

HAND WRITTEN DIGIT RECOGNITION USING MACHINE LEARNING AND DEEP LEARNING ALGORITHMS

The Project Report submitted in partial fulfilment of the requirements for the award of the Degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND SYSTEMS ENGINEERING

Submitted by

AKUNDI SAHITHI	317114110001
ARUGULA GAYATHRI	317114110002
BALAKA SUPRAJA	317114110003
BATHINA LAKSHMI RISHITHA	317114110004

Under the esteemed guidance of

Prof. B. PRAJNA

Professor

Head of Department

Department of Computer Science and Systems Engineering



**DEPARTMENT OF COMPUTER SCIENCE AND SYSTEMS
ENGINEERING**

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN

Visakhapatnam

CERTIFICATE

ANDHRA UNIVERSITY COLLEGE OF ENGINEERING FOR WOMEN

VISAKHAPATNAM



This is to certify that the main project entitled “**Hand written digit recognition using Machine Learning and Deep Learning Algorithms**” is the bonafide work carried out by **A. SAHITHI (317114110001)**, **A. GAYATHRI (317114110002)**, **B. SUPRAJA (317114110003)**, **B. L. RISHITHA (317114110004)** submitted in the partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology in Computer Science and Systems Engineering during March 2021 – June 2021.

Project Guide

Prof. B. Prajna

Professor

Head of the Department

Department of CS & SE

Head of the Department

Prof. B. Prajna

Professor

Head of the Department

Department of CS & SE

ACKNOWLEDGEMENT

We take this opportunity to express deep gratitude to the people who have been instrumental in the successful completion of the project.

We would like to thank our project guide **Prof. B. Prajna**, Professor, Head of the Department, Computer Science and Systems Engineering, for the supervision and support, which was greatly helpful in the progression and smoothness of the entire project work. We would like to show our gratitude to our beloved Principal, **Prof. S. K. Bhatti** for her encouragement in the aspect of our course, Andhra University College of Engineering for Women, who gave us official support for the progress of our project.

We would also like to extend our gratitude to Lab Technicians and our sincere gratitude towards all the faculty members and Non-Teaching Staff in the Department of Computer Science and Systems Engineering for supporting us. We would also like to thank our parents and friends, who have willingly helped us out with their abilities in completing the project work.

With gratitude,

A. SAHITHI	317114110001
A. GAYATHRI	317114110002
B. SUPRAJA	317114110003
B. L. RISHITHA	317114110004

DECLARATION

We hereby declare that the project work entitled “**Handwritten digit recognition using Machine Learning and Deep Learning Algorithms**”, is a bonafide work done by us, under the esteemed guidance of Prof. B. Prajna, Professor, Head of Department of CS&SE, **Andhra University College of Engineering for Women**. This project report is being submitted in the partial fulfilment of the requirements for the award of Degree of Bachelor of Technology in Computer Science and Systems Engineering during the academic year 2020-2021. This project possesses originality as it is not extracted from any source and it has not been submitted to any other institution or university.

NAME	REGD. NO	SIGNATURE
A. SAHITHI	317114110001	
A. GAYATHRI	317114110002	
B. SUPRAJA	317114110003	
B. L. RISHITHA	317114110004	

Visakhapatnam

Date:

INDEX

S.no	Title	Page no.
	ABSTRACT	i
	LIST OF TABLES	ii
	LIST OF FIGURES	iii-iv
	LIST OF ABBREVIATIONS	v
1.	Introduction	1-4
1.1	Project Overview	3-4
1.2	Project Deliverables	4
1.3	Project Scope	4
2.	Review of Literature	5
3.	Problem Analysis	6
3.1	Proposed System	6
3.2	Advantages	6
3.3	Limitations	6
4.	System Analysis	7-12
4.1	System Requirement Specifications	7-8
4.2	Feasibility Study	8-10
4.3	Scenarios	10-11
4.3.1	Use Case Diagram	10
4.4	System Requirements	12
5.	System Design	13-30
5.1	Introduction	13
5.2	UML Diagrams	13-21
5.2.1	Sequence Diagram	13-16
5.2.2	State Chart Diagram	17-19
5.2.3	Activity Diagram	20-21
5.3	System Architecture	22-29
5.4	User Interface	29-30
6.	System Implementation	31-37
6.1	Technology Description	31-35
6.2	System Modules	35-36
6.3	Sample Code	36-37
7.	Testing	38-41
7.1	Introduction	38-41
7.2	Test cases	41
8.	Output	42-47
9.	Conclusion	48
10.	Future Scope	49

ABSTRACT

Digitalisation has become prominent in today's world. The need for digitization today whether or not you are in the information technology is high. The need for storing the information in computers is increasing. Converting the handwritten documents into digital form by humans is often difficult and time consuming. With the rapid development of technology human's reliance on machines to do the time-consuming tasks also greatly increased.

Machine learning and Deep learning are the major fields in Computer Science which have developed intelligent algorithms to train machines to do a set of repetitive tasks. Handwritten digit recognition is one of the significant areas of research and development with a streaming number of possibilities that could be attained.

Handwritten Digit Recognition is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. This project illustrates handwritten digit recognition with the help of MNIST datasets using Support Vector Machines (SVM) and Convolution Neural Network (CNN) models. The main objective of this paper is to compare the accuracy of the models stated above and develop a Graphical User Interface (GUI) application with the most accurate model.

LIST OF TABLES

S. No	Table Name	Page No.
1	Graphical Representation of Use Case Diagram	10
2	Graphical Notations for Sequence Diagram	15
3	Graphical Notations for State Chart Diagram	18
4	Test Cases Representation	41

LIST OF FIGURES

S. No	Figure Name	Page no.
1	Handwritten Digits Sample	2
2	Phases of Handwritten Digit Recognition using ML	3
3	Plotting of some MNIST dataset digits	8
4	Graphical representation of Use Case diagram	10
5	Use Case diagram	11
6	Graphical Notations for Sequence Diagram	15
7	Sequence diagram	16
8	Graphical Notations for State Chart diagram	18
9	State Chart diagram	19
10	Flow of control of Activity diagram	20
11	Activity diagram	21
12	System Architecture	22
13	Support Vector Machine	23
14	Flowchart representing SVM	24
15	Convolution Neural Network	26
16	Flowchart representing CNN	27
17	Opening Jupyter Notebook	42
18	Extracting dataset	42
19	Exploring data distribution in MNIST dataset	43
20	Accuracy of SVM	43
21	Accuracy of CNN	43
22	Confusion Matrix of SVM	44
23	Confusion Matrix of CNN	44

24	Graph illustrating train Vs test loss and accuracy	45
25	GUI interface	45
26	Sample prediction-1	46
27	Sample prediction-2	46
28	Sample prediction-3	47
29	Sample prediction-4	47

LIST OF ABBREVIATIONS

- | | | |
|----------|---|--|
| 1. AI | - | Artificial Intelligence |
| 2. ML | - | Machine Learning |
| 3. DL | - | Deep Learning |
| 4. SVM | - | Support Vector Machine |
| 5. CNN | - | Convolution Neural Network |
| 6. MNIST | - | Modified National Institute of Standards and Technology database |
| 7. GUI | - | Graphical User Interface |

1. INTRODUCTION

Intelligent image processing is an enticing study area in Artificial Intelligence it is also essential for a range of existing accessible research challenges. **Hand-written Digit Recognition** is a well-researched sub-area of the field that discusses the detection of pre-segmented hand-written digits with learning models. It is along with several other disciplines in artificial intelligence, one of the most critical issues of machine learning, data retrieval, deep learning, and pattern recognition. The major application of machine learning approaches has been effective over the last decade in conforming to definitive systems that compete with human performance and perform substantially better than traditional artificial learning methods built manually. Moreover, not all the aspects of these individual models have previously been inspected.

A significant effort has been made by researchers in data mining and machine learning to achieve successful approaches to the approximation of data recognition. Hand-written digits Recognition correspondence has its norm in the twenty-first century and is used much of the time in everyday life as a medium of discourse and capturing the details to be communicated with others. The variety and distortion of the hand-written digit collection are one of the difficulties in the overall recognition of hand-written digits since different cultures will use multiple handwriting types and control to extract the characters identical patterns from their known language.

One of the main tasks in the area of the digital recognition system is the identification of digits from which the best discriminating characteristics can be extracted. In pattern recognition, various methods of area sampling strategies are used to identify certain areas. The difficulty in the identification of hand-written digits is primarily triggered by the wide variety in human writing styles. To enhance the efficiency of a hand-written digit recognition device, robust feature extraction is therefore quite necessary. In the field of pattern recognition device sewing to its use in different areas, hand-written digit recognition has now achieved a lot of attention. In the next few days, by digitizing and manipulating existing paper records, the character recognition technology may serve as a foundation for initiating a paperless world. Hand-written digits datasets are vague, because sharp and perfectly straight lines may not always exist. Feature extraction is the key objective of digit recognition to eliminate the uncertainty from the data and achieve a more powerful embodiment of the term symbol from a series of numerical attributes. It deals with the retrieval from raw picture details of much of the critical information. In comparison, the curves, like the written digits, are not always flat. In comparison, digit datasets may be drawn in multiple sizes and orientations that are often meant to be written in an upright or downright point on a checklist. Consequently, by considering these limitations, an effective hand-written recognition system can be developed. It's very exhausting to remember handwriting digits often since it can be shown that most people cannot even identify their own printed texts. Therefore, there is a restriction for a writer to compose for hand- written digit appreciation.



Figure 1: Handwritten Digits Sample

Hand-written digits identification is a challenging work in a machine vision environment, and it is key to many modern technologies. The identification of hand-written digits is becoming extremely relevant in the developed world because of its realistic applications in our technological experiences. Recent years have seen the implementation of multiple recognition systems in many applications where high classification performance is needed. It lets us tackle more difficult challenges, and allows our jobs simpler. Machine learning and computer vision scientists have been commonly used to incorporate practical applications such as for the identification of zip code (postal code) an early-stage hand-written digit identification has been developed. Online routing of bank accounts, the postal address is commonly used in hand-written digit identification programs. A general tendency has been given to a human being to differentiate various artifacts with differences including numbers, letters, ears, speech. Executing a computerized system for some forms of duties is a very challenging task, and also a complicated and demanding problem in this modern world. Besides, pattern recognition is the basic component in computer-vision and a framework focused on artificial intelligence.

Applications of handwritten digit recognition:

- Postal mail sorting
- Courtesy amounts on cheques
- Formation of data entry

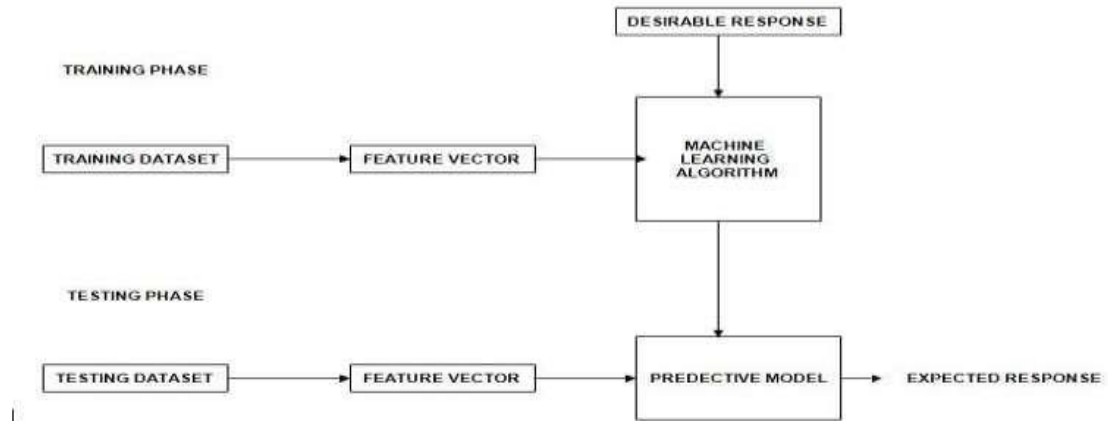


Figure 2: Phases of Handwritten Digit Recognition using Machine Learning

1.1 PROJECT OVERVIEW:

Handwritten digit recognition is the ability of a computer to recognize the human handwritten digits from different sources like images, papers, touch screens, etc., and classify them into 10 predefined classes (0-9). This has been a topic of boundless-research in the field of deep learning and machine learning. In Handwritten digit recognition, everyone faces many challenges because of different styles of writing of different peoples as it is not an Optical character recognition.

Machine Learning provides various methods through which human efforts can be reduced in recognizing the manually written digits. ML algorithms utilize a variety of techniques to handle large amounts of complex data to make decisions. These algorithms complete the task of learning from data with specific inputs given to the machine. Deep Learning is a machine learning method that trains computers to do what easily falls into place for people: learning through examples. Using deep learning, the computer learns to carry out classification works from pictures or contents from any document. DL models can accomplish state-of-art accuracy, beyond the human level performance.

This project provides a comprehensive comparison between different machine learning and deep learning algorithms for the purpose of handwritten digit recognition while using the Support Vector Machine and Convolutional Neural Network for the same purpose. The comparison between these algorithms is carried out on the basis of their accuracy, errors, confusion matrices and testing- training time corroborated by plots that have been constructed using matplotlib for visualization.

The accuracy of any model is paramount as more accurate models make better decisions. The models with low accuracy are not suitable for real-world applications. Ex- For an automated bank cheque processing system where the system recognizes the amount and date

on the check, high accuracy is very critical. If the system incorrectly recognizes a digit, it can lead to major damage which is not desirable. That's why an algorithm with high accuracy is required in these real-world applications. Hence this paper deals with providing a comparison of different algorithms based on their accuracy so that the most accurate algorithm with the least chances of errors can be employed in various applications of handwritten digit recognition.

1.2 PROJECT DELIVERABLES:

- Project Information
- Project Documentation
- Proposed System
- Requirements List
- Program

1.3 PROJECT SCOPE:

The Future development of the applications based on algorithms of deep and machine learning is practically boundless. In the future, works on denser or hybrid algorithms rather than the current set of algorithms with more manifold data to achieve the solutions to many problems may be possible.

Future studies might consider using the architecture of the convolution network which gave the best result on the MNIST database and the proposed recognition system is implemented on handwritten digits. Such more system can be designed for handwritten characters recognition, object recognition, image segmentation, handwriting recognition, text language recognition, and future studies also might consider on hardware implementation on online digit recognition system with more performance and efficiency with live results from live testing case scenarios.

2. REVIEW OF LITERATURE

1. **Paper Name:** Handwritten Digits Identification using MNIST Database Via Machine Learning Models

Authors: Birjit Gope, Sagar Pande, Nikhil Karale, Shivani Dharmale, Pooja Umekar

Overview: The identification of hand-written digits is among the most significant issue in the applications for pattern detection. The applications of digits recognition include the center of the issue is the need to construct an appropriate algorithm that can identify hand-written digits that the users can upload through a smartphone and scanner or any other digital devices. In this paper, they took a repository of MNIST, which is a sub-set of the database of NIST results. The method proposed in this paper is centered on numerous machine learning methods to perform hand-written digit detection that is off-line in the python language platform. The primary objective of this paper is to render hand-written digits recognition reliable and precise. For the identification of digits using MNIST many machine learning algorithms have been used including Support Vector Machine, Multilayer Perceptron, Decision Tree, Naïve Bayes, K-Nearest Neighbor, and Random Forest.

For the identification of hand-written numerals, numerous machine learning algorithms were used in this paper. Among all Machine learning models they got the SVM (Support Vector Machine) recognition method, the total highest accuracy of 95.88 percent. This study is carried out as an initial effort, and the purpose of the paper is to make it simpler to identify hand-written digits without using any common methods for classification.

2. **Paper Name:** Handwritten Digit Recognition Using CNN

Authors: Vijayalaxmi, R Rudraswamimath, Bhavanishankar K

Overview: Digit Recognition is a noteworthy and important issue. As the manually written digits are not of a similar size, thickness, position and direction, in this manner, various difficulties must be considered to determine the issue of handwritten digit recognition. The aim of this project is to implement a classification algorithm to recognize the handwritten digits. The after effects of probably the most broadly utilized Machine Learning Algorithms like SVM, KNN and RFC and with Deep Learning calculation like multilayer CNN utilizing Keras with Theano and Tensorflow.

Utilizing these deep learning techniques, a high amount of accuracy has been obtained. Compared to other research methods, this method focused on which classifier works better by improving the accuracy of classification models by more than 99%. Using Keras as backend and Tensorflow as the software, a CNN model is able to give accuracy of about 98.72%. In this initial experiment, CNN gives an accuracy of 98.72%, while KNN gives an accuracy of 96.67%, while RFC and SVM are not that outstanding.

3. PROBLEM ANALYSIS

3.1: PROPOSED SYSTEM:

- The proposed system uses MNIST dataset which consists of about 70,000 handwritten digits to train the SVM and CNN models and obtain a model with higher accuracy.
- Plots and confusion matrices are used to differentiate between the accuracies of the models and to determine the model with higher accuracy
- Then a Graphical User Interface (GUI) is used to test the model with higher accuracy to test the real time data.

3.2: ADVANTAGES:

- Once we have trained the model, we can save the model and use it for designing various other applications as well.
- The GUI application can be converted into a mobile application which can be used by people to write digits on the forms which needs handwritten data.
- The application can further be developed to identify different character data as well.
- You can further change the application to scan the digits from the paper which makes the digitalisation process easier.

3.3: LIMITATIONS:

- A handwritten digit dataset is vague in essence because there may not always be perfectly straight lines, and different people writings are more or less slopped.
- The curves are not necessarily smooth like printed characters.
- The recognition system sometimes shows inconsistent results due to the similarly shaped numerals.
- All handwritten digital images that are finally tested do not automatically detect boundaries and cropping as well.

4. SYSTEM ANALYSIS

System analysis is a problem-solving technique that decomposes a system into its component pieces for the purpose of studying how well those component parts work and interact to accomplish their purpose. System analysis is the process of studying a procedure in order to identify its goals and purposes and create systems and procedures that will achieve the result in an efficient way.

The development of a computer-based information system includes a system analysis phase which produces or enhances the data-model which itself is a precursor to creating or enhancing a database. There are a number of different approaches to system analysis. When a computer-based information system is developed, systems analysis would constitute the following steps:

- The development of a feasibility study, involving determining whether a project is economically, socially, technologically and organizationally feasible.
- Conducting fact-finding measures, designed to ascertain the requirements of the system's end-users. These typically span interviews, questionnaires, or visual observations of work on the existing system.
- Gauging how the end-users would operate the system (in terms of general experience in using computer hardware or software), what the system would be used for and so on.

4.1 SYSTEM REQUIREMENT SPECIFICATION:

System requirements specification is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers on what the software product is to do as well as what it is not expected to do. software requirements specifications permit a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

4.1.1 FUNCTIONAL REQUIREMENTS:

- MNIST dataset (Modified National Institute of Standards and Technology database) is the subset of the NIST dataset which is a combination of two of NIST's databases: Special Database 1 and Special Database 3.
- Special Database 1 and Special Database 3 consist of digits written by high school students and employees of the United States Census Bureau, respectively.
- MNIST contains a total of 70,000 handwritten digit images (60,000 - training set & 10,000 - test set) in 28x28 pixel bounding box and anti-aliased.



Figure 3: Plotting of some MNIST dataset digits

- Training and testing data by using Machine learning algorithm: Support Vector Machine (SVM) and Deep learning algorithm: Convolution Neural Networks (CNN)
- Finally implementing the model having higher accuracy using a Graphical User Interface (GUI) and testing using the real time data.

4.1.2 NON-FUNCTIONAL REQUIREMENTS:

Non-functional requirements define how a system is supposed to be non-functional that specify criteria that can be used to judge the operation of a system, rather than specific behaviours. Non-functional requirements are in the form of an overall property of the system as a whole or of a particular aspect and not a specific function.

- Performance - How things are going so that we can have early warning of problems that might get in the way of achieving project objectives and so that we can manage expectations.
- Reliability - Application can run with almost zero down-time as long as the internet connection stays active.
- Platform Independent - Application can run on any platform provided that they support the hardware and software requirements
- Accuracy – Application can achieve the maximum accuracy up to 99.1%

4.2 FEASIBILITY STUDY:

The feasibility study is an evaluation and analysis of the potential of a proposed project. It is based on extensive investigation and research to support the process of decision making. Feasibility studies aim to objectively and rationally uncover the strengths and weaknesses of an existing or proposed system, opportunities and threats present in the environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

A well-designed feasibility study should provide a historical background of a project, a description of a service, and details of the operations. Generally, feasibility studies precede technical development and project implementation. A feasibility study evaluates the project's potential for success. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based.

Scalability: Ability to process huge amounts of data.

Reliability: It is an efficient way of processing data without loss.

Technical Feasibility: Technical feasibility also involves evaluation of the hardware and the software requirements of the proposed system.

Economic Feasibility: It serves as an independent project assessment, and enhances project credibility. Economic feasibility is an assessment which typically involves cost/ benefits from the analysis of the project.

Operational Feasibility: It measures how well the proposed system solves problems and takes advantage of the opportunities identified during scope definition. Operational feasibility studies analyse how the project plan satisfies the requirements identified in the requirements analysis phase of system development. To ensure success, desired operational outcomes must inform and guide design and development. These include such design-dependent parameters such as reliability, maintainability, supportability, usability, disposability, sustainability, affordability and others.

Scheduling Feasibility: It is an important factor for project success. A project will fail if it is not completed on time. In Scheduling feasibility, we estimate how much time the system will take to complete and with our technical skills we need to estimate the period to complete the project using various methods of estimation.

Benefits of Conducting a Feasibility Study:

Conducting a feasibility study is always beneficial to the project as it gives you and other stakeholders a clear picture of your idea. Below are the key benefits of conducting a feasibility study:

- Gives project teams more focus and provides an alternative outline.
- Narrows the business alternatives.
- Identifies a valid reason to undertake the project
- Enhances the success rate by evaluating multiple parameters.
- Aids decision-making on the project.

Object-Oriented Analysis:

Object Oriented Analysis is a popular technical approach for analysing, designing an application, system or business by applying the object-oriented paradigm and visual modelling

throughout the development lifecycle to faster, better, stakeholder communication and product quality. In the case of object-oriented analysis, the process varies. But these two are identical at use case analysis. Actually, the steps involved in the analysis phase are

- Identify the actors.
- Classification-develops a static UML class diagram.
- Develop use cases.
- Identify classes, relationships, attributes, methods.

4.3 USE CASE SCENARIOS:

Use Case Model: A Use case is a description of the behaviour of the system. That description is written from the point of a user who just told the system to do something particular.

4.3.1 USE CASE DIAGRAM:

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions that some systems should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

Graphical Notation: The basic components of Use Case diagrams are the Actor, the Use Case, and the Association.




Actor	An actor in the Unified Modelling Language specifies a role played by a user or any other system that interacts with the subject.	
Use Case	A Use Case is the functionality provided by the system. Use Cases are depicted with an ellipse. The name of the Use Case is written in ellipse.	
Association	Association is a relationship between classifiers which is used to show that instances of classifiers could be either linked to each other or combined logically or physically into some aggregation.	

Figure 4: Graphical representation of Use Case diagram

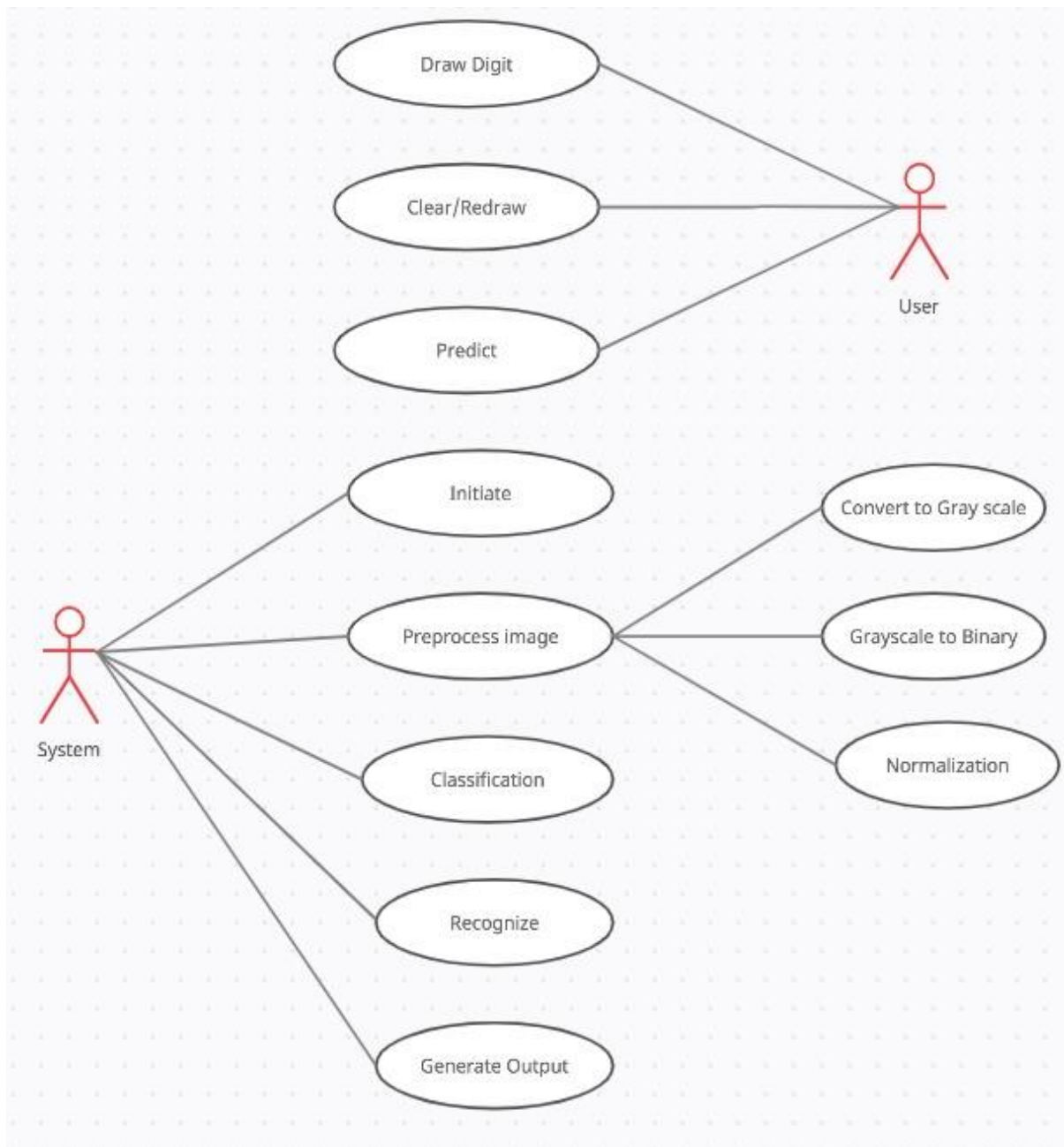


Figure 5: Use Case diagram

4.4 SYSTEM REQUIREMENTS:

System requirements specification is a detailed statement of the effects that a system is required to achieve. A good specification gives a complete statement of what the system is to do, without making any commitment as to how the system is to do it.

A system requirements specification is normally produced in response to a user requirements specifications or other expression of requirements, and is then used as the basis for system design. The system requirements specification typically differs from expression of requirements in both scope and precision the latter may cover both the envisaged system and the environment in which it will operate, but may leave many broad concepts unrefined.

4.4.1 SOFTWARE REQUIREMENTS:

- Operating System: Windows 7 and above.
- Language used: Python Version (3.6.9)
- Dataset: MNIST dataset containing 70,000 handwritten images
- Libraries: Keras, NumPy, Pandas, Tensorflow, Scikit learn, Pillow, Joblib, Tkinter

4.4.2 HARDWARE REQUIREMENTS:

- System: Minimum Pentium IV 2.4 GHz. Recommended Intel core i3 3.3 GHz or more.
- Hard Disk: Minimum 200 GB. Recommended 500 GB or more.
- Ram: Minimum 4 GB. Recommended 4 GB or more.

26 5. SYSTEM

5. SYSTEM DESIGN

5.1 INTRODUCTION:

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

Systems design mainly concentrates on defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

Systems design implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach, but either way the process is systematic wherein it takes into account all related variables of the system that needs to be created—from the architecture, to the required hardware and software, right down to the data and how it travels and transforms throughout its travel through the system. Systems design then overlaps with systems analysis, systems engineering and systems architecture.

The systems design approach first appeared right before World War II, when engineers were trying to solve complex control and communications problems. They needed to be able to standardize their work into a formal discipline with proper methods, especially for new fields like information theory, operations research and computer science.

5.2 UML DIAGRAMS:

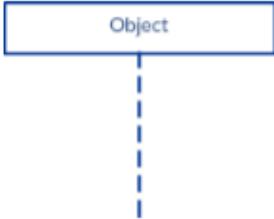




UML is a standard language for specifying, visualizing, constructing and documenting the artifacts of software systems. The UML uses mostly graphical notations to express the design of software projects. It is a very important part of developing object-oriented software and software development process.

5.2.1 SEQUENCE DIAGRAM:

A sequence diagram shows object interactions arranged in time sequence. It depicts - the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram describes an interaction among a set of objects participated in a collaboration (or scenario), arranged in a chronological order; it shows the objects participating in the interaction by their "lifelines" and the messages that they send to each object.

Object	Objects are instances of classes and are arranged horizontally. The pictorial representation for an Object is class (a rectangle) with the name prefixed by the object name	
Actor	Actor can also communicate with objects so they too can be listed as a column. An Actor is modelled using the stick figure.	
Lifeline	The Lifeline identifies the existence of the object over time. The notation for a life time is a vertical dotted line extending from an object	
Activation	Activation modelled as rectangular boxes on the lifeline indicate when the object is performing an action	
Message	Messages modelled as horizontal arrows between Activations indicate the communication between the objects.	

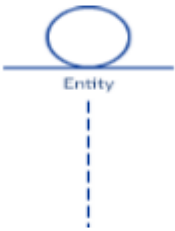
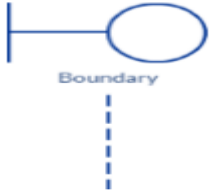

Entity	A lifeline with an entity element represents system data.	 The diagram shows a circle (entity element) on a horizontal line. A vertical dashed line (lifeline) extends downwards from the circle. The word "Entity" is written below the horizontal line.
Boundary	A lifeline with a boundary element indicates a system boundary/ software element in a system	 The diagram shows a circle (boundary element) on a horizontal line. A vertical dashed line (lifeline) extends downwards from the circle. A horizontal line segment with a T-shaped end (activation bar) is attached to the left of the circle. The word "Boundary" is written below the horizontal line.
Control	lifeline with a control element indicates a controlling entity or manager. It organizes and schedules the interactions between the boundaries and entities and serves as the mediator between them.	 The diagram shows a circle (control element) on a horizontal line. A vertical dashed line (lifeline) extends downwards from the circle. A curved arrow (self-message) is drawn on the circle, pointing from the top to the bottom. The word "Control" is written below the horizontal line.

Figure 6: Graphical Notations for Sequence Diagram

A sequence diagram shows, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.

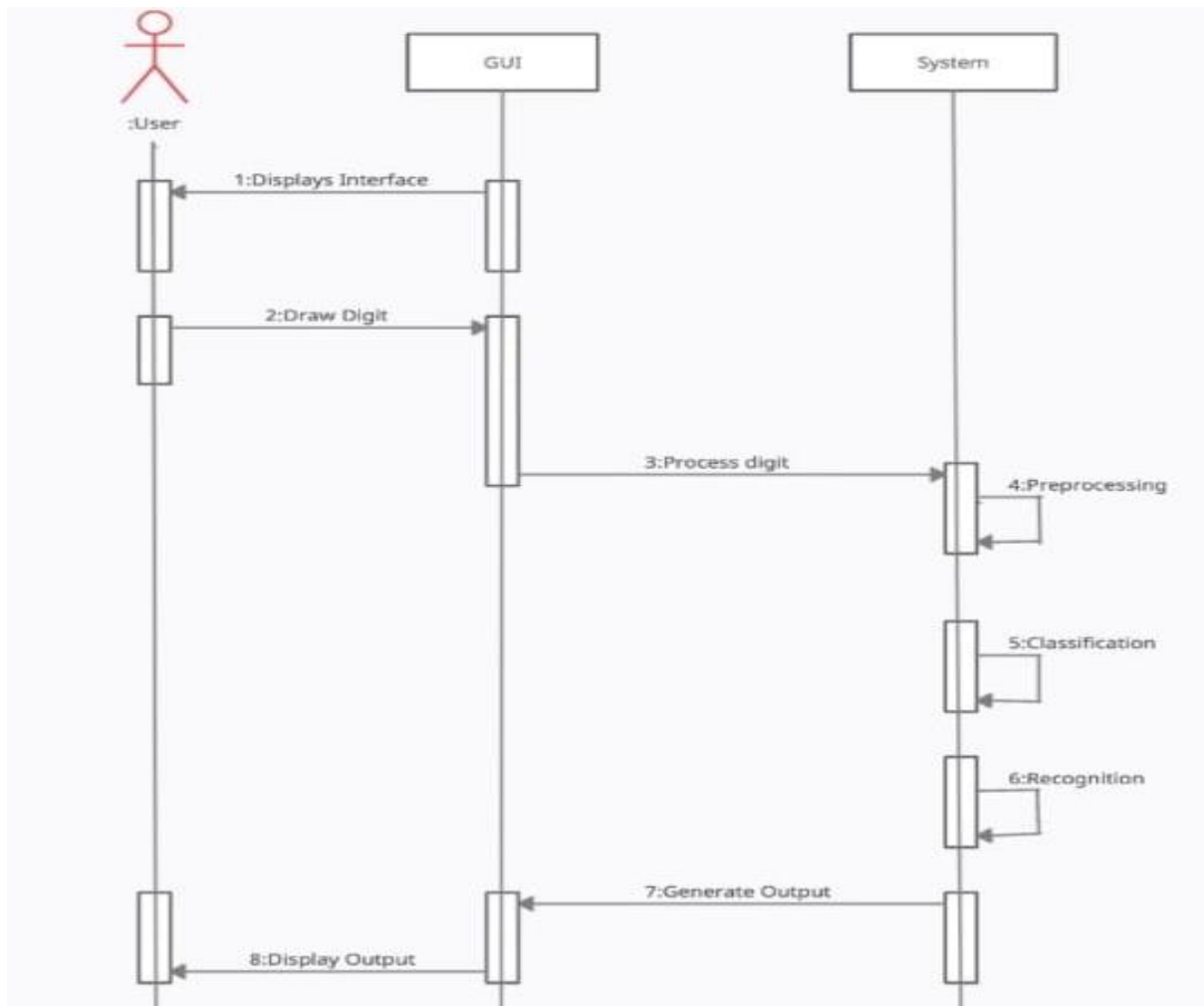


Figure 7: Sequence diagram

5.2.2 STATE CHART DIAGRAM:




A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. A state diagram is used to describe the behaviour of systems. This behaviour is analysed and represented in series of events that could occur in one or more possible states. State diagrams require that the system described is composed of a finite number of states. Sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

The main purposes of using State Chart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

The Main Usage can be described as:

- To Model the object states of a System.
- To Model the reactive System.
- To identify the events responsible for state changes.
- Forward and Reverse Engineering

Initial State	The initial state represents the source of all objects. A filled circle followed by an arrow represents the object's initial state.	
State	State represents situations during the life of an object. Rectangular boxes with curved edges represent a state	
Transition	A transition represents the change from one state to another. A solid arrow represents the path between different states of an object	


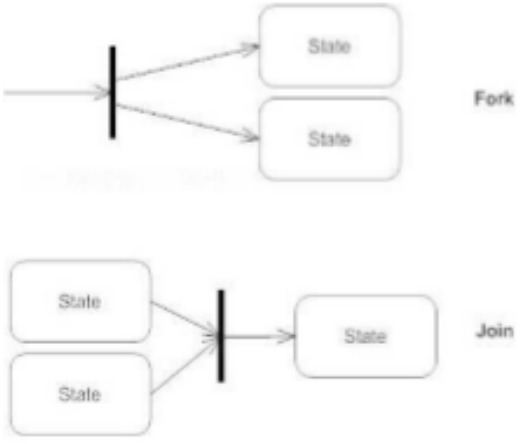
Final State	The final state represents the end of an object's existence. A final state is not a real state, because objects in this state don't exist anymore. A filled circle represents the final state	 Final state
Synchronization and Splitting	A short heavy bar with two transitions entering it represents a synchronization of control. The first bar is called a fork where a single transition splits into concurrent multiple transitions. The second bar is called a join, where the concurrent transitions reduce back to one	

Figure 8: Graphical Notations for State Chart diagram

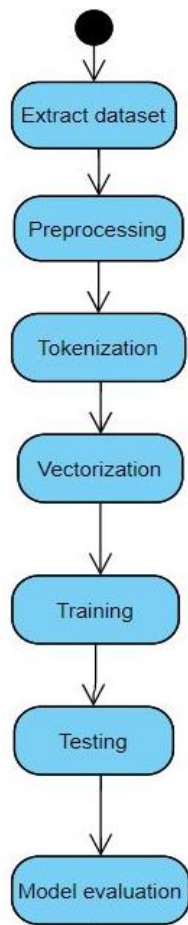


Figure 9: State chart diagram

5.2.3 ACTIVITY DIAGRAM:

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modelling. They can also describe the steps in a use case diagram. Activities modelled can be sequential and concurrent.

Activity Diagram Notations: Activity diagrams symbol can be generated by using the following notations:

- **Initial states:** The starting stage before an activity takes place is depicted as the initial state
- **Final states:** The state which the system reaches when a specific process ends is known as a Final State
- **State or an activity box:** It represents the current state or the activity of the system.
- **Decision box:** It is a diamond shape box which represents a decision with alternate paths. It represents the flow of control.

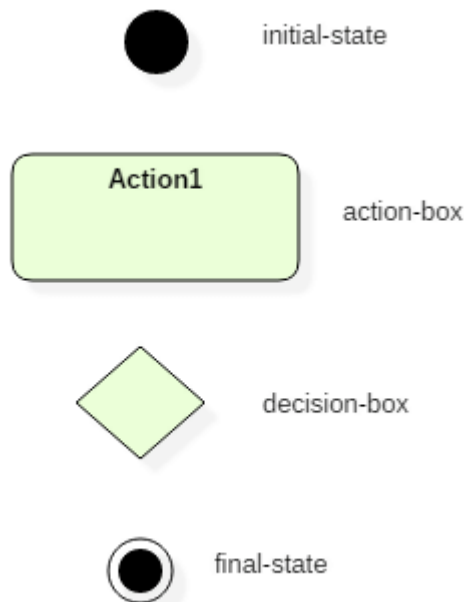


Figure 10: Flow of control of Activity diagram

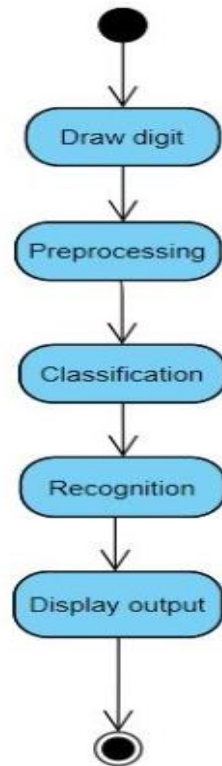


Figure 11: Activity diagram

5.3 SYSTEM ARCHITECTURE:

Systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system.

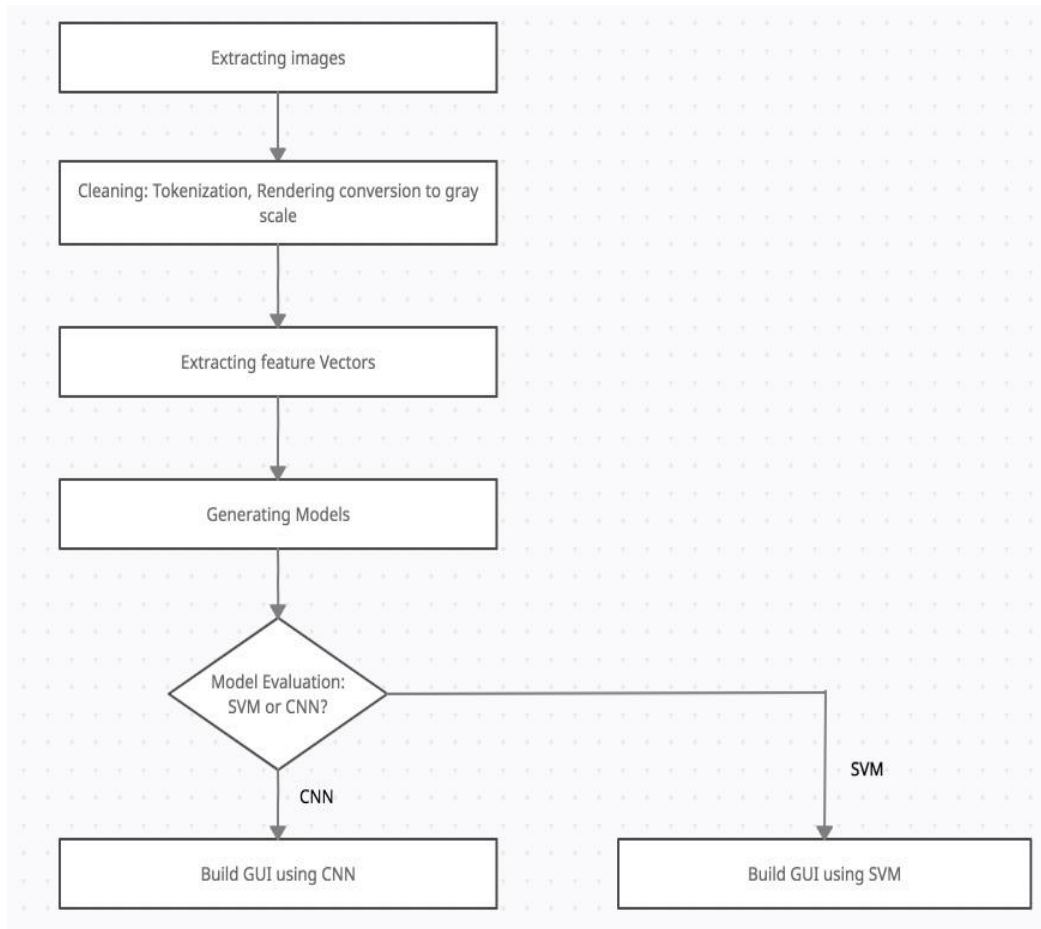


Figure 12: System Architecture

5.3.1 ALGORITHM SPECIFICATION:

Description: Here we are using two different algorithms. One ML algorithm called SVM (Support Vector Machine) and one DL algorithm called CNN (Convolution Neural Network).

SUPPORT VECTOR MACHINE ALGORITHM:

Support Vector Machine:

Support Vector Machine (SVM) is a supervised machine learning algorithm. There is generally plotting of data items in n-dimensional space where n is the number of features, a particular coordinate represents the value of a feature, we perform the classification by finding the hyperplane that distinguishes the two classes. It will choose the hyperplane that separates the classes correctly.

SVM chooses the extreme vectors that help in creating the hyperplane. These extreme cases are called support vectors, and hence the algorithm is termed as Support Vector Machine. There are mainly two types of SVMs, linear and non-linear SVM. We have used Linear SVM for handwritten digit recognition.

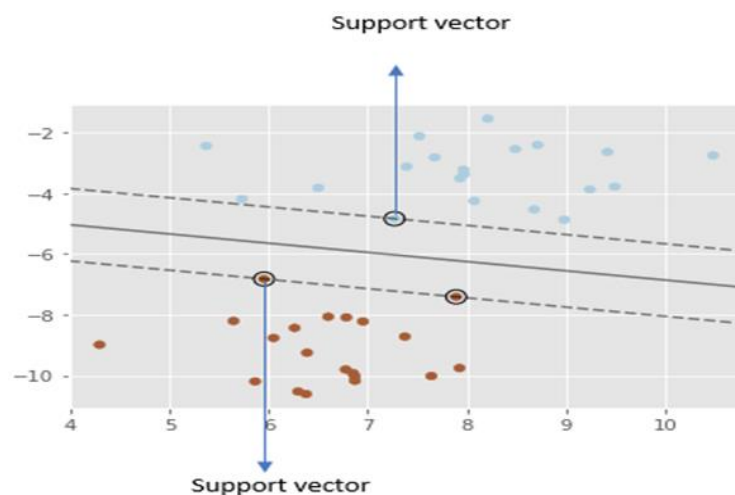


Figure 13: Support Vector Machine

Algorithm:

Step 1 -Import the dataset and libraries: The dataset is available in the MNIST website. Unzip the files and extract the data. Also import the required libraries.

Step 2- Explore the data and figure out what it looks like: Using the imported dataset plot a bar graph to understand the distribution of labels.

Step 3- Pre-process the data: Here the images are converted into grayscale and the pixel data is obtained. The obtained pixel data is normalised for usage in the model.

Step 4- Split the data into test and train datasets: Usually the data is split into training and testing in the ratio 7: 3 or 8:2. Here 60,000 images are used for training and 10,000 images for testing.

Step 5- Train the SVM model: The model used here is linear SVM. This model is inbuilt into the scikit-learn library. Import the model and feed the data to the model. Calculate accuracy.

Step 6- Predict the labels using the model: Using the evaluated model predict the labels of the test data and plot a confusion matrix.

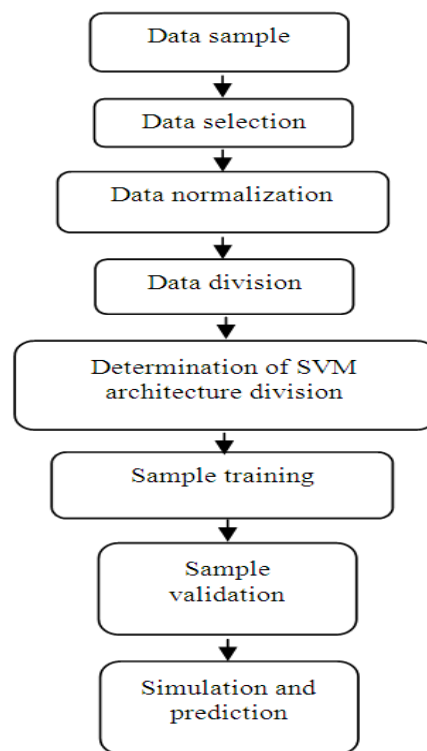


Figure 14: Flowchart representing SVM algorithm

Evaluation of SVM model:

Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results. Simply building a predictive model is not your motive. It's about creating and selecting a model which gives high accuracy on out of sample data. Hence, it is crucial to check the accuracy of your model prior to computing predicted values.

We obtain Classification Report and calculate accuracy percentage through this by finding Precision, Recall, F1 score. We also obtain confusion matrix.

1. **Precision:** Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives. False positives are cases the model incorrectly labels as positive that are actually negative, or in our example, individuals the model classifies as terrorists that are not. While recall expresses the ability to find all relevant instances in a dataset, precision expresses the proportion of the data points our model says was relevant actually were relevant.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

2. **Recall:** The ability of a model to find all the relevant cases within a dataset. The precise definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives. True positives are data points classified as positive by the model that actually are positive (meaning they are correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect).

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

3. **F1 Score:** In some situations, we might know that we want to maximize either recall or precision at the expense of the other metric. However, in cases where we want to find an optimal blend of precision and recall we can combine the two metrics using what is called the F1 Score.

The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

4. Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of classes being predicted. Here are a few definitions, you need to remember for a confusion matrix:

- **Accuracy:** the proportion of the total number of predictions that were correct.
- **Positive Predictive Value or Precision:** the proportion of positive cases that were correctly identified.
- **Negative Predictive Value:** the proportion of negative cases that were correctly identified.

- **Sensitivity or Recall:** the proportion of actual positive cases which are correctly identified.
- **Specificity:** the proportion of actual negative cases which are correctly identified.

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	Positive Predictive Value	$a/(a+b)$
	Negative	c	d	Negative Predictive Value	$d/(c+d)$
		Sensitivity	Specificity	Accuracy = $(a+d)/(a+b+c+d)$	
		$a/(a+c)$	$d/(b+d)$		

It also helps in visualizing the precision and recall.

CONVOLUTION NEURAL NETWORK ALGORITHM:

Convolution Neural Network:

CNN is a deep learning algorithm that is widely used for image recognition and classification. It is a class of deep neural networks that require minimum pre-processing. It inputs the image in the form of small chunks rather than inputting a single pixel at a time, so the network can detect uncertain patterns (edges) in the image more efficiently.

CNN contains 3 layers namely, an input layer, an output layer, and multiple hidden layers which include Convolutional layers, Pooling layers (Max and Average pooling), Fully connected layers (FC), and normalization layers.

CNN uses a filter (kernel) which is an array of weights to extract features from the input image. CNN employs different activation functions at each layer to add some non-linearity. Further, we observe the height and width decrease while the number of channels increases. Finally, the generated column matrix is used to predict the output.

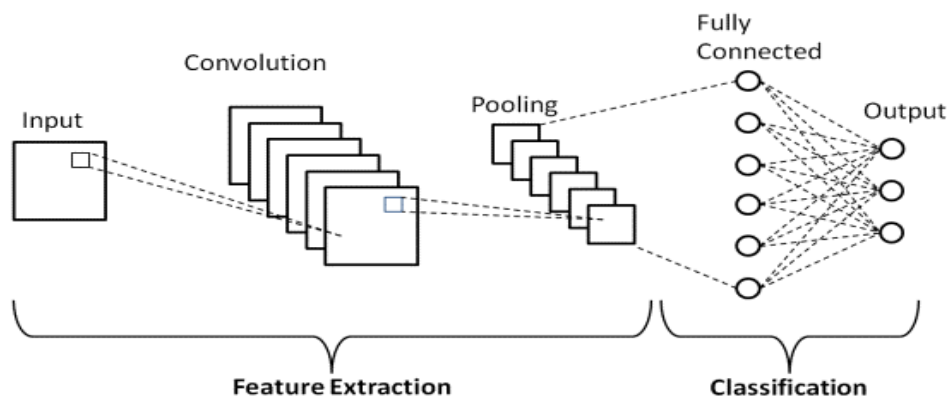


Figure 15: Convolution Neural Network

Algorithm:

Step 1- Import the dataset and libraries: The dataset is available as inbuilt in the keras library. Import other important libraries.

Step 2- Explore the data and figure out what it looks like: Using the imported dataset plot a bar graph to understand the distribution of labels.

Step 3- Pre-process the data: Here the images are converted into grayscale and the pixel data is obtained. The obtained pixel data is normalised for usage in the model. The images are also reshaped to fixed size to avoid complications.

Step 4- Split the data into test and train datasets: Usually the data is split into training and testing in the ratio 7: 3 or 8:2. Here 60,000 images are used for training and 10,000 images for testing.

Step 5- Train the CNN model: The keras library has inbuilt CNN model which can be modified according to the requirements. Various features like number of layers, epochs, activation layers and number of neurons can be specified. Feed the data to the model and train it. Calculate accuracy.

Step 6- Predict the labels using the model: Using the evaluated model predict the labels of the test data and plot a confusion matrix.

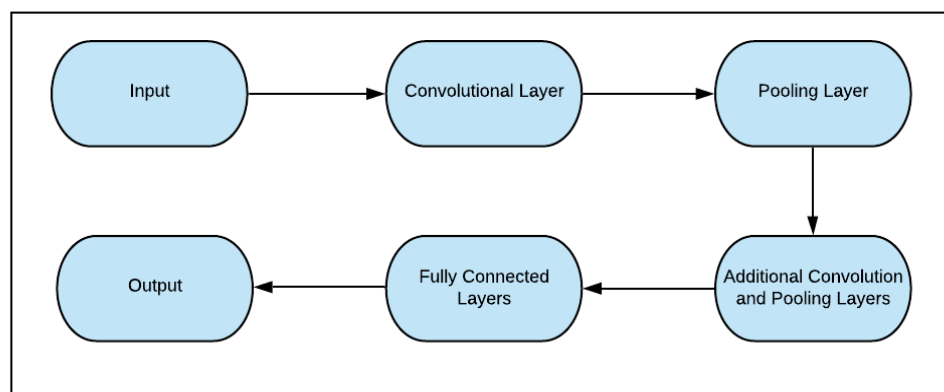


Figure 16: Flowchart representing CNN algorithm

Evaluation metrics for CNN:

Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results. Simply building a predictive model is not your motive. It's about creating and selecting a model which gives high accuracy on out of sample data. Hence, it is crucial to check the accuracy of your model prior to computing predicted values.

We obtain Classification Report and calculate accuracy percentage through this by finding Precision, Recall, F1 score. We also obtain confusion matrix.

1. **Precision:** Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives. False positives are cases the model incorrectly labels as positive that are actually negative, or in our example, individuals the model classifies as terrorists that are not. While recall expresses the ability to find all relevant instances in a dataset, precision expresses the proportion of the data points our model says was relevant actually were relevant.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

2. **Recall:** The ability of a model to find all the relevant cases within a dataset. The precise definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives. True positives are data point classified as positive by the model that actually are positive (meaning they are correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect).

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

3. **F1 Score:** In some situations, we might know that we want to maximize either recall or precision at the expense of the other metric. However, in cases where we want to find an optimal blend of precision and recall we can combine the two metrics using what is called the F1 Score.

The F1 score is the harmonic mean of precision and recall taking both metrics into account in the following equation:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

4. Confusion Matrix

A confusion matrix is an N X N matrix, where N is the number of classes being predicted. Here are a few definitions, you need to remember for a confusion matrix:

- **Accuracy:** the proportion of the total number of predictions that were correct.
- **Positive Predictive Value or Precision:** the proportion of positive cases that were correctly identified.
- **Negative Predictive Value:** the proportion of negative cases that were correctly identified.

- **Sensitivity or Recall:** the proportion of actual positive cases which are correctly identified.
- **Specificity:** the proportion of actual negative cases which are correctly identified.

Confusion Matrix		Target			
		Positive	Negative		
Model	Positive	a	b	<i>Positive Predictive Value</i>	$a/(a+b)$
	Negative	c	d	<i>Negative Predictive Value</i>	$d/(c+d)$
		<i>Sensitivity</i>	<i>Specificity</i>	Accuracy = $(a+d)/(a+b+c+d)$	
		$a/(a+c)$	$d/(b+d)$		

It also helps in visualizing the precision and recall.

5.4 USER INTERFACE:

A user interface, also called a "UI" or simply an "interface," is the means in which a person controls a software application or hardware device. A good user interface provides a "user-friendly" experience, allowing the user to interact with the software or hardware in a natural and intuitive way.

The user interface in the industrial design field of human–computer interaction is the space where interactions between humans and machines occur. The goal of this interaction is to allow effective operation and control of the machine from the human end, whilst the machine simultaneously feeds back information that aids the operators' decision-making process. Examples of this broad concept of user interfaces include the interactive aspects of computer operating systems, hand tools, heavy machinery operator controls, and process controls. The design considerations applicable when creating user interfaces are related to or involve such disciplines as ergonomics and psychology.

Generally, the goal of user interface design is to produce a user interface which makes it easy (self-explanatory), efficient, and enjoyable (user-friendly) to operate a machine in the way which produces the desired result. This generally means that the operator needs to provide minimal input to achieve the desired output, and also that the machine minimizes undesired outputs to the human. With the increased use of personal computers and the relative decline in societal awareness of heavy machinery, the term user interface is generally assumed to mean the graphical user interface, while industrial control panel and machinery control design discussions more commonly refer to human-machine interfaces.

Nearly all software programs have a graphical user interface, or GUI. This means the program includes graphical controls, which the user can select using a mouse or keyboard. A typical GUI of a software program includes a menu bar, toolbar, windows, buttons, and other controls. The Macintosh and Windows operating systems have different user interfaces, but they share many of the same elements, such as a desktop, windows, icons, etc. These common elements make it possible for people to use either operating system without having to completely relearn the interface. Similarly, programs like word processors and Web browsers all have rather similar interfaces, providing a consistent user experience across multiple programs.

Most hardware devices also include a user interface, though it is typically not as complex as a software interface. A common example of a hardware device with a user interface is a remote control. A typical TV remote has a numeric keypad, volume and channel buttons, mute and power buttons, an input selector, and other buttons that perform various functions. This set of buttons and the way they are laid out on the controller makes up the user interface. Other devices, such as digital cameras, audio mixing consoles, and stereo systems also have a user interface.

While user interfaces can be designed for either hardware or software, most are a combination of both. For example, to control a software program, you typically need to use a keyboard and mouse, which each have their own user interface. Likewise, to control a digital camera, you may need to navigate through the on-screen menus, which is a software interface. Regardless of the application, the goal of a good user interface is to be user-friendly.

6. SYSTEM IMPLEMENTATION

6.1 TECHNOLOGIES DESCRIPTION:

6.1.1 WINDOWS:

Windows is a series of operating systems developed by Microsoft. Each version of Windows includes a graphical user interface, with a desktop that allows users to view files and folders in windows. For the past two decades, Windows has been the most widely used operating system for personal computers PCs.

Features:

1. **Speed:** Even aside from incompatibilities and other issues that many people had with Vista, one of the most straightforward was speed – it just felt too sluggish compared to XP, even on pumped up hardware. Windows 7 brings a more responsive and sprightlier feel and Microsoft has spent a lot of time and effort getting the Start Menu response just right.

Microsoft has also recognized the need for improved desktop responsiveness, which gives the impression that the computer is responding to the user and that they are in control –something that was often lacking with Vista. You can also expect faster boot times. And the boot sequence is now not only prettier than it was with Vista, but it's speedier too.

2. **Compatibility:** In simple terms, compatibility on Windows 7 will be far better than it was with Vista. Many programs that individuals and companies used on Windows XP did not work immediately and required updates, but with Windows 7 almost all applications that work on Vista should still run.
3. **Lower Hardware Requirements:** Vista gained a reputation for making even the beefiest hardware look rather ordinary. Windows 7, however, will run well on lower end hardware, making the transition from Window XP less painful.

Microsoft is even pushing Windows 7 for netbooks. This could provide a modern replacement for Windows XP, which has found a new lease of life as the OS of choice on netbooks, supplanting Linux. The downside is that Windows 7 Starter Edition, as it will be called, will be limited to only three applications running at the same time.

4. **Search and Organization:** One of the best things about Windows 7 is the improved search tool, which now rivals Mac OS X's Spotlight to be able to find what you need quickly and easily. For example, typing 'mouse' will bring up the mouse option within the control panel or typing a word will display it and split it up neatly into files, folders and applications.
5. **Safety and Security:** New security features in Windows include two new authentication methods tailored towards touchscreens (PINs and picture passwords), the addition of antivirus capabilities to Windows Defender (bringing it in parity with Microsoft Security

Essentials) Smart Screen filtering integrated into Windows, and support for the "Secure Boot" functionality on UEFI systems to protect against malware infecting the boot process. Family Safety offers Parental controls, which allows parents to monitor and manage their children's activities on a device with activity reports and safety controls.

6.1.2 JUPYTER NOTEBOOK:

The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

Getting Up and Running with Jupyter Notebook:

The Jupyter Notebook is not included in Python, so if you want to try it out, you will need to install Jupyter.

There are many distributions of the Python language. This is will focus on just two of them for the purpose of installing Jupyter Notebook. The most popular is Cpython, which is reference version of python that you can get from their websites. It is also assumed that you are using Python 3.

6.1.3 PYTHON:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level Programming language. Python is designed to be highly readable. It uses English keywords frequently where as other languages. Python is easy to learn yet powerful and versatile scripting language, which makes it for Application development.

- **Python is interpreted:** Python is processed at runtime by the interpreter. You don't need to compile your programs before executing it. This is similar to PHP.
- **Python is Interactive:** you can actually sit at python prompt and interact with the interpreter directly to write your programs.
- **Python is Object Oriented:** Python supports Object-oriented style programming that encapsulates code within objects.
- **Python is Beginner's Language:** Python is a great language for the beginner-level and support development of a wide range applications from simple text processing to WWW browsers to games.

Applications of Python:

- **Web Applications:** It Provides Libraries to handle internet protocols such as HTML and XML, JSON, Email Processing, request, beautiful Soup, Feed Parser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web-based applications. Some important developments are: Python Wiki Engines, Pocoo, Python Blog Software.
- **Desktop GUI Applications:** Python provides Tk GUI Library to develop user interface in python-based application. Some other toolkits such as wxWidgets, Kivy, pyqt that are useable on several platforms. The kivy is popular for writing multitouch applications.
- **Software Development:** Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.
- **Business Applications:** Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high-level application platform.
- **Console Based Applications:** We can use Python to develop console-based applications. Eg: IPYTHON
- **Audioor Video Based Applications:** Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of the real applications are: Time Player, cplay etc.,
- **3D CAD Applications:** To create CAD applications Fandango is a real application which provides full features of CAD.
- **Enterprise Application:** Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications: OpenErp, Tryton, Picalo.etc,

Features of Python Programming:

- **Easy to code:** Python is high level programming language. Python is very easy to learn language as compared to other language like c, c#, java script, java etc. It is very easy to code in python language and anybody can learn python basic in a few hours or days. It is also a developer-friendly language.
- **Free and Open Source:** Python language is freely available at the official website. Since, it is open-source, this means that source code is also available to the public. So, you can download it as, use it as well as share it.
- **Object-Oriented Language:** One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation etc.
- **GUI Programming Support:** Graphical Users interfaces can be made using a module such as PyQt5, PyQt4, wxPython or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.
- **High-Level Language:** Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

- **Extensible feature:** Python is an Extensible language. We can write our python code into C or C++ language and also, we can compile that code in C/C++ language.
- **Python is Portable language:** Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix and Mac then we do not need to change it, we can run this code on any platform.
- **Python is Integrated language:** Python is also an integrated language because we can easily integrate Python with other languages like C, C++ etc.
- **Interpreted Language:** Python is an Interpreted Language because python code is executed line by line at a time. Unlike other languages C, C++, Java etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called Bytecode.
- **Large Standard Library:** Python has a large standard library which provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers etc.
- **Dynamically Typed Language:** Python is dynamically-typed language. That means the type (for example- int, double, long etc) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

Libraries used for this application:

- **OS-** The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.
- **Sys-** Import sys in python is loading the module named sys into the current namespace so that you can access the functions and anything else defined within the module using the module name.
- **Pandas-** In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.
- **Numpy-** NumPy is a Python package which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc. The NumPy arrays takes significantly less amount of memory as compared to python lists. It also provides a mechanism of specifying the data types of the contents, which allows further optimisation of the code.
- **Matplotlib.pyplot-** Plotting library for 2D graphics plotting.
- **Seaborn-** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **SciPy-** SciPy is a free and opensource library used for scientific computing and technical computing. It contains modules for optimization, integration, interpolation, special functions, linear algebra, image processing.

- **Random-**It is a built-in module to generate the pseudo random variables. It can be used perform some action randomly such as to get a random number, selecting a random element from a list, shuffle elements randomly, etc.
- **TensorFlow-**It is a library in python for fast numerical computing. It allows the developers to create large-scale neural networks with many layers. Used for classification, perception, understanding, discovering, prediction and creation.
- **Keras-**It is an open-source library that provides Python interface for artificial neural networks. Keras act as an interface to the TensorFlow library.
- **Pillow-**It is a Python Imaging Library (PIL), which adds support for opening, manipulating, saving images.
- **Tkinter-**In python, Tkinter is a standard GUI (Graphical User Interface) package. It is set of wrappers that implement the Tk widgets as Python classes.
- **Joblib-**It is set of tools to provide lightweight in pipelining in Python. It is optimized to be fast and robust in particular on large data and has specific optimizations for Numpy arrays.
- **SKLearn-** Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, Pandas, and Matplotlib.

6.2 SYSTEM MODULES:

1. Import the libraries and load the dataset: We are going to import all the modules or libraries that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. So, we can easily import the dataset and start working with it.

2. Pre-process the data: The image data cannot be fed directly into the models so we need to perform some operations and process the data to make it ready for our models. Each model will require data supplied to it in different formats. The CNN model will require one more dimension so we reshape the matrix to shape (60000,28,28,1). The SVM model will require the data in pixel format.

3. Create the model: Create our CNN model in Python data science project. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures. Also create our SVM model using the scikit learn inbuilt model. Here we are using the linear SVM model.

4. Train the model: The `model.fit ()` function of Keras will start the training of the model. It takes the training data, validation data, epochs, and batch size. The `model.fit ()` function of scikit learn also fits the model to train using the dataset supplied.

5. Evaluate the model: We have 10,000 images in our dataset which will be used to evaluate how good our model works. The testing data was not involved in the training of the data therefore, it is new data for our model. Evaluate both SVM and CNN models. Check for the accuracies. Also plot graphs and confusion matrices to determine the best model. Now save the parameters of the best model.

6. Create GUI to predict digits: Now for the GUI, we have created a new file in which we build an interactive window to draw digits on canvas and with a button, we can recognize the digit. We use the model with the highest accuracy we have previously saved for the backend purpose of the GUI interface.

6.3 SAMPLE SOURCE CODE:

```
import keras

from keras.datasets import mnist

from keras.utils import to_categorical

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras import backend as K

# The data, split between train and test sets

(x_train, y_train), (x_test, y_test) = mnist.load_data()

print(x_train.shape, y_train.shape)

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

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

input_shape = (28, 28, 1)

# Convert class vectors to binary class matrices

y_train = keras.utils.to_categorical(y_train, 10)

y_test = keras.utils.to_categorical(y_test, 10)

x_train = x_train.astype('float32')

x_test = x_test.astype('float32')

x_train /= 255

x_test /= 255

print('x_train shape:', x_train.shape)

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

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

```

batch_size = 128

num_classes = 10

epochs = 50

model = Sequential()

model.add(Conv2D(32, kernel_size=(5, 5),activation='relu',input_shape=input_shape))

model.add(MaxPooling2D(pool_size=(2, 2)))

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

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.3))

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])

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")

score = model.evaluate(x_test, y_test, verbose=0)

print("Test loss:", score[0])

print("Test accuracy:", score[1])

model.save('mis.h5')

print("Saving the model as mis.h5")

```


7. TESTING

7.1 INTRODUCTION:

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test have the following

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs,
- Performs its functions within an acceptable time,
- Is sufficiently usable,
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects).

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable.

Types of Testing:

A software engineering product can be tested in one of two ways:

- Black box testing
- White box testing

Black box testing: Knowing the specified function that a product has been designed to perform, determine whether each function is fully operational.

White box testing: Knowing the internal workings of a software product determine whether the internal operation implementing the functions perform according to the specification, and all the internal components have been adequately exercised.

Testing Strategies:

Four Testing Strategies that are often adopted by the software development team include:

- Unit Testing
- Integration Testing
- Validation Testing
- System Testing

Unit Testing:

We adopt white box testing when using this testing technique. This testing was carried out on individual components of the software that were designed. Each individual module was tested using this technique during the coding phase. Every component was checked to make sure that they adhere strictly to the specifications spelt out in the data flow diagram and ensure that they perform the purpose intended for them.

All the names of the variables are scrutinized to make sure that they are truly reflected of the element they represent. All the looping mechanisms were verified to ensure that they were as decided. Beside these, we trace through the code manually to capture syntax errors and logical errors.

Integration Testing:

After finishing the Unit Testing process, next is the integration testing process. In this testing process we put our focus on identifying the interfaces between components and their functionality as dictated by the DFD diagram. The Bottom-up incremental approach was adopted during this testing. Low level modules are integrated and combined as a cluster before testing.

The Black box testing technique was employed here. The interfaces between the components were tested first. This allowed identifying any wrong linkages or parameters passing early in the development process as it just can be passed in a set of data and checked if the result returned is an accepted one.

Validation Testing:

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

System Testing:

System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all the work should verify that all system elements have been properly integrated and perform allocated functions. System testing also ensures that the project works well in the environment. It traps the errors and allows convenient processing of errors without coming out of the program abruptly.

Recovery testing is done in such a way that failure is forced to a software system and checked whether the recovery is proper and accurate. The performance of the system is highly effective.

Software testing is critical element of software quality assurance and represents ultimate review of specification, design and coding. Test case design focuses on a set of technique for the creation of test cases that meet overall testing objectives. Planning and testing of a programming system involve formulating a set of test cases, which are similar to the real data that the system is intended to manipulate. Test cases consist of input specifications, a description of the system functions exercised by the input and a statement of the extended output.

In principle, testing of a program must be extensive. Every statement in the program should be exercised and every possible path combination through the program should be executed at least once. Thus, it is necessary to select a subset of the possible test cases and conjecture that this subset will adequately test the program.

Guidelines for developing test cases:

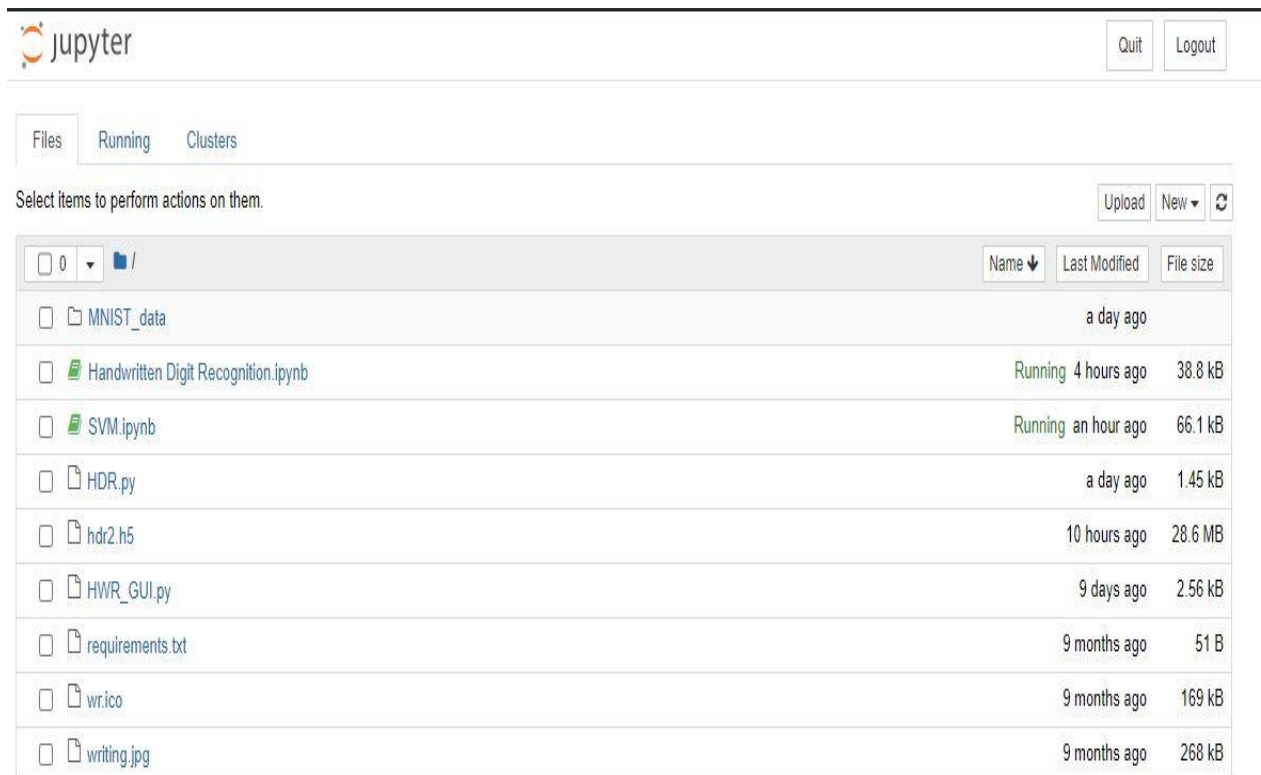
- Describe which feature or service your test attempts to cover
- If the test case is based on a use case it is a good idea to refer to the use case name.
- Remember that the use cases are the source of test cases. In theory the software is supposed to match the use cases not the reverse. As soon as you have enough use cases, go ahead and write the test plan for that piece
- Specify what you are testing and which particular feature. Then specify what you are going to do to test the feature and what you expect to happen.
- Test the normal use of the object's methods. Test the abnormal but reasonable use of the object's methods.
- Test the abnormal but unreasonable use of the object's methods.
- Test the boundary conditions. Also specify when you expect error dialog boxes, when you expect some default event, and when functionality till is being defined.
- Test object's interactions and the messages sent among them. If you have developed sequence diagrams, they can assist you in this process
- When the revisions have been made, document the cases so they become the starting bases for the follow-up test

Attempting to reach agreement on answers generally will raise other what-if questions. Add these to the list and answer them, repeat the process until the list is stabilized, then you need not add any more questions.

7.2 TEST CASES:

S. No	Description	Input	Expected Value	Actual Value	Result
1.	Predicting 0 as 0	0	0	0	PASS
2.	Predicting 1 as 1	1	1	1	PASS
3.	Predicting 2 as 2	2	2	2	PASS
4.	Predicting 3 as 3	3	3	3	PASS
5.	Predicting 4 as 4	4	4	4	PASS
6.	Predicting 5 as 5	5	5	5	PASS
7.	Predicting 6 as 6	6	6	6	PASS
8.	Predicting 7 as 7	7	7	7	PASS
9.	Predicting 8 as 8	8	8	8	PASS
10.	Predicting 9 as 9	9	9	9	PASS

8. OUTPUTS



The image shows the JupyterLab interface. At the top, there's a header with the Jupyter logo and 'Quit' and 'Logout' buttons. Below the header, there are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, showing a file browser. The file browser has a search bar and a 'Select items to perform actions on them.' prompt. Below this, there's a table of files and folders. The table has columns for 'Name', 'Last Modified', and 'File size'. The files listed are:

Name	Last Modified	File size
<input type="checkbox"/> MNIST_data	a day ago	
<input type="checkbox"/> Handwritten Digit Recognition.ipynb	Running 4 hours ago	38.8 kB
<input type="checkbox"/> SVM.ipynb	Running an hour ago	66.1 kB
<input type="checkbox"/> HDR.py	a day ago	1.45 kB
<input type="checkbox"/> hdr2.h5	10 hours ago	28.6 MB
<input type="checkbox"/> HWR_GUI.py	9 days ago	2.56 kB
<input type="checkbox"/> requirements.txt	9 months ago	51 B
<input type="checkbox"/> wr.ico	9 months ago	169 kB
<input type="checkbox"/> writing.jpg	9 months ago	268 kB

Figure 17: Opening Jupyter notebook

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

Figure 18: Extracting dataset

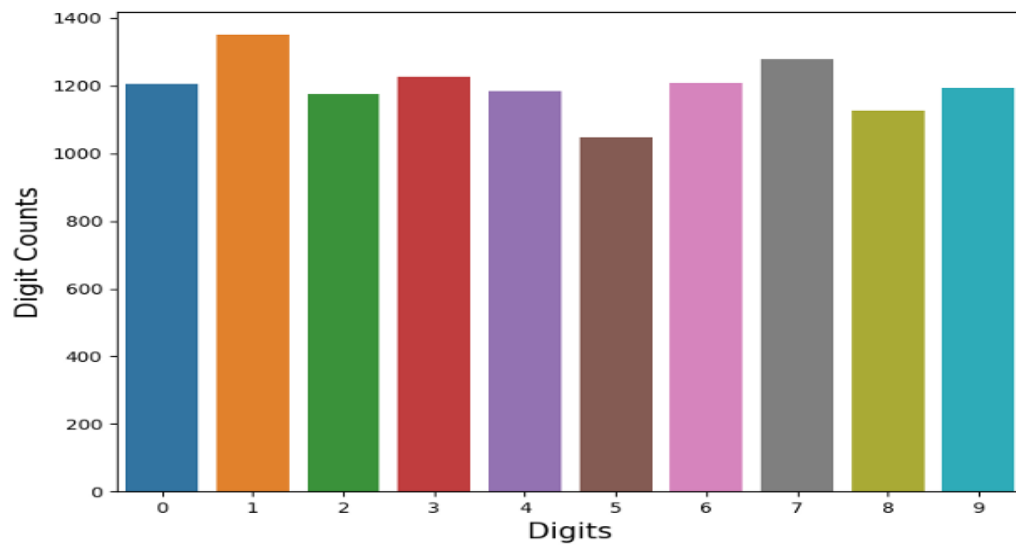


Figure 19: Exploring data distribution in MNIST dataset

```
pred = svm.predict(test)
accuracy_score(tslab, pred)
```

0.9183

Figure 20: Accuracy of SVM

```
print("Test loss: ",score[0])
print("Test accuracy: ",score[1])
```

Test loss: 0.02539978411577631
Test accuracy: 0.9929999709129333

Figure 21: Accuracy of CNN

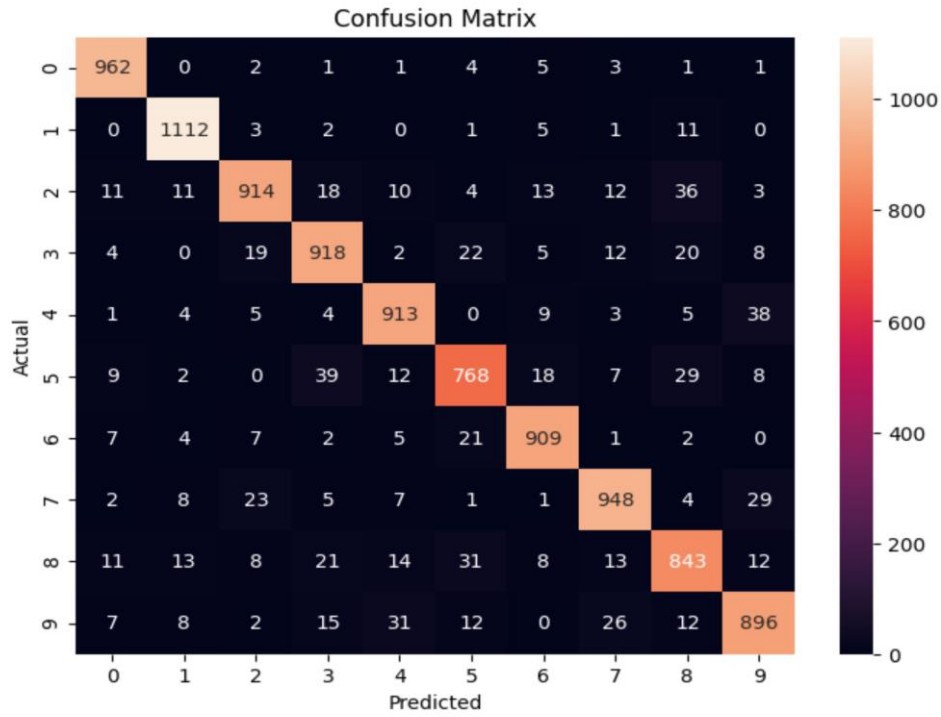


Figure 22: Confusion matrix of SVM

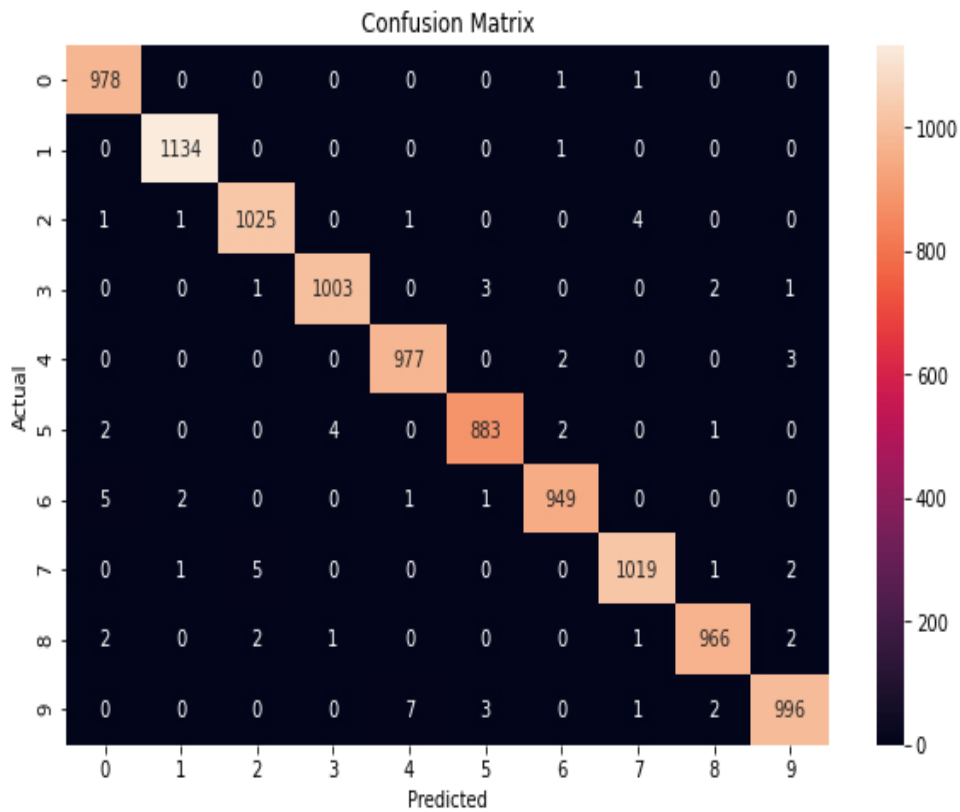


Figure 23: Confusion matrix of CNN

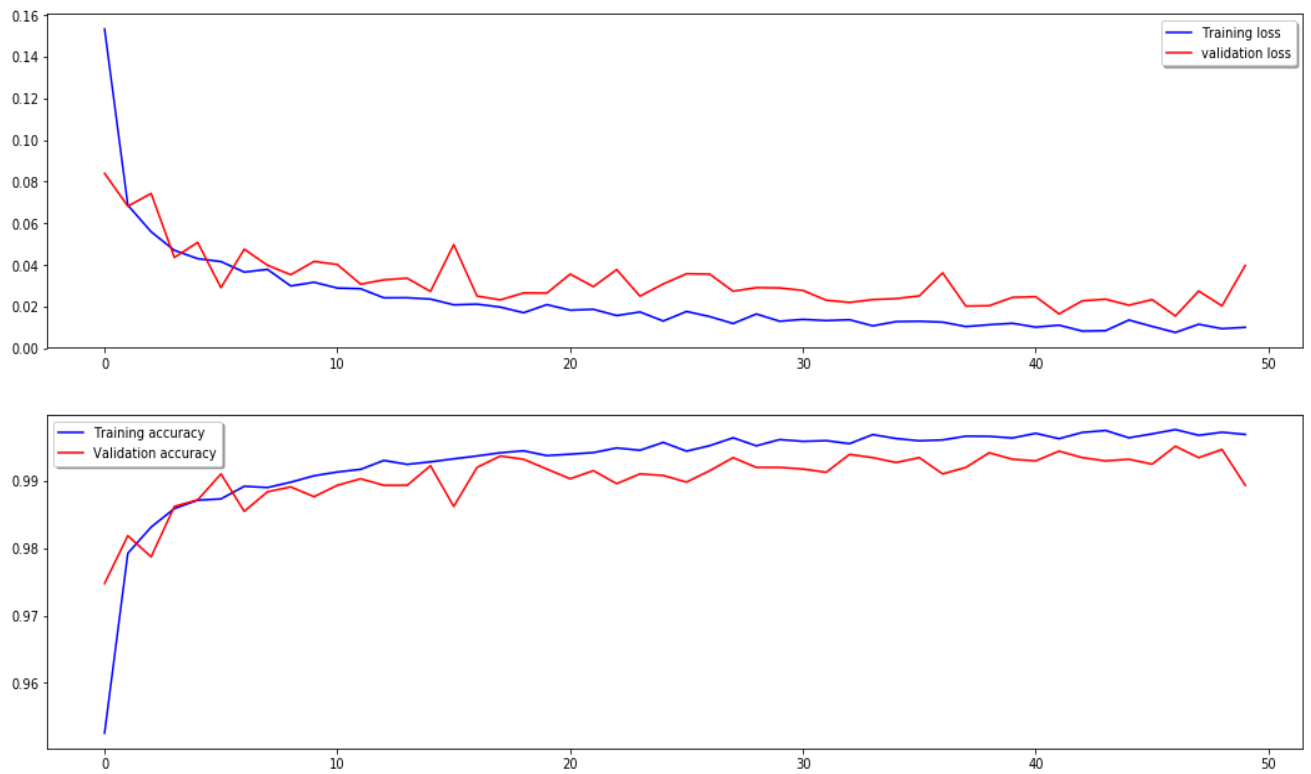


Figure 24: Graph illustrating training Vs testing loss and accuracy

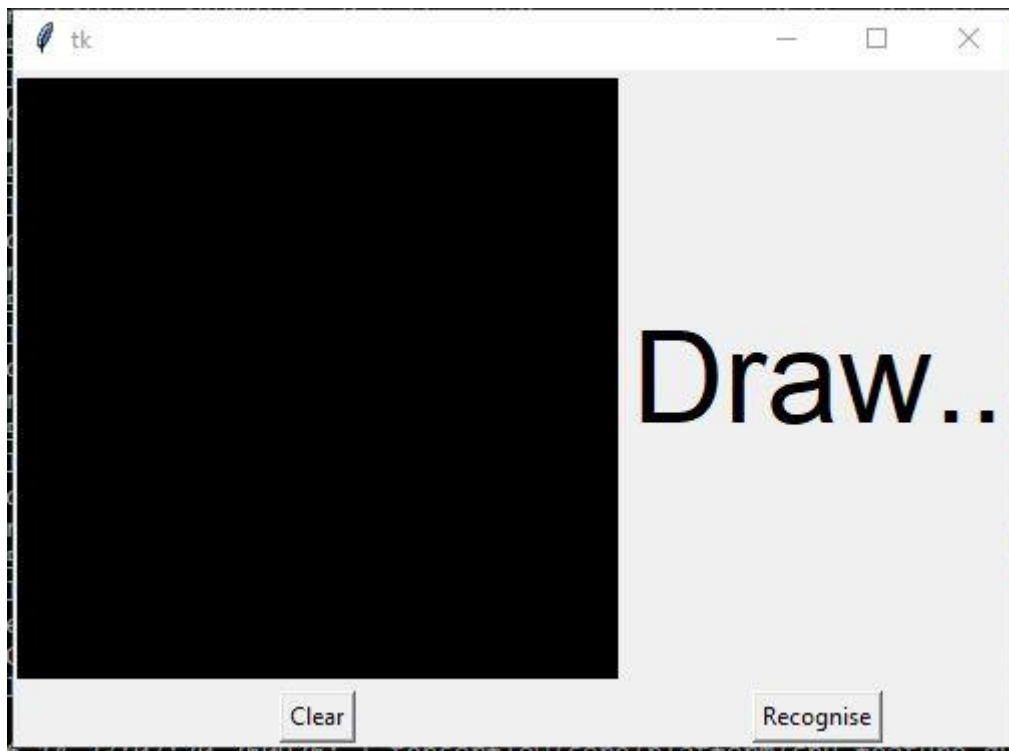


Figure 25: GUI interface

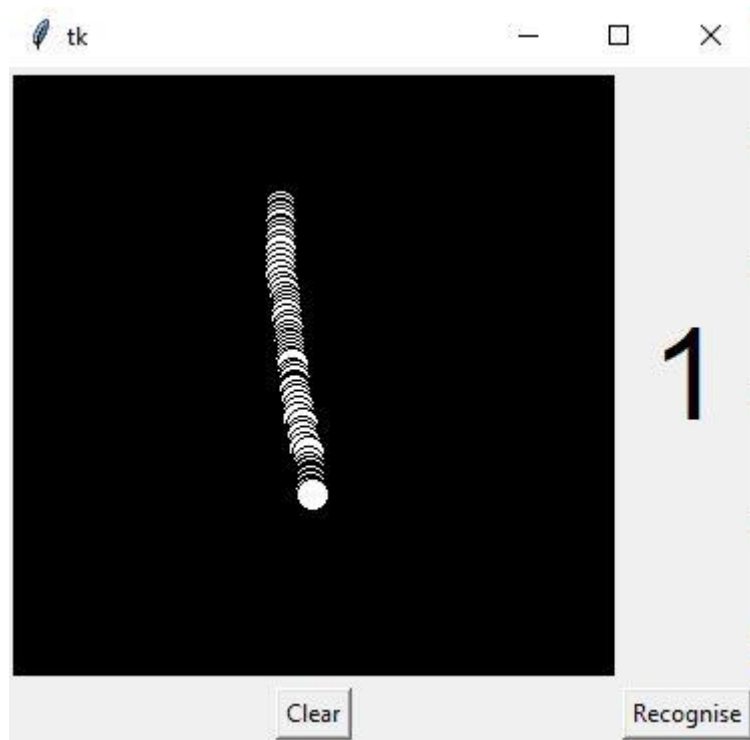


Figure 26: Sample prediction-1

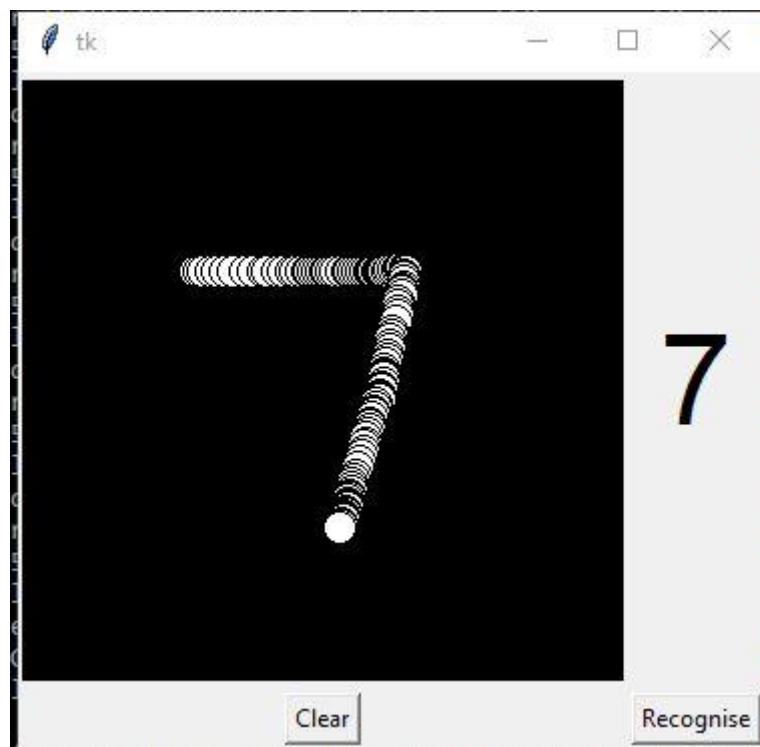


Figure 27: Sample prediction-2

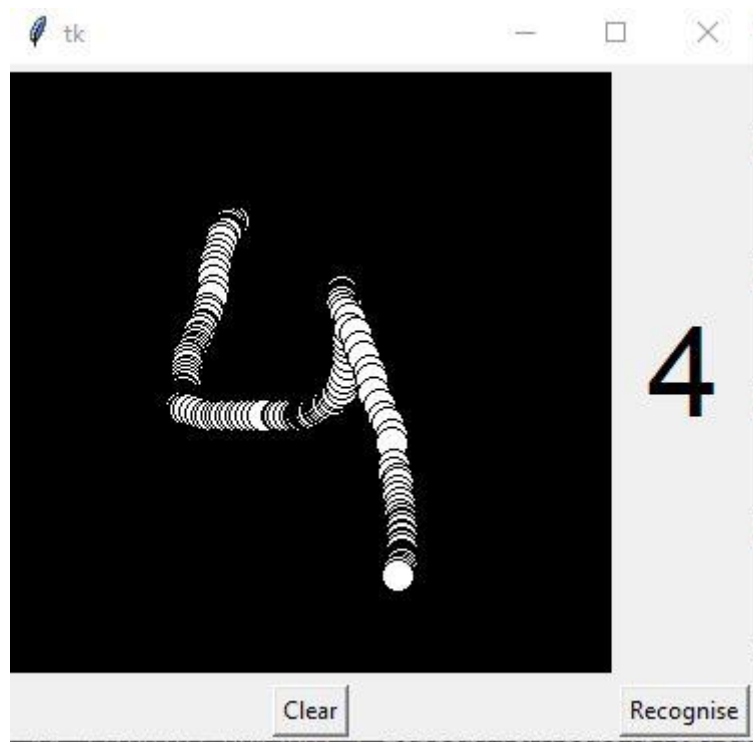


Figure 28: Sample prediction-3

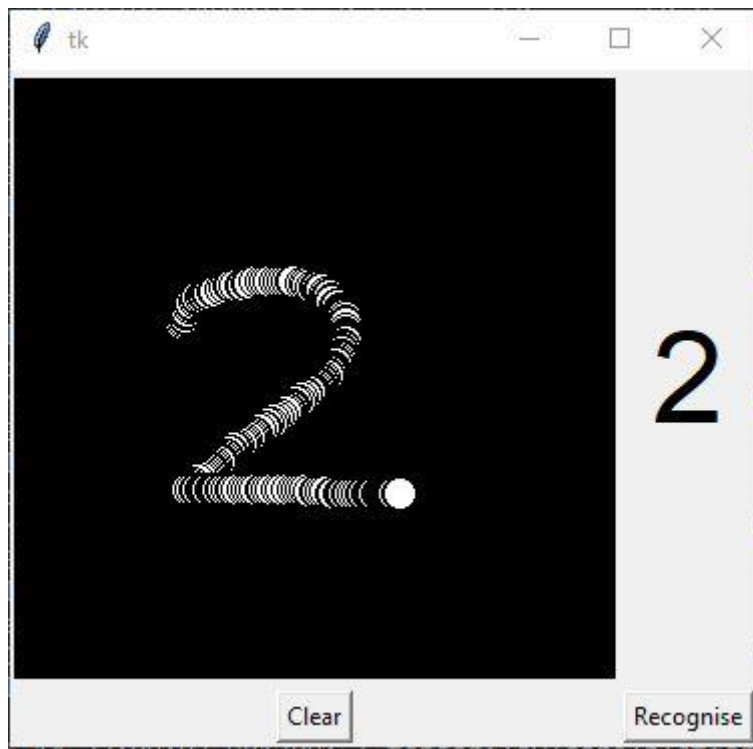


Figure 29: Sample prediction-4

9. CONCLUSION

This project has implemented two models namely Support Vector Machine and Convolutional Neural Network for handwritten digit recognition using MNIST datasets. It compared them based on their working accuracy, execution time and complexity among them and concluded that:

SVM is one of the basic classifiers that's why it's faster than most algorithms and in this case, gives the maximum training accuracy rate but due to its simplicity, it's not possible to classify complex and ambiguous images as accurately as achieved with CNN.

It has been found that CNN gave the most accurate results for handwritten digit recognition but the only drawback is that it took an exponential amount of computing time.

Increasing the number of epochs without changing the configuration of the algorithm is useless because of the limitation of a certain model and it has been noticed that after a certain number of epochs the model starts overfitting the dataset and gives us the biased prediction.

So, it solidifies that CNN is the best candidate among the SVM and CNN based on the accuracy rate, for any type of prediction problem including image data as an input.

10. FUTURE SCOPE

The future development of the applications based on algorithms of deep and machine learning is practically boundless. In the future, work on a denser or hybrid algorithm than the current set of algorithms with more manifold data to achieve the solutions to many problems can be done. Better GUI interfaces could also be developed which can use the scanner facility of mobile phones to recognise the hand-written digits.

10. BIBLIOGRAPHY

- [1] "Handwriting recognition", https://en.wikipedia.org/wiki/Handwriting_recognition
- [2] "What can a digit recognizer be used for?", <https://www.quora.com/What-can-a-digit-recognizer-be-used-for>
- [3] "Handwritten Digit Recognition using Machine Learning Algorithms", S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair.
- [4] "Handwritten Digit Recognition Using Deep Learning", Anuj Dutt and Aashi Dutt.
- [5] "Handwritten recognition using SVM, KNN, and Neural networks", Norhidayu binti Abdul Hamid, Nilam Nur Binti Amir Sharif.
- [6] "Recognition of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and Comparison of Performance for Various Hidden Layers", Fathma Siddique, Shadman Sakib, Md. Abu Bakr Siddique.
- [7] "Advancements in Image Classification using Convolutional Neural Network" by Farhana Sultana, Abu Sufian & Paramartha Dutta.
- [8] "Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18 F-FDG PET/CT images" by Hongkai Wang, Zongwei Zhou, Yingci Li, Zhonghua Chen, Peiou Lu, Wenzhi Wang, Wanyu Liu, and Lijuan Yu.
- [9] rishikakushwah16, "SVM_digit_recognition_using_MNIST_dataset", GitHub, [Online]. Available: https://github.com/rishikakushwah16/SVM_digit_recognition_using_MNIST_dataset
- [10] "Support Vector Machine Algorithm", <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>
- [11] "Crash Course on Multi-Layer Perceptron Neural Networks" <https://machinelearningmastery.com/neural-networks-crash-course/>
- [12] "An Introduction to Convolutional Neural Networks" Available: https://www.researchgate.net/publication/285164623_An_Introduction_to_Convolutional_Neural_Networks
- [13] "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning", Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Available: <https://arxiv.org/pdf/1811.03378.pdf>
- [14] "Basic Overview of Convolutional Neural Network" <https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17>
- [15] dixitritik17, "Handwritten-Digit-Recognition", GitHub, [Online]. Available: <https://github.com/dixitritik17/Handwritten-Digit-Recognition>

- [16] “Support vector machines”, <https://scikit-learn.org/stable/modules/svm.html>
- [17] “LIBSVM”, <https://en.wikipedia.org/wiki/LIBSVM>
- [18] Flame-Atlas, “Handwritten_Digit_Recognition_using_MLP”, GitHub, [Online]. Available:
- [19] https://github.com/Flame-Atlas/Handwritten_Digit_Recognition_using_MLP/blob/master/ANN.py
- [20] “Image Classification using Feedforward Neural Network in Keras” <https://www.learnopencv.com/image-classification-using-feedforward-neural-network-in-keras/>
- [21] “How Do Convolutional Layers Work in Deep Learning Neural Networks” available: <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/>
- [22] “Rectified Linear Unit (ReLU)”, <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>
- [23] “Pooling Layers for Convolutional Neural Networks”, <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/>
- [24] “Dropout in (Deep) Machine learning”, <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>
- [25] “Basic Overview of Convolutional Neural Network (CNN)”, <https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17>
- [26] “Understand the Softmax Function”, <https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e8>
- IJCATM*: www.ijcaonline.org