

# **HANDWRITTEN DIGIT RECOGNITION USING CNN**

**A Project Report submitted in partial fulfillment of the requirements  
for the award of the degree of**

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

**VUNNA ALEKHYA  
(121710308058)**

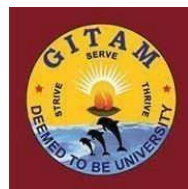
**POREDDY AJAY KUMAR REDDY  
(121710308037)**

**KODUGANTI SUMANTH SAI RAM  
(121710308021)**

**SARIPELLA AKHIL VARMA  
(121710308046)**

**Under the esteemed guidance of**

**Ms. A. Navya Sri  
(Assistant Professor)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM INSTITUTE OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**

**VISAKHAPATNAM**

**2017-2021**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**GITAM INSTITUTE OF TECHNOLOGY**  
**GITAM**  
**(Deemed to be University)**  
**VISAKHAPATNAM**



**DECLARATION**

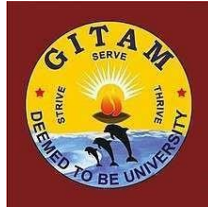
We hereby declare that the project report entitled “**Handwritten Digit Recognition using CNN**” is an original work done by **VUNNA ALEKHYA (121710308058)**, **POREDDY AJAY KUMAR REDDY (121710308037)**, **KODUGANTI SUMANTH SAI RAM (121710308021)**, **SARIPELLA AKHIL VARMA (121710308046)**, and provide this opportunity by the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University), Visakhapatnam submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. In Computer Science and Engineering. The work has not been submitted to any other college or university to award any degree or diploma.

<b>Registration No:</b>	<b>Name</b>	<b>Signature</b>
121710308058	VUNNA ALEKHYA	
121710308037	POREDDY AJAY KUMAR REDDY	
121710308021	KODUGANTI SUMANTH SAI RAM	
121710308046	SARIPELLA AKHIL VARMA	

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM INSTITUTE OF TECHNOLOGY**

**GITAM (Deemed to be University)**



**CERTIFICATE**

This is to certify that the project entitled “**HANDWRITTEN DIGIT RECOGNITION USING CNN**” is a bonafide record of work carried out by **VUNNA ALEKHYA (121710308058), POREDDY AJAY KUMAR REDDY (121710308037), KODUGANTI SUMANTH SAI RAM (121710308021), SARIPELLA AKHIL VARMA (121710308046)** submitted in partial fulfillment of the requirement for the award of the degree of Bachelors of Technology in Computer Science and Engineering.

**PROJECT GUIDE**

**HEAD OF THE DEPARTMENT**

**Ms. A. Navya Sri**  
**(Assistant Professor)**

**Dr. R. Sireesha**  
**(Professor)**

## ACKNOWLEDGEMENT

The final year project is a golden opportunity for learning and self-development. We consider very honored to have so many wonderful people who led us through the completion of the project. The satisfaction that accompanies the successful completion of the project would be incomplete without mentioning the people who made it possible and whose constant guidance and encouragement crowned all the efforts with success.

We express a profound sense of gratitude to our project guide **Ms. A. Navya Sree**, Assistant Professor, Department of Computer Science and Engineering, GITAM, for the expert guidance and motivation given to us throughout our work. She spared her valuable time for patiently listening to our problems and helping us find solutions. Our project report would not have been shaped this form without her constant encouragement and cooperation in all aspects.

We also express our thanks to the project reviewers, **Dr. Y. Srinivas**, Professor, and **Dr. S. Praveen Kumar**, Assistant Professor, Department of Computer Science and Engineering, GITAM, for their valuable suggestions and guidance during our project reviews.

We also express our deep gratitude to our beloved **Dr. R. Sireesha**, Head of the Department, Department of Computer Science and Engineering, for the encouragement and support.

We take this opportunity to place on record our sincere thanks to our honorable principal **Dr. C. Dharma Raj**, GITAM Institute of Technology, GITAM, for providing all the required resources for completing the project till this stage. Finally, we thank everyone who supported us directly and indirectly in making the project a successful one.

V ALEKHYA (121710308058)

P AJAY KUMAR REDDY (121710308037)

K SUMANTH SAI RAM (121710308021)

S AKHIL VARMA (121710308046)

## ABSTRACT

In the field of computer vision, handwritten digit recognition is increasingly widespread. The way to perceive and anticipate manually written numbers from 0 to 9 is easier and more accurate. We have tried in this project to understand and convert handwritten numbers into a standard format. Convolutional neural networks are a type of forwarding mechanism supporting multilayer that is taken into consideration. We will use the MNIST dataset, containing 70,000 images, to prepare and identify them. Each digit has 28 to 28 grey pixel intensities to achieve higher efficiency.

Until being forwarded to hidden layers, such as convolution and pooling, the data will be fed into the input layers of CNN. Finally, the numbers are projected onto the fully connected layer and labelled using a softmax classifier. To create the network, we will use the Python deep learning library in Keras. Moreover, we have created a web application to enhance user participation. The customer provides input to the Web application. The performance will be generated by the model linked to the web app and displayed on the web app screen.

**Keywords-** MNIST dataset, Convolution neural network, Deep learning.

## TABLE OF CONTENTS

<b>S.no.</b>	<b>CONTENT</b>	<b>PAGE NO.</b>
1	Chapter 1-Introducction	1-11
	1.1-Python	1-4
	1.2-Deep learning	4-9
	1.3-Neural network	9-11
2	Chapter 2- Literature Survey	12-17
	2.1-Introduction	12
	2.2-Related work	12-17
3	Chapter 3- Analysis	18-35
	3.1-Objective	18
	3.2-Purpose	18
	3.3-Software and Hardware requirements	18
	3.4-Problem statement	19
	3.4- Dataset Description	19-21
	3.5- Algorithm and Methodology	22-35
	3.5.1- Convolutional neural network	22-24
	3.5.2- Layers of CNN	24-35
4	Chapter 4- Design	36-44
	4.1- Uml diagrams	39-44

5	Chapter 5- Implementation	45-54
6	Chapter 6-Testing	55-57
7	Conclusion	58
8	Future scope	59
9	References	60
10	Appendix	61-78

## LIST OF FIGURES

Fig.No.	Name	Page no.
1.1	ML vs. DL	5
1.2	Neural network	10
3.1	Files in dataset	20
3.2	Digits in dataset	20
3.3	Labels	21
3.4	LeNet architecture	24
3.5	Input and kernel	25
3.6	Feature map	26
3.7	Convolution process	27
3.8	Zero padding	28
3.9	Max pooling	30
3.10	Dropping of neurons	32
3.11	Flattening	33
3.12	Layers of CNN	35
4.1	Flow chart	36
4.2	Splitting the dataset	37
4.3	Use-case diagram	40
4.4	Class diagram	41



4.5	Sequence diagram	42
4.6	Statechart diagram	43
4.7	Deployment diagram	44
5.1	Importing dataset & libraries	45
5.2	Loading the dataset	46
5.3	Reshaping the data	47
5.4	Data normalization	47
5.5	Before normalizing	48
5.6	After normalizing	49
5.7	One-hot encoding	49
5.8	Model building	51
5.9	Model training	53
6.1	Testing	55
6.2	Web app	55
6.3	Recognize	56
6.4	Clear	57

## **CHAPTER 1**

### **INTRODUCTION**

A handwritten digit sensing device uses machines to automatically train or recognize real-life numbers from various sources, such as email, search, notes, photographs, etc., to detect online manuscripts on tablets or systems for automatic inspection of vehicles, processing control quantities, and hand filling forms. To ensure such real-life conditions in computer tablets or system handwriting recognition, consider numeric platforms for vehicles, operating checks, and number details. This project has developed the model of recognizing, transforming, educating, and validating manuscript numbers. To execute this project, we have used the CNN algorithm.

#### **1.1 PYTHON**

Python might be an extremely powerful programming language used for several completely different applications. Over time, the massive community around this open language has created quite a few tools to work with Python efficiently. In recent years, quite tools are designed specifically for information processing. As a result, analyzing information with Python has never been easier. Python will be a programing language that permits you to work quickly and integrate systems additional with efficiency. There is a combination of major Python versions, Python 2 and Python 3.

Vogue, contract, and logic programming are among the paradigms supported by extensions. Dynamic writing and a combination of reference investigation and a cycle-detecting garbage collector are used by Python to handle memory. During the execution of the program, it uses dynamic name resolution (late binding) to bind technique and variable names. In the Lisp tradition, Python's vogue provides some support for sensible programming. Its filter, map, and decrease functions; list comprehensions, dictionaries, sets, and generator expressions; and list comprehensions, dictionaries, sets, and generator expressions.

Python provides code that is concise and easy to read. Machine learning and AI are based on complex algorithms and versatile workflows, but Python's simplicity allows engineers to type in

reliable frameworks. Designers get to put all their exertion into understanding an ML issue rather than centering on the specialized subtleties of the language.

The main reasons why Python is that the most popular language for machine learning is its allows several libraries. A library could be a assortment of functions and routines that a artificial language will use. Having access to numerous libraries permits developers to perform complicated tasks while not the necessity to rewrite several code lines. Since machine learning heavily depends on mathematical optimisation, chance and statistics, Python libraries facilitate knowledge scientists perform varied studies simply. Here are a unit a number of the libraries you'll be able to use with Python:

Pandas – for high-level knowledge structures and analysis.

Keras – for deep learning.

Matplotlib – to form second plots, histograms, charts, etc.

StatsModels – for applied mathematics algorithms and knowledge exploration and plenty of others.

## **History of Python**

Python can be a standard, high-level language for programming. It was initially programmed and developed by Python Software Foundation by Guido van Rossum in 1991. Its syntax helps programmers to accurate concepts with fewer lines. It is designed chiefly to highlight the readability of code.

History was about to be published in the late 1980s. It was then that Python began performing. Soon, Guido Van Rossum started the Centrum Wiskunde & Informatica (CWI), located in the Netherlands on an application-based addition in December 1989. First, it began as a leisure project, so he wanted to find an important project for Christmas to keep him busy. ABC programming language, which was interfaced with Amoeba tools and which featured an exceptional handler, is supposed to be an artificial language of Python itself. In his early career, he already helped to build ABC, and he saw some problems with ABC but most of the features he enjoyed. He did quite smart subsequently. He had adopted the ABC syntax and some of its commodities. There were several criticisms, too, but he fully solved these problems and

developed a proper scripting language to delete all the flaws. The BBC's "Monty Python's 3 Flying Circuit" was the basis for its theme because it was an immense fan of the TV show. He wanted his discovery to have a short, special, and somewhat secret name, and so he called it Python! BDFL before he left office because of the tyrant on 12 July 2018. He was "the benevolent emperor for living" (BDFL).

Finally, in 1991 the language was written. When released, after comparison with Java, C++ & C, it used far fewer codes for basic concepts. The theory of architecture was pretty sweet. Its primary focus is the readability of code and the advanced competitiveness of developers. It was able to provide classes with legacy. Multiple key data formed exceptions and functions when it was launched.

### **Advantages of Python**

- It is simple to read, learn, and write.  
Python may be a language with a high degree of English syntax programming. It makes reading and understanding the code simpler. This is so many people are recommending Python to newcomers; Python is very easy to pick up and understand. You need fewer lines on of the code than other key languages such as C/C++ and Java to do the same operation.
- Improved Productivity  
Python is a language with great productivity. Thanks to Python's versatility developers can specialise in solving the problem. You don't have to waste very long learning the programming language vocabulary behaviour. You write fewer codes and get better.
- The ASCII text file can be downloaded, amended, and also distributed for Python version.

### **Disadvantages of Python**

- Code running on a line-by-line basis also leads to slow running. Python's complex design is also blamed for Python's sluggish pace when it does the extra work during code execution.

- Python needs make a compromise to make the creator simpler. A larger volume of storage is used by the Python programming language. When we create programs, this can be a challenge when we choose optimized memory.
- The server-side programming typically uses Python.  
Because of the following explanations, we do not know Python on the customer or Smartphone applications.

### **Python's advantages over other languages**

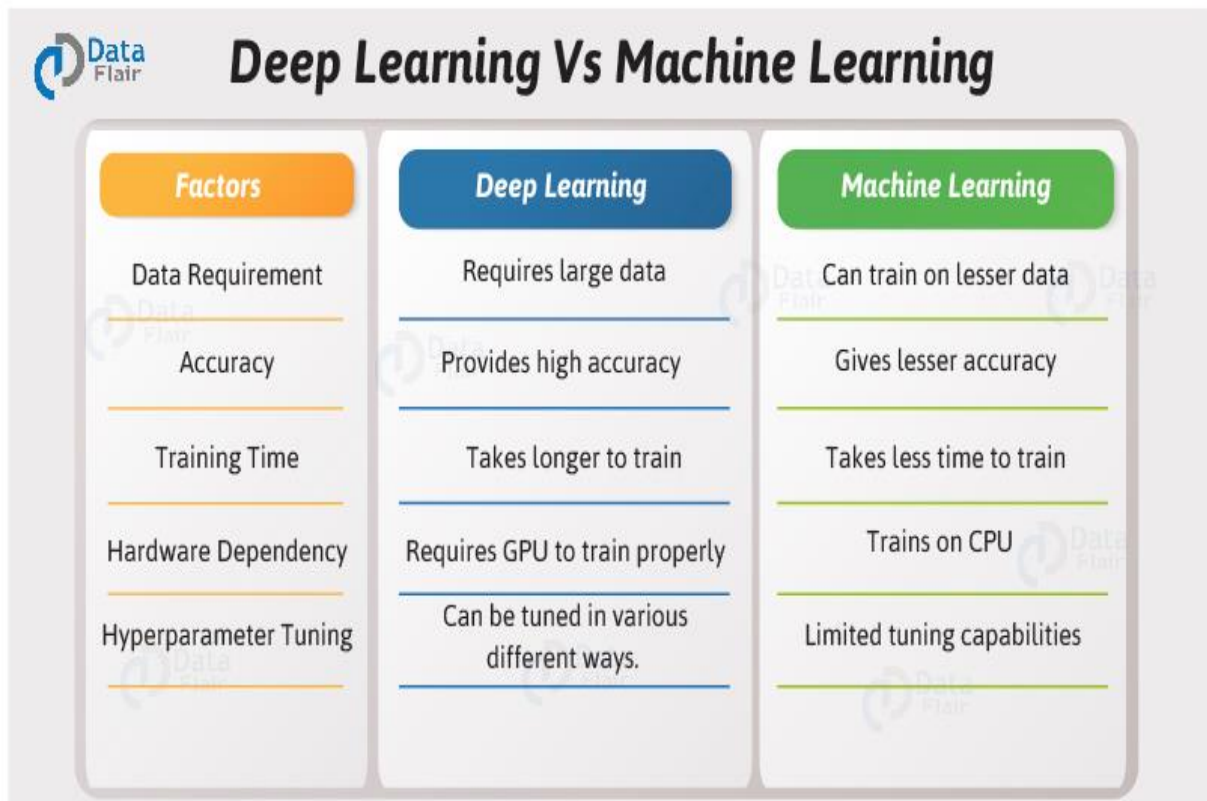
Less coding, most functions, have been removed when the same job is performed in other languages, python needs less coding. Python also has outstanding standard library support, so no third-party libraries need to search for the work. Often that's the reason many people recommend that beginners study python. Python is open; thus, people, small companies or big organisations may use free access to application software. Python is standard and used often, thereby providing greater community support. For anyone, python is if the programming works on a computer, Linux, Mac, or Windows would run.

## **1.2 DEEP LEARNING**

Deep learning may be a deep learning methodology that builds neural networks to imitate human brain structure and output. Suppose that the pc file is also a pixel matrix. Typical pixels are abstracted from the first layer, and functional edges are identified in the image. Easy elements such as leaves and branches may be made in the next layer. Then a tree may be recognized by the following layer. The knowledge passed from one layer to the next is a transition that transforms the 1-layer output into the next input. The computer will also understand which features the data to each layer correlates to a certain abstraction level.

The knowledge passed from one layer to the next is a transition that transforms the 1-layer output into the next input. The computer will also learn the characteristics of the data placed at which layer/level on its own, and each layer coincides with a certain abstraction. Deep learning means approaching and extracting complex non-linear functions with minor errors via multilayer IP. In deep learning, evolutionary and recurrent neural networks and new neural networks play a

functional role. In statistical predictions, recurrent neural networks have better accuracy. But the gradient of the neural network may disappear with in-depth training.



The infographic is titled "Deep Learning Vs Machine Learning" and features the "Data Flair" logo in the top left corner. It is structured as a table with three columns: "Factors", "Deep Learning", and "Machine Learning". The "Factors" column lists five criteria, each with a horizontal line. The "Deep Learning" column provides details for each factor, and the "Machine Learning" column provides corresponding details. The background is light gray, and the columns are color-coded: orange for factors, blue for Deep Learning, and green for Machine Learning.

Factors	Deep Learning	Machine Learning
Data Requirement	Requires large data	Can train on lesser data
Accuracy	Provides high accuracy	Gives lesser accuracy
Training Time	Takes longer to train	Takes less time to train
Hardware Dependency	Requires GPU to train properly	Trains on CPU
Hyperparameter Tuning	Can be tuned in various different ways.	Limited tuning capabilities

**Figure 1.1:** ML vs. DL

### Deep Learning Importance

This learning algorithm is also an innovator in the analysis of vast amounts of information as the machine's accuracy increases as more data is analysed. As information grows, even secret patterns within the data are better recognised by the computer. When the computer learns from the stored data, it can automatically derive features and abstract data with little to no human input.

## **Advantages of Deep Learning**

- **Use of unstructured data as maximum:**

Deep learning algorithms will not be used to train various algorithms but will also gain useful lessons for the purposes of coaching. Profound learning algorithms are used to Find all existing relationships between a manufacturing summary, social networking, or more to predict the coming stock prices of a given company

- **Eliminating the need for feature engineering:**

One of the main benefits of a profound learning method is the capacity to carry out functions on its own. During this technique, the information would be scanned by an algorithm to spot features that connect them with rapid learning without being expressly instructed to attempt. The opportunity to minimise wasteful work helps data scientists.

- **Ability to have responses of the highest quality:**

People die or are tired and make blunders. This is not the case from time to time, even though it includes neural networks. Once properly educated, a deep learning algorithm would be able to do thousands of regular and repeat activities in a relatively shorter timeframe than what someone believes to be. Furthermore, the work quality should not degrade until the work details contain detail that is not the issue you are attempting to unravel.

- **Unnecessary expense elimination:**

Deep learning helps diagnose arbitrary faults that are difficult to coach, such as minor product-labelling faults etc. Deep learning models can also recognise deficiencies which may otherwise be difficult to find. When clear photos are difficult due to various factors, deep learning may take these differences into account and learn useful features to make inspections robust.

- **Data labelling can be a expensive and time consuming task:**

Elimination of the need to mark data. The need for good-labelling data becomes redundant with a deeper learning approach, as the algorithms exceed learning without any guideline. Other kinds of approaches to machine learning are not almost so effective as well.

### **Disadvantages of Deep Learning**

- To achieve greater results than other methods, it takes a lot of experience. Coaching of complicated data structures is expensive. In addition, deep learning needs costly GPUs and a large number of computers. This raises users' costs.
- You cannot find the best departmental teaching tools by default because the topology, instruction and other criteria are needed. It is also impossible for fewer qualified citizens to follow it.
- You need classifiers in order to explain the output that is provided merely by learning. These operations are carried out by revolutionary neural network-based algorithm.

### **Deep Learning Applications**

- **Virtual Assistants**

Virtual Assistants that include natural speech commands and extensive activities for users are cloud-based. Amazon Alexa, Cortana, Siri, and Google Assistant are typical examples of automatic helps. They need internet connected devices to operate in all their capacity. You prefer to provide a cleaner user experience, based on prior meetings, while you have an assistant command, using deep learning algorithms.

- **Chatbots**

Chatbots will answer customer problems in seconds. A buggy is an AI chat-to-text app. It can talk and do actions like human beings. Customer participation, social network ads and immediate customer message are commonly used in Chatbots. It automatically



generates device inputs. Machines and deep learning algorithms are used to generate different types of response.

- **Healthcare**

There has been a note of the use of profound medical education. Computer-aided diseases and computer-aided diagnosis have become possible with profound learning. It is frequency to be used in surgery, medical research and detection of life-threatening diseases such as cancer and diabetic retinopathy by medical imaging.

- **Entertainment**

The following give their clients movies and music and content reviews. Netflix, Amazon, YouTube and Spotify. This is largely attributed to deep schooling. Internet streaming providers give you advice that can help you in your search experience, interest and behaviour in making product and service choices. Deep learning technology is also used to apply sound to silent movies and to generate subtitles automatically. First, News Aggregation is our next important application for profound learning.

- **News Aggregation and Fake News Detection**

Deep Learning lets you tailor the news to your reader's style. You may supplement and filter news coverage in line with demographic, regional, economic and individual preferences. Neural networks promote the creation of classification systems to recognise and remove fake and biased news. You also warn of possible privacy breaches.

- **Image Coloring**

Deep study has seen significant advances in the colorization of images. The image colouring provides a gray-scale input and a coloured picture. An example of a ChromaGAN picture colorization model. A perceptive and semantime meaning of both the class distributions and colours is formed into a network of generatives by an adverse paradigm that learns to colour.

- **Robotics**

In the construction of robotics Deep Learning is commonly used for performing human duties. Deep Learning robots use real time warnings to identify obstacles and instantly plan their path. It can be used for transportation of supplies in hospitals, warehouses, storage, warehousing, manufacturing material, etc. Boston Dynamics robots respond to people as they walk about what they can unload a dishwasher, raise and perform other activities.

- **Image Captioning**

Photo captioning is the way to make a textual image description. Computer vision is used to correctly view the image content and the language model. A recurring neural network such as LSTM is used to render the marks a consistent argument. A textual file description bot has been created by Microsoft to allow you to upload the image or url of any image.

- **Advertising**

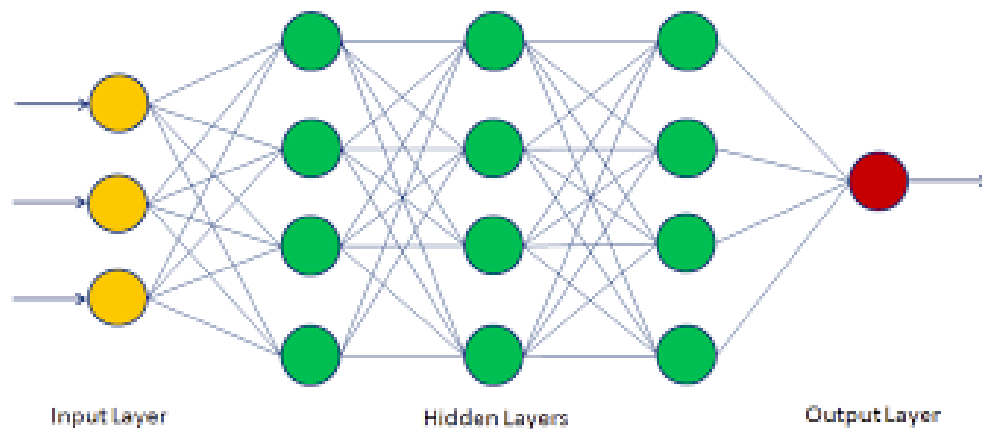
Deep learning allows optimization of the user experience in advertising. Deep Learning increases advertising's relevance and encourages advertisers and advertisers to adopt advertising strategy. It allows ad networks to save prices, reducing the cost of buying a drive from \$60 to \$30. Predictive advertising can be developed based on the data, real-time ads and advertising for target screens can be provided.

## **1.3 NEURAL NETWORKS**

Mathematical patterns using brain-based computing algorithms to store information are neural networks. Neural networks are called the 'artificial neural network' collectively, since they are used in machinery. Neural neurons are built like the brain with several neurons with many connections. In many implementations, neural networks are used to model uncertain associations between different parameters accompanied by many examples. Examples of effective neural

network implementations include handwritten digit classifications, voice recognition, and consequently inventory price prediction.

In addition, in medical applications, the use of neural networks is increasing. In 1943, Warren McCulloch and Walter Pitts, a young mathematician, produced a study on the workings of neurons. This article was the first step in the creation of artificial neural networks. A basic circuit-neural electricity network was modelled. During the 1950s, the efforts of Rosenblatt led to a two-layer sensory network adjusting the weight of the relation to the classification of the two layers. In the early 1980s, researchers showed a revival of interest in neural networks.



**Figure 1.2:** Neural network

All this indicates that neural networks operate on a picture. Deep learning networks, also referred to as convolutional networks, are more accurately recognised like artificial neural networks. The CNN combines learning and data entry with 2D convolutional layers, which make it better suitable for manipulation of 2D data like pictures. CNN can't determine which features are used to identify images, so the extraction of handheld attributes does not take place.

CNN works by directly removing photo features. The associated characteristics are not pre-trained, but discovered on a photo set as network trains. This automated extraction of features

makes deep learning algorithms particularly precise for computer vision tasks such as object recognition. Deep learning also reduces human efforts in the areas of interpretation, research, prediction and various other areas. The manuscript digits (0-9) of the popular MNIST dataset are recognised in this project.

The MNIST (Modified National Institute of Standards and Technology) dataset provided handwritten numbers totalling 70,000 photographs, with 60,000 in the training set and 10,000 in the test set of numbered 10 digit images (0 to 9). Handwritten figures are 28\*28 grey images which mirror a picture, each of which has a first column photograph (0 to 9). In order to test and optimise the learning models for the handwritten classification, Yann Lecun, Corinna Cortes and Christopher Burges have developed the MNIST data collection. Because everybody is writing in an interesting and exciting way, handwriting is a recognition. This is the power of the machine to automatically identify and decipher manuscripts or characters.

## **CHAPTER 2**

### **LITERATURE SURVEY**

#### **2.1 INTRODUCTION**

A literature survey is part of the project report that shows the different analyses and analyses generated and the results already printed taking into account different project requirements and scale of the project. This is the most critical part of your report because it lets you direct your analysis. It helps you to achieve a goal for your research and hence gives you your statement of issue.

#### **2.2 RELATED WORK**

[1]Handwritten Digit Recognition Using Convolutional Neural Networks [2017]

##### **-T SIVA AJAY**

In the area of machine view, handwritten numerical recognition is very common. A more accurate method is used for perceiving and anticipating manually numbered numbers from 0 to 9. Current networks are a multilayer mechanism of advancement. A CCN beats other Artificial Neural Networks in order to collect and use information on features, improve information on the 2D structure better and prevent localisation, scales and other distortions. LeNet engineering was first described in a LeCun et al. paper. LeNet was mostly carried out by its creators dependent on numbers and ch. LeNet engineering is straightforward and clear, making CNN implementation simple.To plan and classify them, we will use the MNIST data array. The main purpose of this dataset is to identify handwritten numbers 0-9. A total of 70,000 instructional and research photographs are available to us. In order to increase precision, each digit shows 28 by 28 grey pixel intensity. The numbers are entered in the LeNet input layers with two sets of convolution layers, induction layers and

pooling layers, followed by opaque layers comprising two sets of overall layers. A softmax classifier is then placed in the completely linked layer by means of a project.

[2]Handwritten Digit Recognition System Based on Convolutional Neural Network  
[2020]

**-Jinze Li, Gongbo Sun, Leiye Yi, Qian Cao, Fusen Liang, Yu Sun**

The paper says that the identification of the symbol is commonly used in machine vision today. A widespread form of image detection is digital reconnaissance. Today, automated identity online technology is much advanced relative to offline systems. This article focuses on a convolutional offline digital neural network recognition method for handwriting. The programme uses the MNIST dataset as a reference sample and uses the Opencv Toolkit to prepare the imaged. The digital image functions handwritten are then collected by the LeNet-5, transformed and pulled into the single-dimensional vector via the convolutional nerve network. Finally, the highest chance to quantify the result for handwritten digit recognition is found using the Softmax regression algorithm.

[3]Effective Handwritten Digit Recognition using Deep Convolution Neural Network  
[2020]

**-Yellapragada SS Bharadwaj, Rajaram P, Sriram V.P, Sudhakar S, Kolla Bhanu Prakash**

The paper suggested a basic neural network approach to written digit recognition using a convolutional neural network. Digits are regarded as an insolvable challenge for master learning algorithms like KNN, SVM/SOM due to their distinct expressive nature. In this paper, Convolution Neural Networks were used with 70000 digit MNIST datasets and with 250 various written forms. The technique suggested for preparing the actual written digit calculation at 60000 digits and 10,000 under validation has obtained 98.51% accuracy with only 0.1% error.

[4]Handwritten Digit Recognition using CNN [2019].

**-Vijayalaxmi R Rudraswamimath, Bhavanishankar K.**

Digit Recognition was explored in this article as an interesting and important topic. Since the written digits do not seem to be of the same height, spacing, position or direction, the difficulty of writing the digit identity can be evaluated in different ways. The presentation and instance of the numbers are also dependent on the various and singular compositional forms of several individuals. This is a system by which transcribed numbers are perceived and written. It contains a variety of functions, including pre-programmed bank checks, correspondence and tax documents. The aim of this project is to build a classification algorithm to identify written numbers.

[5]Handwritten Digit Recognition Using Machine Learning: A Review [2019]

**-Anchit, Shrivastava, Isha Jaggi, Shefali Gupta, Deepali Gupta.**

The handwritten recognition method has proved complicated thanks to a vast variety of written forms. In this way, we have tried to lay the basis for further field research in order to deal with the problems that remain today. In order to find the most appropriate and safest solution for digital identification, the current methods and solutions have been explored. A total of 60,000 images were used as training sets with a 28x28 pixel scale. Pictures and training sets were coupled to the original picture. After rigorous testing and measurement, a low error rate of just 0.32% was found in the classifier ensemble process.

[6]Handwritten Digit Recognition using Machine Learning Algorithms [2018]

**-S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair.**

Handwritten identification of characters is one of the most important subjects for pattern recognition systems. Digital identity has a variety of applications including mail sorting, business processing letters, entry of type material, and so on. At the

heart of the issue is the creation, by printer, tablet and other portable devices, of a cheap formulation which recognises written digits and is sent by the client. This paper discusses a manual digit recognition system for off-line application of a combination of machine learning techniques. The primary purpose of this paper is to show the efficient and accurate ways to make manuscript digits more appealable. Multilayer Perceptron, support vector machine, Nave Bayes, Net Bayes, Random forest, J48 and the Random Tree are used in many machine learning algorithms for the recognition of weka digits.

[7]Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers [2018]

**-Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique.**

Deep learning has brought about an enormous twist of machinery by making the development of artificial neural networks even more intelligent (ANN). In a wide range of applications in different fields, including criminal investigation, health, medicine, sports, robots and drones, deep learning is commonly used. Neural networks (CNN), which combine Artificial Neural Networks(ANN) with advanced approaches to deep learning, are at the front of spectacular deep learning advances. The implementations are just a few: pattern recognition, sentence recognition, speech recognition, facial recognition, categorising text, identity documents, scene recognition and handwritten digit recognition. The aim of this analysis is to compare the precise classification of handwritten figures with the use of different cached layers and counts of time. For this CNN success study, we performed our experiment with MNIST.



[8]A Comparative Study on Handwriting Digit Recognition Using Neural Networks [2017]

**-Mahmoud M. Abu Ghosh, Ashraf Y. Maghari.**

Handwritten numerical recognition has been one of the most well-known applications for machine learning and computer vision. A number of machine learning approaches have been used to resolve the written digit recognition problem. Neural network methods are the topic of this article. The three most well-known NN approaches are deep neural networks, profound creed networks (DBNs) and convolutional neural networks. In this article, the three NN methods for different variables such as precision and performance are compared and evaluated. However, accuracy and efficiency of identification are not the only criterion in the testing process, as are other considerations of concern such as runtime. Experiments are conducted with handwritten numbers using both random and regular datasets. The results demonstrate that DNN with a 98.08 percentage accuracy score is the most accurate of three NN processes. On the other hand, the execution time of DNN is the same as the other two algorithms. Instead every algorithmic rule has a 1–2 percentage error rating, in fact, due to the resemblance of digit forms, with the numbers (1,7), (3,5), (3,8), (8,5) ,(9,5) and (6,9).

[9]Improved Method of Handwritten Digit Recognition [2017]

**-Ernst Kussul, Tatyana Baidyk.**

Users will compare different handwritten digit recognition processes using the MNIST database. There are many facts about different recognition scores for the classifier, and our neural classifier was second (recognition rate 99.21 percentage ). We are actively working to improve the structure of neural networks and algorithms with regard to handwriting digit recognition. The improved classification score is 99.37 points. This is the most successful outcome. In this post, we clarify the general shape of our classification and the latest developments.

[10] Handwritten Digit Recognition Using Deep Learning [2017].

**- Anuj Dutt, AashiDutt**

The development of different machine learning, Deep Learning and Computer Vision algorithms has recently given the researchers great interest in hand-written digit recognition. In this study I compare the results of some of the most commonly used machine learning algorithms such as SVM, KNN & RFC and using Keras with Theano and Tensorflow as a deep learning algorithm like CNN multi-layered. With them, I have been able to achieve the accuracy of CNN (Keras+Theano) of 98,70% compared to SVM of 97.91%, KNN of 96,67%, and RFC of 96.89%.

## **CHAPTER 3**

### **ANALYSIS**

#### **3.1 OBJECTIVE**

The objective of our project is to identify the handwritten numbers which are given to the system in the form of images. We are making use of the Convolutional Neural Network algorithm for recognizing and classifying the digits into a specific categorical type.

#### **3.2 PURPOSE**

As handwriting varies from person to person, handwritten numbers do not always have the same height, weight, orientation or marginal justification. The general problem is to distinguish the digits due to the similarity of digits such as 1 and 7, 5 and 6, 3 and 8, 2, five, two and seven. We would like to produce a model that transforms the handwritten digits into a typical type in various types, so that no misunderstanding is created.

#### **3.3 SOFTWARE & HARDWARE REQUIREMENTS**

We require an IDE for implementing CNN. The IDE which we chose is Anaconda prompt (Jupyter Notebook) version 3.6.5 and we implemented CNN using python. For the web application, we used Spyder IDE. We can use windows 7 or older or macOS or Linux operating system for implementation. We also need minimum of 1GHz processor, 4 GB RAM and minimum of 32 GB hard drive and a Local Area Network or Wi-Fi.

### **3.4 PROBLEM STATEMENT**

As handwritten numbers do not tend to be consistent in the same height, weight or direction, including margins, the general drawback is when you distinguish the digits by the consistency of one to 7, 5 to 6, 3 and 8, 2 and 5, 2, 2 and 7, etc. This problem also occurs when a group of people in different handwriting types write a single digit. Finally, the creativeness and choice of different handwriting was inspired by the formation and presentation of the numbers. As a consequence, we developed a written digit detection model to avoid the problem.

### **3.5 DATASET DESCRIPTION**

A total of 70,000 hand written digits from the MNIST data collection is shown, with 60,000 training examples and 10,000 evaluation examples, all with numeric images 10 digitally (0 to 9).





The scale was normalized to match a 20\* 20pixel box without adjusting the ratio of a small segment from the comprehensive NIST set. Handwritten digits are pictures of 28\*28 images that represent an entity, with a label (0 to 9) for each image in the primary column. The same is true for the test kit, consisting of ten thousand images with a mark of 0 to 9.

In order to evaluate and improve machine learning models for the manuscript digit classification query, Yann Lecun, Corinna Cortes and Christopher Burges have developed this MNIST dataset. The MNIST dataset was created by a combination of unique database 3 (USA Census Office workers) and unique database 1, the binary representations of handwritten digits in NIST's unique datasets (high school students).The instruction was considered SD-3 (unique database –3) and the study considered SD-1 (unique data database –1) and it was simpler to understand SD-3.The NIST data collection was used to maintain the stuff in various learning classifications that are interesting, divided and equivalent.

The array of MNIST instruction contains 30,000 SD-3 designs and 30,000 SD-1 motifs. Our testing kit consisted of 5,000 SD-3 patterns and 5,000 SD-1 patterns. The set of 60,000 models contained some 250 drawings by poets. A research set consists of 5,000 SD-3 samples and 5,000 SD-1 samples. A number of digit photographs were taken, standardised and justified as cantered

from a range of digits. This provides an excellent dataset for the analysis of models and enables machine learners to concentrate on profound learning and machine learning with little or no data cleaning..

This PC > Local Disk (D:) > MINST DATASET

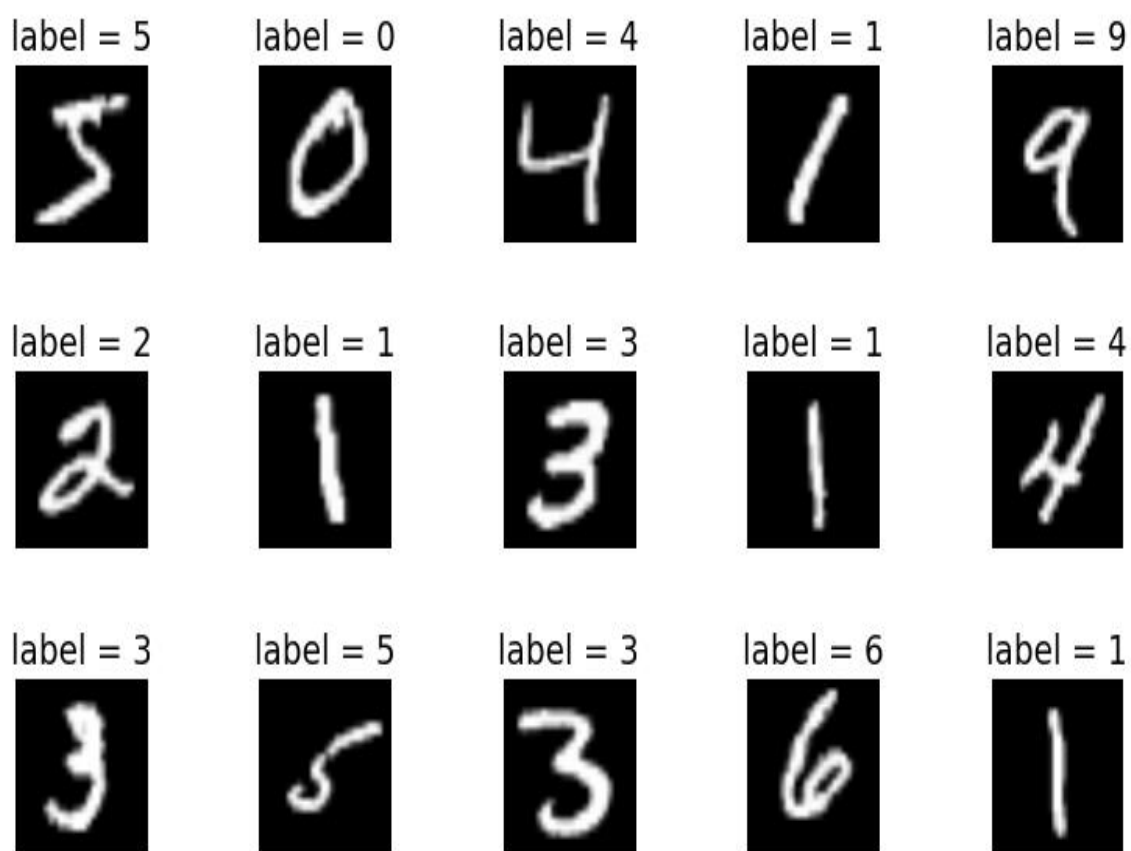
Name	Date modified	Type	Size
 t10k-images.idx3-ubyte	31-10-2020 15:50	IDX3-UBYTE File	7,657 KB
 t10k-labels	31-10-2020 15:50	IDX1-UBYTE File	10 KB
 train-images.idx3-ubyte	31-10-2020 15:51	IDX3-UBYTE File	45,938 KB
 train-labels	31-10-2020 15:51	IDX1-UBYTE File	59 KB

**Figure 3.1:** Files in dataset



**Figure 3.2:** Digits in dataset.

Pixels are ranked from 0 to 255 RGB colour code, with a set backdrop to white (0 RGB) and a set backdrop to black (255 value from RGB). Digital marks are broken down into 10 groups with digits 0 to 9.



**Figure 3.3:** Labels

### **3.6 ALGORITHM AND METHODOLOGY:**

In this project, we have used CNN. PYTHON can be used to implement convolutional neural networks. In our execution, we use Python, and our Python is incorporated with the Keras deep learning repository. We'll use Keras models to construct our network, and we'll write a driver program to call the network for inputs in the data set. Instructions, algorithms, and datasets for analysis are also included in the driver program.

MNIST data sets are the simplest, most widely accessible, and simplest to interpret data in the computer vision industry, and they are the machine we will use for our deep learning journey. With less preparation time, our network will categorise the digits up to 98 percent of the time after implementation. This implantation may often be removed by either CPU or GPU-enabled systems, but the CPU needs more planning than the GPU.

We'll use 66% of the data to prepare our network in order to test it. The MNIST data set has 28 grey images that are per digit taken as 28 and 28 and can be downloaded directly. The grayscales' pixel intensities range from 0 to 255. With a light background, all digits are shown as black, white, the digit itself, and various grey shades.

#### **3.6.1 CONVOLUTION NEURAL NETWORK**

The word "convolutional neural network" refers to a network that employs convolutional measurement. "Convolution" refers to the operation and summation of weight multiplying variable values. In at least one layer, convolutional networks are used to evaluate special neural networks instead of the general mathematical form. A convolutional neural network is a type of deep neural network widely used to examine visual eidetic processes. Since Facebook's predictive tagging algorithms is built on neural networks, CNN's use was obnoxious. Google for Amazon's pic-scan and search infrastructure, Instagram for product analysis for home feed personalization. For pattern recognition and image method problems, square neural network measurements are commonly used.

CNN is made up of the input, output, and hidden layers within the input and output layers. In any neural feed-forward network, the activation process and final convolution hide the inputs and

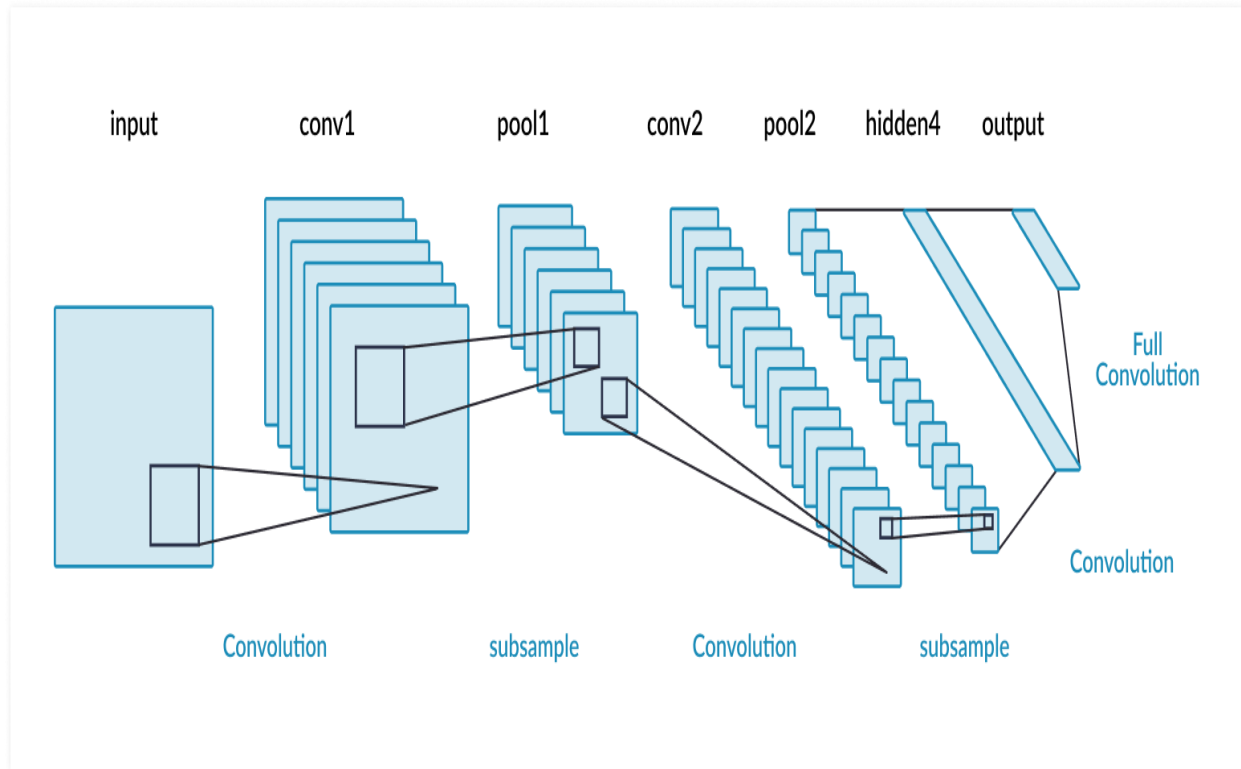
outputs of any intermediate-level area unit known as hidden. In an extremely convolutional neural network, the hidden layers embrace layers and perform convolutions. This typically includes the multiplying layer or other scalar items, which are usually ReLU enabled. Pooling layers, fully linked layers, and standardisation layers are examples of alternative layers.

Since CNN has parameter sharing and dimensionality reduction choices, it is a healthier network than a feed-forward network. When all neurons in a function map share weights, this is known as parameter sharing. This reduces the number of parameters in the whole method and makes the computation more cost-effective. Dimensionality reduction is a term that describes strategies for reducing the number of input variables in a dataset. Another reason CNN is superior is that it detects critical information without the need for human intervention.

Yann LeCun, a postdoctoral science researcher, introduced the first ground breaking neural networks, known as ConvNets, in the 1980s. LeCun engineered fictitious fame, a basic image recognition neural network, by Kunihiko Fukussima, a Japanese organisation affiliated with the United Nations, a few years ago. For the first version, referred to as LeNet, CNN will accept written numbers (after LeCun).

In our implementation of Convolutional networks, we use Lenet engineering, which is based on locally suitable fields and kernels, as well as sub-sampling or pooling for the invariance of several distortions. A typical convolutional neural network for digital classification and recognition is shown in Figure 3.4. The input layer provides images from the MNIST dataset, which are downloaded by available libraries. They are oriented and standardised. With local receptive fields, neurons may learn various optical characteristics such as terminals, aligned boundaries, and curves. The layers that follow extract these characteristics more precisely.





**Figure 3.4:** LeNet Architecture

### 3.6.2 LAYERS OF CNN

**1. Input:** The images which contain pixel values were given as input to the CNN.

**2. Convolutional layer:**

Convolution may be a linear process when running a convolutional neural network that is like a traditional neural network, which is the multiplication of weights collected using data. These strategies were created to multiply a two-dimensional weight list known as a filter or a kernel between the grade array of the input file and the filter. The filter is smaller than the file input, the size of the input filter can be compounded, and the filter can also be a scalar product.

The use of a non-filtered filter is intentional so the same filter can be extended by the input array multiple times at completely different input points. The filter is used consistently, particularly for any patch in the whole file half or filter size from left to right.

And if the filter is multiplied by the input field once will this be a single value. The result is a two-dimensional output series that represents the filtering of the data, as a filter is applied many times to the input array. As such, the function map is known as this operation's two-dimensional output chain.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

1	0	1
0	1	0
1	0	1

Filter / Kernel

**Figure 3.5:** Input and kernel

For instance, on the left side of Figure 3.5, the input of the convolution layer is the input image. The filter, also known as the kernel, is located at the right. We'll use these terms interchangeably. This can be called a 3x3 convolution thanks to its filter type. Unknown

space is called the receptive zone anywhere convolution happens. Because of the filter dimension, the receiving area is also 3x3.

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Input x Filter

4		

Feature Map

**Figure 3.6:** Feature map

The filter at the top left of the following function diagram represents the output of the convolution operation "4". The filter is then moved to the right and the process is repeated, with the output being added to the map. We'd like to go this route, and the blend results in the feature map. Because of the Non-Linearity of completely connected layer outputs, each value is passed to a function map like a ReLU after a working map has been drawn.

### Activation function:-

It determines whether or not a weighted total and a preference can trigger the new neurons. The objective of the activation is to provide nonlinearity to the neuron production.

### Rectified linear unit [ReLU]:

Perhaps linear activation or ReLU will be briefly updated piece by piece, which would directly yield feedback if it were positive. The default activation mechanism is the product of a process that uses it for coaching and achieves better results for several different forms of neural networks.



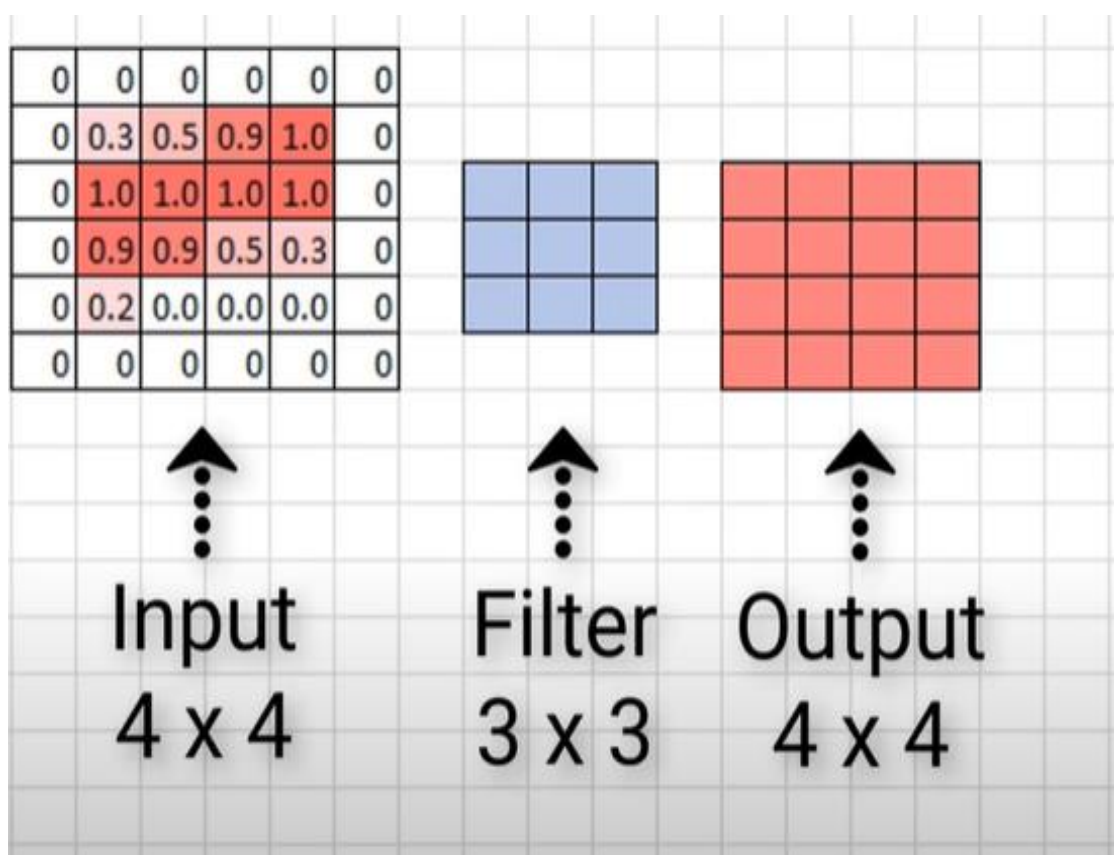
Figure 3.7: Convolution process

- It doesn't matter if the output here is smaller than the input (input size is 28\*28, output size 26\*26), after the convolution process. In the example above, the convolution layer is seven as an input.
- So if we have useful data at the edges of a picture then only one filter converts this image. This is when the output becomes smaller and smaller as the input is

convolutionized by more and more filters as it moves further. We have not lost data or something because more important data can be present in the middle but not at the ends.

- If a tiny image starts, the display will become useless after a layer of convolution. The filter is not as convolving as its input parts are converging we will loss our precious results.
- To overcome this problem, we use null padding. Zero padding: this technique allows you to keep the original input size. There is no padding when we apply a pixel strip with a zero value around the entry borders (outside of an image).

**Example:**



**Figure 3.8:** Zero padding

- The entry is passed by a 0 valued pixel boundary, a zero border is inserted by the size 4x4 and, as before, we have a 3x3-size filter.
- The initial 3x3-block dispositive in the entry are one by one on the right, and the other 3x3 blocks are then followed.
- The whole input block is continued through this phase. This allowed us to cross our 3x3 filter four times, and we also could use our filter four times by this method of padding.
- We can get 4x4 size performance through this operation. The input and output sizes of 4x4 are therefore the same. In order to preserve the initial input dimension, depending on the filter size and output size, we also have to add more than a boundary of zeroes.

### **3. Pooling layer:**

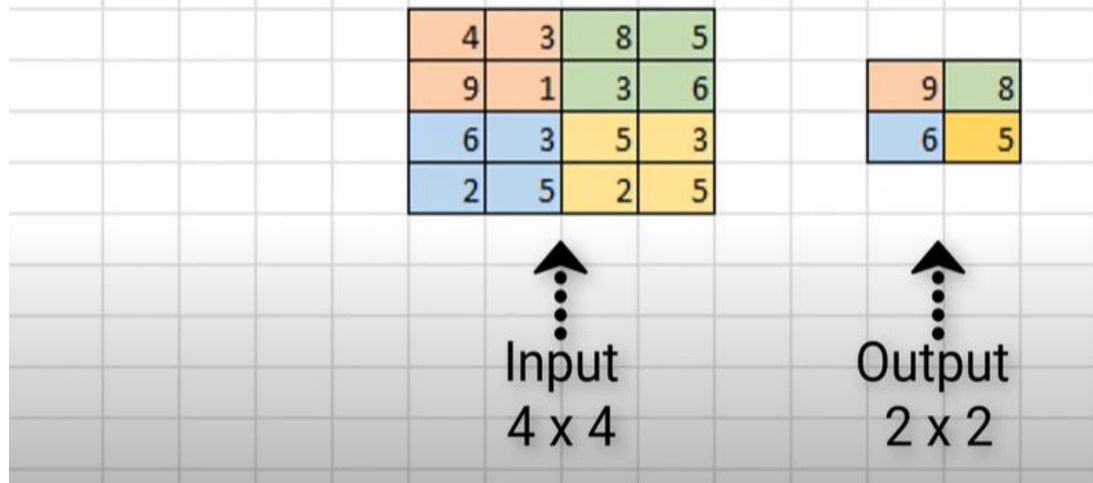
According to the convolutions, using a pooling layer to order layers in a convention neural network is a standard design that can be repeated once or more in a given model. On each feature map, a completely new set of identically grouped characteristics is handled individually. Similar to how a filter for feature maps operates, pooling involves deciding on a pooling operation. The filter operation or filter size is smaller than the standard map scale; in practise, a two-pixel string is almost always used with a filter operation of 2 pixels per two pixels.

To put it another way, a pooling layer reduces the size of a characteristic map by two elements, such as halving every dimension and halving the pixel or value sum on each characteristic map. A pooling layer applied to a 6 to 6 (36 pixel) feature map will result in a pooled 3 to 3 feature map performance, for example (9 pixels).

A pooling layer and the creation of down samples or clustered feature charts are two ways to summarise the features discovered by feedback. Minor changes in the location of the feature in the feedback observed through the convolutional layer result in a combined feature map with the feature in the same area, which is useful.

**Example:**

Filter size =  $2 \times 2$   
Stride = 2



**Figure 3.9:** Max pooling

- Max Pooling is a CNN Convolution operation for each layer. By decreasing pixel count when applied to a model Max Poolings reduces the image size.
- We can make filters of any size we like, and I've used 2x2 as an example.
- Then we'll set a stride, which is the number of pixels by which we want our filter to pass.
- We will take the first input area of 2x2 to find the maximum value in this 2x2 block for each of the values.
- We then save the value that is used in the max pooling process to recover all the result.
- The max value is determined and saved for the first 2x2 block.
- Afterwards, the number of pixels we have set out in our steps will move over the right side (here, the stride value is two, so we will move by 2).

- The values will then be saved with the same maximum pooling process as before. We will continue in the same direction until we reach the right edge.
- After that we step down and repeat a max pooling procedure by the number of pixels we defined in our step (Calculate the max value and store the value).
- We can see these numbers as pools, and we're going to call max pooling because we take the maximum values.
- This process is repeated for the whole input image and once finished, a new representation of the image will be displayed. This output is then used as a next layer input.

#### **4. Dropout layer:**

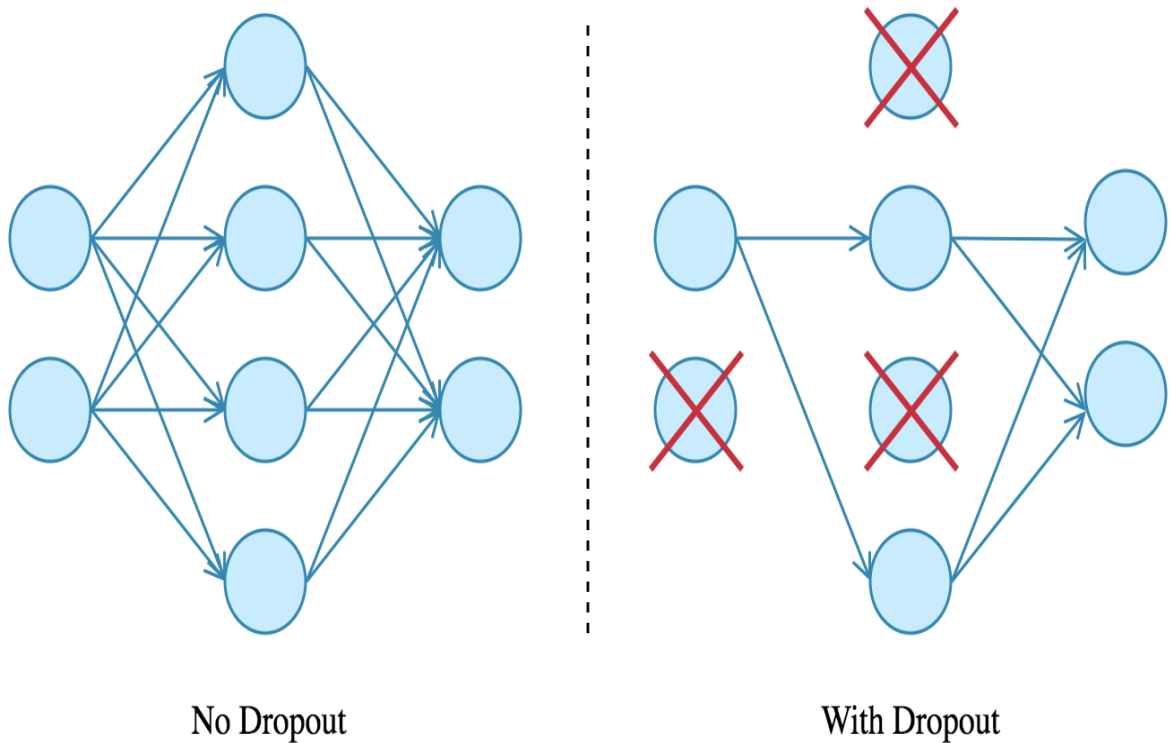
Deep neural networks with wide range of parameters for machine learning. However, it can be a huge disadvantage to over fit those networks. Overfitting happens after our training sets are classified or predicted that our models would be smart. The definition of non-qualified details is not sensitive, however. The large networks are slow to utilize and make it impossible to overrun the trauma because the checking time combines many massive network forecasts. Dropout can be a disadvantage management strategy. The central plan is to exclude the units during training from the neural network.

Dropout is considered to be the most general method of deep neural networks regularization. Even the most modern models with a precision of 95 percentage raise with just a 2-percent decrease.

To prevent overfitting, dropout is used, and the theory is clear. A "dropped-out" or damaged neuron is present briefly during planning for any iteration. This means that all inputs and outputs of the neuron are deactivated at the present iteration. Each step of training will test the dropped neurons with a probability  $p$  in order to activate the dropped neuron at one point at a time. The hyper parameter  $p$  is called the drop-out rate, normally approximately 0.5, which equates to 50% of the nerve.



We deliberately disable neurons and the network functions well. The reason for this is that network reduction prevents the dependence and functionality of too many neurons..



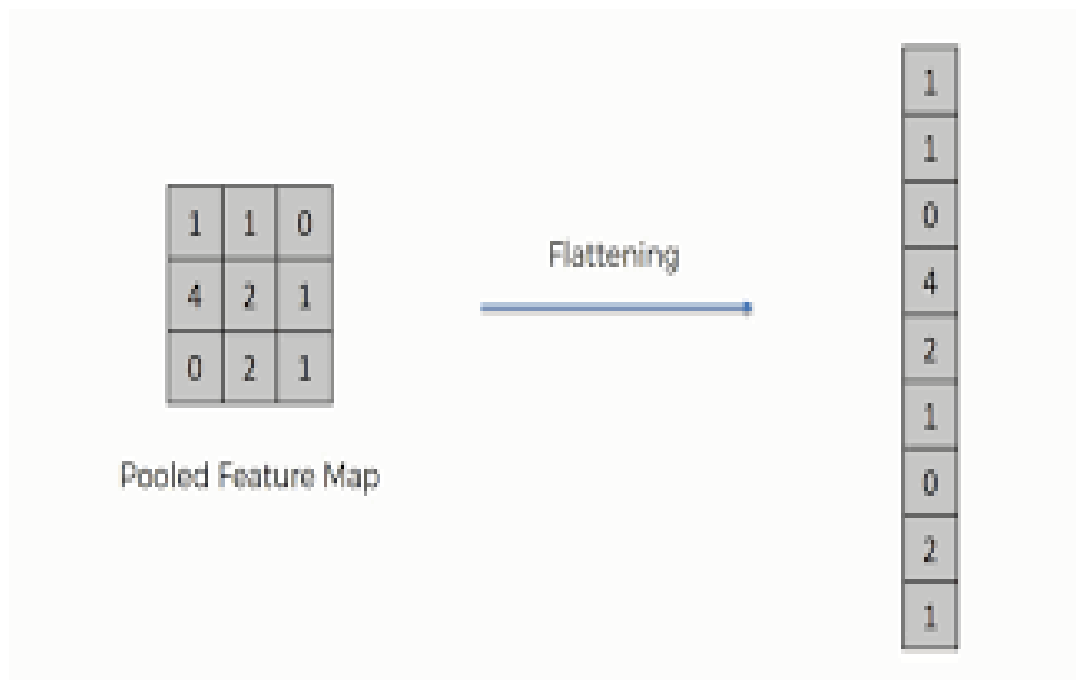
**Figure 3.10:** Dropping of neurons

Dropout is used to prevent overfitting and preparation is often convenient. At any iteration during workout, a neuron is temporarily "deleted" or disabled with probability  $P$ . This means that all inputs and outputs of the current neuron are deactivated during the current iteration. Downloaded neurons have been examined with probability  $p$  at every training stage so that a dropped-out neuron can take part in a future stage. The hyper parameter  $p$  is called the drop rate, which is usually between 0.5 and 50% of the neurons.

## 5. Flatten layer:

Convolution and pooling layers output 3D quantities, while a fully connected layer needs a 1D vector of numbers as input. As a result, we transform the output of the final pooling layer into the input of the fully connected layer.

The knowledge for input to the next layer is flattened into a one-dimensional series. We tend to flatten the convolutional layer contribution to have a long function vector. The ultimate classification model, known as a completely connected layer, is connected. To put it another way, all component data is gathered into a single line and linked to the highest layer.



**Figure 3.11:** Flattening

## **6. Fully connected layer:**

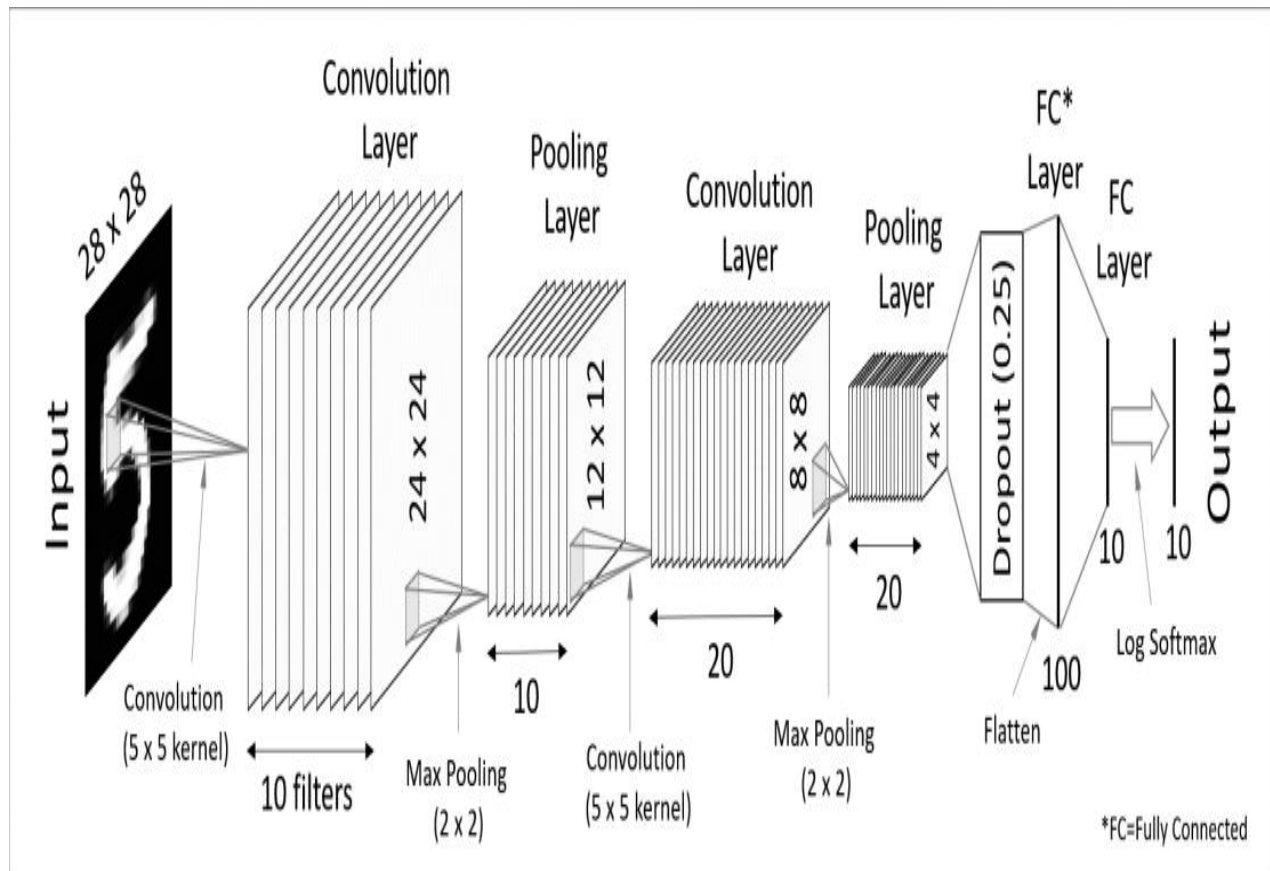
The input function vector is generated by the CNN's completely connected layer (FC). This function vector/tensor/layer contains information that is critical to the input. This vector is used to translate into another type of output for classification, regression, or input into another network like RNN once a network has been trained. As an embedded vector associate, it's being used together. This function vector is used in training to visualise the loss and encourage network training.

Information about local input features such as corners, blobs, forms, and so on has been provided in convolution layers until FC layers. Each layer has a number of filters that each represent a different characteristic of the region. All of the most significant convolutional layers' data is combined and aggregated in the FC layer.

The last pooling layer at the tip of a CNN is used as an input for the ostensibly fully linked layer. The properties of the previous layers that have been replaced are used to classify this layer. This layer is typically a regular CNN with a softmax activation that produces a likelihood for each mark predicted by the model.

### **Softmax classifier:**

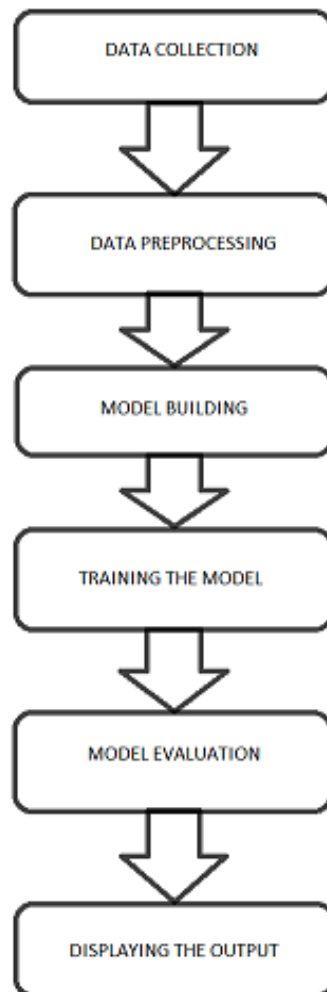
It returns all performance groups probabilities. It is used for performance standardization.



**Figure 3.12:** Layers of CNN

## CHAPTER 4

### DESIGN



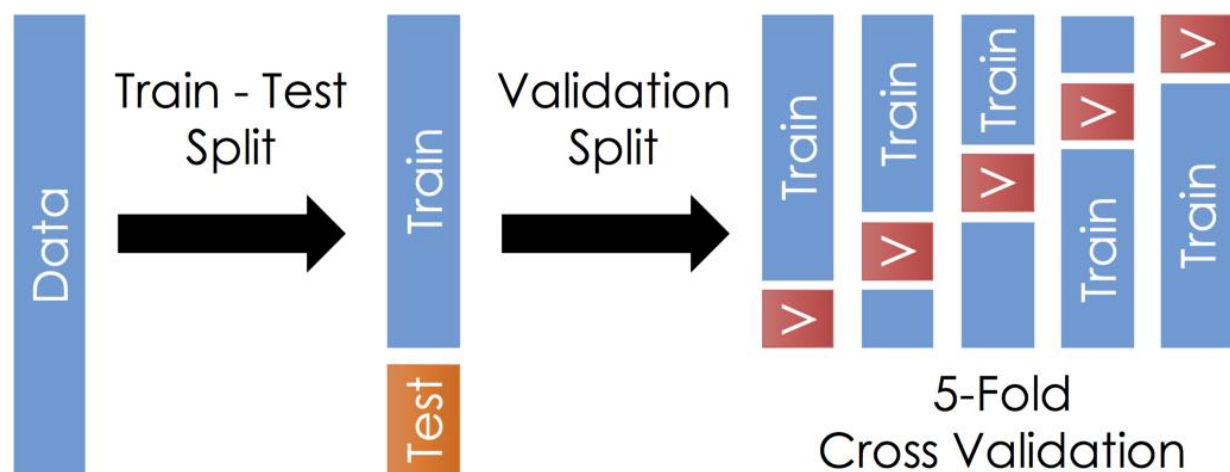
**Figure 4.1:** Flow chart

#### STEP 1: DATA COLLECTION

You must import the appropriate libraries and we must acquire and import the necessary data collection.

## STEP 2: SPLITTING THE DATASET

We divide the dataset into three parts: 'Training data,' 'Validation data,' and 'Testing data.' The 'training data set' is used to train the classifier, the 'validation set' is used to tune the parameters, and the 'test data set' is used to test the classifier's performance on unknown data. It's worth noting that only the training and validation sets are required while the classifier is being trained. The test data collection must not be used while the classifier is being trained. The test set will only be usable while the classifier is being tested.



**Figure 4.2:** Splitting the dataset

## STEP 3: DATA PREPROCESSING

In pre-processing data, the raw data for a learning model is prepared and adapted. This is the first step in building an apprenticeship model. Real-world data consist usually of sounds and missing values and cannot be used directly in an unusable format in machine learning models. In order to clean the data, data pre-processing is important to ensure it is suitable for a model of learning and to increase accuracy and efficiency of the model.

#### **STEP 4: MODEL BUILDING**

We must use the appropriate algorithm to construct a model. The dilemma is dependent on the selection of the algorithm.

#### **STEP 5: TRAINING THE MODEL**

At first, the data collection was divided into train and test data during the pre-processing phase. We'll train the model here with the data from the train.

#### **STEP 6: MODEL EVALUATION**

We will use the test data to test the model here. Then we will use a few metrics to test the model. Accuracy is one of the measures we used. Exactness is shown according to the proportion of accurate test data estimates. It is essentially determined by dividing the number of right forecasts by the total number of forecasts.

#### **STEP 6: WEB APPLICATION**

For better user access, a web application is available. As developers we will understand and use the code for our purposes, but interact with the user interface instead of code is more useful for other users. Now we are creating a web application and connecting to our learning model. The web program will now be entered by the user. On the web application screen the performance determined by the model attached to the web application is shown.

## **4.1 UML DIAGRAMS:**

The unified modeling language encourages the coder to use the modeling notation driven by syntactic-semantic and pragmatic principles to determine a particular research model. A UML scheme uses five different viewpoints, which interpret the system in a markedly different way. The following is the description of each view by a gaggle of diagrams.

### **User Model View**

From the consumer viewpoint, this view reflects the device. The presentation of the report explains a situation from the viewpoint of the end-user.

### **Structural Model**

Data and capabilities are provided from inside the device during this model, in this view. This model view shapes the static designs.

### **Behavioral Model**

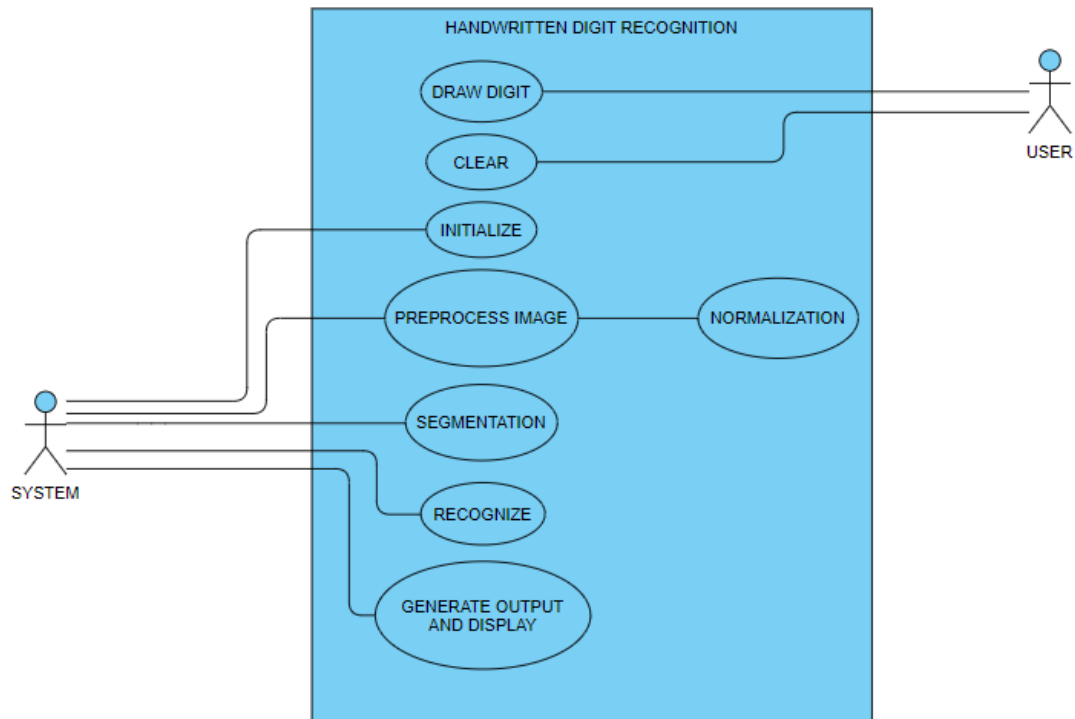
View It shows the dynamic of behavior as part of the structure, showing relationships within the user model between certain design elements and the vision of the structural model.

### **Implementation Model View**

This represents the structural and behavioral aspects of the structure as it is to be designed.

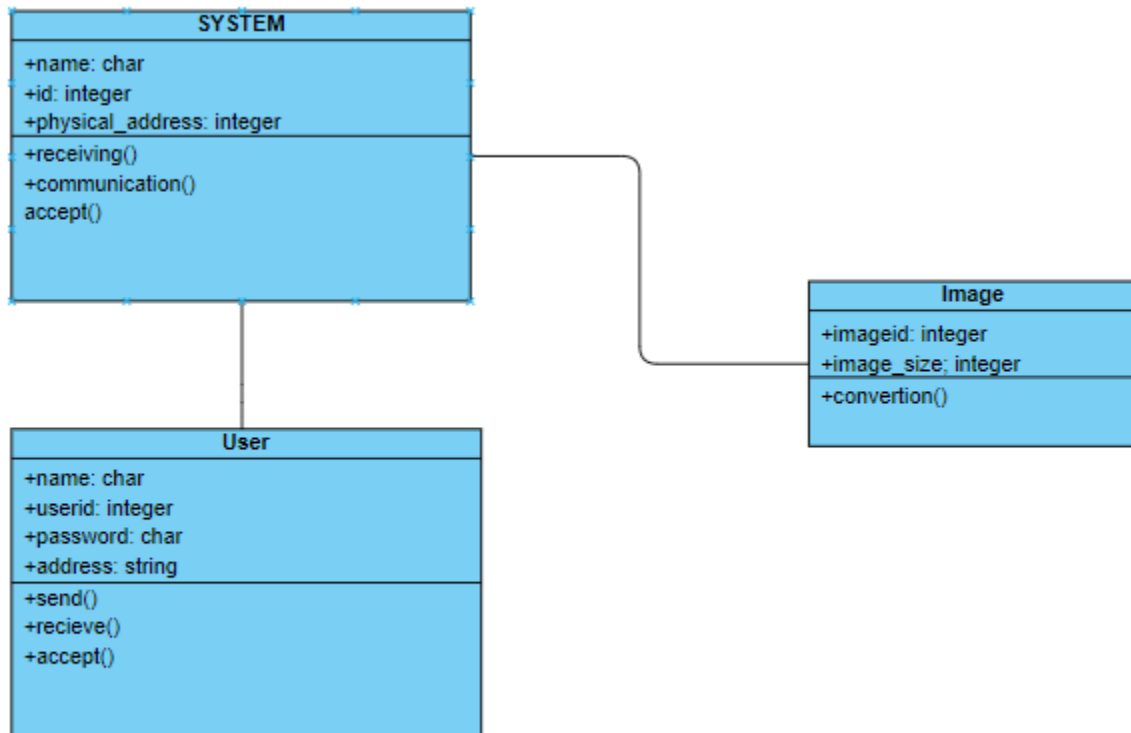


## USE CASE DIAGRAM:



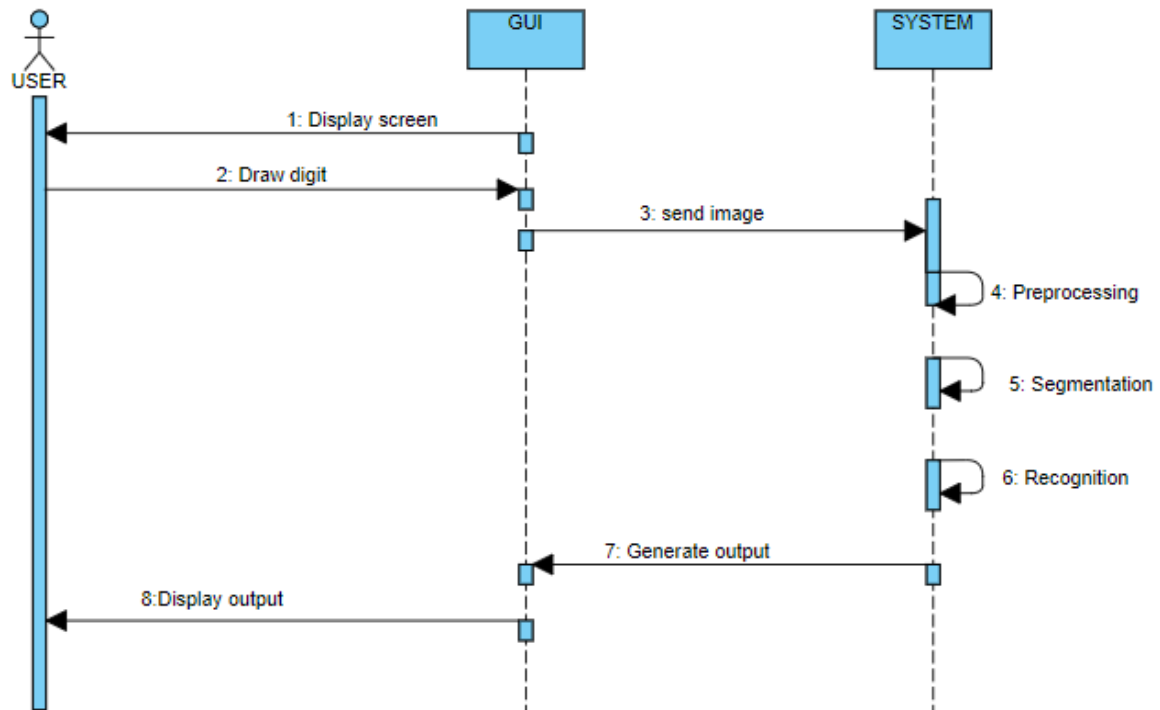
**Figure 4.3:** Use case diagram

## CLASS DIAGRAM:



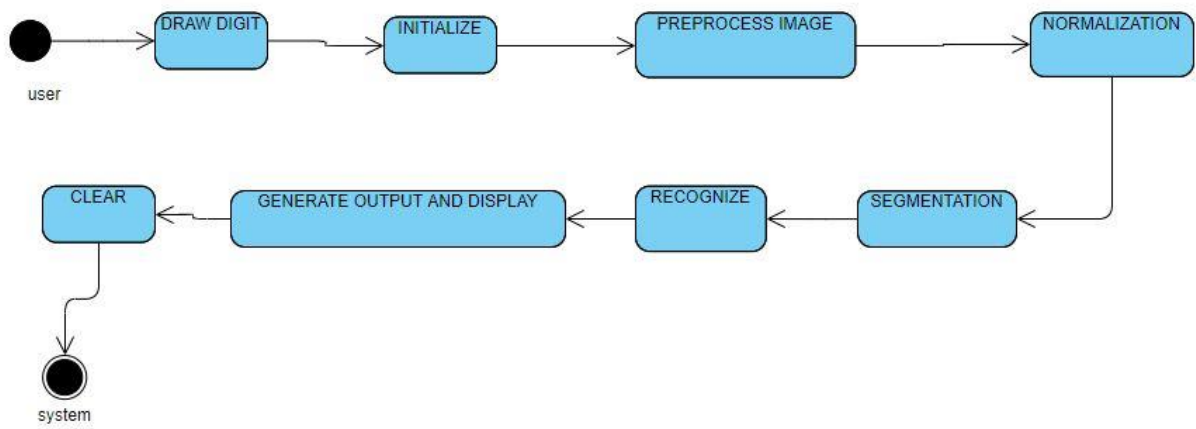
**Figure 4.4:** Class diagram

## SEQUENCE DIAGRAM:



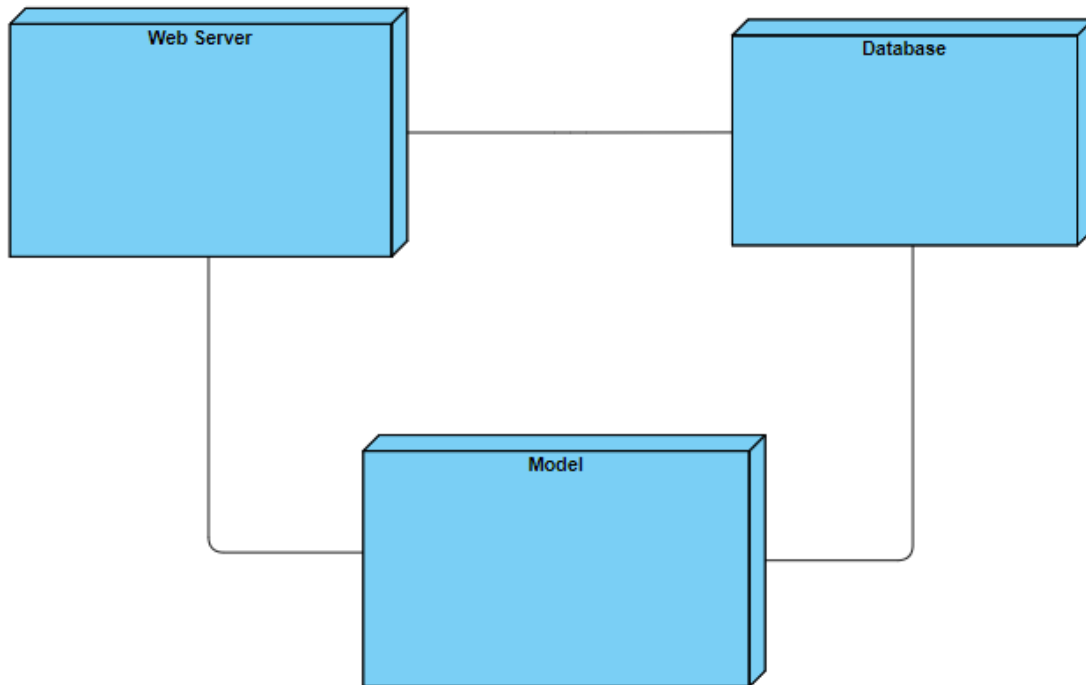
**Figure 4.5:** Sequence diagram

## STATE CHART DIAGRAM:



**Figure 4.6:** Statechart diagram

**DEPLOYMENT DIAGRAM:**



**Figure 4.7:** Deployment diagram

## CHAPTER 5

### IMPLEMENTATION

#### STEP 1: IMPORTING VARIOUS LIBRARIES

```
In [1]: import keras
        from keras.datasets import mnist
        from keras.models import Sequential
        from keras.layers import Dense, Dropout, Flatten
        from keras.layers import Conv2D, MaxPooling2D
```

**Figure 5.1:** Importing dataset & libraries

- ❖ We used the Keras library for our research. Keras is a Python library that can be used for profound learning by Theano or TensorFlow. Deep learning frameworks are designed for the quickest and most effective application in research and development.
- ❖ The dataset we used to train our model is the Mnist dataset predefined in the Keras library. So now we have imported the mnist dataset from the dataset package of Keras library.
- ❖ Here we have used a sequential model, which is a type of CNN model. So we have imported it from the models package of Keras library.
- ❖ And later on, in the CNN model, we have used five layers. They are convolutional layer, max-pooling layer, dropout layer, flatten layer, and dense layers. Now we have imported these five layers from the layers package of Keras library.

## STEP 2: DATA PREPROCESSING

- ❖ As our dataset is a predefined one, we can directly load it into our model using the `load_data()` function.

```
In [2]: (x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
In [3]: x_train.shape
```

```
Out[3]: (60000, 28, 28)
```

```
In [4]: x_test.shape
```

```
Out[4]: (10000, 28, 28)
```

```
In [5]: x_train.shape[0]
```

```
Out[5]: 60000
```

**Figure 5.2:** Loading the dataset

- ❖ While loading, the dataset is automatically split into four parts, say `x_train`, `x_test`, `y_train`, and `y_test`.
- ❖ The train part of the dataset represents the training images and training labels.
- ❖ The test part of the dataset represents the testing images and labels.
- ❖ And `x` represents images, whereas `y` represents.
- ❖ The MNIST dataset contains 70,000 images; these 70,000 images are split into 60,000 train images and 10,000 test images.
- ❖ We have checked the shape of `x_test` and `x_train` using the `shape()` function.
- ❖ The `x_test.shape` was given as (10000, 28, 28) and the `x_train.shape` was given as (60000, 28, 28).
- ❖ We realized that there are 60,000 train images and 10,000 test images by this shape function with 28x28 pixel size.

```
In [6]: x_train = x_train.reshape((x_train.shape[0], 28, 28, 1))
```

```
In [7]: x_test = x_test.reshape((x_test.shape[0], 28, 28, 1))
```

```
In [8]: x_train = x_train.astype('float32')  
x_test = x_test.astype('float32')
```

**Figure 5.3:** Reshaping the data

- ❖ The data of images cannot be immediately entered as input into the algorithm, so certain operations need to be performed and processed to prepare our neural network.
- ❖ The shape of the training has 3 variables, but the CNN requires another dimension, so we have reshaped our train data by adding the last dimension as “1”.
- ❖ The last one of the pixels for RGB is 3 that shows the components of red, green, and blue, and for each colored image will be like the input of three pixels. In MNIST, where the pixel values are grayscale, the pixel dimension is set to 1.
- ❖ Later on, we have to normalize the results, so train and test elements have been transformed into floating form.

### STEP 3: NORMALIZATION

```
In [10]: x_train = x_train / 255  
x_test = x_test / 255
```

**Figure 5.4:** Data normalization



- ❖ The pixel values of the images range between zero to 255 grayscale. It is almost always a good strategy to do input scaling for neural network models once. So we have quickly normalized the data by dividing the values by the maximum value that is 255. So now the values in x\_train are changed to the range zero to one.
- ❖ Normalization is commonly a method used as part of machine learning data processing. Normalization seeks to change numerical column values inside the dataset to use a standard scale without distorting the differences between the values or data losses.

```
In [9]: print(x_train[0])
```

```
[ 0.]
[ 0.]
[ 0.]
[ 0.]
[ 0.]
[ 3.]
[ 18.]
[ 18.]
[ 18.]
[126.]
[136.]
[175.]
[ 26.]
[166.]
[255.]
[247.]
[127.]
[ 0.]
```

**Figure 5.5:** Before normalizing

- ❖ Data before normalization is represented in figure. 5.5.
- ❖ And the data after normalization is represented in figure. 5.6.

```
In [11]: print(x_train[0])
[0.         ]
[0.         ]
[0.         ]
[0.01176471]
[0.07058824]
[0.07058824]
[0.07058824]
[0.49411765]
[0.53333336]
[0.6862745 ]
[0.10196079]
[0.6509804 ]
[1.         ]
[0.96862745]
[0.49803922]
[0.         ]
[0.         ]
```

**Figure 5.6:** After normalizing

```
In [12]: y_train
Out[12]: array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)

In [13]: y_train = keras.utils.to_categorical(y_train)
         y_test = keras.utils.to_categorical(y_test)

In [14]: y_train
Out[14]: array([[0., 0., 0., ..., 0., 0., 0.],
                [1., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 1., 0.]], dtype=float32)
```

**Figure 5.7:** One-hot encoding

- ❖ An example of categorical variables as binary vectors is a hot encoding. The values to be mapped to numeric values are initially required. Then of number value is imaged as a binary vector, all of which is zero except the integer index marked one.
- ❖ In this case, the output value is an integer ranging from 0 to 9. This is a sorting challenge for several classes. It is best to use the one-hot encoding of the class values, which involves converting the vector of class integers into a binary matrix.
- ❖ For example let us take the last value in the `y_train` that is 8 which is converted to binary row `[0,0,0,0,0,0,0,0,1,0]`. Let the value be `I`, then the  $(i+1)^{\text{th}}$  position in the binary row is filled with 1, and the remaining places are filled with 0's.

#### **STEP 4: MODEL BUILDING**

- ❖ Using the sequence model of the CNN algorithm we have developed our model. First of all; we declared and started a few variables for this purpose. They are batch size, number classes, input shapes and epochs.
- ❖ The batch size is regarded as a hyper parameter defining the number of samples to be processed before modifying the parameters of the internal model.
- ❖ Consider the batch as an iteration of one or more samples and predict.
- ❖ The estimates are matched with the predicted result variables at the tip of the batch, and a mistake is determined. This mistake is used to optimise the model using the algorithmic update programme.
- ❖ The number of classes included in the model is the number of classes. Our project is to divide the picture into numbers

```

In [16]: batch_size = 128
         num_classes = 10
         epochs = 10
         input_shape = (28, 28, 1)

In [17]: model = Sequential()
         model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
         model.add(Conv2D(64, (3, 3), activation='relu'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Dropout(0.25))
         model.add(Flatten())
         model.add(Dense(256, activation='relu'))
         model.add(Dropout(0.5))
         model.add(Dense(num_classes, activation='softmax'))

In [18]: model.compile(loss=keras.losses.categorical_crossentropy,
                       optimizer=keras.optimizers.Adam(),
                       metrics=['accuracy'])

```

**Figure 5.8:** Model building

- ❖ We've got ten numbers saying 0-9. So in our project we have 10 classes.
- ❖ The number of epochs is a hyper parameter which indicates how much a study algorithm will iterate over the whole research data set.
- ❖ Each sample can take at least one epoch in the training data set to make sure internal pattern parameters are modified. Any era consists of one or more lots. In our project there are ten epochs.
- ❖ A loop on the number of times that each loop yields over the training dataset can be considered. This for-loop is another for-loop, which iterates through all sample groups, the listed one lot has the requisite sample "batch size" set.
- ❖ The input shape variables describe the form of the input during the first layer.
- ❖ The photographs of the input shape vector are 28x28 pixels in dimension and grey scale (28,28,1).

- ❖ We used the CNN sequential model in this project. A single stack of layers is designed for a sequential model that has precisely one input tensor and one output tensor for each layer.
- ❖ And five layers we used. They are conv, max-pooling, dropout, flat and dense.
- ❖ We have first introduced a convolutional layer as the input layer, for which the neuron no. is 32 and the kernel size (3,3), and have given the activation function for RELU.
- ❖ We have then introduced another neuron no. layer with the same RELU activation function as 64 and kernel size (3,3).
- ❖ We have also added a pool-size max-pooling layer (2,2). A down-sampling process is performed along the measurements of the pooling sheet.
- ❖ Next, we added a dropout layer with a drop of 25% and a flattened layer.
- ❖ Later, we also added a dense layer with a RELU activation and 256 neurons and added another dropout layer with the same 25% drop-off.
- ❖ Finally, a dense layer of ten groups and a softmax classification has now been introduced. Softmax is a math function that converts a number vector into a probability vector. The odds of each value previously measured are equal to each value's relative size. And then the output is printed on the basis of the probabilities.
- ❖ We built our model using the categorical cross-entropy loss function and Adam optimizer following model construction.
- ❖ We used categorical cross-entropy here because of the encoded names. Cross Entropy is a calculation of how two probability distributions are calculated.
- ❖ Adam optimizer is the application of the Adam optimization algorithm. Adam is a stochastic approach to descent gradients, which calculates adaptive learning speeds for different parameters based on averages at moments of initial and secondary levels.

```

In [19]: hist = model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(x_test, y_test))
print("The model has successfully trained")
model.save('mnist.h5')
print("Saving the model as mnist.h5")

Epoch 1/10
469/469 [=====] - 91s 193ms/step - loss: 0.1916 - accuracy: 0.9410 - val_loss: 0.0450 - val_accuracy: 0.9850
Epoch 2/10
469/469 [=====] - 90s 191ms/step - loss: 0.0653 - accuracy: 0.9798 - val_loss: 0.0347 - val_accuracy: 0.9888
Epoch 3/10
469/469 [=====] - 91s 193ms/step - loss: 0.0464 - accuracy: 0.9851 - val_loss: 0.0316 - val_accuracy: 0.9889
Epoch 4/10
469/469 [=====] - 92s 195ms/step - loss: 0.0376 - accuracy: 0.9882 - val_loss: 0.0272 - val_accuracy: 0.9914
Epoch 5/10
469/469 [=====] - 88s 189ms/step - loss: 0.0290 - accuracy: 0.9905 - val_loss: 0.0283 - val_accuracy: 0.9908
Epoch 6/10
469/469 [=====] - 94s 200ms/step - loss: 0.0275 - accuracy: 0.9908 - val_loss: 0.0269 - val_accuracy: 0.9916
Epoch 7/10
469/469 [=====] - 99s 211ms/step - loss: 0.0224 - accuracy: 0.9928 - val_loss: 0.0266 - val_accuracy: 0.9918
Epoch 8/10
469/469 [=====] - 93s 198ms/step - loss: 0.0203 - accuracy: 0.9931 - val_loss: 0.0284 - val_accuracy: 0.9912
Epoch 9/10
469/469 [=====] - 93s 199ms/step - loss: 0.0181 - accuracy: 0.9941 - val_loss: 0.0255 - val_accuracy: 0.9924
Epoch 10/10
469/469 [=====] - 93s 199ms/step - loss: 0.0160 - accuracy: 0.9947 - val_loss: 0.0302 - val_accuracy: 0.9913
The model has successfully trained
Saving the model as mnist.h5

```

**Figure 5.9: Model training**

- ❖ The model which was built was later on trained with 60,000 images of the MNIST dataset and validated using 10,000 test images of the MNIST dataset. Here we have provided 10 epochs and verbose as one. After that the model was saved as mnist.h5.

## STEP 6: WEB APPLICATION.

- ❖ Moreover, we have created a web application to enhance user participation. The customer provides input to the Web application. The performance will be generated by the model linked to the web app and displayed on the web app screen.
- ❖ We have Spyder IDE for our web application. We used tkinter library in our web application implementation.

- ❖ Tkinter is the quality GUI closet for python. Python is a quick and practical way to construct Tkinter's GUI applications. Tkinter provides an object-focused interface with the powerful Tk GUI toolkit. In an excessively GUI program, the Tkinter offers a range of controls, such as buttons, labellings and text panes. This controls are usually referred to as widgets. Currently there are 15 different widgets in Tkinter. Tkinter is one of you most of the time, but there are a similar number of alternate modules. During the “\_tkinter” binary module the Tkinter interface was discovered. This module includes the low level interface to Tkinter which should not be used by application programmers. It still has a shared library, but can be statically connected to the python interpreter in some situations.

## CHAPTER 6

### TESTING

#### 6.1 ACCURACY

```
In [20]: score = model.evaluate(x_test, y_test, verbose=0)
          print('Test loss:', score[0])
          print('Test accuracy:', score[1])
```

Test loss: 0.030246563255786896  
Test accuracy: 0.9912999868392944

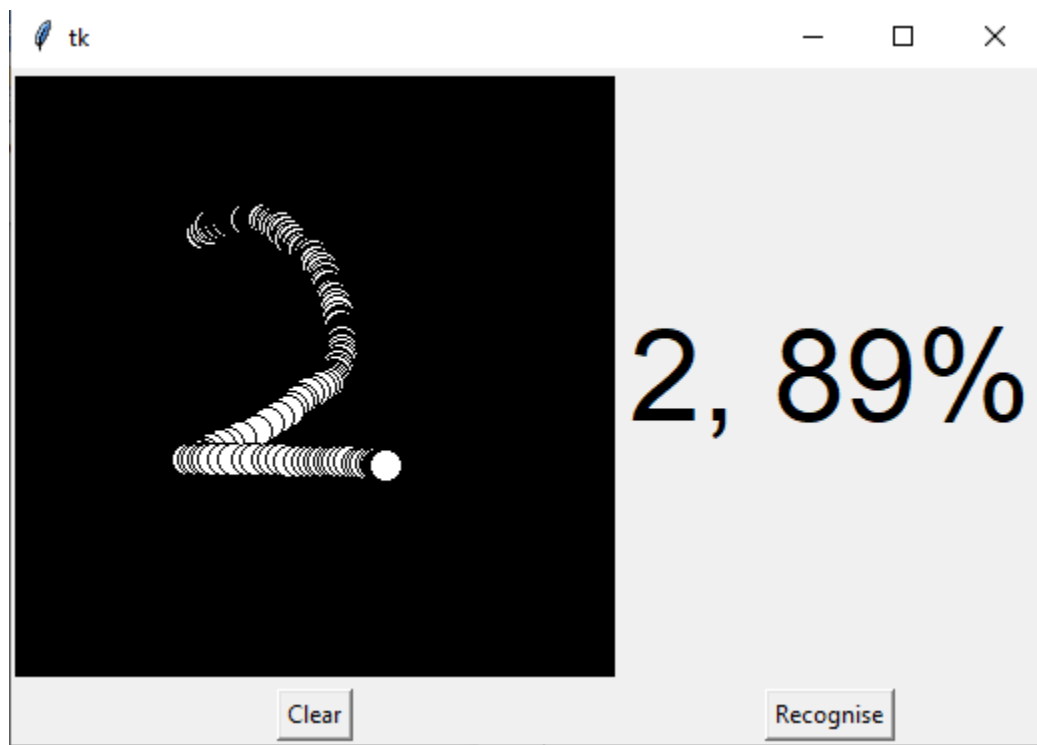
**Figure 6.1:** Testing

#### 6.2 WEB APPLICATION

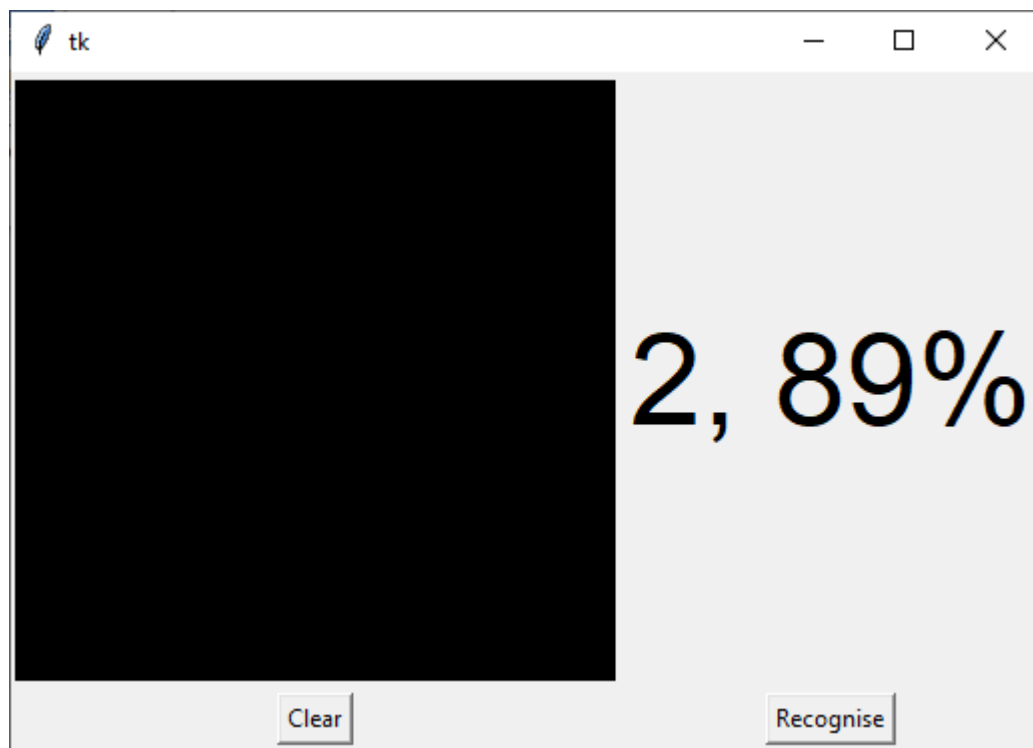


**Figure 6.2:** Web app





**Figure 6.3:** Recognize



**Figure 6.4:** Clear

## CONCLUSION

The performance of the network is calculated by different factors such as low memory requirements, short duration and better accuracy. The main objective of this paper, however, is to improve the precise definition. While artificial neurons are more precise, deeper learning features, such as convolutional neural networks, are now heavily used in computer vision. Research in this field is continuing and several different variants of LeNet architecture such as LeNet-1, LeNet-4, LeNet-4 Boost and KNN's are being built by researchers. Yet our LeNet architecture has long been regarded as state-of-the-art. A number of additional approaches, including Tangent Distance Classifier, were developed using the LeNet architecture. This project's primary objective is to address one of its methods. They can be done in different ways with different systems like the mat laboratory and the octave. The primary objective of the artificial intelligence machine vision division is to build a stronger network for any performance measure and to provide results for any kind of dataset that can be studied, educated and remembered.

## **FUTURE SCOPE**

In different applications, fixed-size Convolutional Networks (CNNs) have been used, such as manuscript digital identification, machine-printed character identification and online handwriting recognition. CNN may also be used for signature verification. The more training simulations, the higher the networking results. Unattended deep learning has been enabled by revolutionary neural networks. CNN's future work involves compressing or generating the same results through optimization tricks and more continuous functional learning from smaller networks. There is no fluttering of the input images. The major 3D vision networks can be studied using LeNet design and bio conformity methods; the hope of the future is unchecked by the CNN.

## REFERENCES

- [1]T SIVA AJAY-Handwritten Digit Recognition Using Convolutional Neural Networks [2017].
- [2]Jinze Li, Gongbo Sun, Leiye Yi, Qian Cao, Fusen Liang, Yu Sun- Handwritten Digit Recognition System Based on Convolutional Neural Network[2020].
- [3]Yellapragada SS Bharadwaj, Rajaram P, Sriram VP, Sudhakar S, Kolla Bhanu Prakash- Effective Handwritten Digit Recognition using Deep Convolution Neural Network[2020].
- [4]Vijayalaxmi R Rudraswamimath, Bhavanishankar K.- Handwritten Digit Recognition using CNN[2019].
- [5]Anchit, Shrivastava, Isha Jaggi, Sheifali Gupta, Deepali Gupta.- Handwritten Digit Recognition Using Machine Learning: A Review[2019].
- [6]S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair.- Handwritten Digit Recognition using Machine Learning Algorithms[2018].
- [7]Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique.- Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers[2018].
- [8]Mahmoud M. Abu Ghosh, Ashraf Y. Maghari.- A Comparative Study on Handwriting Digit Recognition Using Neural Networks[2017].
- [9]Ernst Kussul, Tatyana Baidyk- Improved Method of Handwritten Digit Recognition [2017].
- [10]Anuj Dutt, AashiDutt- Handwritten Digit Recognition Using Deep Learning [2017].

## APPENDIX

### ML CODE:

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
```

```
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
```

```
X_train.shape
```

```
OUTPUT: (60000, 28, 28)
```

```
X_test.shape
```

```
OUTPUT: (10000, 28, 28)
```

```
X_train.shape[0]
```

```
OUTPUT: 60000
```

```
X_train = X_train.reshape((X_train.shape[0], 28, 28, 1))
```

```
X_test = X_test.reshape((X_test.shape[0], 28, 28, 1))
```

```
X_train = X_train.astype('float32')
```

```
X_test = X_test.astype('float32')
```

```
print(X_train[0])
```

OUTPUT:

```
[ 0.]
[ 0.]
[ 0.]
[ 0.]
[ 3.]
[ 18.]
[ 18.]
[ 18.]
[126.]
[136.]
[175.]
[ 26.]
[166.]
[255.]
[247.]
[127.]
[ 0.]
[ 0.]
[ 0.]
```

`X_train = X_train / 255`

`X_test = X_test / 255`

`print(X_train[0])`

OUTPUT:

```
[0.          ]
[0.          ]
[0.01176471]
[0.07058824]
[0.07058824]
[0.07058824]
[0.49411765]
[0.53333336]
[0.6862745 ]
[0.10196079]
[0.6509804 ]
[1.          ]
[0.96862745]
[0.49803922]
[0.          ]
[0.          ]
```

Y\_train

```
OUTPUT: array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

```
Y_train = keras.utils.to_categorical(Y_train)
```

```
Y_test = keras.utils.to_categorical(Y_test)
```

Y\_train

```
OUTPUT: array([[0., 0., 0., ..., 0., 0., 0.],
               [1., 0., 0., ..., 0., 0., 0.],
               [0., 0., 0., ..., 0., 0., 0.],
               ...,
               [0., 0., 0., ..., 0., 0., 0.],
               [0., 0., 0., ..., 0., 0., 0.],
               [0., 0., 0., ..., 0., 1., 0.]], dtype=float32)
```

```
print('X_train shape:', X_train.shape)
```

```
print('Y_train shape:', Y_train.shape)
```

```
print(X_train.shape[0], 'train samples')
```

```
print(X_test.shape[0], 'test samples')
```

```
OUTPUT: X_train shape: (60000, 28, 28, 1)
```

```
Y_train shape: (60000, 10)
```

```
60000 train samples
```

```
10000 test samples
```

```
batch_size = 128
```

```
num_classes = 10
```

```
epochs = 10
```

```
input_shape = (28, 28, 1)
```

```
cnnmodel = Sequential()
```

```
cnnmodel.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
```

```
cnnmodel.add(Conv2D(64, (3, 3), activation='relu'))
```



```

cnnmodel.add(MaxPooling2D(pool_size=(2, 2)))
cnnmodel.add(Dropout(0.25))
cnnmodel.add(Flatten())
cnnmodel.add(Dense(256, activation='relu'))
cnnmodel.add(Dropout(0.5))
cnnmodel.add(Dense(num_classes, activation='softmax'))
cnnmodel.compile(loss=keras.losses.categorical_crossentropy,
                  optimizer=keras.optimizers.Adam(),
                  metrics=['accuracy'])

hist = cnnmodel.fit(X_train,
                    Y_train, batch_size=batch_size, epochs=epochs, verbose=1, validation_data=(X_test, Y_test))
print("The model has successfully trained")
model.save('mnist.h5')
print("Saving the model as mnist.h5")

```

## OUTPUT:

```

Epoch 1/10
469/469 [=====] - 109s 232ms/step - loss: 0.1955 - accuracy: 0.9410 - val_loss: 0.0461 - val_accuracy:
0.9843
Epoch 2/10
469/469 [=====] - 102s 217ms/step - loss: 0.0652 - accuracy: 0.9801 - val_loss: 0.0357 - val_accuracy:
0.9875
Epoch 3/10
469/469 [=====] - 104s 221ms/step - loss: 0.0473 - accuracy: 0.9858 - val_loss: 0.0307 - val_accuracy:
0.9901
Epoch 4/10
469/469 [=====] - 99s 211ms/step - loss: 0.0370 - accuracy: 0.9883 - val_loss: 0.0343 - val_accuracy:
0.9883
Epoch 5/10
469/469 [=====] - 93s 198ms/step - loss: 0.0318 - accuracy: 0.9896 - val_loss: 0.0287 - val_accuracy:
0.9898
Epoch 6/10
469/469 [=====] - 98s 209ms/step - loss: 0.0262 - accuracy: 0.9915 - val_loss: 0.0291 - val_accuracy:
0.9909
Epoch 7/10
469/469 [=====] - 103s 220ms/step - loss: 0.0219 - accuracy: 0.9925 - val_loss: 0.0289 - val_accuracy:
0.9906
Epoch 8/10
469/469 [=====] - 92s 196ms/step - loss: 0.0204 - accuracy: 0.9928 - val_loss: 0.0299 - val_accuracy:
0.9910
Epoch 9/10
469/469 [=====] - 96s 206ms/step - loss: 0.0185 - accuracy: 0.9936 - val_loss: 0.0267 - val_accuracy:
0.9927
Epoch 10/10
469/469 [=====] - 91s 194ms/step - loss: 0.0173 - accuracy: 0.9943 - val_loss: 0.0291 - val_accuracy:
0.9914
The model has successfully trained
Saving the model as mnist.h5

```

```

score = cnnmodel.evaluate(X_test, Y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
OUTPUT: Test loss: 0.02910645119845867
Test accuracy: 0.9914000034332275

```

## WEB APPLICATION:

```

from keras.models import load_model
from tkinter import *
import tkinter as tkrr
import win32gui
from PIL import ImageGrab, Image
import numpy as np
model = load_model('mnist.h5')
def predictdigit(img):
    img = img.resize((28,28))
    img = np.array(img)
    img = img.reshape(1,28,28,1)
    img = img/255.0
    result = model.predict([img])[0]
    return np.argmax(result), max(result)
class App(tkrr.Tkr):
    def __init__(self):
        tkrr.Tkr.__init__(self)
        self.X = self.Y = 0
        self.canvas = tkrr.Canvas(self, width=300, height=300, bg = "black", cursor="cross")
        self.label = tkrr.Label(self, text="Thinking..", font=("Helvetica", 48))
        self.classify_btn = tkrr.Button(self, text = "Recognise", command =
self.classifyhandwriting)
        self.button_clear = tkrr.Button(self, text = "Clear", command = self.clearall)

```

```

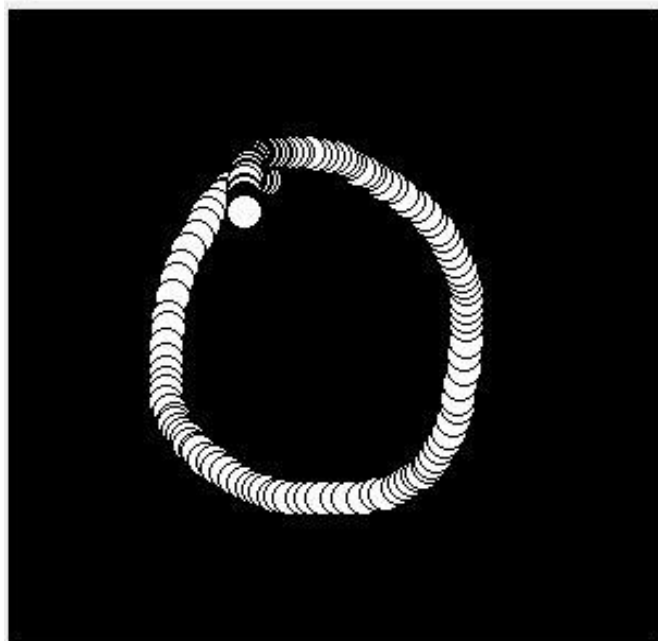
self.canvas.grid(row=0, column=0, pady=2, sticky=W, )
self.label.grid(row=0, column=1, pady=2, padx=2)
self.classify_btn.grid(row=1, column=1, pady=2, padx=2)
self.button_clear.grid(row=1, column=0, pady=2)
self.canvas.bind("<B1-Motion>", self.drawlines)
def clearall(self):
    self.canvas.delete("all")
def classifyhandwriting(self):
    hand = self.canvas.winfo_id()
    rectangle = win32gui.GetWindowRect(hand)
    image = ImageGrab.grab(rectangle)
    digit, acc = predictdigit(image)
    self.label.configure(text= str(digit)+' '+ str(int(acc*100))+'%')
def drawlines(self, event):
    self.X = event.X
    self.Y = event.Y
    R=8
    self.canvas.create_oval(self.X-R, self.Y-R, self.X + R, self.Y + R, fill='white')
app = App()
mainloop()

```

OUTPUT:



tk

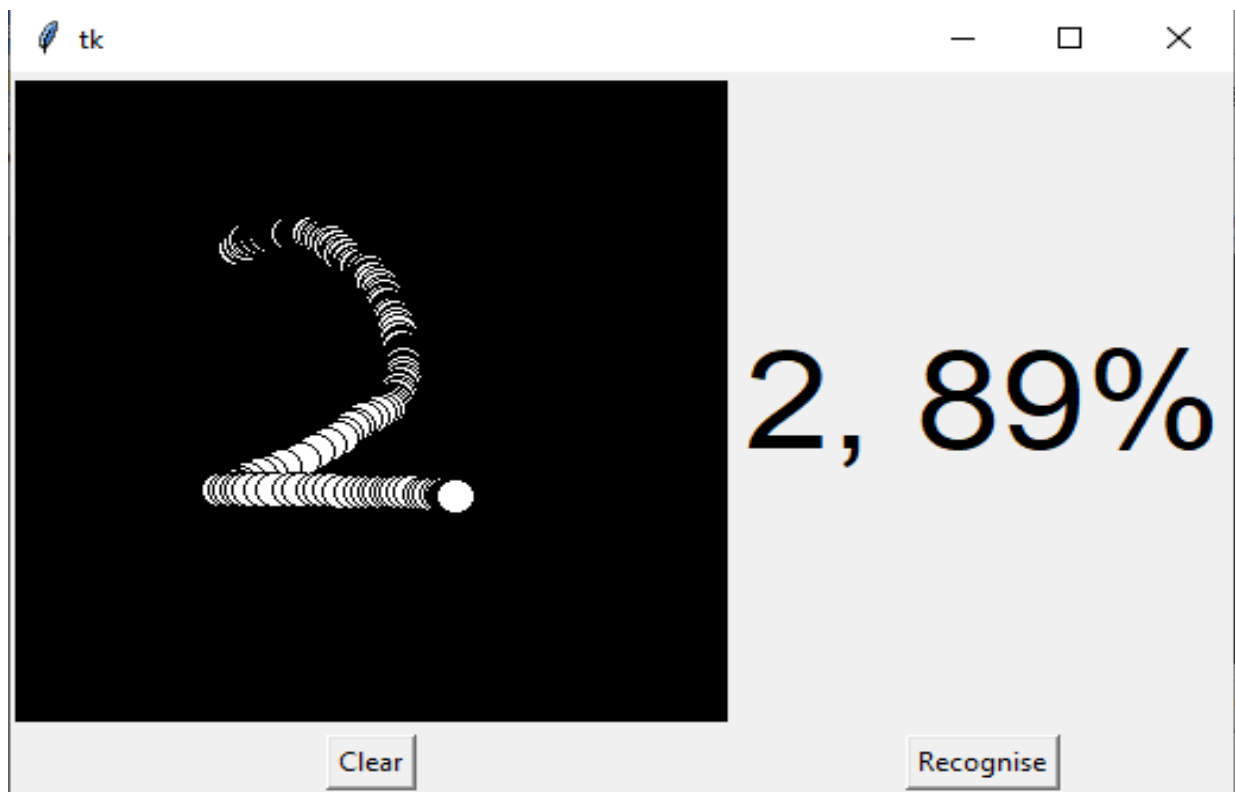


Clear

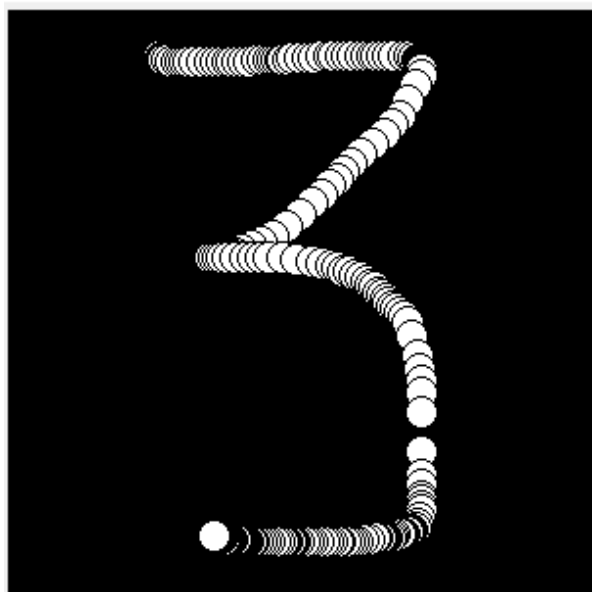
Recognise

0, 99%





tk



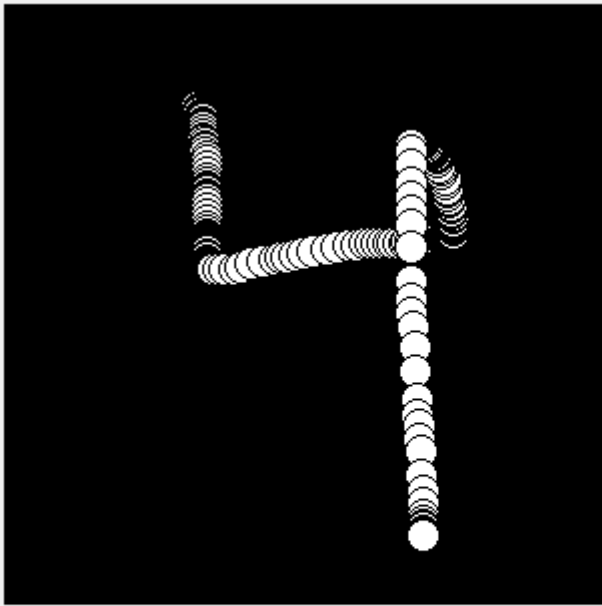
Clear

3, 96%

Recognise



tk

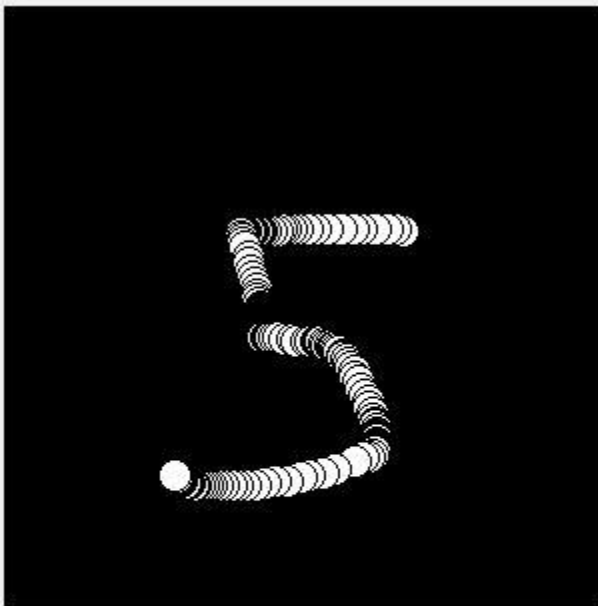


Clear

4, 56%

Recognise

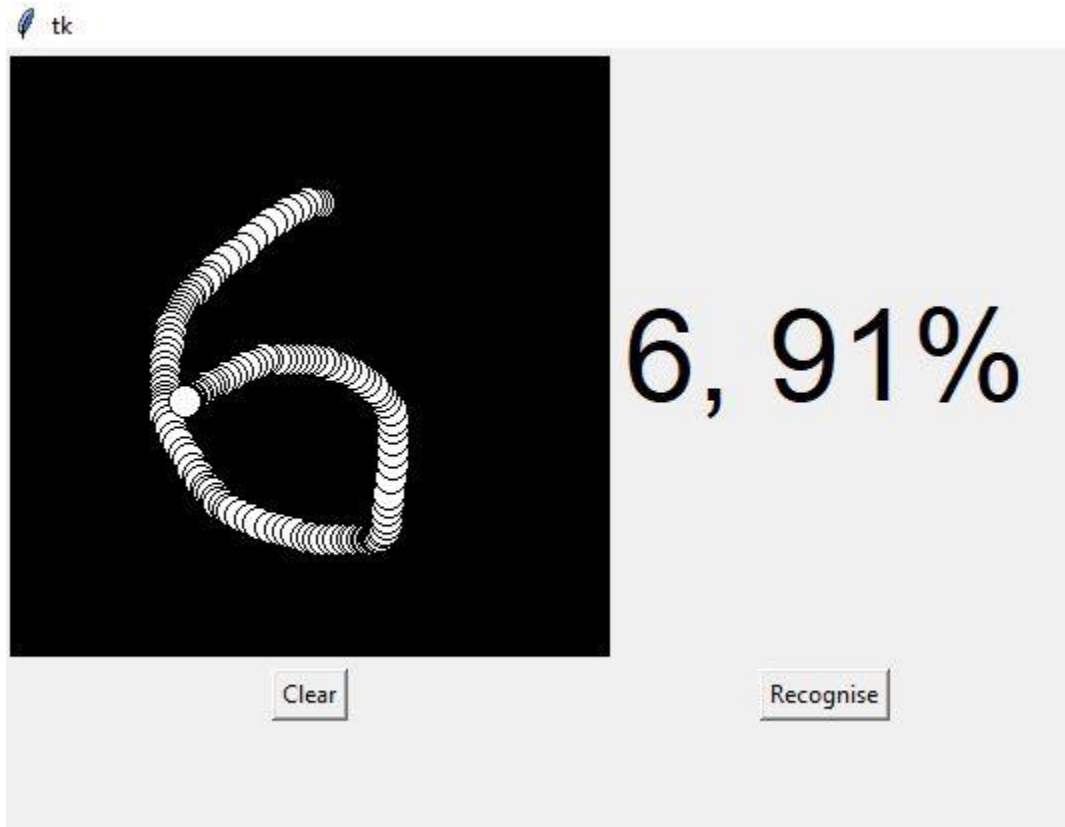
tk



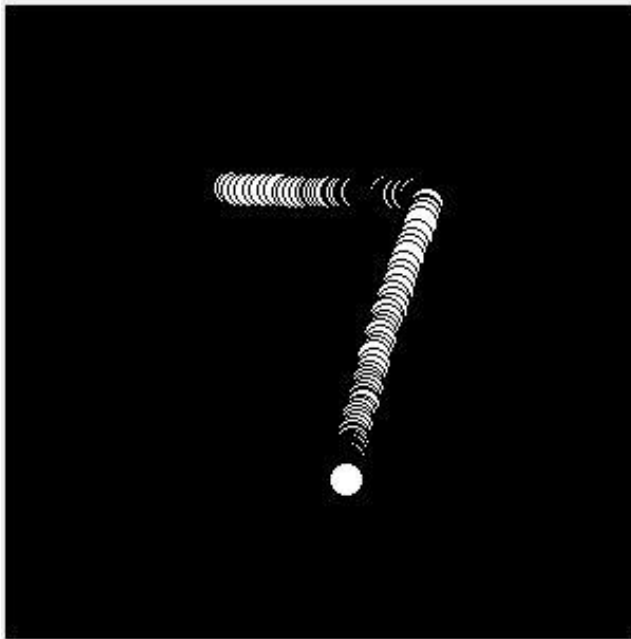
5, 81%

Clear

Recognise



tk

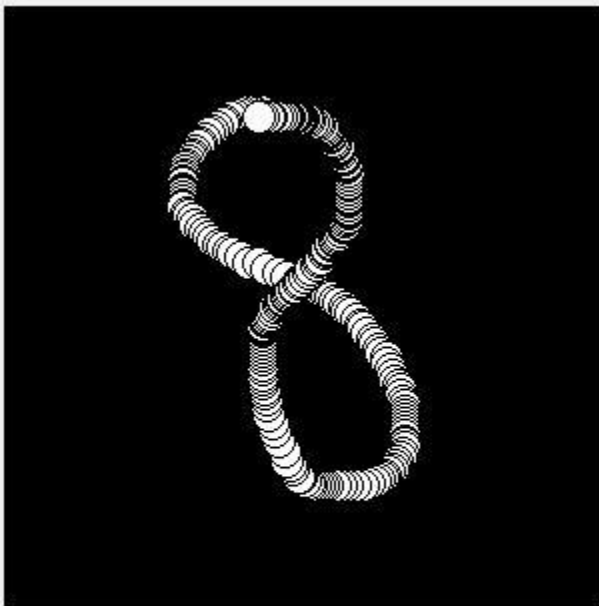


7, 72%

Clear

Recognise

tk

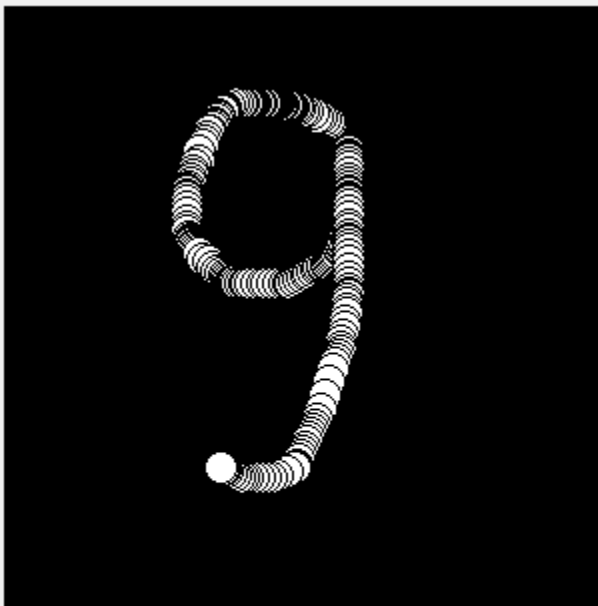


8, 93%

Clear

Recognise

tk



Clear

9, 30%

Recognise

