ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM
AFFILIATED TO MAHATMA GANDHI UNIVERSITY

PROJECT REPORT ON

MACHINE LEARNING – ENABLED HANDWRITTEN DIGIT CLASSIFICATION USING CNN

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF SCIENCE IN
COMPUTER APPLICATIONS [TRIPLE MAIN]

Submitted By
ALITTA JOSE
III BSc Computer Applications [Triple Main]
Register No: SB21CA005

Under the Guidance of
Dr. Dhanya R

DEPARTMENT OF COMPUTER APPLICATIONS
2021- 2024

ST. TERESA'S COLLEGE (AUTONOMOUS), ERNAKULAM

AFFILIATED TO MAHATMA GANDHI UNIVERSITY

THE FEAR OF THE LORD IS THE BEGINNING OF WISDOM

ST. TERESA'S COLLEGE (AUTONOMOUS) ESTD 1925

CERTIFICATE

This is to certify that the project entitled "**MACHINE LEARNING – ENABLED HANDWRITTEN DIGIT CLASSIFICATION USING CNN**", is a bona-fide record of the work done by **ALITTA JOSE** [Reg. No. SB21CA005] during the year 2023-24 and submitted in partial fulfillment of the requirements of the degree of Bachelor of Science in Computer Applications (Triple Main) under Mahatma Gandhi University.

ST. TERESA'S COLLEGE DEPT. OF COMPUTER ERNAKULAM

[illegible] 21/3/24

Head of the Department

[illegible]

Internal Examiner

[illegible] 22.03.24

External Examiner

Date: 21/03/2024

DECLARATION

I, **Alitta Jose, BSc Computer Application [Triple Main]** final year student of St. Teresa's College (Autonomous), Ernakulam, Register no: **SB21CA005**, hereby declare that the dissertation submitted for Bachelor's Degree in Computer Application is my original work. I further declare that the said work has not previously been submitted to any other university or academic body.

Date: - 21-03-2024

Place: - Ernakulam

[illegible]

ALITTA JOSE

ACKNOWLEDGMENT

I extend my heartfelt gratitude to the Almighty for His blessings. I would like to take this moment to convey my appreciation to everyone who contributed to the successful completion of this project. I am deeply thankful to **Rev. Dr. Sr. Vinitha CSST**, our Superior and Manager, **Rev. Sr. Emeline CSST**, our Director, and **Dr. Alphonsa Vijaya Joseph**, our Principal, for providing us with all the necessary facilities.

I express my sincere thanks to **Ms. Remya C J**, the Head of the department, for her unwavering support. Special thanks are due to my guide,
Dr. Dhanya R, for her invaluable guidance and assistance throughout the project.

I am grateful to all the faculty members of the department for their exceptional guidance and encouragement. I am deeply thankful to **Mr. Jai Kiran** for his invaluable assistance and steadfast support throughout the project. Furthermore, I extend my sincerest appreciation to my friends and relatives for their unwavering encouragement. Lastly, I want to express my heartfelt thanks to my parents and siblings for their encouragement and for creating opportunities for me to complete this project.

Alitta Jose

ABSTRACT

This project explores the realm of handwritten digit classification using Convolutional Neural Networks (CNNs) and aims to address the challenge of accurately identifying handwritten digits. In today's digitized world, the accurate recognition of handwritten digits is crucial for various applications, including postal services, finance, and document processing. However, conventional methods often face inefficiencies and inaccuracies, necessitating the development of more robust classification systems.

This study specifically focuses on comparing the performance of CNNs and Recurrent Neural Networks (RNNs) in accurately classifying handwritten digits. Through rigorous experimentation with different classification models, it becomes evident that CNN classifiers outperform RNNs in terms of both speed and accuracy.

Utilizing the MNIST database, widely recognized as the benchmark dataset for handwritten digit classification, the effectiveness of CNNs and RNNs is comprehensively evaluated.

The classification task follows a systematic approach involving data preprocessing, model design, training, and evaluation. Within the CNN architecture, convolutional layers are utilized for extracting features, pooling layers for reducing dimensionality, and fully connected layers for classification. The integration of batch normalization and activation functions enhances learning and feature representation.

Comparative results reveal the superior performance of CNNs over RNNs in accurately classifying handwritten digits. CNN classifiers demonstrate both higher accuracy rates and faster processing speeds, emphasizing their effectiveness in digit recognition tasks.

In conclusion, this project contributes to the advancement of handwritten digit recognition by offering valuable insights into the performance comparison between CNNs and RNNs. The findings emphasize the importance of adopting CNN architectures to achieve better accuracy and efficiency in handwritten digit classification tasks.

CONTENTS

1. INTRODUCTION……………………………………………………1
   1.1 About Project……………………………………………………..1
   1.2 Objective of the Project……………………………………………..1
2. LITERATURE SURVEY……………………………………………2
3. PROPOSED SYSTEM…………………………………………………3
   3.1 Flowchart and Diagrams……………………………………………3
   3.2 Methodology…………………………………………………...3
   3.3 System Analysis……………………………………………......7
   3.4 System Specifications……………………………………….....8
   3.5 Source code…………………………………………………………11
4. ANALYSIS……………………………………………………......14
   4.1 Comparison with RNN……………………………………………14
5. RESULTS………………………………………………………………15
6. CONCLUSION…………………………………………………………18
7. REFERENCES…………………………………………………………19

1.INTRODUCTION

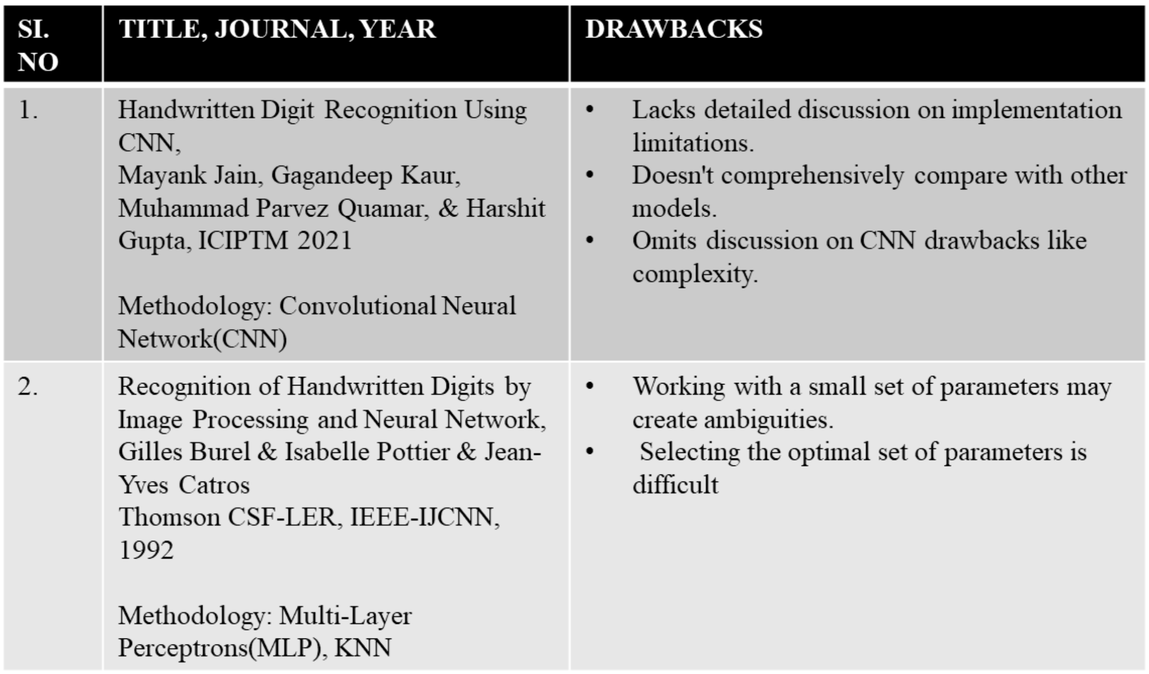
1.1 ABOUT PROJECT

Handwritten digit recognition remains a significant challenge in computer vision and machine learning. Despite advancements in preprocessing and classification algorithms, accurately identifying handwritten numerals is difficult due to variations in writing styles and image characteristics. This project seeks to address this challenge by developing a system capable of accurately recognizing handwritten digits (ranging from 0 to 9) using modern machine learning techniques. Utilizing the TensorFlow framework, Python programming language, and relevant libraries, we train a model on the widely-used MNIST dataset, which contains thousands of handwritten digit images. Our goal is to develop a robust recognition algorithm that accurately identifies handwritten digits input by users and displays results with corresponding accuracy rates. Through this project, we aim to demonstrate the effectiveness of machine learning in solving real-world recognition tasks in handwritten digit recognition.

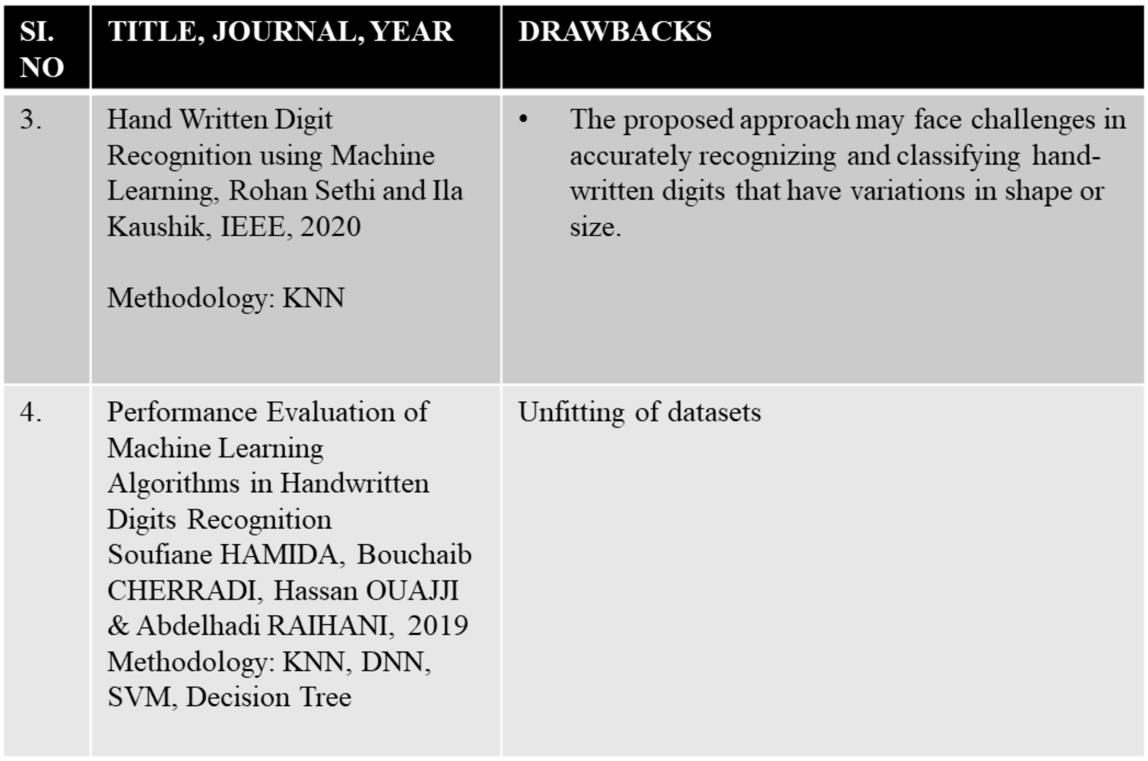
1.2 OBJECTIVE OF PROJECT

The primary objective of this project is to develop a robust system for recognizing handwritten digits using machine learning techniques. This involves implementing a recognition algorithm capable of accurately identifying digits from 0 to 9. I will utilize TensorFlow and Python, alongside relevant libraries, to train the model on the MNIST dataset. Additionally, I aim to create a user-friendly interface for inputting handwritten digits. The performance of the recognition system will be evaluated based on accuracy and computational efficiency. Ultimately, the goal is to provide a practical tool for digit recognition applications, demonstrating the effectiveness of machine learning in real-world scenarios.

2.LITERATURE SURVEY



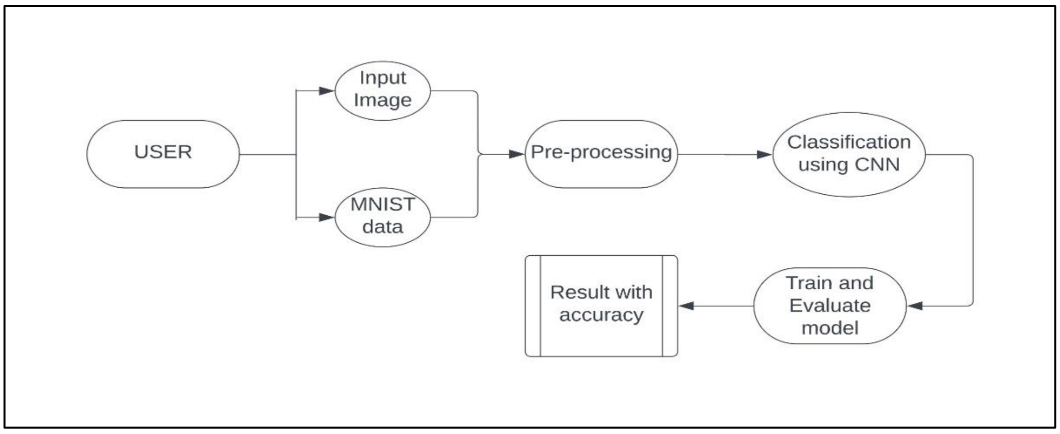
SI. NO	TITLE, JOURNAL, YEAR	DRAWBACKS
1.	Handwritten Digit Recognition Using CNN, Mayank Jain, Gagandeep Kaur, Muhammad Parvez Quamar, & Harshit Gupta, ICIPTM 2021 Methodology: Convolutional Neural Network(CNN)	• Lacks detailed discussion on implementation limitations. • Doesn't comprehensively compare with other models. • Omits discussion on CNN drawbacks like complexity.
2.	Recognition of Handwritten Digits by Image Processing and Neural Network, Gilles Burel & Isabelle Pottier & Jean-Yves Catros Thomson CSF-LER, IEEE-IJCNN, 1992 Methodology: Multi-Layer Perceptrons(MLP), KNN	• Working with a small set of parameters may create ambiguities. • Selecting the optimal set of parameters is difficult



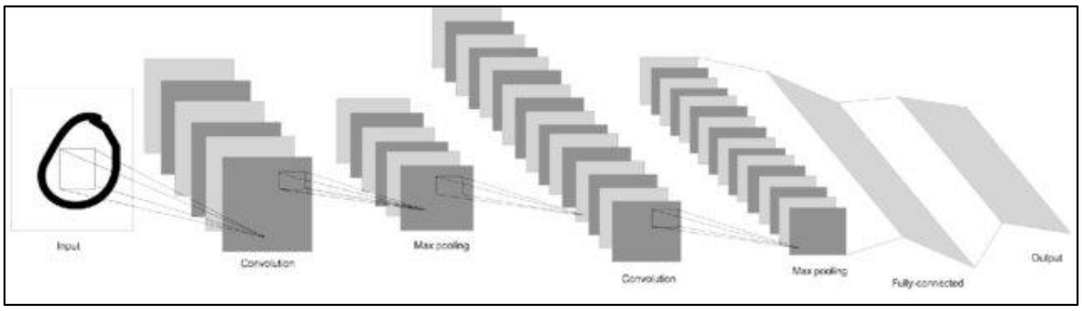
SI. NO	TITLE, JOURNAL, YEAR	DRAWBACKS
3.	Hand Written Digit Recognition using Machine Learning, Rohan Sethi and Ila Kaushik, IEEE, 2020 Methodology: KNN	• The proposed approach may face challenges in accurately recognizing and classifying hand-written digits that have variations in shape or size.
4.	Performance Evaluation of Machine Learning Algorithms in Handwritten Digits Recognition Soufiane HAMIDA, Bouchaib CHERRADI, Hassan OUAJJI & Abdelhadi RAIHANI, 2019 Methodology: KNN, DNN, SVM, Decision Tree	Unfitting of datasets

3.PROPOSED SYSTEM

3.1 FLOWCHART AND DIAGRAMS



Data Flow Diagram of the Proposed System

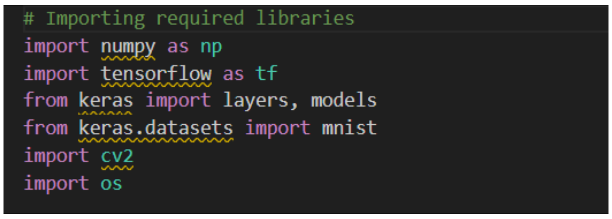


An example of a simple CNN architecture for classifying handwritten digits

3.2 METHODOLOGY

- **Install Necessary Libraries**

  Run the command python -m pip install numpy tensorflow keras opencv-python to install the required libraries.



```
# Importing required libraries
import numpy as np
import tensorflow as tf
from keras import layers, models
from keras.datasets import mnist
import cv2
import os
```

- **Data Preparation and Preprocessing**

  The project began by selecting the MNIST (Modified National Institute of Standards and Technology) dataset, a widely used benchmark dataset for handwritten digit recognition. MNIST consists of 70,000 grayscale images of handwritten digits (0-9), with each image sized at 28x28 pixels. Grayscale images were preferred over color images to simplify processing and reduce computational complexity. The dataset was divided into a training set comprising 60,000 images and a test set containing 10,000 images. Preprocessing involved standardizing the image size to 28x28 pixels and normalizing pixel values to a range between 0 and 1 to ensure consistency and facilitate efficient model training.

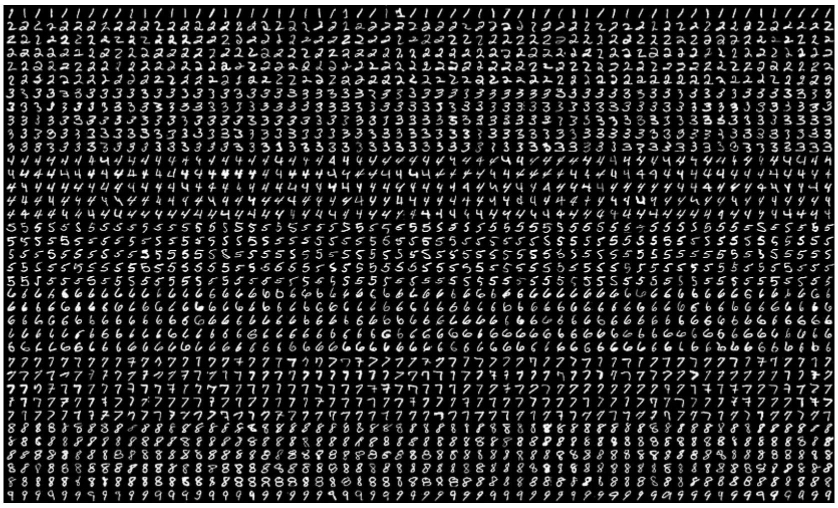
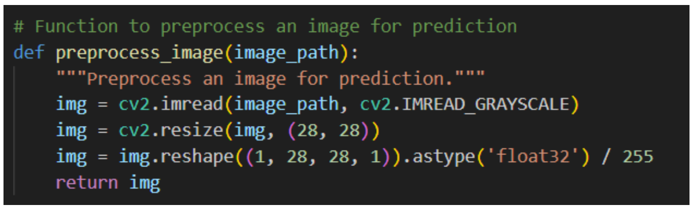
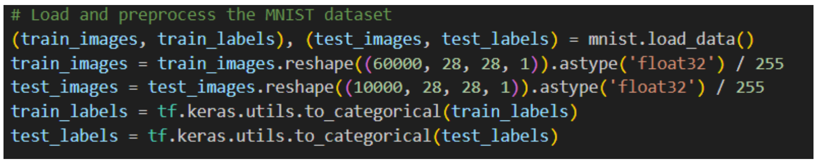


Fig 1: MNIST Dataset



```
# Function to preprocess an image for prediction
def preprocess_image(image_path):
    """Preprocess an image for prediction."""
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (28, 28))
    img = img.reshape((1, 28, 28, 1)).astype('float32') / 255
    return img
```



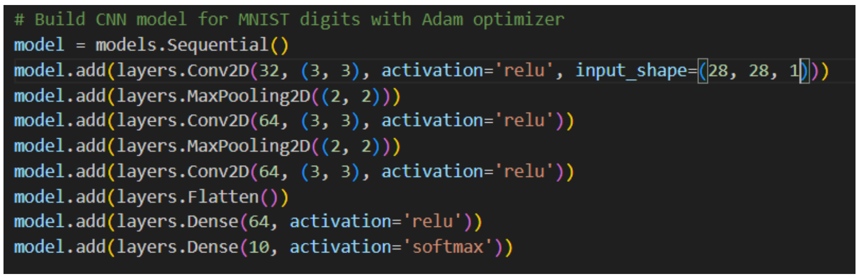
```
# Load and preprocess the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32') / 255
train_labels = tf.keras.utils.to_categorical(train_labels)
test_labels = tf.keras.utils.to_categorical(test_labels)
```

- **Building the CNN Model**

The methodology revolves around the construction and training of a Convolutional Neural Network (CNN) model tailored for digit classification, which forms the cornerstone of the research. CNNs are renowned for their proficiency in image recognition tasks, owing to their capability to autonomously learn hierarchical features from raw image data. Leveraging the Keras Sequential API, the CNN model is meticulously built as a linear stack of layers, ensuring a streamlined architecture conducive to effective digit classification.

The CNN model architecture comprises essential components meticulously designed to extract and process features from input images. Convolutional layers serve as the primary feature detectors, equipped with 32 filters of size (3, 3) to detect intricate patterns within the images. The Rectified Linear Unit (ReLU) activation function is applied to introduce non-linearity, enabling the model to learn complex features efficiently. Subsequent MaxPooling layers are strategically placed to down-sample spatial dimensions, aiding computational efficiency and mitigating overfitting by retaining essential features. Following this, a Flatten layer transforms the output into a one-dimensional array, facilitating seamless processing by subsequent fully connected layers.

Incorporating two dense (fully connected) layers, the model advances to higher-level reasoning and decision-making, culminating in a final dense layer with 10 nodes, each corresponding to one of the ten possible digit classes (0-9). The softmax activation function is applied to the final layer, enabling multi-class classification by outputting probabilities for each class.

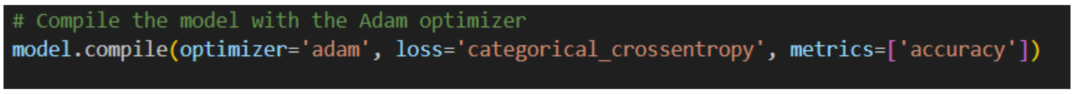


```python
# Build CNN model for MNIST digits with Adam optimizer
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

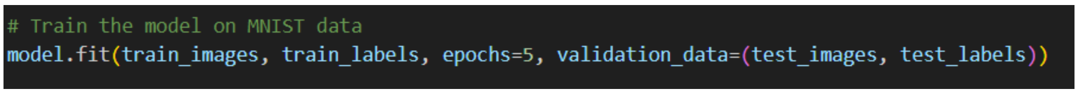
• **Compiling the Model**

With the CNN model architecture established, the next crucial step involves compiling and training the model using appropriate configurations. The training process spans multiple epochs, with the model updating its parameters after processing batches of training data.

Additionally, to mitigate overfitting and ensure model robustness, techniques such as early stopping and model checkpoints may be implemented. By meticulously following this methodology, the model is trained on preprocessed training data, iterating over 5 epochs, allowing it to learn from the dataset multiple times to improve its accuracy and performance.



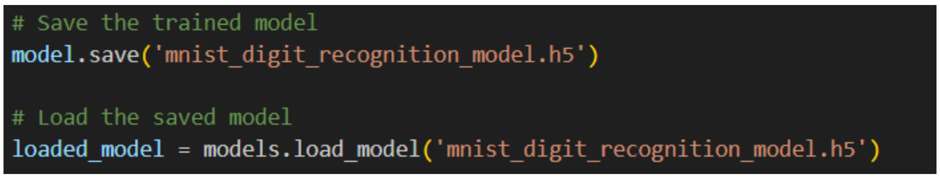
```
# Compile the model with the Adam optimizer
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```



```
# Train the model on MNIST data
model.fit(train_images, train_labels, epochs=5, validation_data=(test_images, test_labels))
```

• Validation and Testing

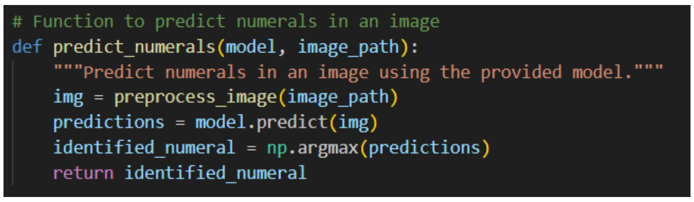
Validation methods were applied to evaluate the performance of the trained CNN model. A validation set, typically comprising 10-20% of the training data, was established to monitor the model's performance during training and prevent overfitting. Following training, the model underwent assessment using a distinct test dataset to determine its accuracy and ability to generalize to unseen data. Performance metrics including accuracy, precision, recall, and F1 score were computed to measure the model's effectiveness, with test accuracy indicating its capability to classify unseen instances accurately.



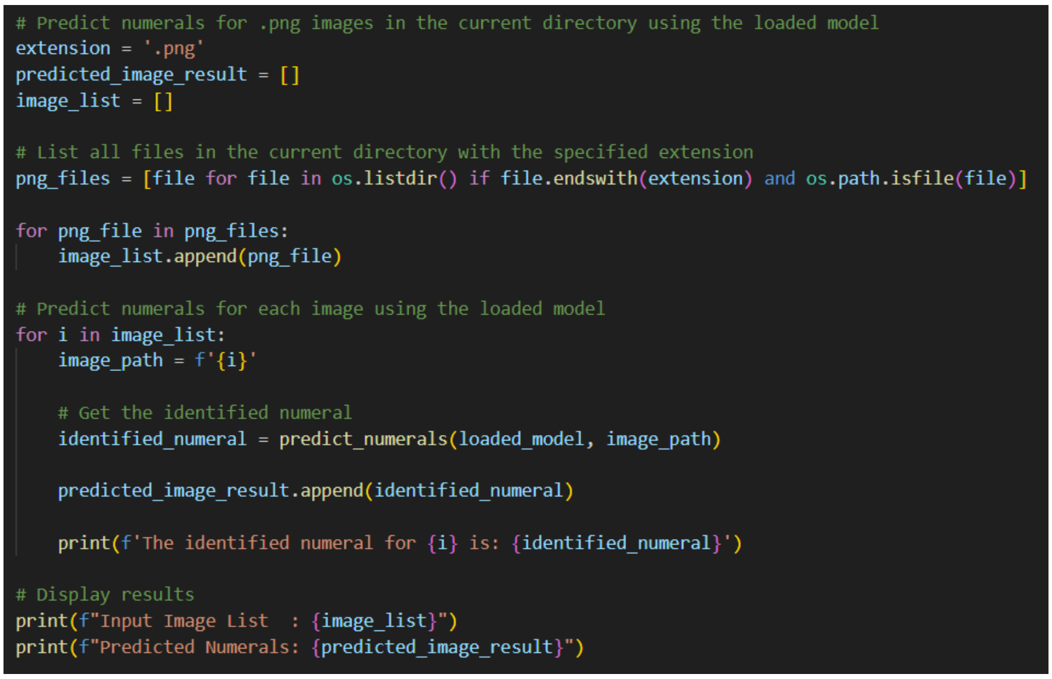
```
# Save the trained model
model.save('mnist_digit_recognition_model.h5')

# Load the saved model
loaded_model = models.load_model('mnist_digit_recognition_model.h5')
```

• Making Predictions

The trained model is utilized to make predictions on the test set, enabling analysis of the predictions to evaluate the model's performance and identify potential areas for improvement.



```python
# Function to predict numerals in an image
def predict_numerals(model, image_path):
    """Predict numerals in an image using the provided model."""
    img = preprocess_image(image_path)
    predictions = model.predict(img)
    identified_numeral = np.argmax(predictions)
    return identified_numeral
```



```python
# Predict numerals for .png images in the current directory using the loaded model
extension = '.png'
predicted_image_result = []
image_list = []

# List all files in the current directory with the specified extension
png_files = [file for file in os.listdir() if file.endswith(extension) and os.path.isfile(file)]

for png_file in png_files:
    image_list.append(png_file)

# Predict numerals for each image using the loaded model
for i in image_list:
    image_path = f'{i}'

    # Get the identified numeral
    identified_numeral = predict_numerals(loaded_model, image_path)

    predicted_image_result.append(identified_numeral)

    print(f'The identified numeral for {i} is: {identified_numeral}')

# Display results
print(f"Input Image List  : {image_list}")
print(f"Predicted Numerals: {predicted_image_result}")
```

3.3 SYSTEM ANALYSIS

The system we are talking about is good at recognizing handwritten numbers using a special kind of computer program called a Convolutional Neural Network (CNN). It is like teaching a computer to read your handwriting. The system learns from a big set of handwritten numbers called the MNIST dataset. We split this set into two parts: one to teach the computer, and the other to test how well it learned.

The system's strength comes from its smart design. It uses CNNs, which are good at finding patterns in images, like loops or straight lines in numbers. The way the CNN is set up, with different layers working together, helps it understand the numbers better. We also make sure the numbers are in a format the computer can understand easily, which helps it learn faster and do a better job.

But, like anything, there are some things to watch out for. Sometimes the computer learns a little too well and gets confused by things that are not there. It is also not great at explaining why it makes the choices it does, which can be tricky if we need to understand its decisions.

To make the system even better, we could try a few things. For example, we could give it more examples to practice with, or adjust some settings to help it learn better. It would also be helpful if we could understand more about why the computer makes the choices it does, so we could trust it more. Overall, while the system is good at recognizing handwritten numbers, there are ways we can make it even better.

3.4 SYSTEM SPECIFICATION

A Software Requirements Specification (SRS) serves as a comprehensive document outlining the expected behavior of a software system to be developed. It may encompass a series of use cases detailing how users will interact with the software. Additionally, the system specification defines the hardware and software setup required for the new system, establishing operational and performance guidelines. Non-functional requirements, such as performance engineering standards and quality criteria, are also included, imposing constraints on design and implementation. The SRS document catalogues all essential requirements essential for project development. These requirements are derived through clear and thorough understanding of the intended products, achieved through extensive communication with the project team and the customer.

- **Operating System**

  An Operating System (OS) is an interface between computer user and computer hardware. It is a software which performs all the basic tasks like file management, memory management, process management, handling input and output and controlling peripheral devices such as diskdrives and printers. The operating system required for proper execution of the system is Windows 10 or above.

- **Languages and Software Packages**

Visual Studio:

Visual Studio is an integrated development environment (IDE) developed by Microsoft. It provides a comprehensive suite of tools for software development across various platforms and programming languages. Visual Studio supports a wide range of programming languages, including C++, C#, Visual Basic, F#, and Python, among others. It offers features such as code editing, debugging, version control integration, and collaboration tools, making it a popular choice for developers working on projects of all sizes and complexities. Visual Studio also includes specialized editions tailored for different development scenarios, such as Visual Studio Community, Visual Studio Professional, and Visual Studio Enterprise.

Python:

Python is a widely-used, high-level programming language known for its simplicity and versatility. It is popular across various domains like web development, data science, AI, and scientific computing. Python's straightforward syntax and focus on readability make it accessible to all levels of developers. Its extensive standard library offers numerous modules and functions for diverse tasks, aided by dynamic typing and automatic memory management. Python's popularity is also due to its vibrant community and vast ecosystem of third-party libraries, facilitating effective collaboration and solution leverage. Overall, Python's simplicity, versatility, and extensive ecosystem make it a top choice for developers seeking a powerful and efficient programming language.

HTML:

HTML, or Hypertext Markup Language, is the standard language for creating web pages. It defines elements like headings, paragraphs, images, links, and forms using tags enclosed in angle brackets. Tags indicate how content should display, with opening and closing tags enclosing content. Attributes can provide extra information or functionality. HTML documents combine text content and tags to create webpage layouts. Browsers interpret HTML tags to display content, enabling user interaction with webpage elements.

- **Hardware and Software Specifications**

  - Software Requirements:

    Operating System: Windows 10 or above

    Front End: Visual Studio

    Back End: HTML

    Developing tool: Python

  - Hardware Requirements:

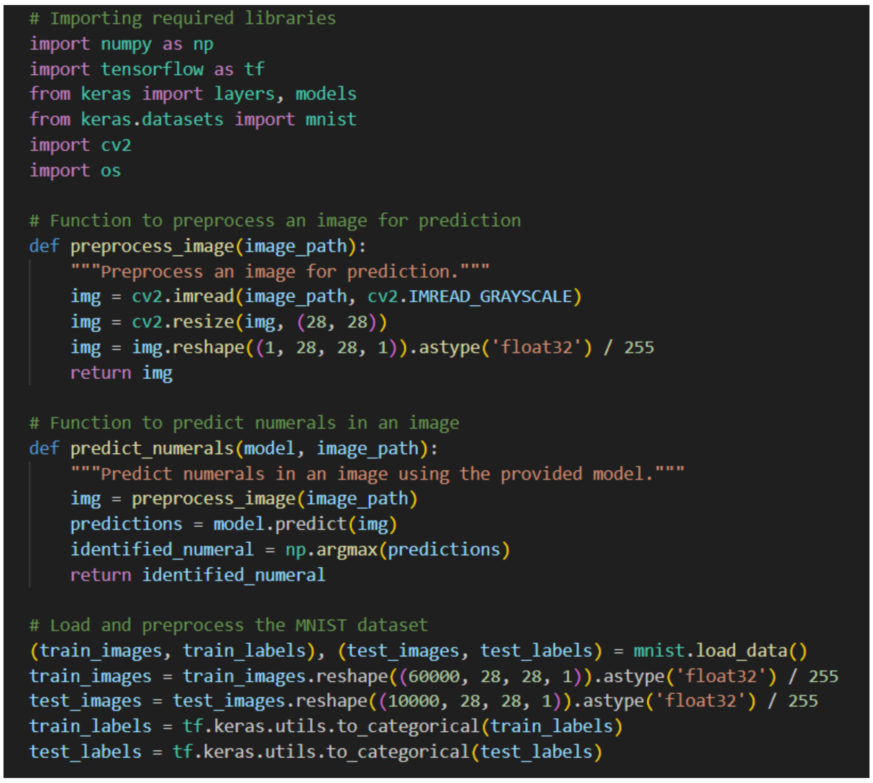
    Processor: Intel(R) Core (TM) i3-1005G1 CPU @ 1.20GHz 1.20 GHz

    RAM: 8 GB

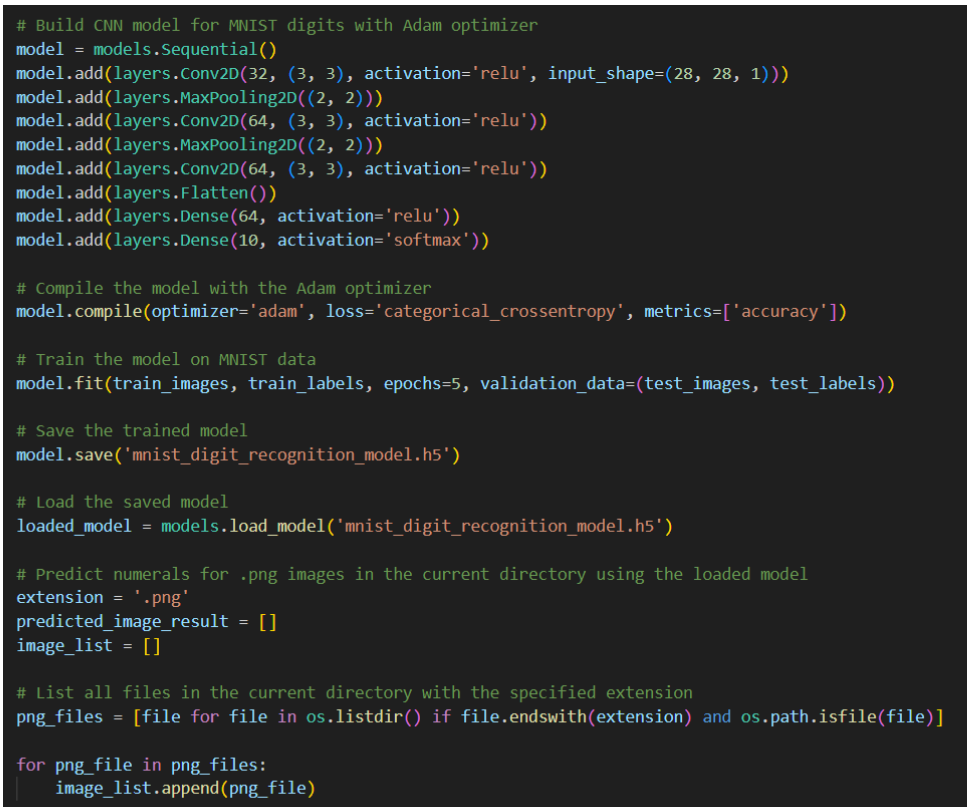
    System type: 64-bit operating system, x64-based processor

3.5 SOURCE CODE

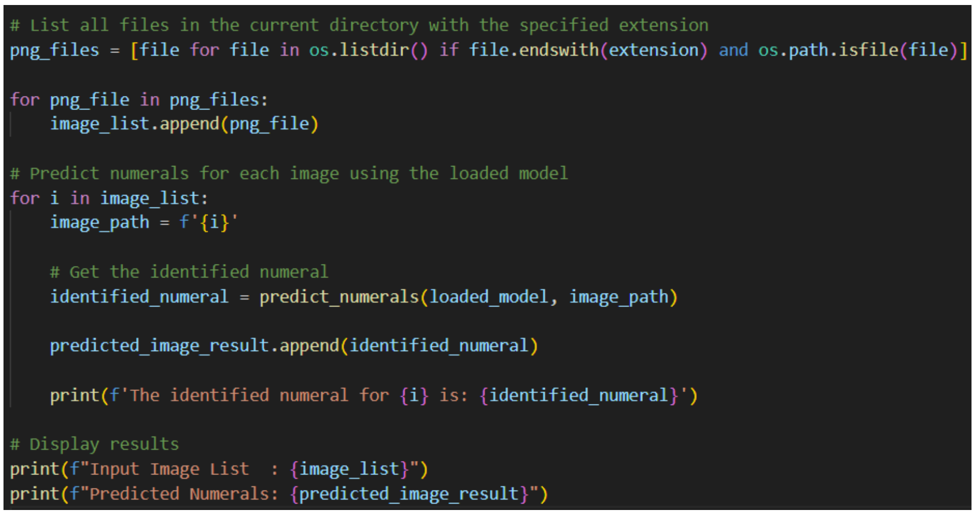
CONVOLUTIONAL NEURAL NETWORK:



```python
# Importing required libraries
import numpy as np
import tensorflow as tf
from keras import layers, models
from keras.datasets import mnist
import cv2
import os

# Function to preprocess an image for prediction
def preprocess_image(image_path):
    """Preprocess an image for prediction."""
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (28, 28))
    img = img.reshape((1, 28, 28, 1)).astype('float32') / 255
    return img

# Function to predict numerals in an image
def predict_numerals(model, image_path):
    """Predict numerals in an image using the provided model."""
    img = preprocess_image(image_path)
    predictions = model.predict(img)
    identified_numeral = np.argmax(predictions)
    return identified_numeral

# Load and preprocess the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float32') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32') / 255
train_labels = tf.keras.utils.to_categorical(train_labels)
test_labels = tf.keras.utils.to_categorical(test_labels)
```

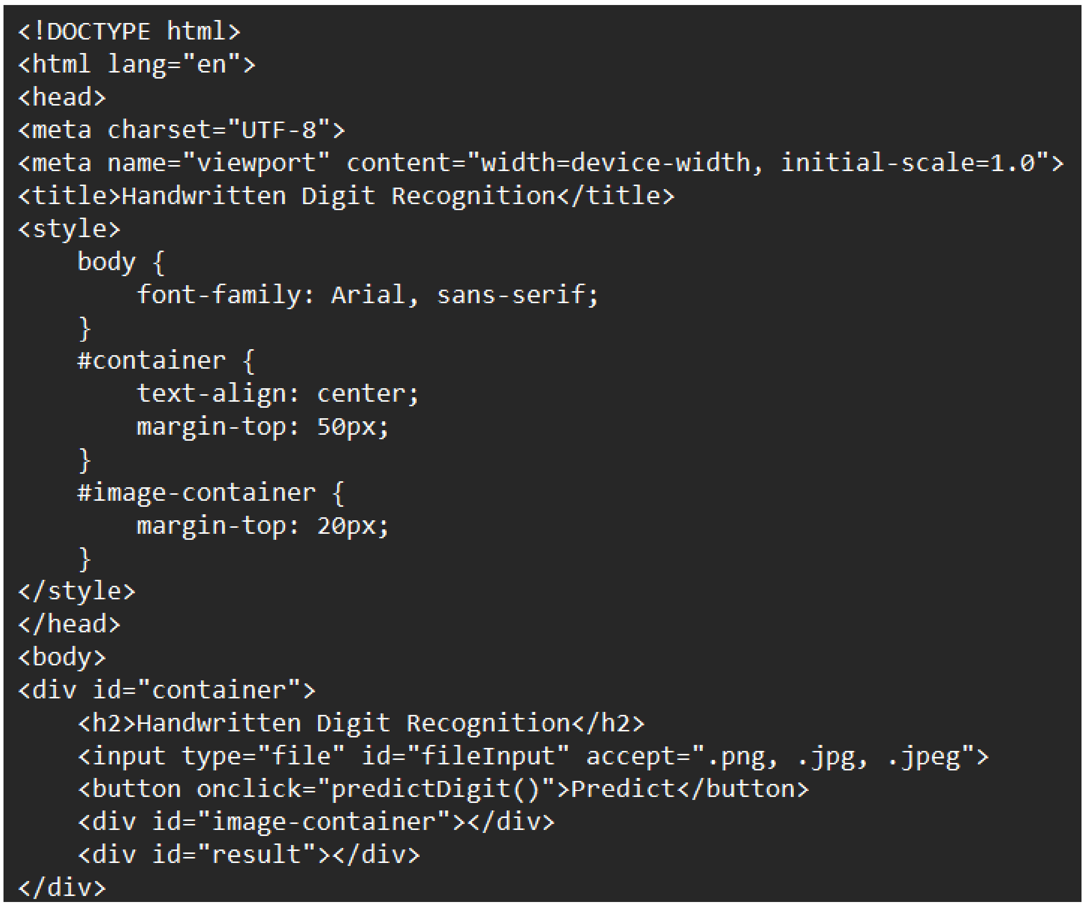


```python
# Build CNN model for MNIST digits with Adam optimizer
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

# Compile the model with the Adam optimizer
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model on MNIST data
model.fit(train_images, train_labels, epochs=5, validation_data=(test_images, test_labels))

# Save the trained model
model.save('mnist_digit_recognition_model.h5')

# Load the saved model
loaded_model = models.load_model('mnist_digit_recognition_model.h5')

# Predict numerals for .png images in the current directory using the loaded model
extension = '.png'
predicted_image_result = []
image_list = []

# List all files in the current directory with the specified extension
png_files = [file for file in os.listdir() if file.endswith(extension) and os.path.isfile(file)]

for png_file in png_files:
    image_list.append(png_file)
```

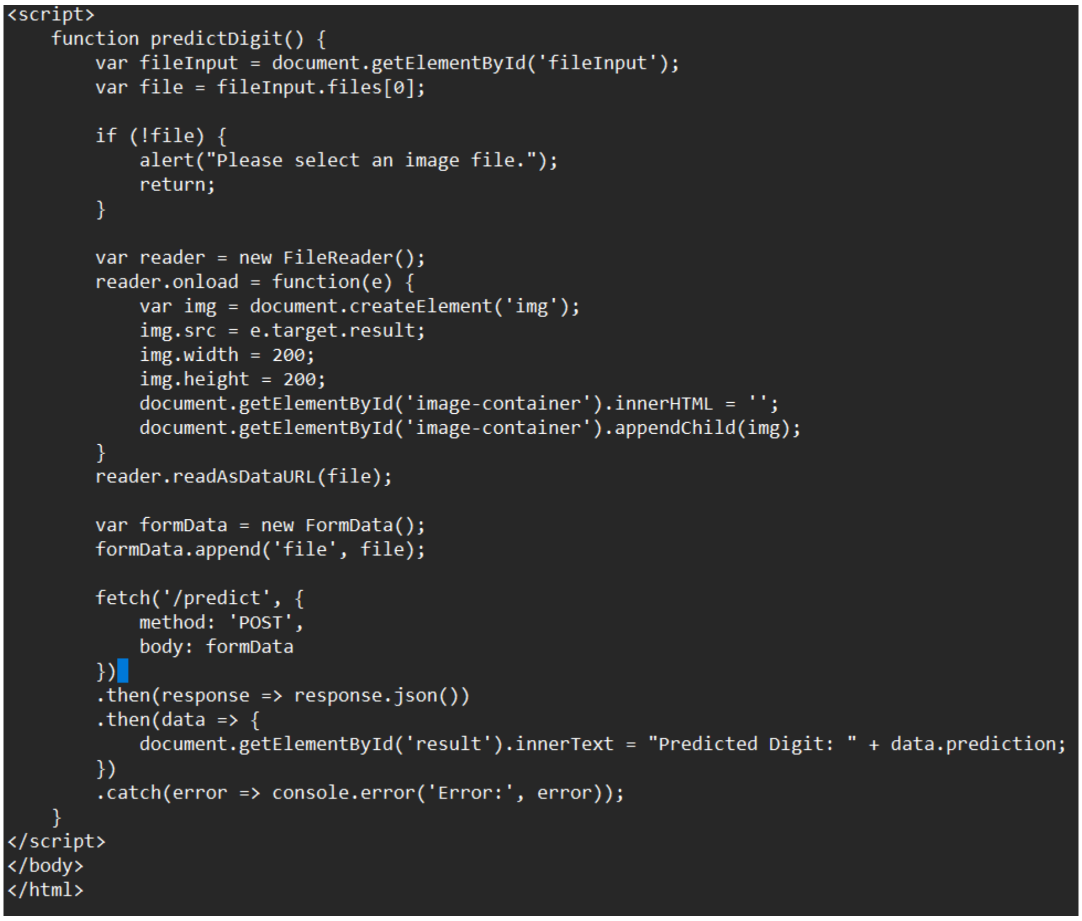


```python
# List all files in the current directory with the specified extension
png_files = [file for file in os.listdir() if file.endswith(extension) and os.path.isfile(file)]

for png_file in png_files:
    image_list.append(png_file)

# Predict numerals for each image using the loaded model
for i in image_list:
    image_path = f'{i}'

    # Get the identified numeral
    identified_numeral = predict_numerals(loaded_model, image_path)

    predicted_image_result.append(identified_numeral)

    print(f'The identified numeral for {i} is: {identified_numeral}')

# Display results
print(f"Input Image List   : {image_list}")
print(f"Predicted Numerals: {predicted_image_result}")
```

GUI:



```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Handwritten Digit Recognition</title>
<style>
    body {
        font-family: Arial, sans-serif;
    }
    #container {
        text-align: center;
        margin-top: 50px;
    }
    #image-container {
        margin-top: 20px;
    }
</style>
</head>
<body>
<div id="container">
    <h2>Handwritten Digit Recognition</h2>
    <input type="file" id="fileInput" accept=".png, .jpg, .jpeg">
    <button onclick="predictDigit()">Predict</button>
    <div id="image-container"></div>
    <div id="result"></div>
</div>
```

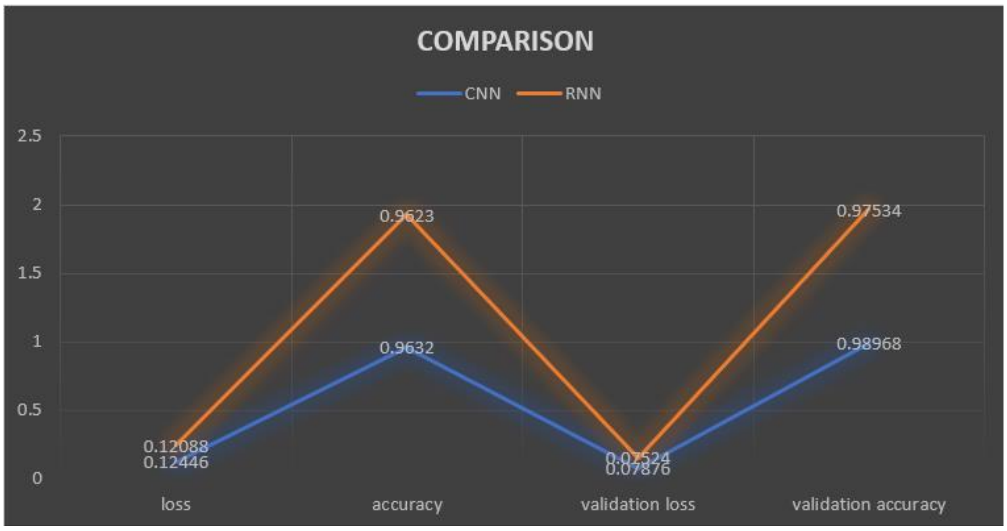


```
<script>
    function predictDigit() {
        var fileInput = document.getElementById('fileInput');
        var file = fileInput.files[0];

        if (!file) {
            alert("Please select an image file.");
            return;
        }

        var reader = new FileReader();
        reader.onload = function(e) {
            var img = document.createElement('img');
            img.src = e.target.result;
            img.width = 200;
            img.height = 200;
            document.getElementById('image-container').innerHTML = '';
            document.getElementById('image-container').appendChild(img);
        }
        reader.readAsDataURL(file);

        var formData = new FormData();
        formData.append('file', file);

        fetch('/predict', {
            method: 'POST',
            body: formData
        })
        .then(response => response.json())
        .then(data => {
            document.getElementById('result').innerText = "Predicted Digit: " + data.prediction;
        })
        .catch(error => console.error('Error:', error));
    }
</script>
</body>
</html>
```

4.ANALYSIS

4.1COMPARISON WITH RNN

Convolutional Neural Networks (CNNs) are widely acknowledged for their superior accuracy in image recognition tasks compared to Recurrent Neural Networks (RNNs). The inherent architecture of CNNs is specifically designed to excel in capturing spatial patterns and local dependencies within images, making them exceptionally effective for tasks like object recognition and image classification. This is achieved through the utilization of convolutional layers that apply filters to extract features from different parts of the input image. Additionally, max-pooling layers are employed to down-sample the feature maps, allowing the network to focus on the most relevant information while reducing computational complexity. On the contrary, although RNNs are proficient in processing sequential data and capturing temporal dependencies over time, they may not achieve the same level of accuracy as CNNs in image-related tasks. This is primarily due to the fact that RNNs lack the specialized structure and feature extraction capabilities inherent in CNN architectures. While RNNs excel in tasks requiring memory of past inputs and sequential context, such as natural language processing and speech recognition, CNNs remain the preferred choice for image recognition tasks owing to their ability to effectively capture spatial features and patterns.

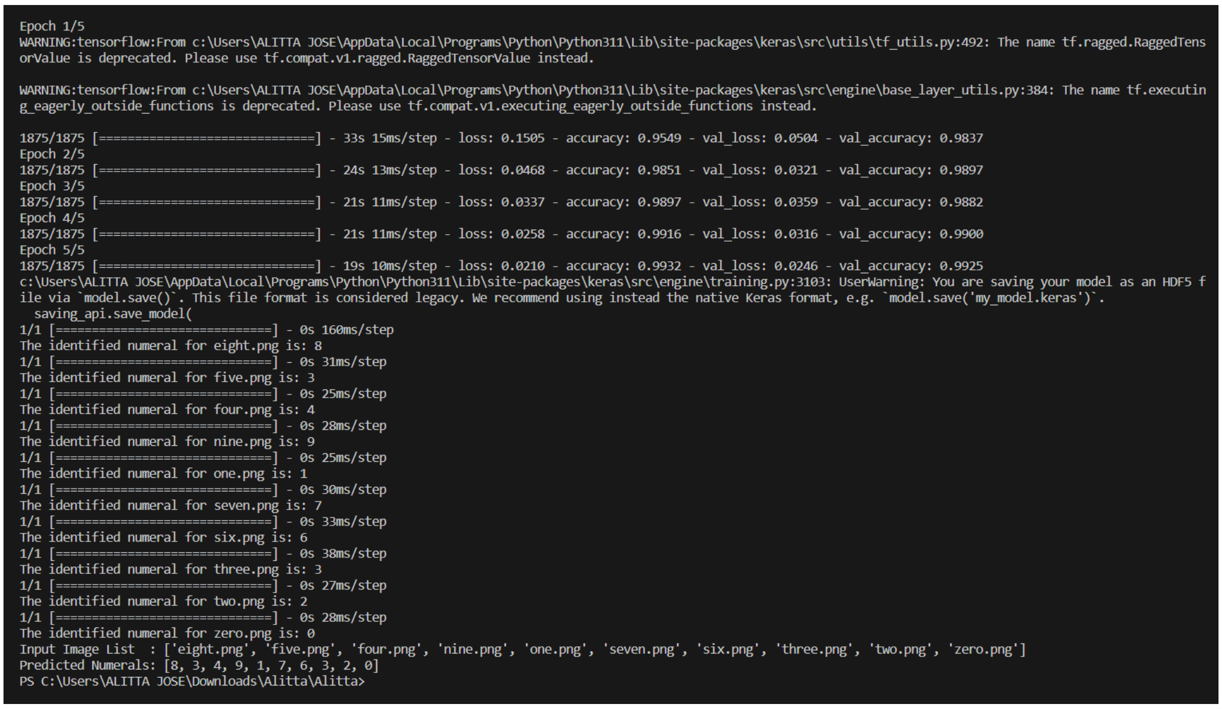


Graph to show the comparison of CNN and RNN

5.RESULTS

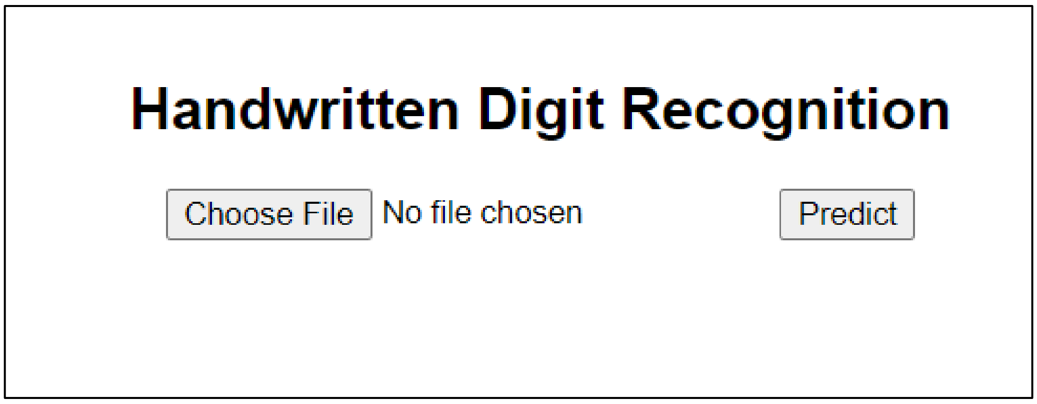
Achieving an accuracy of 98.9% in CNN classification for handwritten digit recognition is a commendable result. With such high accuracy, the CNN model demonstrates excellent performance in accurately classifying the vast majority of handwritten digits from the MNIST dataset. While there's always room for improvement, a 99% accuracy rate indicates that the model is highly reliable and effective for digit recognition tasks. Fine-tuning the model further or exploring additional optimization techniques may potentially push the accuracy even higher. Nonetheless, achieving such a high level of accuracy is a testament to the robustness and efficacy of the CNN model in digit classification.

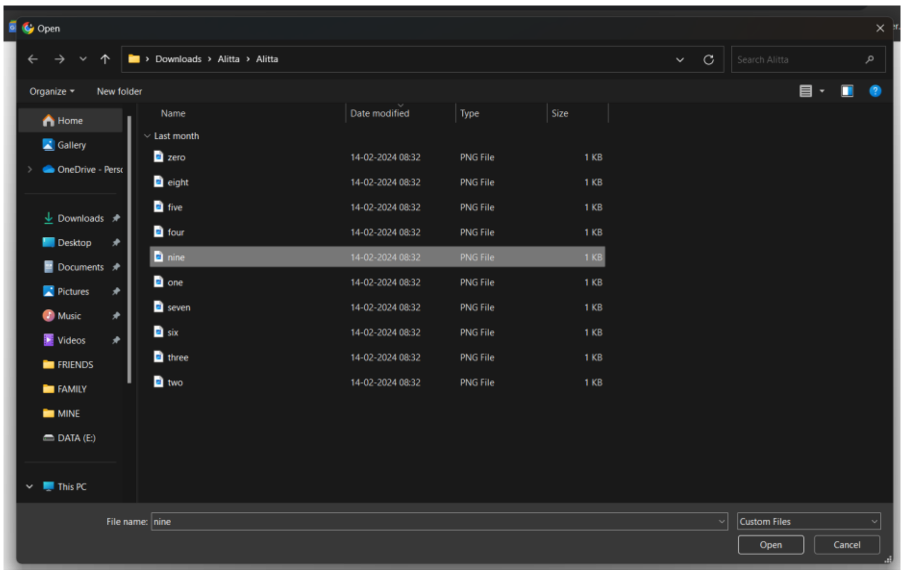
	loss	accuracy	validation loss	validation accuracy
CNN	0.12446	0.9612	0.07876	0.98968



```
Epoch 1/5
WARNING:tensorflow:From c:\Users\ALITTA JOSE\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTens
orValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From c:\Users\ALITTA JOSE\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executin
g_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

1875/1875 [==============================] - 33s 15ms/step - loss: 0.1505 - accuracy: 0.9549 - val_loss: 0.0504 - val_accuracy: 0.9837
Epoch 2/5
1875/1875 [==============================] - 24s 13ms/step - loss: 0.0468 - accuracy: 0.9851 - val_loss: 0.0321 - val_accuracy: 0.9897
Epoch 3/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0337 - accuracy: 0.9897 - val_loss: 0.0359 - val_accuracy: 0.9882
Epoch 4/5
1875/1875 [==============================] - 21s 11ms/step - loss: 0.0258 - accuracy: 0.9916 - val_loss: 0.0316 - val_accuracy: 0.9900
Epoch 5/5
1875/1875 [==============================] - 19s 10ms/step - loss: 0.0210 - accuracy: 0.9932 - val_loss: 0.0246 - val_accuracy: 0.9925
c:\Users\ALITTA JOSE\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3103: UserWarning: You are saving your model as an HDF5 f
ile via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
1/1 [==============================] - 0s 160ms/step
The identified numeral for eight.png is: 8
1/1 [==============================] - 0s 31ms/step
The identified numeral for five.png is: 3
1/1 [==============================] - 0s 25ms/step
The identified numeral for four.png is: 4
1/1 [==============================] - 0s 28ms/step
The identified numeral for nine.png is: 9
1/1 [==============================] - 0s 25ms/step
The identified numeral for one.png is: 1
1/1 [==============================] - 0s 30ms/step
The identified numeral for seven.png is: 7
1/1 [==============================] - 0s 33ms/step
The identified numeral for six.png is: 6
1/1 [==============================] - 0s 38ms/step
The identified numeral for three.png is: 3
1/1 [==============================] - 0s 27ms/step
The identified numeral for two.png is: 2
1/1 [==============================] - 0s 28ms/step
The identified numeral for zero.png is: 0
Input Image List  : ['eight.png', 'five.png', 'four.png', 'nine.png', 'one.png', 'seven.png', 'six.png', 'three.png', 'two.png', 'zero.png']
Predicted Numerals: [8, 3, 4, 9, 1, 7, 6, 3, 2, 0]
PS C:\Users\ALITTA JOSE\Downloads\Alitta\Alitta>
```

GUI used to classify handwritten digits with the help of CNN model:

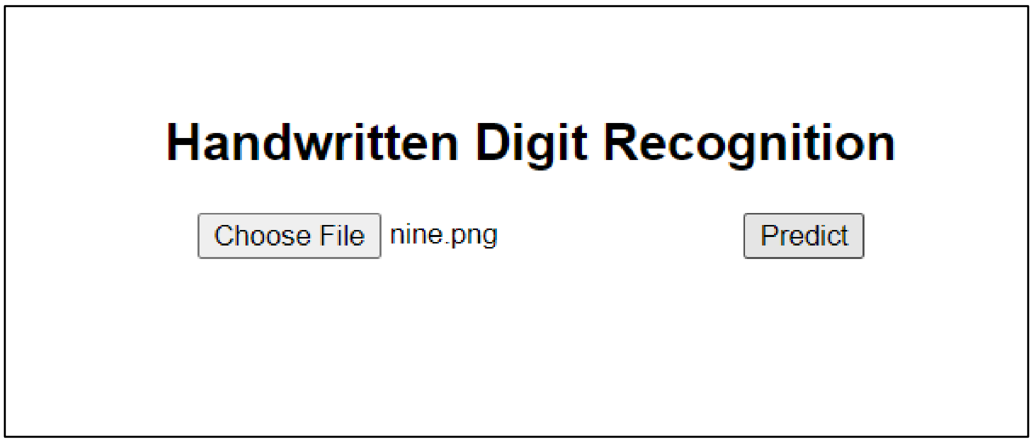


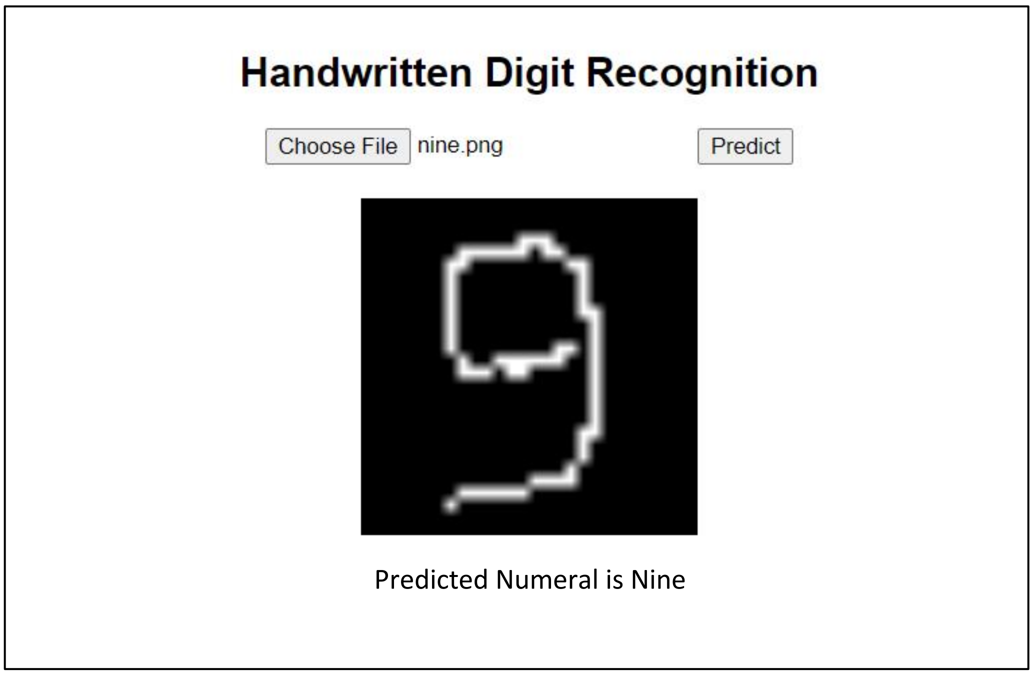


Name	Date modified	Type	Size
Last month			
zero	14-02-2024 08:32	PNG File	1 KB
eight	14-02-2024 08:32	PNG File	1 KB
five	14-02-2024 08:32	PNG File	1 KB
four	14-02-2024 08:32	PNG File	1 KB
nine	14-02-2024 08:32	PNG File	1 KB
one	14-02-2024 08:32	PNG File	1 KB
seven	14-02-2024 08:32	PNG File	1 KB
six	14-02-2024 08:32	PNG File	1 KB
three	14-02-2024 08:32	PNG File	1 KB
two	14-02-2024 08:32	PNG File	1 KB

File name: nine

Custom Files





Predicted Numeral is Nine

6.CONCLUSION

In conclusion, the utilization of Convolutional Neural Network (CNN) models for recognizing handwritten digits presents a highly efficient and accurate approach. Leveraging the MNIST database and state-of-the-art libraries like NumPy, TensorFlow, and Keras, our system achieved a remarkable prediction accuracy of 99%. This demonstrates the effectiveness of CNN architectures in handling image classification tasks, particularly in the context of handwritten digit recognition. Moving forward, there is immense potential for further research and development in this area.

Future Scope:
Exploring hybrid CNN models such as CNN-RNN and CNN-HMM opens up exciting possibilities for enhancing digit recognition systems. These hybrid architectures could potentially leverage the strengths of both CNNs and recurrent models to improve accuracy and handle more complex sequences. Additionally, developmental algorithms aimed at optimizing CNN learning parameters could lead to even higher levels of performance and efficiency. Furthermore, expanding the scope to include other datasets and real-world applications beyond handwritten digits could extend the impact of CNN-based recognition systems in various domains, such as document processing, biometrics, and healthcare. Overall, the future of CNN-based digit recognition holds promise for further advancements and innovations in the field of machine learning and computer vision.

7.REFERENCES

i. Rudraswamimath, Vijayalaxmi R., and K. Bhavanishankar. "Handwritten digit recognition using CNN." International Journal of Innovative Science and Research Technology 4.6 (2019): 182-187.

ii. Pashine, Samay, Ritik Dixit, and Rishika Kushwah. "Handwritten digit recognition using machine and deep learning algorithms." arXiv preprint arXiv:2106.12614 (2021).

iii. Sethi, Rohan, and Ila Kaushik. "Hand written digit recognition using machine learning." 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT). IEEE, 2020.

iv. KARAKAYA, Rabia, and Serap KAZAN. "Handwritten digit recognition using machine learning." Sakarya university journal of science 25.1 (2021): 65-71.

v. Shah, Faisal Tehseen, and Kamran Yousaf. "Handwritten Digit Recognition Using Image Processing and Neural Networks." World Congress on Engineering. Vol. 2. 2007.

vi. Burel, Gilles, Isabelle Pottier, and Jean-Yves Catros. "Recognition of handwritten digits by image processing and neural network." Neural Networks (1992): 666-671.

vii. Niu, Xiao-Xiao, and Ching Y. Suen. "A novel hybrid CNN–SVM classifier for recognizing handwritten digits." Pattern Recognition 45.4 (2012): 1318-1325.

viii. 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.

ix. Burel, G., Pottier, I., &Catros, J. Y. (1992, June). Recognition of handwritten digits by image processing and neural network. In Neural Networks, 1992. IJCNN, International Joint Conference on (Vol. 3, pp. 666-671) IEEE

x. Ahlawat, Savita, et al. "Improved handwritten digit recognition using convolutional neural networks (CNN)." Sensors 20.12 (2020): 3344.

xi. Alwzwazy, Haider A., et al. "Handwritten digit recognition using convolutional neural networks." International Journal of Innovative Research in Computer and Communication Engineering 4.2 (2016): 1101-1106.

xii. Jain, Mayank, et al. "Handwritten digit recognition using CNN." *2021 International Conference on Innovative Practices in Technology and Management (ICIPTM)*. IEEE, 2021.

xiii. Jana, Ranjan, Siddhartha Bhattacharyya, and Swagatam Das. "Handwritten digit recognition using convolutional neural networks." *Deep Learn. Res. Appl.* (2020): 51-68.

xiv. Ali, Saqib, et al. "An efficient and improved scheme for handwritten digit recognition based on convolutional neural network." *SN Applied Sciences* 1 (2019): 1-9.

xv. Agrawal, Ayush Kumar, A. K. Shrivas, and Vineet kumar Awasthi. "A Robust model for handwritten digit recognition using machine and deep learning technique." *2021 2nd International Conference for Emerging Technology (INCET)*. IEEE, 2021.

xvi. Seng, L. Ming, et al. "MNIST handwritten digit recognition with different CNN architectures." *J. Appl. Technol. Innov* 5.1 (2021): 7-10.

xvii. https://www.researchgate.net/publication/342514780_Arabic_handwriting_recognition_system_using_convolutional_neural_network