HANDWRITTEN CHARACTER RECOGNITION AND DIGIT CHARACTER RECOGNITION USING DEEP LEARNING

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/

Information Technology

By

Manish Gupta (191553)

Abhishek (191441)

Under the supervision of

Mr. Praveen Modi

Department of Computer Science & Engineering and InformationTechnology

Jaypee University of Information Technology

Waknaghat,Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled **"Handwritten Character Recognition and Digit Character Recognition Using Deep Learning"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own workcarried out over a period from August 2022 to May 2023 under the supervision of **Mr. Praveen Modi, Assistant Professor, CSE/IT**. I also authenticate that I have carried out the above-mentioned project work under the proficiency stream **Machine Learning.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Manish Gupta (191553)
Abhishek (191441)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Mr. Praveen Modi
Assistant Professor
Computer Science & Engineering and Information Technology
Dated:

Plagiarism Certificate

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ________________________ __**Department:** _______________ **Enrolment No** ________

Contact No. ___________________________**E-mail.** __________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): ______________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am grateful and wish my profound indebtedness to Supervisor **Mr. Praveen Modi,** Assistant Professor, Department of CSE & IT, Jaypee University of Information Technology, Waknaghat. The deep Knowledge & keen interest of my supervisor in the field of "**Machine Learning"** to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to **Mr. Praveen Modi,** Department of CSE & IT, for her kind help to finish my project. I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which havedeveloped their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Manish Gupta **Abhishek**

191553 **191441**

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Abbreviations	**Meaning**
CNN	Convolutional Neural Network
RNNs	Recurrent Neural Networks
OCR	Optical Character Recognition
AI	Artificial Intelligence
CNTK	Cognitive Toolkit
GANs	Generative Adversarial Networks
FC	Fully Connected
SGD	Stochastic Gradient Descent
ANN	Artificial Neural Network

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Handwritten character recognition and digit character recognition are important tasks in image processing and machine learning. Deep learning techniques have shown great success in achieving high accuracy in these tasks, especially with the use of Convolutional Neural Networks (CNNs).

CNNs are a type of deep neural network that can effectively learn and extract features from images. They are well suited for image classification tasks, as they can detect patterns and features at different levels of abstraction. In handwritten character recognition and digit character recognition, CNNs can be used to learn and recognize the unique features of each character or digit.

In the training phase of the model, the CNN learns from a large dataset of labeled images of handwritten characters or digits. The model adjusts its internal parameters, known as weights and biases, to minimize the difference between its predicted output and the true label of each image. In the testing phase, the model uses these learned weights and biases to predict the label of new, unseen images.

One key advantage of using deep learning techniques with CNNs is their ability to automatically learn and extract features from the input images. This reduces the need for manual feature engineering, which can be time-consuming and error-prone.

In conclusion, deep learning techniques using CNNs have shown great success in achieving high accuracy rates in handwritten character recognition and digit character recognition. These techniques have the potential to be used in a wide range of applications, including document analysis, handwriting recognition, and automated data entry.

Chapter 1 – INTRODUCTION

1.1 Introduction

Artificial intelligence is utilized to identify handwritten characters, such as numbers and letters, using deep learning algorithms. Artificial neural networks, the foundation ofdeep learning algorithms, are capable of learning from experience and upgrading themselves without the aid of explicit programming.

To identify handwritten characters, a photo of the handwritten character must be implemented, and it needs to be turned into a digital representation that a computer can process. Typically, an optical character recognition (OCR) system is used for this. A deep learning model that has been trained on a sizable dataset of handwritten characters is given the digital image. This training data will be utilized by the model to determine the character in the image and output that character.

For handwritten character recognition, deep learning models frequently consist of multiple levels of synthetic neurons, each of which analyses an individual part of the input data. Large datasets of handwritten characters may be used to train these models, allowing them to discover patterns and properties that are shared by many characters. Convolutional neural network (CNNs), which are made particularly for processing pictures, and neural networks with recurrent connections (RNNs), which can detect temporal relationships in the input data, are a couple of methods that can be used to improve the performance of these models.

Handwriting-based authentication systems, automated form processing, and digitalization of old documents are only a few applications of deep learning-based handwritten character recognition in the real world. Despite the complexity of the problem, recent advances in deep learning have greatly improved the precision of handwritten character identification, opening up a possible area for further investigation and research.

Due to the availability of enormous amounts of annotated data and the development of powerful computer resources, deep learning-based handwritten character recognition has become more and more popular recently. Deep learning algorithms have surpassed traditional machine learning algorithms, especially for difficult tasks like handwriting detection.

One of the most popular deep learning models for handwritten character recognition is the convolutional neural network (CNN). Since CNNs are designed to analyses visual data, they may effectively capture the spatial relationships between pixels in a picture. CNNs are particularly useful for recognizing handwriting patterns since they have been shown to perform at the cutting edge in handwritten character recognition tasks.

Recurrent neural networks (RNNs) are another deep learning method that might be applied to the recognition of handwritten characters. RNNs are effective in recognizing handwriting because it is just a sequence of strokes, and they are designed to handle sequential input. RNNs are able to record and distinguish the temporal correlations between strokes even when characters are written in an unusual way.

To train a deep learning model for handwritten character recognition, a large amount of labelled data is required. This may be done by collecting handwriting samples from a variety of sources, such as private papers, old documents, and public databases. The data is then appropriately categorized before being used to build a deep learning model.

Handwriting-based authentication systems, automated form processing, and document digitization are a few examples of real-world applications for deep learning-based handwritten character recognition. Handwritten characters on scanned paper may be automatically detected and converted into digital text to make it easier to search for and analyses a document's content.

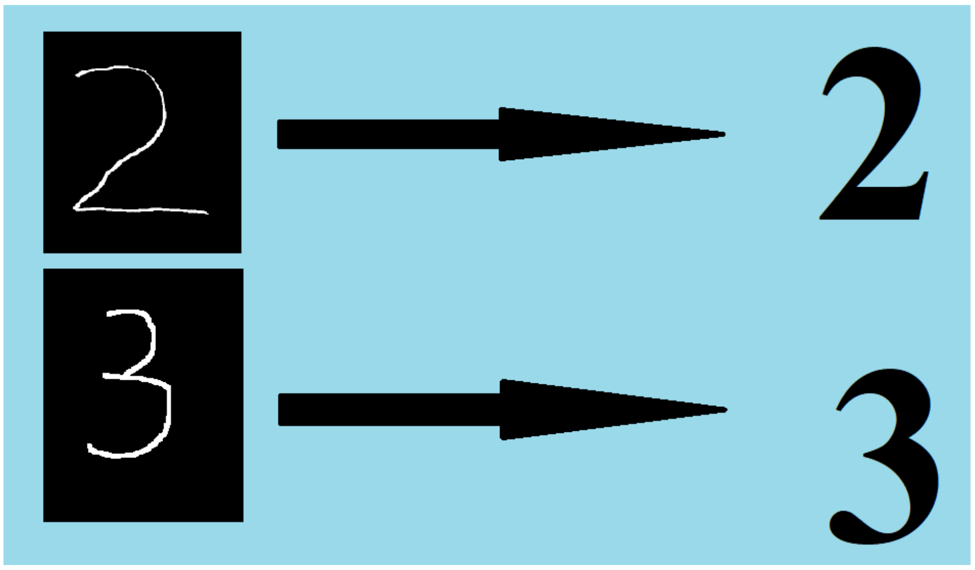


Fig 1: Handwritten Digit Recognition with CNN

As we can see in the Fig 1 depicts a scenario where a user inputs the characters 2 and 3, and a CNN model is used to classify them. In addition to manually created features, many contemporary practical implementations of machine learning employ linear classifiers. A feature vector's component parts are weighted together using a two-class linear classifier. The weighted total determines which category the input falls into if itis higher than the threshold. The input-output function must, however, be insensitive to unimportant adjusting parameters, such as changes in an object's location, orientation, or lighting, or changes inside the tone or intonation of voice, while also being extremely sensitive to particular minor differences.

The back-propagation method equation may be repeatedly employed to transport gradients throughout each of the sections, starting at the outcome so at submit. It is simple to determine the gradients with regard to a weight of each module after these gradients have been calculated. In many deep learning applications, feedforward neural networks are often employed. In these applications, they are trained to transfer a fixed-size input (such as an image) to a fixed-size output (such as the probabilities for various categories). In these networks, processing nodes are arranged in a series of linked layers that accept incoming data and output it. Information only travels through the network in a forward manner, from the input layer via the hidden levels to the output layer, thanks to the one-way connections between the layers. In order to reduce the difference between its output and the desired output, the network learns to modify the weights of these connections during training.

The unit set computes the weight value of its inputs from the preceding layer and feeds the result to a nonlinear function to advance to the subsequent layer. A rectified unit (Rectified linear), which is just a half-wave rectifier, is now the most well-known nonlinear function (z, 0). ReLU often learns significantly quicker in networks with multiple layers, allowing a thoroughly supervised system to be taught without uncontrolled pre training. In the past, neural networks employed smaller nonlinearities like tanh(z) or 1/(1+exp(z)). Hidden units are typically referred to as units that are not located in the output or input layer.

The hidden layers may be viewed as non-linearly bending the input such that the final layer can linearly separate the categories. By the late 1990s, the fields of machine learning, computer vision, and voice recognition had mostly abandoned neural networks and backpropagation. There was a general consensus that it was impossible to learn practical, multi-step feature extraction with minimal prior knowledge.

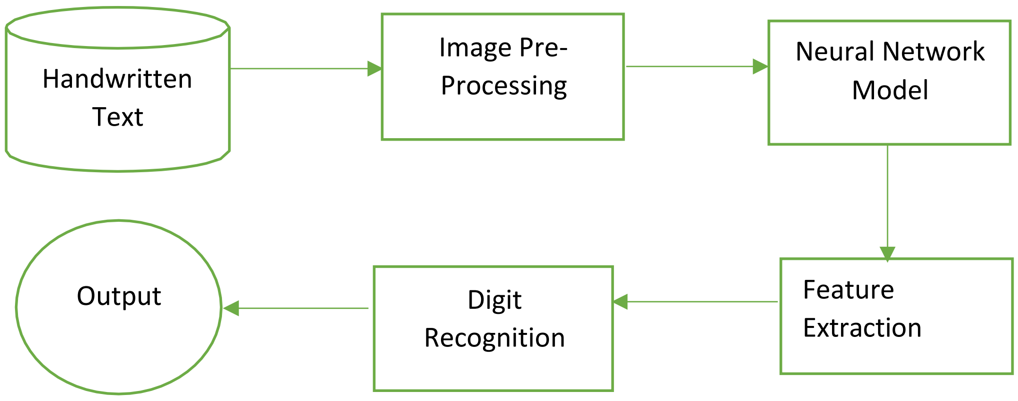


Fig 2: Overview of System [8]

As we can see that in Fig 2 that the first step in the process is image pre-processing, which involves preparing the handwritten text image for analysis. This step may involve adjusting the image for orientation, size, and resolution, as well as removing any noise or artifacts that could affect the accuracy of the recognition process. After the image has been pre-processed, the next step is feature extraction. This involves identifying the key features of the characters in the image that can be used to distinguish them from one another. Common techniques for feature extraction include edge detection, blob detection, and texture analysis. Once the features have been extracted, a neural network model is trained using the extracted features as input. The neural network is typically trained using a large dataset of labelled examples, which allows it to learn to recognize patterns and make accurate predictions. After the neural network has been trained, the next step is digit recognition. This involves using the trained neural network to recognize the characters in the input image and classify them according to their respective classes. Finally, the output is presented to the user in a readable format, such as text or a digital representation of the original handwriting.

However, compared to deep feedforward networks with complete connection between neighboring layers, there was one specific form that was considerably simpler to train and generalized far better. The neural network used here was convolutional. At a period when artificial neural networks were out of favor, it has numerous successful applications and has subsequently been warmly embraced by the field of computer vision.

The feature map's filtering process is mathematically a discrete convolution, thus the name. Although the convolutional layer's job is to find local conjunctions of the preceding layer's components, the association layer's job is to combine semantically related features into a single entity. The theme may be consistently identified using the coarse granularity of each feature's location, despite the fact that the exact locations of the parts that make up the pattern may fluctuate somewhat. In a single feature map, a typical pooling unit calculates the largest local array of units (or multiple feature maps). Neighbour pooling units use input form arrays that have been moved by more than a row or column to reduce the representation's dimensionality and produce small-scale invariance. Convolutional and fully linked layers are added after two or three steps of convolution, nonlinearity, and pooling. Convent’s gradient backpropagation makes training all weights throughout all filter banks as easy as training a normal deep network.

Computers can recognize and decipher numbers written by humans using a technique called handwritten digit recognition. However, it is difficult for robots to precisely recognize and distinguish between handwritten digits as they might vary in appearance and are not always flawless. The technique that employs handwritten character recognition is the answer to this issue identifies the digit that is presented in the picture of a digit.

One of the challenges in deep learning-based handwritten character identification is the variability of handwriting. It can be easy to spot individual distinctions in handwriting, which can be impacted by factors including age, gender, and writing style. Due to this variability, it is difficult to develop algorithms that can accurately recognise all types of handwritten.

To overcome this challenge, scholars have developed techniques like data augmentation, which artificially generates new handwriting samples by rotating and scaling the existing examples. This increases the variety of training data and improves the model's ability to recognise different handwriting styles.

Another challenge is that deep learning models need a lot of tagged data to be trained. In particular, character-level tagging is required for handwriting identification, which may be a time-consuming and expensive process. To get around this problem, researchers have developed semi-supervised and unsupervised learning algorithms that may use unlabelled input to supplement the labelled data and improve the performance of the model.

In addition to recognising individual characters, deep learning models can also distinguish phrases, sentences, and paragraphs. Unlike word recognition, which recognises a group of letters as a single word, sentence recognition recognises a string of words as a whole phrase. Natural language processing and automatic translation are two fields where these systems are helpful.

In conclusion, handwritten character recognition based on deep learning is a challenging and promising field with several practical applications. It is projected that the further growth of this discipline will be pushed by deep learning algorithms, data labelling techniques, and increases in computer power, leading to improved performance and new applications.

1.2 Problem Statement

To effectively identify and categories handwritten characters from photographs for handwritten character recognition using CNN. The shape, size, and style of handwritten characters might change, making it difficult to recognize them. In order to efficiently extract features from the input image, learn to recognize patterns, and accurately forecast the identification of the handwritten character, a deep learning model must be created. The most important properties for character recognition are taught to the CNN model using a huge dataset of labelled instances. The end objective is to create a model that is extremely precise and reliable and can be used to a number of tasks, including automatic form processing, optical character recognition, and digitizing handwritten documents.

1.3 Objectives

1. The system's main goal is to correctly identify and categories handwritten characters from photographs. In order to correctly recognize and categories the characters, the CNN model need to be able to learn and extract pertinent characteristics from the input image.
2. System must handle various alterations of the picture in order to identify the patterns, style of the input entered by the user and in order to achieve that target the model is trained on a diverse dataset.
3. For any real-time problem solving the system must always try to keep the use of minimum resources and in order to do that least processing power must be used in order to train and test the model.

1.4 Methodology

The methodology of handwritten character recognition using CNNs are:-

1. Data Gathering: In order to train model the most basic requirement is the data. The data that contain various pictures with different type of styles, patterns etc. must be gathered and used for the training purpose. For each image in the collection, the appropriate character class has to be assigned.
2. Pre-Processing: The data that is gathered in the initial phase must be pre-processed in order to prepare the data for analysis and this step requires converting RGB photos to grayscale in order to reduce the dimensionality of the input data and get rid of any colour changes that could reduce the accuracy of the identification process.
3. CNN Classifier: A CNN model must be created and trained to recognize the handwritten characters. This entails specifying the model's architecture, such as the quantity and kind of convolutional, pooling, and fully connected layers. The model is then trained using the pre-processed dataset, with the weights of the model being adjusted via backpropagation to reduce the loss function.
4. Feature Extraction: The trained CNN model is used to extract features from the input image. This involves passing the input image through the layers of the model to generate a feature map that captures the relevant characteristics of the image.
5. Recognized Character: Finally, the recognized character is determined by passing the feature map through one or more fully connected layers to generate a probability distribution over the possible character classes. The class with the highest probability is then selected as the recognized character.

Overall, the methodology of handwritten character recognition using CNNs involves collecting and pre-processing the input data, training a CNN model to extract relevant features and classify the characters, and using the trained model to recognize the characters in new input images. The accuracy of the system can be improved by using a larger and more diverse dataset, optimizing the architecture of the CNN model, and fine-tuning the model using techniques such as transfer learning.

1.4.1 Software Requirements

Python 3.9

Python 3.9 is the final version to offer various Python 2 data integrity layers, giving python project maintainers extra time to prepare the deletion of Python 2 functionality and the inclusion of supports for Python 3.9.

Advantages of Using Python

1. Independence across platforms

Python is chosen by developers over other coding because it can run without change across a variety of platforms. Since Python is cross-platform and compatible with Windows, Linux, and macOS, little modifications are necessary. Python is a coding language that works practically across all platforms; thus, a Python expert isn't really needed to explain the code.
Python's basic excitability makes software deployment easy and allows the development and usage of independent apps. The only technology that has the ability to create software from scratch is Python. Because other programmer requires other languages to finish the project in order to be declared complete, it is advantageous for developers.
Python's platform neutrality is advantageous to developers who typically require several resources to complete a single project.

2. Consistency and simplicity

Python is a haven for the vast majority of coders who seek consistency or simplicity in their job. The presentation process is made simpler by the concise and comprehensible Python code. Developers can write code using this language more quickly and clearly than with other programming languages. It enables the developer to get input from nearby developers in order to enhance the product or service.
Python is simpler than other programming languages, making it simpler for beginners to learn and master. Furthermore, seasoned people have found it

straightforward to construct solid systems and can focus their efforts on expanding their creativity and using machine learning to address real-world problems.

3. Frameworks and libraries variety

Libraries / frameworks are necessary to offer a suitable programming environment. Python frameworks and modules' trustworthy environment drastically accelerates programming development. Developers may use libraries to essentially use prewritten code to accelerate development while working on huge projects.
Simple-to-use techniques for use in deep - learning or deep learning applications are provided by PyBrain, a Python-based modular deep learning toolbox. For the best and most dependable development solutions, the Python frameworks / libraries offer the right framework and a tried-and-true environment.

Why Python is Most Preferable for our project

Due to the drive for automation, machine learning or AI a whole is quickly gaining in popularity. Search engines, spam filters, and recommendation systems are a few examples of how artificial intelligence (AI) may be applied to provide novel solutions to everyday issues.
Artificial intelligence (AI) is essential for solving practical issues and automating labor-intensive processes. Since Python is simple and dependable in comparison to other programming languages, it is regarded as a great tool for automating these operations. Programmers may discuss projects and offer suggestions on how to improve their code easily thanks to Python's vibrant development community.
Even the most accomplished developers and programmers still have a great deal to learn the about complex software development world. It is helpful to have connection to a vibrant community where individuals can talk about and swap opinions about projects. Given that it has a sizable developer community,

Python is a well-liked programming language for machine learning as well as other activities like data analysis, regression, web development, etc.
The forums for Python developers actively foster the growth of the overall artificial intelligence community. The forums help students improve their knowledge of Python-based deep learning, which expands the pool of experts. Python's popularity among large organizations is rising as a result of its efficiency and simplicity. Spotify and Google both utilize the computer language to crawl websites.

Software components or libraries used in building project

NumPy 1.21.2

NumPy is the name of a Python module that enables computation and manipulation of both multidimensional and the one array elements. A set of tools for working with an elevated multidimensional array object is also included.
Travis Oliphant created the NumPy package in 2005 by combining the advantages of the progenitor module Numeric with the features of another module, NumPy array. NumPy implements matrices, multidimensional arrays, and other complex data structures. These data structures enable the quickest computation of arrays and matrices. NumPy may be used to operate logically and mathematically on arrays. NumPy is supported and used by several other well-known Python programming, including pandas and matplotlib.

Matplotlib

The Matplotlib package may be used to generate 2D diagrams & plots using Python content. It provides a module called Pyplot that streamlines everything for plotting and includes components to manage line styles, textual style credits, arranging tomahawks, and other highlights. It provides a huge variety of graphs and plots, including histograms, bar graphs, power spectra, and error graphs. It is used with NumPy to provide a robust open-source MATLAB alternative environment. This one work well with both python and Pyplot, two graphical toolkits. Pyplot is the name of a plotting package for 2D graphics inside the Python programming language. Shells, web application servers, Python scripts, or other graphical user interface toolkits may all utilize it.

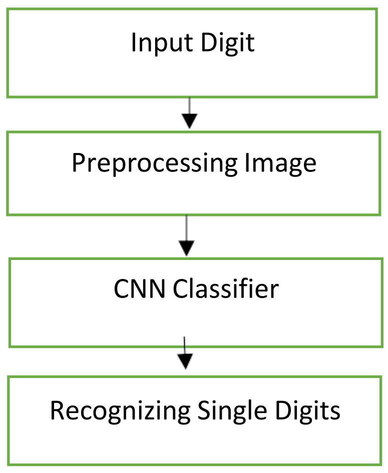


Fig 3: Handwritten Character Recognition Using CNN [7]

As we can see that in Fig 3 that firstly, the input data must be prepared, which includes collecting a large dataset of labelled handwritten digit images. This dataset can be obtained from various sources, such as publicly available databases or through manual digitization of handwritten digits. The dataset should include a wide range of variations in handwriting styles, sizes, and orientations to improve the accuracy and robustness of the model.

Next, the image pre-processing stage involves normalizing the images and removing any noise or artifacts that could affect the accuracy of the recognition system. Techniques such as image resizing, normalization, and filtering can be applied to improve the quality of the image and enhance the features.

The CNN classifier is then built, which typically consists of convolutional layers for feature extraction, pooling layers for down-sampling, and fully connected layers for classification. The model is trained using the pre-processed dataset and an optimization algorithm such as stochastic gradient descent. The weights of the model are adjusted during training to minimize the error between the predicted output and the true labels.

To recognize single digits, the pre-processed image is fed into the trained CNN model, which extracts the relevant features and classifies the digit into its corresponding label. The output of the model is a probability distribution over the possible labels, and the label with the highest probability is selected as the predicted label for the input digit.

1.5 Organizations

Chapter 1: Introduction
It covers the various project related things such as, introduction, Problem statement, motivation and tells us about the reasons behind the choice of this project.

Chapter 2: Literature Survey
The chapter covers the literature surveyed as well as talks about the concepts that have been studied and understood.

Chapter 3: System Design & Development
This chapter talks about the system design and development of the project. The chapter covers how the whole model has been created and how the whole project works.

Chapter 4: Experiments & Result Analysis
This chapter covers the experiments and results for the project. Basically, talks about the different test cases for which we have tested the project.

Chapter 5: Conclusion
This chapter talks about the Conclusion of the whole project and also gives the future scope of the whole project and describes the application of the project.

Chapter 2 – LITERATURE SURVEY

- **Avanaei and A. S. Maida [1],** This study presents a new type of spiking convolutional neural network (CNN) that takes inspiration from biological systems. The network is trained in a layer-by-layer approach, with a convolutional/pooling layer followed by a feature discovery layer, both using bio-inspired learning methods. The kernels in the convolutional layer are trained with a sparse, spiking auto-encoder that represents primary visual features, while the feature discovery layer uses a probabilistic spike-timing-dependent plasticity (STDP) rule. The layer uses leaky, integrate-and-fire (LIF) neurons to represent complex visual features in a Winner-Takes-All (WTA) thresholded manner. The network is tested on the MNIST digit dataset using both clean and noisy images. Results show that the convolutional layer is suitable for multi-layer learning, and the network achieves recognition performance above 98% for clean images, thanks to the informative visual features extracted from the hierarchical convolutional and feature discovery layers. The network is also shown to be robust to additive noise, with a performance loss ranging from 0.1% to 8.5%.

- **Kale, Karbhari V., Prapti D. Deshmukh, Shriniwas V. Chavan, Majharoddin M. Kazi, and Yogesh S. Rode. [2],** this paper presents a method for recognizing handwritten compound characters using a Zernike moment feature descriptor and a classification system based on SVM and k-NN. The approach involves pre-processing and normalizing 27,000 handwritten character images into 30x30 pixel images and dividing them into zones. The pre-classification step categorizes the characters into three classes based on the presence or absence of a vertical bar. Zernike moment feature extraction is then applied to each zone. The proposed system achieves an overall recognition rate of up to 98.37% using SVM and 95.82% using k-NN classifiers.

- **R. Sethi and I. Kaushik [3],** This paper aims to showcase the progress made in recognizing hand-written characters and digits, which is a challenging task due to variations in shape and size. To address this challenge, the paper proposes the use of vertical and horizontal projection methods for segmentation and SVM for classification. Additionally, the convex hull algorithm is utilized for feature extraction.

- **LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998) [4].,** The paper examines various methods utilized for recognizing handwritten characters and digits, and compares their performance on a standard task. The study shows that convolutional neural networks (CNNs), which are specifically designed to handle the variability of 2D shapes, outperform other techniques. Real-life document recognition systems consist of multiple modules, and the paper introduces a new learning approach called graph transformer networks (GTN) that allows these modules to be trained globally using gradient-based methods. The paper also presents two systems for online handwriting recognition, demonstrating the advantages of global training and the flexibility of GTN. A graph transformer network is described that utilizes CNN character recognizers for reading bank cheques, achieving high accuracy and deployed commercially for reading millions of cheques daily.

- **M. Hanmandlu,O.V. Ramana Murthy, Vamsi Krishna Madasu, [5],** This paper describes an OCR (Optical Character Recognition) system for recognizing handwritten Devnagari characters. The system uses a neural classifier to recognize basic symbols and employs four feature extraction techniques, namely intersection, shadow feature, chain code histogram, and straight line fitting features. Shadow features are computed globally for the entire character image, while the other features are computed by dividing the character image into different segments. The classification decision obtained from four Multi-Layer Perceptron (MLP) based classifiers is combined using a weighted majority voting technique. The system achieved an overall recognition rate of 92.80% on a dataset of 4900 samples, considering the top five choices. Comparison with other recent methods for handwritten Devnagari character recognition indicates that this approach has a better success rate than the others.

Chapter 3 – SYSTEM DESIGN & DEVELOPMENT

3.1 System Development

The process of developing a system that recognises and interprets handwritten characters, such as letters and numbers, requires employing software and algorithms. The following steps are often included in the process as shown in Fig. 4:

1. Data gathering: Gathering a dataset of handwritten characters for the recognition system's training and testing purposes.
2. Preparing the acquired data for use in training the recognition system by preparing it using methods like noise reduction, normalisation, and binarization.
3. To build a collection of representative characteristics that may be used to train the recognition system, a set of features from the pre-processed data are extracted, such as the form and orientation of the letters.
4. Training: The process of instructing a machine learning algorithm to use the extracted features as training data for a recognition system.
5. Testing and evaluation: To assess the performance and accuracy of the trained recognition system, the system was tested on a different dataset.
6. Deployment: Setting up the recognition system for usage in practical applications, such reading text from scanned documents or using handwriting as an input method for digital devices.

Overall, creating a handwritten character recognition system may be difficult and sophisticated, requiring knowledge of data processing, computer vision, and machine learning. However, handwriting recognition systems are becoming more crucial for a variety of applications due to the growing demand for digitization and automation in numerous industries.

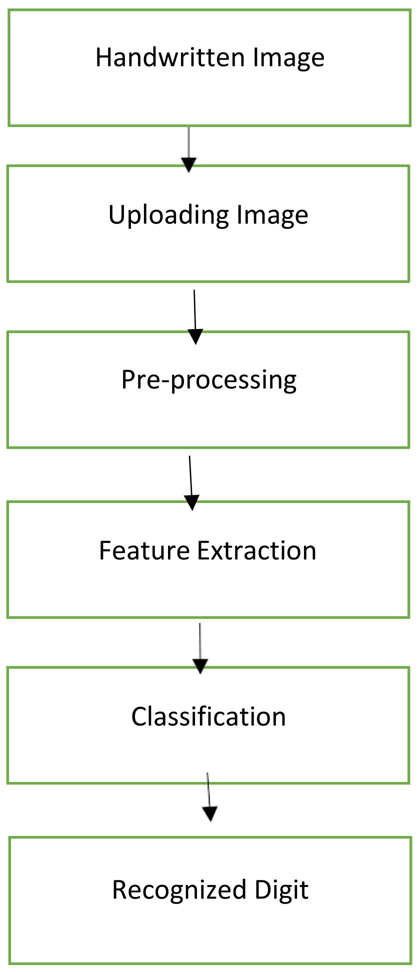


Fig 4: Handwritten Character Recognition Using Deep Learning [7]

3.2 Model Development

The process of model development includes various steps. The various steps are: -

3.2.1 Importing the libraries:

The required libraries to create the model are:

3.2.1.1 Keras

The Python-based high-level neural network API Keras may be used on top of a number of different deep learning frameworks, including TensorFlow, Microsoft Cognitive Toolkit (CNTK), or Theano. It was created with the intention of facilitating speedy experimentation, allowing users to quickly construct and test deep learning models.

Deep learning models may be defined and trained with ease because to Keras' straightforward, understandable, and versatile syntax. It supports several different neural network designs, such as feedforward, convolutional, recurrent, and more complex models like generative adversarial networks (GANs).

3.2.1.2 Seaborn

A well-liked data visualisation package based on Matplotlib is called Seaborn. With a focus on usability and aesthetics, it offers a higher-level interface for producing interesting and appealing statistical visuals.

For generating popular visualisations including scatter plots, line plots, histograms, bar charts, and heatmaps, Seaborn has a number of built-in functions. Additionally, it supports more complex visualisations like kernel density estimates, box plots, and violin plots.

The tools for statistical analysis and modelling that Seaborn offers in addition to its charting features include regression graphs, factor plots, and distribution plots. In conclusion, Seaborn is a useful tool for anybody dealing with data, offering a strong and adaptable foundation for producing educational and aesthetically pleasing visualisations.

3.2.2 Loading the dataset:

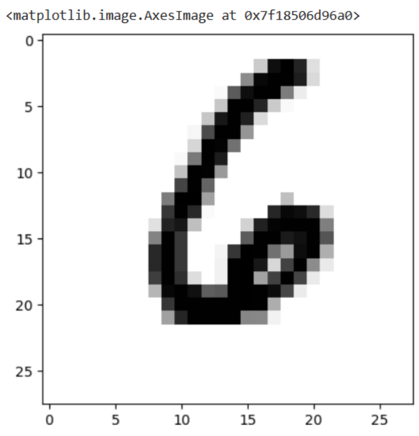
3.2.2.1 MNIST dataset

Datasets for computer vision are frequently obtained from the National Institute of Standards and Technology (NIST), which are then used to create and evaluate a variety of frameworks. Two different handwritten digit databases may be found in the MNIST dataset, which is kept by NIST. 250 pictures of digits make up the preparation set, with half of them written by Registration Department personnel and the other half by secondary school students. MNIST is frequently used as a dataset to show how robust machine learning methods are in comparison to other datasets.

The dataset consists of 10,000 photos used for testing and cross-validation and a second set of 60,000 images used for training. A fixed grayscale number in the centre of each image's 28x28 pixel region. With each element denoting the pixel intensity, the image may be represented as a 784-dimensional vector.

3.2.2.2 Retrieving the images:

The photo labels are loaded. The images must then be adjusted to (28,28) because they all require a different size in order to be recognised as shown in Fig. 5. Make a NumPy matrix from the images after that.



```
<matplotlib.image.AxesImage at 0x7f18506d96a0>
```

```
[[  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   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   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   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   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   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   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  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0 136 253 253 253 212 135 132  16   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   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   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   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]]
```

Fig 5: Retrieving the images

3.2.2.3 Compile Model:

We can compile the model now that it has been specified. The backend (so-called underpinnings) of the model is constructed using effective numerical libraries like Theano or TensorFlow.

Here, we provide a few requirements for the network's training qualities. By working out, we strive to determine the ideal weights for entrance. The operator used to find the best weights for both the network, the loss algorithm to use to assess the weights and biases, and any alternative kip’s we'd like to record and monitor during training must all be specified.

Table 1: Model Summary

Model: "sequential"

Layer (Type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 64)	640
activation (Activation)	(None, 26, 26, 64)	0
max_pooling2d (MaxPooling2D)	(None, 13, 13, 64)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	36928
activation_1 (Activation)	(None, 11, 11, 64)	0
max_pooling2d_1 (MaxPooling 2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 64)	36928
activation_2 (Activation)	(None, 3, 3, 64)	0
max_pooling2d (MaxPooling2D)	(None, 1, 1, 64)	0
flatten (Flatten)	(None,64)	0
dense (Dense)	(None, 64)	4160
activation_3 (Activation)	(None, 64)	0

dense_1 (Dense)	(None, 32)	2080
activation_4 (Activation)	(None, 32)	0
dense_2 (Dense)	(None, 10)	330
activation_5 (Activation)	(None, 10)	0

Total params: 81,066
Trainable params: 81,066
Non-trainable params: 0

As shows in Table 1, it shows the architecture of a neural network for image recognition. The Conv2D layer performs convolution on the input image to extract features. The Activation layer applies a nonlinear activation function to the output of the Conv2D layer. The Max Pooling layer down samples the output of the previous layer to reduce the number of parameters. Finally, the Dense layer performs classification on the features extracted from the previous layers.

3.2.2.4 Use the model and plot the precision and loss graphs:

We build the model and use the fit function to apply it. The frequency and loss graphs are then drawn. Average training accuracy was 98.3%, while average validation accuracy was 97.6%.

3.2.2.5 Model to be used for the project

Convolution Neural Network

In simpler terms, CNN is an artificial neural network that is particularly good at recognizing patterns, especially in pictures. It works by using filters of different sizes and numbers to detect these patterns. Although they may also be utilized with one-dimensional and three-dimensional data, CNNs are especially made to function with two-dimensional image data. The "convolution" process is carried out by the convolutional layer, which is a fundamental part of a CNN. Considering that it functions well with data that is stored as grids, this layer is frequently utilized in CNN models. By deactivating part of the neurons, dropout layers are also employed during training to reduce overfitting.

To extract features from an image, convolution and max pooling are used. This involves applying 32 3×3 convolutional filters to a 28x28 image, followed by a max-pooling layer with a 2×2 pooling size, and then a second fully connected layer with 64 ×3×3 filters.

Convolutional neural networks (CNNs) use alternating layers of convolutional and subsampling operations, followed by one or more fully connected layers for classification or regression. Convolutional layers apply filters to the input data to extract local features, while subsampling layers reduce the dimensionality of the feature maps. The output of these layers is then passed to fully connected layers that perform the final classification or regression on the extracted features. Flattening is used to convert images into a set of numbers that can be fed into a deep neural network for further processing.

In convolutional layers, a dot product operation is performed to mix an input patch of data with a filter of smaller size. The filter-sized patch of the input and filter is multiplied element-wise to obtain a dot product, and the components of this result are combined to produce a single value. The filter is intentionally larger than the input to allow the same filter (set of weights) to be used repeatedly at different points on the input. Each overlapping section or filter patch of the input is subjected to filtering successively, from left to right and top to bottom.

Working on CNN

Convolutional neural networks are developed as layers of hypothetical neurons. Artificial neurons are mathematical devices with the ability to compute the weighted sum of several information sources and to generate initial respect. These synthetic neurons closely resemble their biological counterparts.

Layers in CNN Model

The fact that these layers occur repeatedly indicates the depth of our network, and this structure is referred to as a deep neural network as shown in Fig. 6.

a) Raw image pixels are given as the input.

b) Convolutional layer: Translates the output of the neuron layer from input layers. The filter that will be applied must be specified. Only 5×5 windows that move across input data and capture pixels having maximum intensities are allowed for each filter.

c) Rectified linear unit [ReLU] layer: offered an encoder on the data taken in the form of an image. ReLU function is employed in the back propagation scenario to prevent changes in pixel values.

d) A down sampling procedure in volume all along dimensions is carried out by the pooling layer (width, height).

e) Fully linked layer: The score class is targeted, and the input digits' maximum score is established.

There is a significant rise in complexity as we dig further and deeper into the levels. However, it would be worthwhile to go since while accuracy might improve, time consumption does, sadly, too.

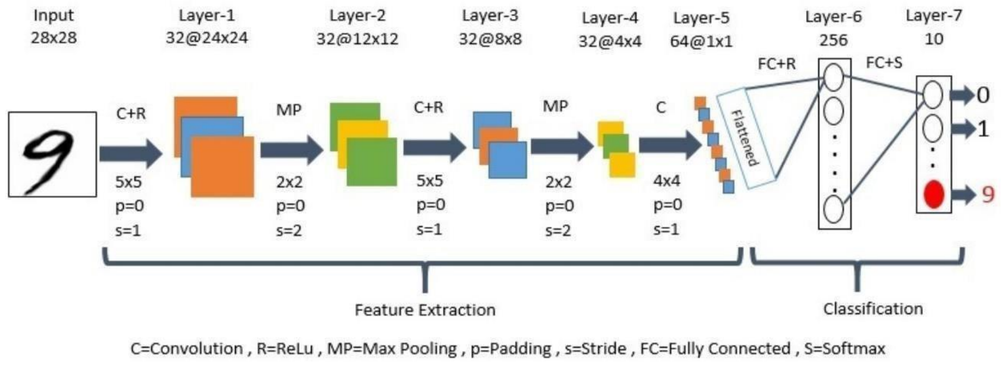


Fig 6: CNN Architecture for Handwritten Character Recognition [4]

1. Convolution Layer

Different characteristics from input photos are extracted by a convolutional neural network's first layer. Convolution is a mathematical technique that does this by slapping a filter over the input picture that has a specific size (M × M). Depending on the filter's size, the filter's components are multiplied by the appropriate components of the input picture to create a new matrix. A variety of filters are used during this process, each of which extracts a distinct characteristic from the original image.

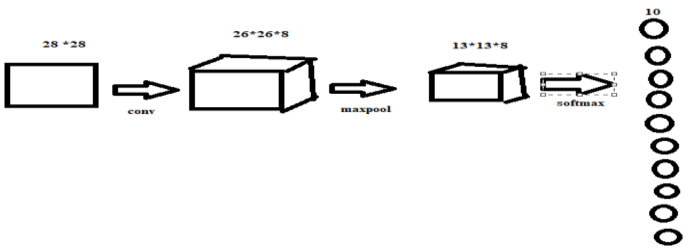


Fig 7: Convolutional Layer

As we can show in Fig 7. Convolutional layers extract features from input data, max pooling reduces dimensionality of feature maps, softmax provides a probability distribution over classes, and the output layer assigns a class label (0-9) to the input based on the highest probability in the softmax output.
The convolutional layer uses convolution to the input picture. The output of the Convolutional layer is down sampled by the MaxPooling layer. The SoftMax layer then classifies the characteristics that were retrieved from the earlier layers.
The outcome, dubbed the "Feature map," offers information on the image, including its corners & borders. Later, additional layers are added to this feature space to uncover new features from the source image.

2.Pooling Layer

After performing a convolutional operation on the input data in a neural network, a pooling layer is frequently included to cut down on the size of the feature maps that are produced. This size decrease aids in lowering the computational expense of supplementary layers. Each feature map is subjected to the pooling process individually, which has the effect of down sampling the map by averaging or maximising values of close values. The network becomes more effective as a result of this down sampling's contribution to reducing the connections between layers.

There are several different types of pooling techniques depending on the technology employed. The biggest help for Max Pooling comes from the feature map. The mean of the components in a section of a picture with a particular size is determined by average pooling. Sum pooling is used to add the elements of the selected part. Typically, the Pooling Layer connects the FC Level and the Deep Neural Layer.

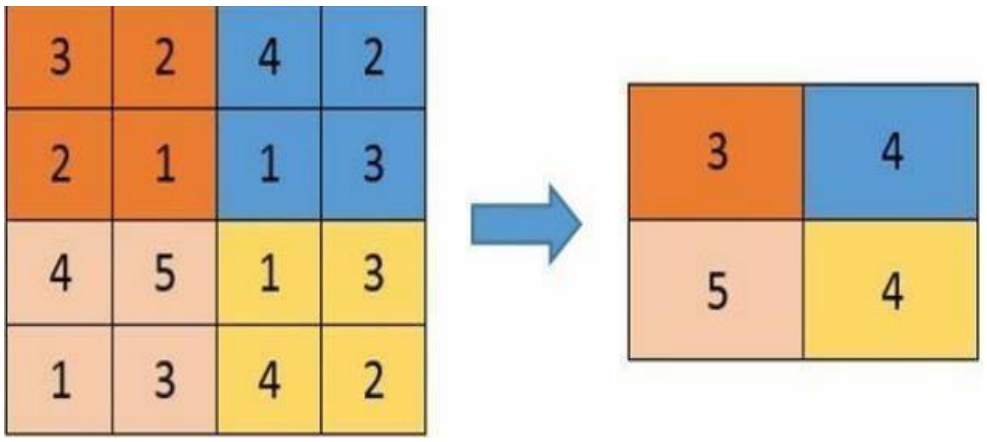


Fig 8: Max Pooling Layer [4]

As we can show in Fig 8 that Max pooling layers are a type of pooling layer commonly used in convolutional neural networks (CNNs) for image recognition tasks. The max pooling layer reduces the spatial size of the feature map by down-sampling the input through selecting the maximum value within a defined kernel window. This operation helps in reducing the computational complexity of the network and also makes the network more robust to slight spatial shifts or distortions in the input data. Max pooling layers also have a regularization effect that can help prevent overfitting. However, excessive pooling can result in loss of information, so it is important to balance the amount of pooling with the accuracy of the network.

3. Fully Connected Layer

Fully Connected (FC) layers are used in Convolutional Neural Networks (CNNs) to connect the cells or nodes in one layer to those in the following layer. These layers, which usually come before the output layer and are near the end of the network, have weights and biases that are learned during training. These FC layers aid in mapping previously discovered characteristics to the final output classes or values.

This section discusses an architecture for a neural network where an input image is first flattened into a vector and then passed through fully connected (FC) layers that manipulate the input mathematically.

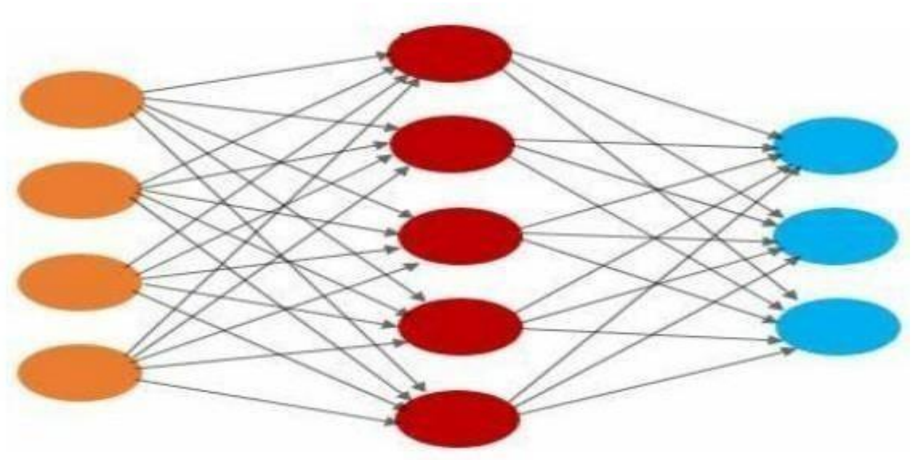


Fig 9: Fully Connected Layer [4]

As we can show in Fig 9 that the FC layers' primary function is to categorise the input image, which means to place it into one of several already established categories. After the input has been processed by the FC levels, the categorization process takes place in the network's later stages.

4. Dropout

An overfitted machine learning model starts to perform poorly when evaluated on fresh, untrained data because it has become overly specialized to the training data. This may occur when the model is too dependent on particular features or patterns in the training data and is unable to generalise to new data.

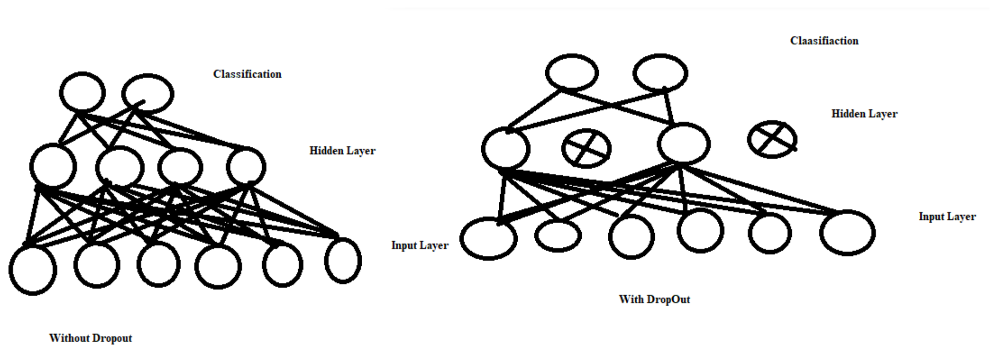


Fig 10: Dropout Layers

As we can show in Fig. 10 that Dropout layers are a regularization technique used in neural networks to prevent overfitting by randomly dropping out some of the neurons during training. Without dropout layers, the network is more likely to overfit to the training data and have poor performance on new, unseen data.

Dropout, a method that includes randomly eliminating a particular percentage of neurons from the input layer, is used to prevent overfitting in neural networks. In this instance, 30% of the input neurons are at random removed from the training population when the dropout is set to 0.3. By doing so, the model can be made smaller and avoid overfitting by not depending too much on a small number of neurons.

5. Activation Functions

- SoftMax: For multi-class classification issues, machine learning frequently uses the mathematical function known as SoftMax. SoftMax can be used to categorise photographs of handwritten characters into one of many potential classes (i.e., the individual alphabetic letters) in the context of handwritten character recognition.

  When given a vector of numerical values as input, the SoftMax function returns a second vector of the same length with each element representing a probability between 0 and 1. These probabilities show how confident the model is in each potential class. The result may be easily understood as a probability distribution across the classes because the probabilities of all classes sum up to 1.

A machine learning model is initially trained on a collection of labelled character image datasets before being applied to the identification of handwritten characters using SoftMax. The model's weights are changed to maximise its accuracy on the training set as it learns to link certain visual elements with particular classes.

Once trained, the model may be used to categorise fresh, undiscovered character pictures. To extract pertinent information, such as the character's shape or the distribution of pixel intensities, the picture is first pre-processed. The model then receives these characteristics and employs the SoftMax function to create a probability distribution across all conceivable classes. The predicted class for the input image is then determined by selecting the class with the highest probability.

Overall, SoftMax is a helpful tool for recognising handwritten characters since it offers a straightforward and comprehensible approach to divide photos into a number of classes with confidence scores.

- ReLU: In neural networks for handwritten character recognition, the Rectified Linear Unit (ReLU) is a frequently employed activation function. Compared to other activation functions like the sigmoid and tanh functions, ReLU provides a number of benefits, such as quicker training convergence, improved activation sparsity, and computing economy.

  The handwritten image's pixel values are fed into a neural network used for character recognition, and the output layer generates the anticipated class label (i.e., the recognised character). There are one or more hidden layers in between the input and output layers that perform a series of linear changes on the input before sending the output via the activation function.

  The ReLU activation function is defined as $f(x) = \max(0,x)$, where x is the input to the neuron, and it performs a non-linear adjustment to each hidden layer neuron's output. If the input value is positive, the ReLU function returns the value; if not, it returns zero. This non-linear modification increases the network's ability to recognise handwritten letters and enables it to learn sophisticated representations of the input data. Due to its simplicity, computational efficiency, and capacity to learn intricate representations of input data, ReLU is a well-liked option.

3.2.2.6 Training & Validation

A neural network model must be trained on a dataset once it is created in order for it to learn and perform better. To decrease mistakes and increase accuracy, optimizers are used to change the model's training parameters, such as the weights and learning rate. By reducing the possible outcomes, optimizers aid in the solution of optimisation issues.
In neural networks, a variety of optimizers are employed, including Adam, Stochastic Gradient Descent (SGD), and Gradient Descent. Each optimizer uses a different approach to change the neural network model's parameters and enhance training results.

3.2.2.6.1 Gradient descent

A well-liked optimization approach called gradient descent is used in machine learning to train models, particularly those for handwritten character recognition. Building a model that can properly divide handwritten character pictures into distinct categories is the aim of handwritten character recognition. The model must understand the patterns and characteristics that set one character apart from another in order to do this.
In order to reduce the difference between the model's predictions and the actual labels, the model modifies its internal parameters while being fed a collection of labelled pictures.
Gradient descent adjusts the model's parameters iteratively in the direction of the loss function's steepest fall. This procedure keeps on until the algorithm minimises the loss function.

Gradient descent may be used to update the weights and biases of a neural network that has been trained on picture data in handwritten character recognition. A picture of a handwritten character is sent into the network, which then processes it through many hidden layers to create an output that matches the expected class label. The gradient descent approach is used during training to update the network's parameters, minimising the difference between the expected and actual labels.

3.2.2.6.2 Adam Optimizer

A well-liked optimisation technique frequently employed in the development of deep learning models, particularly those for handwritten character recognition, is the Adam optimizer.

The objective of handwritten character recognition is to correctly categorise photographs of handwritten characters. A deep learning model, such as a convolutional neural network (CNN), which learns to recognise patterns in the pictures, can be used to do this task. The Adam optimizer modifies the neural network's weights during training to reduce the discrepancy between the expected and actual classifications of the pictures in the training dataset. By calculating the gradients of the loss function with respect to the weights and utilising those gradients to update the weights in a way that minimises the loss, the optimizer is able to do this.

Since the Adam optimizer adjusts the learning rate for each weight based on the gradients calculated during training, it is especially effective for training deep learning models. This aids in keeping the optimizer from becoming stuck in local minima, which can speed convergence and produce better outcomes.

3.3 Activity Diagram

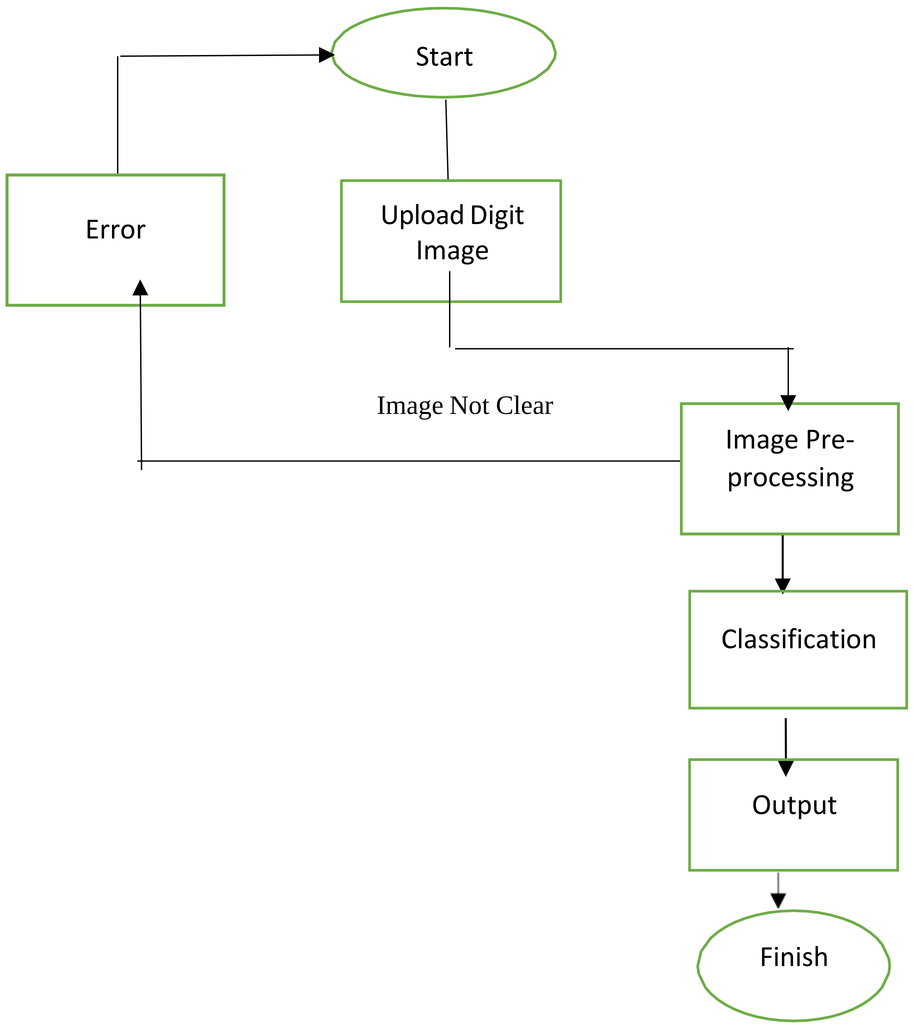
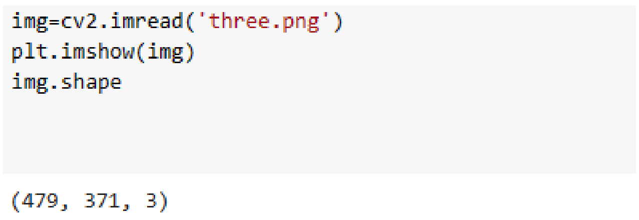


Fig 11: Activity Diagram

An activity diagram is a graphical representation of the steps involved in a process or system. As we can see in Fig 11 that in handwritten digit recognition, the activity diagram would start with the "start" node and proceed to the "upload image" node, where the user uploads the image of the handwritten digit. The diagram would then proceed to the "image pre-processing" node, where the uploaded image is pre-processed to enhance its quality and remove any noise or distortions. Next, the diagram would go to the "classification" node, where the pre-processed image is input to a classification algorithm, such as a neural network, which recognizes the digit and outputs the corresponding numerical value. Finally, the diagram would proceed to the "output" node, where the recognized digit is displayed to the user, before ending with the "finish" node. Overall, the activity diagram provides a clear visualization of the steps involved in the handwritten digit recognition process.

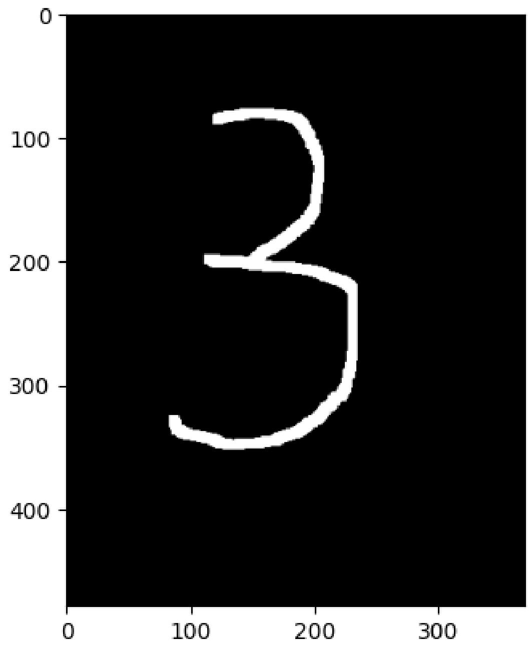
CHAPTER 4- PERFORMANCE ANALYSIS

Taking an input from a user:

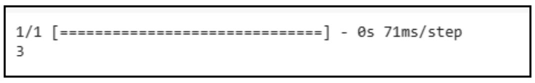


```
img=cv2.imread('three.png')
plt.imshow(img)
img.shape
```

```
(479, 371, 3)
```



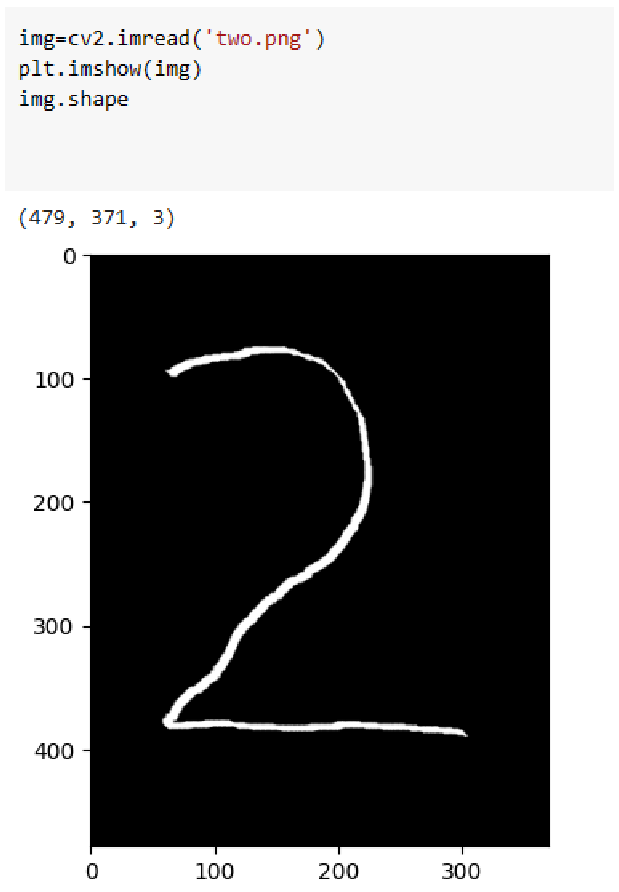
Output:



```
1/1 [==============================] - 0s 71ms/step
3
```

Fig 12: Input & Output-1

Taking an input from a user:



```
img=cv2.imread('two.png')
plt.imshow(img)
img.shape
```

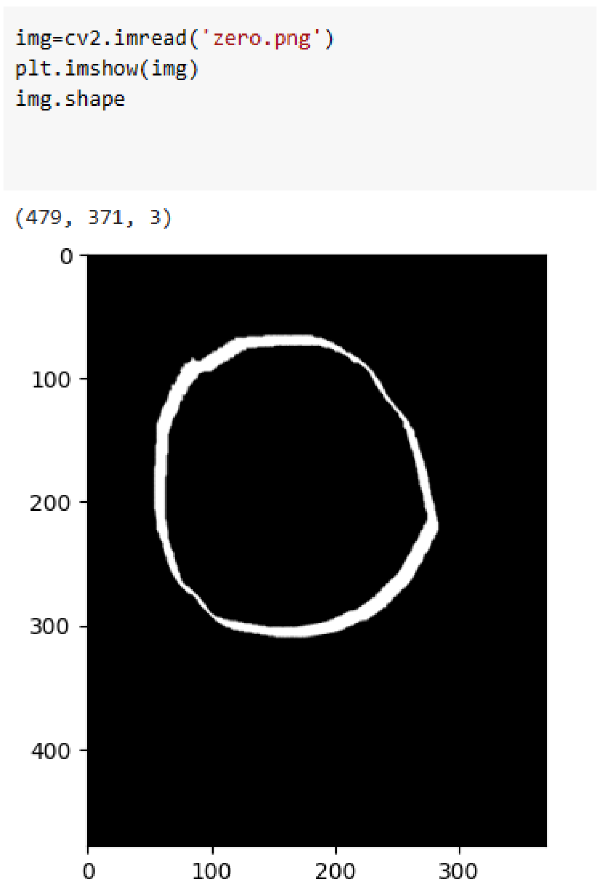
```
(479, 371, 3)
```

Output:

```
1/1 [==============================] - 0s 20ms/step
2
```

Fig 13: Input & Output-2

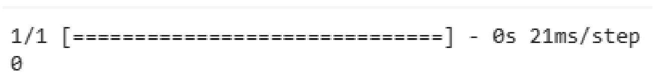
Taking an input from a user:



```
img=cv2.imread('zero.png')
plt.imshow(img)
img.shape
```

```
(479, 371, 3)
```

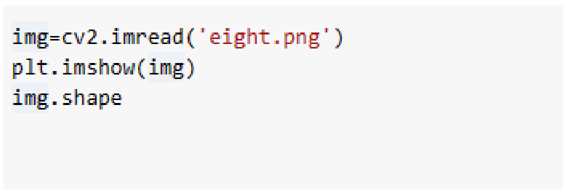
Output:



```
1/1 [==============================] - 0s 21ms/step
0
```

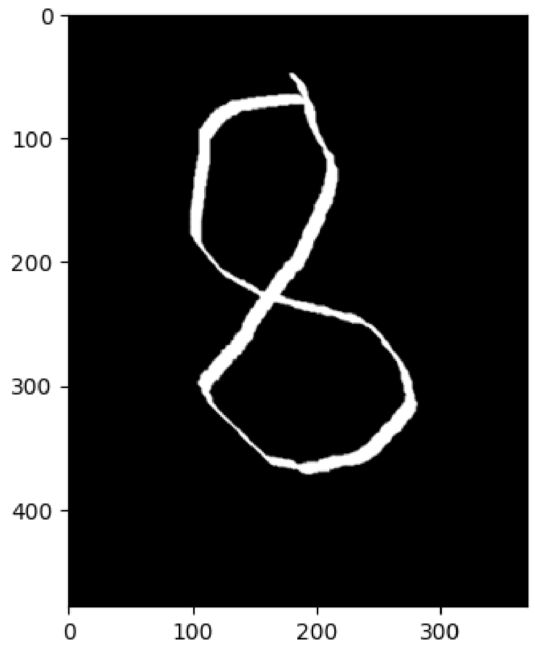
Fig 14: Input & Output-3

Taking an input from a user:

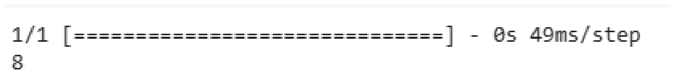


```
img=cv2.imread('eight.png')
plt.imshow(img)
img.shape
```

```
(479, 371, 3)
```



Output:

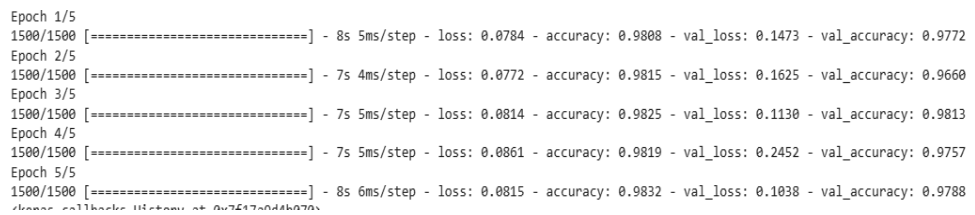


```
1/1 [==============================] - 0s 49ms/step
8
```

Fig 15: Input & Output-4

Accuracy on test set:

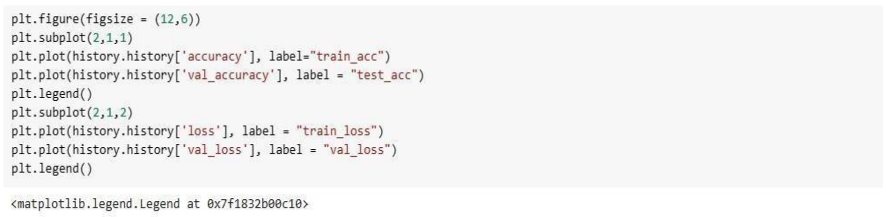
We obtained an accuracy of 97.8% on the test set.



```
Epoch 1/5
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0784 - accuracy: 0.9808 - val_loss: 0.1473 - val_accuracy: 0.9772
Epoch 2/5
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0772 - accuracy: 0.9815 - val_loss: 0.1625 - val_accuracy: 0.9660
Epoch 3/5
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0814 - accuracy: 0.9825 - val_loss: 0.1130 - val_accuracy: 0.9813
Epoch 4/5
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0861 - accuracy: 0.9819 - val_loss: 0.2452 - val_accuracy: 0.9757
Epoch 5/5
1500/1500 [==============================] - 8s 6ms/step - loss: 0.0815 - accuracy: 0.9832 - val_loss: 0.1038 - val_accuracy: 0.9788
```

Fig 16: Accuracy of test net

Accuracy and loss Function:



```
plt.figure(figsize = (12,6))
plt.subplot(2,1,1)
plt.plot(history.history['accuracy'], label="train_acc")
plt.plot(history.history['val_accuracy'], label = "test_acc")
plt.legend()
plt.subplot(2,1,2)
plt.plot(history.history['loss'], label = "train_loss")
plt.plot(history.history['val_loss'], label = "val_loss")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f1832b00c10>
```

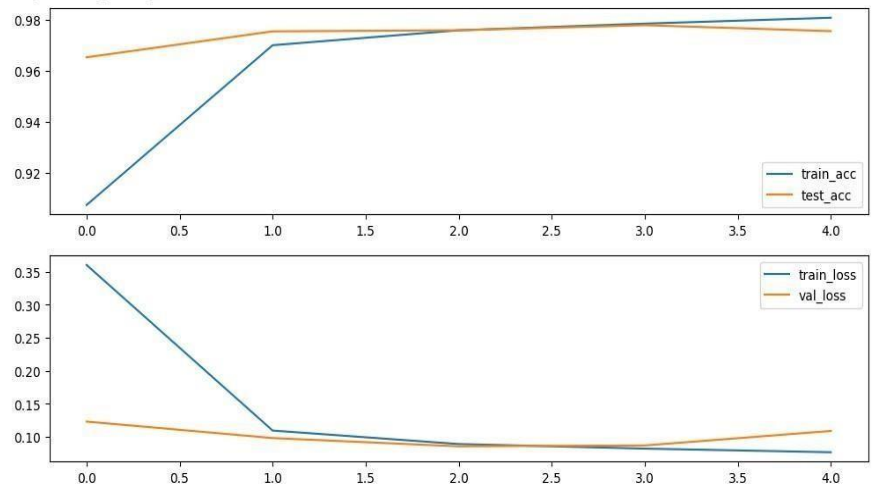


Fig 17: Accuracy and Loss function

CHAPTER 5 - CONCLUSION

5.1 Conclusion

In this work, we evaluated the fluctuation of convolutional neural organisation to avoid the file structures, elevated component extraction, and confounding organisation (mixture of classifiers) course of action of the conventional confirmation template. The goal was to enhance the illustration of transcribed digit confirmation. The current paper proposes the work of several hyper-boundaries through a thorough study utilising the MNIST dataset. Furthermore, we proved that fine-tuning the hyper-boundary is necessary to enhance CNN engineering presentation. We got a validation rate of 99.89% using the Adam analysis for the MNIST collection, which is higher than any earlier published findings.

Through studies, it is clearly demonstrated how increasing the number of convolution operation in the CNN architecture affects how transcribed digit confirmations are given. The current research has the unique advantage of concurrently examining all CNN technology frontiers that offer the highest confirming accuracy for such MNIST dataset. Using a pure CNN model, the joint researchers are still unable to equal this accuracy. With almost equal accuracy to that attained in this study, some analysts have employed the pooled of CNN models for a comparable dataset to increase their accuracy results at the expense of increased computing cost and the high complexity of the testing.

Future research should focus on CNN crossovers, such as CNN-RNN and news models, and unambiguous space recognition frameworks. It is possible to investigate developmental computations, namely the quantity of layers, the rate of learning, and the convolutional channel portion sizes, to improve the learning bounds of CNNs.

5.2 Future Scope

Based on the data the driver's mindset can supply to the system, the on-board technology of the automobile uses facial expression recognition to ensure both the driver's and the customer's safety. The creation of a system for recognising facial expressions using computer vision, along with improvements to its sophisticated extraction and classification capabilities. This technology may be applied to digital safety systems that can recognise a person in any manner of expression.

Applications built using machine and deep learning algorithms have practically infinite potential for future growth. In order to solve numerous difficulties, we may work on a hybrid or denser algorithm in the future that uses more diverse data than the existing collection of methods. Future uses for these algorithms range from the general public to high-level authorities. By differentiating the algorithms described above and continuing to improve them, we may create very useful apps that can be utilised by both secret government agencies and the general public. These algorithms may be used in hospital software for thorough medical diagnosis, therapy, and patient monitoring.

They can also be used in surveillance systems to detect suspicious behaviour, fingerprinting and retina registry filtering programmes, and apps for checking Through the employment of these algorithms in common and sophisticated applications, advancements in this field can assist us in establishing a climate of safety, awareness, and ease (i.e., Or government level). The future of technology will be applications ai - based and deep learning because of this absolute correctness and benefits over several significant issues.

References

1. Avanaei and a. s. maida, "multi-layer unsupervised learning in a spiking convolutional neural network," in 2017 international joint conference on neural networks (ijcnn), 2017, pp. 2023-2030: ieee.
2. Kale, Karbhari V., Prapti D. Deshmukh, Shriniwas V. Chavan, Majharoddin M. Kazi, and Yogesh S. Rode. "Zernike moment feature extraction for handwritten Devanagari compound character recognition." In 2013 Science and Information Conference, pp. 459-466. IEEE, 2013.’
3. R. Sethi and I. Kaushik, "Hand Written Digit Recognition using Machine Learning," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 2020, pp. 49-54, doi: 10.1109/CSNT48778.2020.9115746
4. LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". Proceedings of the IEEE, 86(11): 2278–2324.
5. M. Hanmandlu,O.V. Ramana Murthy, Vamsi Krishna Madasu, “Fuzzy Model based recognition of Handwritten Hindi characters”, IEEE Computer society, Digital Image Computing Techniques and Applications , 2007
6. M. Jain, G. Kaur, M. P. Quamar and H. Gupta, "Handwritten Digit Recognition Using CNN," 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM), Noida, India, 2021, pp. 211-215, doi: 10.1109/ICIPTM52218.2021.9388351.
7. Shrivastava, I. Jaggi, S. Gupta and D. Gupta, "Handwritten Digit Recognition Using Machine Learning: A Review," 2019 2nd International Conference on Power Energy, Environment and Intelligent Control (PEEIC), Greater Noida, India, 2019, pp. 322-326, doi: 10.1109/PEEIC47157.2019.8976601.
8. M. Jain, G. Kaur, M. P. Quamar and H. Gupta, "Handwritten Digit Recognition Using CNN," 2021 International Conference on Innovative Practices in Technology and Management (ICIPTM), Noida, India, 2021, pp. 211-215, doi: 10.1109/ICIPTM52218.2021.9388351.
9. Akhlaghi, M., Ghods, V. Farsi handwritten phone number recognition using deep learning. *SN Appl. Sci.* **2**, 408 (2020). https://doi.org/10.1007/s42452-020-2222-5.

10. “Ayush Purohit et al “A Literature Survey on Handwritten Character Recognition” International Journal of” “Computer Science and Information Technologies, Vol. 7) , 2016.”

Appendices

Data Cleaning

Data cleaning or data cleansing is the process of identifying and correcting (or removing) corrupted or inaccurate records from a record set, table or database and refers to the identification of incomplete, incorrect, inaccurate or irrelevant portions of data and the subsequent replacement, modification or deletion of impure or raw data.

Data cleaning can be done interactively using data wrangling tools or as a batch process using scripting or a data quality firewall. The set of data should be compatible with other sets that are comparable in the system after cleaning. User input mistakes, transmission or storage damage, or disparate database structure definitions of comparable entities in separate repositories might all have contributed to the inconsistencies that were later discovered or fixed.

Data validation is distinct from data cleaning in that validation nearly always entails rejecting data upon entry and is carried out at the moment of input instead of on batches of data.

Typographical mistakes may be removed, or values may be validated and corrected in accordance with a 's specific of entities, as part of the real data cleansing ritual. Any address without a valid postal code would be rejected as invalid. Alternatively, validation might use fuzzy or approximative string match.

Some data cleansing programming purge the data by cross-referencing it with a collection of approved data. Data augmentation, which involves making data more comprehensive by adding relevant information, is a frequent technique in data cleansing. For instance, relating phone numbers to addresses that correspond to those numbers. Data normalization, which is the act of merging data from "various file formats, file names, and columns," and translating it into one, may also be included in data cleaning.

After cleaning, the data set should be consistent with other similar data sets in the system. Inconsistencies found or resolved may have originally been caused by user input errors, transmission or storage corruption, or different data dictionary definitions of similar entitiesin different repositories.

A machine learning dataset is a collection of data that is used to train a model. The data set acts as an example that teaches the machine learning algorithm how to make predictions. Common data types include:

- Text data
- Image data
- Audio data
- Video data
- Numerical data

Purpose of AI/ML Datasets:

One of the crucial steps in training an AI/ML model is the preparation and selection of an appropriate dataset. This step can significantly impact the outcome of an AI/ML development project and determine whether it will be successful or not.

Key Purpose Of an AI/ML Datasets:

To train the model
To measure the accuracy of the model once it is trained

What are the types of ML Datasets.?

The entire collected data set is divided into 3 subsets, which are as follows:

1. Training dataset

The training set is a crucial subset of the entire dataset, representing around 60% of the total data. Its purpose is to provide the initial data used to train the model. This means that it helps to teach the algorithm what features to identify in the data. For instance, an AI system designed to recognize vehicle license plates will be trained on image data that includes location labels (such as front or back of the car) and the format of the license plates.

2. Validation data file

The validation subset is a portion of the total dataset, typically 20%, that is used to assess the performance of a trained model. It allows for the evaluation of model parameters and can reveal potential issues or shortcomings in the model. Additionally, the validation subset can help determine whether the model is overfitting or underfitting the training data.

3. Test data file

The third subset of data, which represents the final 20% of the total dataset, is used as input in the last stage of the training process. This subset contains unknown data that has not been seen by the model before and is used to assess the model's accuracy. Essentially, it evaluates how well the model has learned from the training and validation subsets.

Convolutional Neural Network

Fully Convolutional Networks (FCN) are a type of Artificial Neural Network (ANN) that are commonly used for analyzing visual images. These networks are designed to be Shift Invariant and area Invariant, meaning that they can respond to translated features in the same way, making them useful for image analysis. However, they are not entirely Translation Invariant since they follow a down sampling process during training. CNNs can be applied in many areas, including image segmentation, video and image popularity, natural language processing, and economic time collecting.

CNNs are a modified version of multilayer perceptron, which refers to networks where each neuron in a layer is connected to all neurons in the next layer. This full connectivity makes these networks susceptible to overfitting. To prevent this, CNNs use regularization techniques such as weight decay, trimming connectivity, and dropout. CNNs also utilize the hierarchical patterns in data to collect patterns of increasing complexity using smaller and simpler models, making them less complex than other types of neural networks.

CNNs are inspired by the animal visual cortex, where individual neurons only respond to stimuli in a limited area of the field of view called the receptive field. The receptive fields of different neurons overlap, covering the entire field of vision. Unlike traditional image algorithms that use hand-engineered filters, CNNs optimize the filters through computerized learning, which makes them independent of prior knowledge and human intervention in feature extraction. This is a significant advantage of CNNs, as they require little pre-processing compared to other image analysis algorithms.

Number Recognition

ORIGINALITY REPORT

17%	12%	9%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.researchgate.net Internet Source	3%
2	ijircce.com Internet Source	2%
3	Submitted to Wright State University Student Paper	1%
4	Submitted to Jaypee University of Information Technology Student Paper	1%
5	Charu C. Aggarwal. "Neural Networks and Deep Learning", Springer Science and Business Media LLC, 2018 Publication	1%
6	corporatefinanceinstitute.com Internet Source	1%
7	dokumen.pub Internet Source	1%
8	Submitted to Kingston University Student Paper	<1%