x, y - 5), font, fontScale, color, thickness)
    for i in range(0,len(l1)):
        l2.append(contours[i][0][0][0])
#     for i in range(0,len(l1)):
#         l4.append(contours[i][0][0][1])
    print(l1)
    print(l2)
#     print(l4)
```

Fig 7.10 - Python Code of GUI File (Part 3)

The above image i.e., 7.10 shows how the code loops through the obtained contours and recognizes the digit by creating a bounding rectangle around the digit and converting the image data into its binary matrix format and using this matrix of intensity denoted in the form 1’s and 0’s to recognize the shape of the digit properly.



```
    l3 = [l1 for _,l1 in sorted(zip(l2,l1))]
    print(l3)
    final_number = int("".join(list(map(str,l3))))
    print(final_number)
    np_image = np.array(image)
    ##############################################################
    # MAIN MAIN MAIN cv2.imshow('Predicted Output', image)
    # MAIN MAIN MAIN MAIN cv2.waitKey(0)
    new_window = Toplevel(root)
    new_window.title('Predicted Output')
    new_window.geometry("855x580")
    new_window.resizable(0,0)
    photo = ImageTk.PhotoImage(image = Image.fromarray(np_image))
    panel = Label(new_window, image=photo)
    panel.image = photo
#     panel.pack()
    panel.grid(row=0, column=1)
    ctb = Button(new_window,text='Convert to Binary', command=lambda:set_text(format(final_number, "b")))
    ctb.grid(row=2, column=0, pady=1, padx=1)
#     if t=="Button-1 Clicked":
#         T.delete(0,END)
#         T.insert(0,text)
#     ctb.pack()
    cto = Button(new_window,text='Convert to Octal', command=lambda:set_text(format(final_number, "o")))
    cto.grid(row=2, column=2, pady=1, padx=1)
#     cto.pack()
    cthd = Button(new_window,text='Convert to Hexadecimal', command=lambda:set_text(format(final_number, "X")))
    cthd.grid(row=2, column=1, pady=1, padx=1)
#     cthd.pack()
    T = Entry(new_window, width = 32) #T = Text(new_window, height = 2, width = 32)
#     T.insert(END, binary_number)
    l = Label(new_window,text='Converted Number')
    l.grid(row=4, column=1)
    T.grid(row=5, column=1)
    copy_btn = Button(new_window,text='COPY', command=copy)
    copy_btn.place(x=535,y=534)


btn_save = Button(text='Recognize Number', command=Recognize_Digit)
btn_save.grid(row=2, column=0, pady=1, padx=1)
button_clear = Button(text='Clear Canvas', command=clear_widget)
button_clear.grid(row=2, column=1, pady=1, padx=1)

root.mainloop()
```

Fig 7.11 - Python Code of GUI File (Part 4)

The above image i.e., Fig 7.11 shows how a new window will be created when the user clicks on the “Recognize Number” button. This new window is section two of the GUI which is already explained above in the project report.

This new window will consist of the “Convert to Binary”, “Convert to Hexadecimal”, and “Convert to Octal” buttons and a non-editable text box which will be used to display the result according to the user’s choice alongside which will be an additional “COPY” button that can be used by the user to copy the output which is converted from the decimal format to its binary/hexadecimal/octal equivalent.

The source code explains how these buttons are implemented within the GUI and also shows the coordinates where these buttons are placed according to the size of the new window.

CHAPTER 8. CONCLUSION, LIMITATIONS, AND FUTURE SCOPE

8.1 CONCLUSION

Handwritten digit recognition utilizing machine learning has been implemented in this project report. This problem was taken up so as to create a Convolutional Neural Network to recognize handwritten digits with great accuracy so that it can be used in various technical/non-technical fields but mainly the banking industry. On the MNIST dataset, one of the most widely used machine learning methods, Convolutional Neural Network (CNN), was utilized for training and testing the needed machine learning model. This project achieved a high level of accuracy by utilizing this deep learning technique. This was accomplished by correctly identifying the numerals drawn at various angles and producing the appropriate results as a result of this identification. This is achieved through the following steps: analyzing the handwritten digits drawn on the canvas, displaying the result in a new window with each of the digits' predicted label and the corresponding accuracy, and then giving the options to the user to convert the recognized number into its binary, hexadecimal or octal equivalent. In this project, we try to achieve a high level of accuracy so that the proposed machine learning model can be used in the banking industry for small transactions as these require a lot of manpower which can be saved by using this technology. Even though the implementation of this technology might come with its flaws, the amount of manpower it can save can make the risks proposed by this technology (technical flaws which can cause monetary loss) feasible.

8.2 LIMITATIONS

The fundamental restriction of handwritten digit recognition is that handwriting styles vary, which is a very personal behavior in which there are many different models for numbers based on angles, segment lengths, stress on certain sections of numbers, and so on. This could be a disadvantage because it reduces overall accuracy. Using this technique (handwritten digit recognition) on a big scale could have a number of negative consequences. If a handwritten digit recognition system is not employed with a good intention, it can exacerbate a variety of social problems. People could use such technology to figure out bank pins, ATM pins, and other monetary crimes. On a very large scale if there is great reliance on the algorithm, it may lead to some sort of discrepancy and may also be used to breach classified information in various fields/sectors. During recognition accuracy plays an important role but if the accuracy is too low then there might be some risk in using the proposed machine learning model.

8.3 FUTURE SCOPE

As the world progresses towards automation, [7][8] the use of handwritten digit recognition can have a great impact in the future. It can be used to process checks in banks, sort mail, enter data into forms, verify signatures, understand addresses, analyze and verify documents, and so on. Furthermore, it can be improved to automate the process of scanning digits/text information to help support many complex real-time applications. It can also be used to implement bigger and more complex problems by using it abstractly. It may also be used to comprehend many machine learning algorithms such as [13] CNN, k-NN, Naive Bayes, SVM, and others. It may also be used in electronic form filling. Immediate and direct conversion of handwritten data to printed data can take place which would not only increase productivity, but also reduce overhead cost. This is only possible if handwriting recognition is standardized and improved so that it can be used on any computing device. It can also be used in biometric and forensic and It is well suited for author identification used in forensics and biometrics. Thus, one application of handwriting recognition is resolving disputes over ancient manuscripts. Here, the actual author of the manuscript is identified based on certain characteristics of the author's handwriting. This will help you avoid making false claims about your handwriting or manuscript. It can also be helpful in digitization of palm leaf manuscripts.

REFERENCES

[1] Isha Jaggi, Anchit Shrivastava, Gupta Deepali, "Handwritten Digit Recognition Using Machine Learning", Conference on Power Energy, Environment and Intelligent Control, (2019), unpublished.

[2] Rohan SethiIla, KaushikIla Kaushik, "Hand Written Digit Recognition using Machine Learning", International Conference on Communication Systems and Network Technologies, (2020), pp. 2329-7182.

[3] P. S. Nair, Assegie, T. A., "Handwritten digits recognition with decision tree classification: a machine learning approach," International Journal of Electrical and Computer Engineering, (2019), pp.446-4451.

[4] Rabia KARAKAYA, Serap KAZAN, "Handwritten Digit Recognition Using Machine Learning", Journal of Computer Science and Technology, vol. 25, Issue 1, (2021), pp.65-71.

[5] Pranit Patil, Bhupinder Kaur, "Handwritten Digit Recognition using various Machine Learning Algorithms and Models", International Journal of Innovative Research in Computer Science & Technology, Vol. 8, Issue 4, (2020), ISSN: 2347-5552.

[6] Mayank Jain, Gagandeep Kaur, Muhammad Parvez Quamar, Et al., "Handwritten Digit Recognition Using CNN", International Conference on Innovative Practices in Technology and Management, (2021), unpublished.

[7] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, Hiromichi Fujisawa, "Handwritten digit recognition: Benchmarking of state-of-the-art techniques", Journal of Computer Science and Technology, Vol. 36, Issue 10, (2003), pp.2271-2285.

[8] Priya, Rajendra Singh, Dr.Soni Changlani, "Handwritten Digit Recognition By Proximal SVM", Journal of Emerging Technologies and Innovative Research, Vol-4, Issue-4, (2017), ISSN-2349-5162.

[9] Stephen R. Garner, "WEKA: The Waikato Environment for Knowledge Analysis", Kings College of Engineering, Punalkulam, Pudukottai, Department of Computer Science: University of Waikato, (2021).

[10] P. Ghadekar, S. Ingole and D. Sonone, "Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM Classifier", International Conference on Computing Communication Control and Automation, (2018), pp. 1-6.

[11] Chandrajit Choudhury, Dibyakanti Mahapatra, Ram Kumar Karsh, "Generator Based Methods for Off-line Handwritten Character Recognition", Advanced Communication Technologies and Signal Processing, (2020), pp.1-6.

[12] Aun Irtaza, Ali Javed, Saleh Albahli, “An improved faster-RCNN model for handwritten character recognition”, Journal of Computer Science & Technology, (2020), Vol. 8, Issue- 4, ISSN: 2347-5552.

[13] Viragkumar N. Jagtap , Shailendra K. Mishra, “Fast Efficient Artificial Neural Network for Handwritten Digit Recognition”, International Journal of Computer Science and Information Technologies , Vol. 5, Issue 2, (2014), ISSN: 0975-9646, pp. 2302-2306.

[14] Samay Pashine, Ritik Dixit, Rishika Kushwah, “Handwritten Digit Recognition using Machine and Deep Learning Algorithms”, International Journal of Computer Applications, Vol-176, June (2021).

[15] Aarti Gupta, Rohit Miri, and Hiral Raja, “Recognition of Automated Hand-written Digits on Document Images Making Use of Machine Learning Techniques”, European Journal of Engineering and Technology Research, Vol. 6 Issue-4, May (2021), ISSN: 2736-576X.

APPENDIX A:

CONTRIBUTION OF PROJECT

A-1. Objective and Relevance of Project

One of the most essential and practical difficulties in pattern recognition applications is handwritten digit recognition. HDR is the process of identifying and classifying user-defined handwritten digits into pre-defined digit class labels, such as (0-9). It provides for numerous important uses, including bank check processing, mail sorting, and data entering in forms.

Because of its relevance in allowing one to understand the worldwide renowned machine learning algorithms, the rapid use of this technology can help solve a lot of real-world complex problems in real time.

As a result, the goal of this project is to develop a Handwritten Digit Recognition system with a GUI that is accurate enough to predict handwritten digits produced in real-time on a canvas widget within the GUI with accuracy and robustness.

A-2. Outcome

The outcome of this project is a GUI application created using the standard Tkinter python library. The GUI consists of a canvas widget to draw handwritten digits in real-time and several button widgets that are bound to various functionalities that are triggered according to user events.

The machine learning model will be using the MNIST dataset and data will be preprocessed before being fed into the model to provide better accuracy.

A-2.1. Research Paper

Handwritten Digit Recognition System Using Machine Learning

Ujjwal Jindal*,Vikas Panwar†,Ujjwal Gupta‡, Gaurav Parashar§
Department of Computer Science and Engineering
KIET Group of Institutions
Ghaziabad, India
Email: *ujjwal.1923cs1059@kiet.edu,
†vikas.1923cs1013@kiet.edu, ‡ujjwal.1923cs1179@kiet.edu, §gauravparashar24@gmail.com (ORCID: 0000-0003-4869-1819)

***Abstract*—This research study proposes a model for a handwritten digit identification system based on machine learning which could be used to recognize and identify digits written by a user on a canvas (editable) widget inside a graphical user interface (GUI). The certainty of several machine learning algorithms which can be used to develop such systems is compared in this research study in conjunction with a complete discussion of handwritten digit recognition systems, the approach used by us to implement them, and their applications.**

***Index Terms*—handwritten digit recognition, GUI, machine learning algorithm, handwritten digit recognition system, accuracy, convolutional neural network, support vector machine.**

I. Introduction

As a very practical technique, handwritten digit recognition is one of the most crucial problems in machine learning that should be solved. It also plays an important role in knowing about the various fields that use pattern recognition. Applications for handwritten digit recognition include sorting mail, processing checks in banks, entering data onto forms, and more. Machine learning is one of the most essential ideas to be highlighted in order to meet the constantly expanding everyday demands related to the IT business. We set out to construct a handwritten digit recognition system using machine learning to grasp machine learning better and exercise it in problem-solving. This problem provides a number of use cases that could be advantageous for people or organizations, and it will allow us to study machine learning procedures from the basics. Machine learning includes a variety of learning model types. These are listed below: A. Supervised Learning Supervised learning consists of various algorithms some of which used are as Naive Bayes, Random Forest, K-Nearest Neighbors, Decision Trees, Support Vector Machine, and Linear Regression. In the supervised learning approach, a model is trained with the help of a labeled dataset which consists of two variables, referred to as ”input” and ”output,” that are mapped to one another. B. Unsupervised Learning Unsupervised learning consists of Numerous techniques some of which are Neural Networks, K- Mean Clustering, Multivariate Analysis, and Anomaly Detection. When using an unsupervised learning strategy, the model is trained with the help of an unlabeled dataset, whether it is classified or not. During the learning process, input is not transferred to output; rather, the input dataset which is to be trained is sorted into groups, which helps to forecast the result of the testing dataset. C. Reinforcement Learning Different algorithms which come under reinforcement learning include Q Learning, Negative, Positive, and Markov Decision Process. Reinforcement learning uses Sequential decision-making. Based on collective incentives, an intelligent agent manipulates its surroundings in an effort to maximize these rewards.

II. MNIST DATASET

Yann LeCun, Corinna Cortes, and Christopher Burges created the modified National Institute of Standards and Technology dataset or MNIST dataset. Various digits which are scanned were standardized in size and justified as centered, which made it an exemplary database helping in the evaluation of these models. This made it particularly prominent among academics for building handwritten digit recognition systems utilizing machine learning techniques. The error rate can be greatly decreased by utilizing a variety of classifiers for distinct algorithms and parameters. The MNIST dataset consists of training examples that include a variety of digits that are handwritten. It includes 70,000 images, out of which 60,000 are used to train the model and 10,000 are used in the testing of the model. Both datasets contain snaps of the 10 digits, starting from 0 to 9, that have been accurately classified. The representation of handwritten numbers is a 28*28 grayscale picture. Each MNIST data point consists of two components: an image of the handwritten digit and a target label pertaining to it. When using the MNIST dataset, very little data cleaning is necessary, allowing one to concentrate completely on the goal of their deep learning model or machine learning model.

III. Literature Survey

The handwritten digits are preprocessed in several standard databases, including segmentation and normalisation, so that researchers can compare the recognition outcomes of their techniques on an equal footing, according to Li Deng [1],

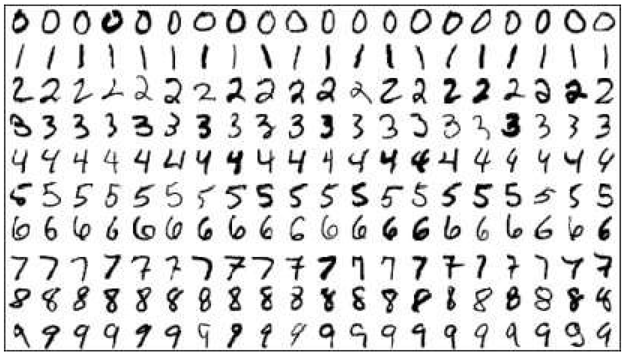


Fig. 1. Dataset shows the different designs of handwritten labels

who claimed that Handwritten Digit Recognition is one of the significant problems in the recognition of an optical character. He came to the conclusion that the MNIST database offers a relatively straightforward and static classification assignment for academics and students to investigate pattern recognition and machine learning since it eliminates the need for pointless data pretreatment and formatting.

By fitting generative models created using deformable B-splines with Gaussian ”ink generators” that are spaced along the length of the spline, M. Revow, C.K.I. Williams, and G.E. Hinton [2] investigated a technique for recognising handwritten digits. They later came to the conclusion that while this method requires high computation, it may be used as a verification stage for faster recognizers to achieve improved performance because there will be little correlation between Additionally, they have shown how generative models can be used to extract additional information from images that is useful for model-driven segmentation.

By making a number of changes to the previously described neural network model neocognitron, Kunihiko Fukushima [3] proposed an improved version and verified its effectiveness by employing a sizable database of handwritten digits (ETL1). The self-organization of line-extracting cells, the highest level of supervised competitive learning, the contrast-extracting layer followed by edge-extracting cells, the inhibitory surround in the connections between S- and C-cells, etc. are a few examples of the many alterations. Later, He came to the conclusion that neocognitron is a simpler network than other ANN. Additionally, the number of repeated presentations of a training set required by neocognitron is much lower than for the network trained by backpropagation, and thanks to the modifications proposed, the accessory circuits present in the earlier versions can be removed.

In order to improve the performance of the existing object and shape recognition model, Mandana Hamidi and Ali Borji [4] suggested adding more biologically inspired attributes as lateral inhibition, feature localisation, and feature sparsification. Later, they came to the conclusion that the modified model performs better than the original model at recognising English and Farsi handwritten digit datasets, and that it can also be used to identify individuals using their palm, iris, or fingerprint biometrics because these features have intricate structural details.

Using the spike-triggered Normalised Approximate Descent (NormAD) technique, Shruti R. Kulkarni and Bipin Rajendran [5] have demonstrated supervised learning in Spiking Neural Networks (SNNs) for the problem of handwritten digit identification. They have ended their effort by offering numerous experimental insights for the improvement of learning parameters and network setup. According to their experiments, even with 3-bit synaptic weights, the developed SNN’s classification accuracy does not decrease by more than 1 percent in comparison to the floating-point baseline. With four times fewer parameters than the state-of-the-art network, their network—which uses neurons operating at sparse biological spike rates below 300 Hz—achieved a classification accuracy of 98.17 percent on the MNIST test database.

An adaptive deep Q-learning technique for handwritten digit recognition was put forth by Junfei Qiao [6]. The suggested technique, known as Q-ADBN (Q-learning Adaptive Deep Belief Network), combines the decision-making abilities of reinforcement learning with deep learning’s capacity to extract features. The Q-ADBN initially utilises an adaptive deep auto-encoder (ADAE) to extract features from the input images before utilising the Q-learning method to determine the final recognition choice by maximising the Q-function. According to the experimental findings, the proposed Q-ADBN works better in terms of recognition accuracy and processing speed than existing approaches of a similar kind.

My Donell [7] discusses a study that demonstrated shallow non-convolutional neural networks trained using the ”Extreme Learning Machine” (ELM) technique may achieve error rates below 1 percent on the MNIST handwritten digit benchmark. On the NORB image database, it was demonstrated that the ELM approach could achieve less than 5.5 percent error rates. The conventional ELM algorithm has been improved by the authors in a number of ways that can greatly boost speed. The study also discovered that the ELM technique is suitable for many practical machine learning applications since it is simple to use and accurate when creating a single-hidden-layer neural network classifier. The study’s findings indicate that the use of deep convolutional networks may involve confirmation bias, as opposed to straightforward single-layer feedforward.

An automated process for producing features for handwritten digit recognition is described by Gader, P.D.[8]. The technique directs the search for features using the two evaluation metrics orthogonality and information. High classification rates are achieved by using the generated features in a neural network that has been trained using backpropagation. On a test set of 1000 digits per class, the classifier is combined with several other high-performance classifiers to achieve recognition rates of about 98 percent.

A deep convolutional extreme learning machine (DC-ELM) technique for image classification problems is presented by Pang, Shan, and Xinyi Yang [9]. It combines the benefits of an extreme learning machine (ELM) and a convolutional neural network (CNN) to improve generalisation performance and speed up training. To extract high-level characteristics from raw input images, the approach uses a number of different convolution and pooling layers. These features are then given to an ELM classifier. In order to decrease feature dimensionality and conserve computational resources, the DC-ELM includes stochastic pooling. The DC-ELM performs better than existing ELM methods and cutting-edge deep learning approaches in tests on handwritten digit recognition tasks in terms of test accuracy and training time. Applications define the best network structure for DC-ELM, and there is a way to find the best structure.

He, Sheng, and Lambert Schomaker [10] offer a deep adaptive learning approach for single-word image-based writer identification. To enforce the emergence of reusable characteristics, the method entails including an auxiliary task during the training phase. To take use of the deep features that were learnt from the auxiliary task, the authors suggest a brand-new adaptive convolutional layer. Three different auxiliary tasks that correspond to the explicit information in handwritten word images are evaluated after the multi-task neural network has been trained from beginning to end. Results demonstrate that the suggested strategy outperforms non-adaptive and straightforward linear-adaptive alternatives in terms of writer identification performance.

A growing amount of interest has been generated in finding a solution to the open problem of handwritten digit recognition, which Ali Alani [11] have shown. Research on handwritten digit recognition in Arabic is scarce, despite the fact that many research have been proposed in the past and most recently to enhance handwritten digit recognition in a variety of languages. In the feature extraction step, we start by using the RBM, a deep learning technique that can extract extremely valuable features from raw data and has been applied to a number of classification issues. Last but not least, a comparison of our findings with those of other investigations on the CMATERDB 3.3.1 Arabic handwritten digit dataset demonstrates that our methodology the highest degree of accuracy.

Suiyang Khoo, Zhihong Man, Kevin Lee, Dianhui Wang, and Zhenwei Cao [12] This research develops an optimal weight learning machine for a handwritten digit image identification using a single hidden layer feedforward network (SLFN). It can be seen that the SLFN's input and output weights have both undergone global optimisation using the batch learning type of least squares. In order to maximise the separability of all nonlinearly separable patterns and achieve a high level of recognition accuracy with a minimal number of hidden nodes in the SLFN, all feature vectors of the classifier can then be positioned at the specified locations in the feature space. A test to see whether handwritten digit images can be recognized. The proposed methodology exhibits good performance and efficacy using both the MNIST database and the USPS database.

According to S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana, and Abdullah Al Jobair [13], one of the practically significant problems in pattern recognition applications is handwritten character recognition. Applications for digit recognition include filling out forms, processing bank checks, and sorting mail. The capacity to create an effective algorithm that can recognise handwritten digits and which is submitted by users via a scanner, tablet, and other digital devices is at the core of the issue. WEKA has been used to recognise digits using a variety of machine learning algorithms, including Multilayer Perceptron, Support Vector Machine, NaFDA5, Bayes, Bayes Net, Random Forest, J48, and Random Tree. The study's findings indicate that the top 90.37 percent accuracy has been obtained for Multilayer Perceptron.

This paper introduces a novel deep learning architecture called DIGITNET and a large-scale digit dataset called DIDA to detect and recognise handwritten digits in historical document images written in the nineteenth century, according to a proposal by Huseyin Kusetogullari, Amir Yavariabdi, Johan Hall, and Niklas Lavesson [14]. Digit ized from historical Swedish handwritten documents authored by many priests in a variety of handwriting styles are collected to create the DIDA collection. Three sub-datasets of this dataset are available: a single-digit subset, a large-scale bounding box annotated multi-digit subset, and a digit string subset containing samples in Red-Green-Blue (RGB) colour spaces. Additionally, DIDA is used to train the DIGITNET network, which consists of the DIGITNET-dect and DIGITNET-rec deep learning architectures, to isolate and identify digit strings in historical handwritten documents.

The adaptive function neural network (ADFUNN) and online snap-drift learning are combined in a novel way by Miao Kang and Dominic Palmer-Brown [15] and used for optical and pen-based recognition of handwritten digits. [E. Alpaydin, F. Alimoglu for Optical Recognition of Handwritten Digits and E. Alpaydin, C. Kaynak for Pen-Based Recognition of Handwritten Digits Snap-drift [S.W. Lee is a quick, unsupervised method appropriate for online learning and non-stationary situations where new patterns are continuously supplied. It combines the complimentary principles of common (intersection) feature learning (referred to as snap) and LVQ (drift towards the input patterns) learning.

A. Existing System

Some of the algorithms used to develop handwritten digit recognition systems are Random Tree, Support Vector Machine (SVM), Proximal Support Vector Machine (PSVM), Multilayer Perceptron, Random Forest, Bayes Net, Naive Bayes, and J48. Previous studies have shown that these algorithms provide accuracy around:
• Proximal SVM Algorithm - 97%
• Multilayer Perceptron Algorithm- 91%
• SVM Algorithm - 88%
• Random Forest Algorithm - 86%
• Bayes Net Algorithm - 85%
• Naive Bayes Algorithm - 82%
• J48 Algorithm - 80%
• Decision Tree Algorithm - 76%

Even while some applications based on this technology may find these algorithms valuable, many other areas of applications, like those in the Finance business, call for superior outcomes that could be obtained by utilizing alternative algorithms in comparison to the algorithms that have already been described.

B. Suggested System

Handwritten digit recognition systems can be developed by using Convolutional Neural Network (CNN) in order to lower error and increase overall efficiency. Our suggested approach makes use of a 3x3-sized kernel and CNN with numerous pooling and convolutional layers to accomplish this. In order to train our model, 60,000 28*28 grayscale photos are used. Our model is trained through a typical 5 epochs to attain correctness around 99.15% rather than conventional techniques used to develop handwritten digit recognition systems, like J48, Decision tree, Native Bayes, etc.

IV. SUGGESTED APPROACH

The aim of our proposed work is to identify handwritten digits defined by the user and recognize the full digit that the user inputs (by default, in the decimal number system), which is then translated to the user's preferred binary, octal, or hexadecimal number system. For this, a graphical user interface (GUI) will be developed in which a canvas widget will be visible to the user which will help him to draw handwritten digit strings that will be recognized and converted appropriately. After that, the area can be cleaned up to continue.

A. Dataset

The suggested model is trained using the MNIST dataset. 70,000 digital photos make up the collection, which may be used to train and evaluate the model. These datasets for training and testing are created using a particular ratio. Then, this image data is cleansed and prepared for advancement.

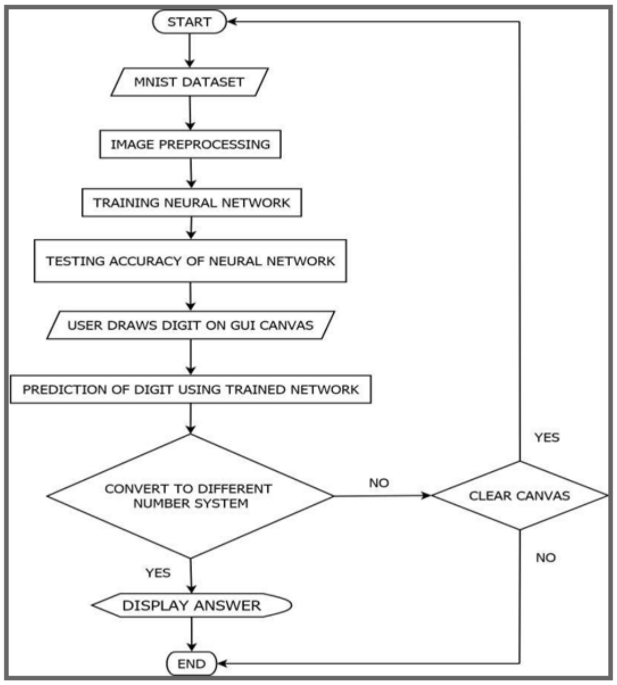

START
MNIST DATASET
IMAGE PREPROCESSING
TRAINING NEURAL NETWORK
TESTING ACCURACY OF NEURAL NETWORK
USER DRAWS DIGIT ON GUI CANVAS
PREDICTION OF DIGIT USING TRAINED NETWORK
CONVERT TO DIFFERENT NUMBER SYSTEM
NO
CLEAR CANVAS
YES
NO
YES
DISPLAY ANSWER
END

Fig. 2. Flowchart of Activities in Suggested project

B. Image Preprocessing

In this phase, several techniques are put into practice, like shrinking the pictures, transforming them into the format of grayscale, and enhancing the picture, in order to make the digital image data useful for our machine learning system.

C. Training Neural Network

The CNN model, which includes various pooling and convolutional layers along with a kernel of size 3x3, would be developed after data preprocessing is finished. The model will then be trained using training and validation data with the help of many pre-loaded libraries of Python, such as Scipy, Theano, TensorFlow, Keras, Pandas, Matplotlib, etc.

D. Testing Accuracy of Neural Network

Testing datasets are then used to gauge the model's effectiveness once it has been trained using the training dataset. MNIST's whole dataset is used to test the suggested model's accuracy.

E. User Draws Digit on GUI Canvas

Once the model has been reviewed, trained, and tested using MNIST dataset, it is then available for use via a Graphical User Interface (GUI) based canvas where a user can draw numbers using the mouse pointer.

F. Recognize Number/Clear Canvas

The user is presented with two alternatives after drawing the desired number of numbers on the GUI canvas-

- Recognize Numbers. In order to forecast the user's string of digits, this option makes use of the CNN model.

- Clear Canvas: With this option, the user may clear the canvas and add new digits to the existing ones.

G. *Convert to Different Number System*

The user is presented with three options following the occurrence of the predicted digits-

- Binary conversion: This option converts the recognized decimal number into binary.
- Hexadecimal conversion: This option converts the recognized decimal number to its hexadecimal.
- Octal conversion: This option converts the recognized decimal number to its octal.

V. IMPLEMENTATION DETAILS

A. *Digit Recognize File*

The handwritten digit recognition system needed for this project will be built using the MNIST dataset. To do this, the MNIST dataset will be added to our digit recognition Python application. The established sequential CNN model will next be extended with convolution and pooling layers. A 3x3-sized kernel will be used to filter the digital image data.
After the data has been processed through pooling and convolution layers, the 'flatten' function is used to reduce the multidimensional data input to a single dimension before moving to a fully connected layer. The activation function that will be utilised to train the suggested CNN model is called ”relu”.
Following this, the image data is binarized, or changed from its original 28*28 grayscale format to a matrix in binary. Then, 60,000 training samples and 10,000 testing samples are separated from the MNIST dataset. The final step in compiling the model is training it through 5 epochs using the 64-batch ”rmsprop” optimizer.
The suggested CNN model's loss and accuracy are then assessed, and the model is saved for use in the GUI Python file at a later time.

B. *GUI File*

After the saved model has been loaded into the GUI Python file, the main GUI window that displays the canvas widget is initially created using the 'Tk' method. A main loop is given to the master window, and it keeps going until the user shuts it. The title of the main GUI window is then assigned to the proposed project. The ”Recognise Number” and ”Clear Canvas” buttons are situated in the main window.
The procedures for putting features into action, such as cleaning the canvas, drawing the numbers, initiating an event to do so, and identifying the numbers, are then defined. The border of an image is defined as a set of contours, or a line connecting all the points with a similar intensity.
The model begins to predict each and every digit one by one once you click the ”Recognise Number” button. The result, which is given in a new window, recognises each digit separately and displays the accuracy with which it was detected. In addition to the three additional options and the indicated number, this new window also has the required title. The feature of translating the detected decimal number to the user's selected binary, hexadecimal, or octal number system is finally enabled by these three options.

VI. LIMITATIONS

The fact that there is so many various handwriting style which is a very personal behavior—makes developing a handwritten digit identification system one of the most challenging tasks. Numbers can be written at different angles, with varying amounts of stress, and with various segment lengths. Although machine learning developers encounter these difficulties, some steps have already been taken, such as fine-tuning already mentioned models and developing cutting-edge classification algorithms for accurately predicting handwritten numbers while decreasing computing cost and time. Additionally, extensive study is being done in this area to support it appropriately.

If this concept is used on a large scale, several problems might occur. If utilized maliciously, the ability to recognize handwritten numbers could eventually result in a number of problems. Such technology could be used by criminals to identify ATM, bank, and other types of pins. Contrarily, even though problems like these might arise, steps could be taken to address them and maintain the use of this technology to computerize a variety of tasks, including banking, address recognition, shipping systems, the postal service, and others, making it advantageous to work, as it will ultimately have more benefit than drawbacks.

VII. CONCLUSION AND FUTURE SCOPE

Our study has led us to the conclusion that machine learning algorithms are extremely effective at identifying trends throughout various writing styles. Handwritten digits can be recognized using a variety of techniques.
It may be said that CNN recognizes and predicts handwritten digits with the highest degree of accuracy. By removing the fine-tuning of the hyper parameters in the pure convolutional neural network (CNN) architecture, the accuracy of these conventional CNNs can be increased even further. This will help lower the total cost and complexity of the model's computations.
This technology can also be implemented on the GPU in addition to the CPU to increase efficiency by speeding up calculation. The suggested machine learning model performs better overall because training time is cut down by using Compute Unified Device Architecture (CUDA) on a GPU.
There are many applications of machine learning-based technology in the banking, shipping, and postal sectors, and it can automate many operations in these sectors. Some of these involve automating the banking procedure and saving time by examining cash slips for account numbers and amounts. The postal and shipping industries could potentially profit from computerized address recognition.

In general, this technology may handle a wide range of complicated applications, some of which are real-time in nature and some are not.

References

[1] Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE signal processing magazine, 29(6), 141-142.

[2] Revow, M., Williams, C. K., & Hinton, G. E. (1996). Using generative models for handwritten digit recognition. IEEE Transactions on pattern analysis and machine intelligence, 18(6), 592-606.

[3] Fukushima, K. (2003). Neocognitron for handwritten digit recognition. Neurocomputing, 51, 161-180.

[4] Hamidi, M., & Borji, A. (2010). Invariance analysis of modified C2 features: case study—handwritten digit recognition. Machine Vision and Applications, 21, 969-979.

[5] Kulkarni, S. R., & Rajendran, B. (2018). Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization. Neural Networks, 103, 118-127.

[6] Qiao, J., Wang, G., Li, W., & Chen, M. (2018). An adaptive deep Q-learning strategy for handwritten digit recognition. Neural Networks, 107, 61-71.

[7] McDonnell, M. D., Tissera, M. D., Vladusich, T., van Schaik, A., & Tapson, J. (2015). Fast, simple and accurate handwritten digit classification by training shallow neural network classifiers with the 'extreme learning machine'algorithm. PloS one, 10(8), e0134254.

[8] Gader, P. D., & Khabou, M. A. (1996). Automatic feature generation for handwritten digit recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(12), 1256-1261.

[9] Pang, S., & Yang, X. (2016). Deep convolutional extreme learning machine and its application in handwritten digit classification. Computational intelligence and neuroscience, 2016.

[10] He, S., & Schomaker, L. (2019). Deep adaptive learning for writer identification based on single handwritten word images. Pattern Recognition, 88, 64-74.

[11] Alani, A. A. (2017). Arabic handwritten digit recognition based on restricted Boltzmann machine and convolutional neural networks. Information, 8(4), 142.

[12] Man, Z., Lee, K., Wang, D., Cao, Z., & Khoo, S. (2013). An optimal weight learning machine for handwritten digit image recognition. Signal Processing, 93(6), 1624-1638.

[13] Shamim, S. M. (2018). Handwritten digit recognition using machine learning algorithms. Global Journal Of Computer Science And Technology, 18(D1), 17-23.

[14] Kusetogullari, H., Yavariabdi, A., Hall, J., & Lavesson, N. (2021). DIGITNET: a deep handwritten digit detection and recognition method using a new historical handwritten digit dataset. Big Data Research, 23, 100182.

[15] Kang, M., & Palmer-Brown, D. (2008). A modal learning adaptive function neural network applied to handwritten digit recognition. Information Sciences, 178(20), 3802-3812.

A-3. Concerns Related to Project

A-3.1. Social Relevance

Handwritten digit recognition is an area of utmost importance critical to research and development, with a seemingly endless list of possibilities.

A machine learning-based handwritten digit identification system is socially relevant since it has a wide range of applications, including number plate recognition, postal mail sorting, and bank check processing.

This benefits society by automating the process of real-time handwritten digit/text recognition for a variety of applications.

However, putting this technology into practice on a wide scale could have a number of negative consequences. Recognizing handwritten numbers could lead to the emergence of various social difficulties if it is not done with good intentions.

A-3.2. Health Concern

Handwritten Digit Recognition using Machine Learning doesn't necessarily pose any health concerns.

The primary objective of this particular project is to develop and test a machine-learning system for automatically recognising handwritten digits. There is no direct impact or health concern related to this project.

It may have an indirect impact on the mental health of a person in a good sense since using this system automates the process of digit recognition and may improve one's efficiency ultimately saving time and manpower.

A-3.3. Legal Aspects

Here, our handwritten digit recognition system using machine learning uses the MNIST which is a free dataset. It uses python as its core programming language which is a free & open-source language. Libraries used for this project (TensorFlow, Keras, Tkinter) are also free and open source.

Hence, this project is completely legal and a risk management plan will be developed to completely tackle all the obstacles faced during the completion of this project.

Also, several laws may need to be implemented by the government alongside spreading awareness to introduce the incorporation of a handwritten digit recognition system worldwide as it may be used by malicious people to cause felonies.

A-3.4. Cultural Aspects:

Handwritten digit recognition does not infer any cultural aspects directly. Rather it may benefit an organization economically as it can be implemented within several software technologies which would ultimately reduce the overhead cost.

On the other hand, people with malicious intentions may use this as a tool to cause harm to other people who are unaware of this technology.