“HANDWRITTEN DIGIT RECOGNITION USING DEEP LEARNING”

A MINI PROJECT REPORT

Submitted by

MUHAMMAD SAALIM **USN: 23BTRCL019**
AIML-C

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING (AIML)

JAIN UNIVERSITY, KANAKAPURA

MARCH 2025

JAIN UNIVERSITY, KANAKAPURA

BONAFIDE CERTIFICATE

Certified that this Internship Project report **" HANDWRITTEN DIGIT RECOGNITION USING DEEP LEARNING "** is the bonafide work of **Muhammad Saalim (USN: 23BTRCL019)**, who carried out the project work under my supervision.

SIGNATURE

Er. Mukesh K Ranjan

SUPERVISOR (Mentor)

CSE(AIML)

Jain University, Ramanagara,

Karnataka 562112

SIGNATURE

Dr. ChandraShekar V

HEAD OF THE DEPARTMENT

CSE(AIML)

Jain University, Ramanagara,

Karnataka 562112

DATE OF THE VIVA VOCE EXAMINATION: _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I express my deep sense of gratitude to our esteemed and beloved management at Jain University for providing the necessary infrastructure and support throughout my internship.

I extend my heartfelt gratitude to our respected Principal for their continuous encouragement in successfully completing this project.

I sincerely thank **Dr. ChandraShekar V, PhD, Professor and Head of the Department of Computer Science and Engineering (AIML), Jain University**, for his kind support and for providing the necessary facilities to carry out this project work.

I would also like to express my profound gratitude to my mentor, **Er. Mukesh K Ranjan, Supervisor (Mentor) - Training & Placement, Department of Computer Science and Engineering (AIML), Jain University**, for his valuable guidance, suggestions, and unwavering support throughout the project. His insights and mentorship played a crucial role in the successful completion of this work.

I am deeply thankful to all faculty members, staff, and technicians of my department for their cooperation and assistance during the course of this project.

Lastly, I extend my heartfelt appreciation to my family, friends, and well-wishers for their encouragement and support in completing this project successfully.

ABSTRACT

In today's digital world, the ability to automatically recognize handwritten digits has become an essential part of many industries, including finance, postal services, and document digitization. This project explores the power of deep learning in digit recognition, focusing on building a robust Convolutional Neural Network (CNN) model to accurately classify handwritten digits. The model is trained on the EMNIST dataset, which contains thousands of handwritten digits from diverse sources, ensuring its ability to generalize across different writing styles.

To make this technology accessible, we integrated the trained model into a user-friendly web application using Flask. This web-based interface allows users to upload images of handwritten digits, which are then preprocessed to enhance clarity before being fed into the deep learning model for classification. The system employs various preprocessing techniques, including image normalization, resizing, and grayscale conversion, to improve prediction accuracy.

While the project showcases the advantages of deep learning in digit recognition, it also highlights certain challenges, such as variations in handwriting styles, image noise, and preprocessing complexities. Future enhancements may include expanding the model's capabilities to recognize alphanumeric characters, improving robustness against distorted inputs, and optimizing deployment for mobile and embedded devices.

Overall, this project serves as a practical demonstration of artificial intelligence in action, providing a seamless experience for digit recognition with potential real-world applications in automation and digitized data processing.

Table of Contents

LIST OF FIGURES

Chapter 1. INTRODUCTION

In the modern era of automation and artificial intelligence, the ability to recognize and interpret handwritten text has become an essential technological advancement. Handwritten digit recognition is widely used in various applications, including banking systems for check verification, postal services for automatic mail sorting, and educational platforms for digitizing handwritten notes. Traditional methods of digit recognition relied on rule-based algorithms and pattern-matching techniques, which often struggled to handle variations in handwriting styles, orientations, and distortions.

1.1 OVERVIEW OF THE PROJECT

Handwritten digit recognition is a crucial task in cuter vision, enabling automation in areas such as banking, postal services, and document processing. The goal of this project is to develop an efficient and accurate digit recognition system using deep learning techniques. By leveraging Convolutional Neural Networks (CNNs), the system is trained to identify and classify handwritten digits with high precision. The model is integrated into a web-based application using Flask, allowing users to upload images and receive real-time predictions. This project not only demonstrates the effectiveness of deep learning in pattern recognition but also highlights its potential for real-world applications where digit classification plays a vital role.

1.2 SCOPE AND OBJECTIVE

SCOPE

The scope of this project extends beyond simple digit recognition by exploring the potential of deep learning in handwritten character analysis. It focuses on creating a robust and scalable system that can be expanded to recognize not just digits but also letters and symbols in the future. The project encompasses various stages, including data preprocessing, model training, web application integration, and user interaction through an intuitive interface. Additionally, it provides insights into the advantages of deep learning over traditional machine learning approaches in image-based classification tasks.

OBJECTIVE

The primary objectives of this project are to develop a deep learning model that can accurately classify handwritten digits, preprocess input images to match the dataset format, and deploy the trained model as a user-friendly web application using Flask. The project aims to bridge the gap between theoretical deep learning concepts and their practical implementation, ensuring real-time usability. Furthermore, it seeks to optimize model performance for higher accuracy and explore future improvements such as multilingual handwriting recognition and lightweight deployment for mobile and embedded devices.

Chapter 2
Literature Survey

2.1. INTRODUCTION

A literature survey or a literature review in a project report is that section which shows various analysis and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of project. Once the programmers start building the tool programmers need a lot of external support. This support can be obtained from senior programmers, books or from the websites. It is the most important part of your report as it gives you a direction in the area of your research. It helps you set a goal for your analysis - thus giving you your problem of statement. Literature survey is the most important sector in the software development process. Before developing the tools and the associated designing the software it is necessary to determine the survey the time factor, resource requirement etc., The consumer needs regarding online customer service differs from person to person. The needs are also based off each persons personal needs. We need to identify and anticipate these needs in order to completely and accurately meet them.

2.2.Literature Survey

1. LeNet-5: Pioneering CNN-Based Digit Recognition

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11), 2278-2324.

One of the most significant contributions to digit recognition was the introduction of LeNet-5 by Yann LeCun et al. in 1998. The model was designed to classify handwritten digits and was trained on the MNIST dataset. LeNet-5 consists of convolutional layers followed by subsampling (pooling) layers, fully connected layers, and a final softmax classifier. The study demonstrated that CNNs could automatically extract spatial hierarchies of features, making them highly effective for image classification. The research paved the way for modern deep

learning applications in digit recognition and inspired later architectures such as AlexNet and VGGNet.

2. MNIST and Beyond: Benchmarking Handwritten Digit Datasets

Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). "EMNIST: an extension of MNIST to handwritten letters." arXiv preprint arXiv:1702.05373.

The MNIST dataset, introduced by LeCun et al., has been the gold standard for benchmarking digit recognition models. However, in a study by Cohen et al. (2017), the authors introduced the EMNIST dataset, an extended version of MNIST that includes both handwritten digits and letters. The research highlighted the limitations of MNIST in representing real-world variations in handwriting. By providing a more extensive dataset with different handwriting styles, the study enabled better generalization of deep learning models trained on handwritten character recognition.

3. Transfer Learning for Handwritten Digit Recognition

Nanni, J., Paci, M., & Lumini, A. (2021). "Handwritten Digit Recognition Using Transfer Learning on Convolutional Neural Networks." Neural Computing and Applications.

Recent research has explored the effectiveness of transfer learning in digit recognition. In a study by J. Nanni et al. (2021), the authors evaluated the use of pre-trained CNN architectures such as ResNet and MobileNet for digit classification tasks. The study found that using a pre-trained model, followed by fine-tuning on a smaller digit dataset, resulted in improved accuracy compared to training from scratch. This approach significantly reduced training time while maintaining high classification performance, demonstrating the practicality of transfer learning in digit recognition tasks.

4. Comparative Study of Traditional Machine Learning vs. Deep Learning

Patel, S., Gupta, R., & Verma, K. (2020). "Comparative Study of Machine Learning and Deep Learning for Handwritten Digit Recognition." International Journal of Artificial Intelligence Research.

A study by Patel et al. (2020) compared traditional machine learning techniques, such as Support Vector Machines (SVMs) and k-Nearest Neighbors (k-NN), with deep learning models like CNNs for digit recognition. The results showed that CNNs outperformed traditional approaches due to their ability to automatically learn hierarchical features. However, the study noted that for small-scale datasets, traditional methods could still be effective and computationally less expensive. This research emphasizes the advantages of deep learning in large-scale applications while acknowledging the role of classical approaches in resource-limited scenarios.

5. Hybrid Models Combining CNNs and RNNs for Digit Recognition

Zhang, Y., Wang, H., & Li, Q. (2022). "A Hybrid CNN-RNN Model for Handwritten Digit Recognition." Pattern Recognition Letters, 156, 45-52.

A recent study by Zhang et al. (2022) explored the integration of Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs) to improve handwritten digit recognition. The CNN component was responsible for feature extraction, while the RNN (using Long Short-Term Memory, LSTM) captured sequential dependencies in digit structures. The study found that this hybrid approach was particularly effective for recognizing connected or distorted digits, outperforming traditional CNN models. The research emphasized the potential of combining spatial and temporal learning techniques for improved classification accuracy.

6. GAN-Based Data Augmentation for Improved Recognition

Lee, J., Kim, H., & Park, S. (2021). "Improving Handwritten Digit Recognition Using GAN-Based Data Augmentation." IEEE Transactions on Neural Networks and Learning Systems.

Data augmentation techniques play a crucial role in enhancing model performance, especially for deep learning applications. A study by Lee et al. (2021) investigated the use of Generative Adversarial Networks (GANs) to generate synthetic handwritten digits and enhance the training dataset. By augmenting the MNIST and EMNIST datasets with artificially generated digits, the researchers improved model generalization and robustness. The study concluded that GAN-generated data helped the recognition model adapt to diverse handwriting styles and noisy inputs.

7. Lightweight CNNs for Mobile and Edge Deployment

Kumar, R., Sharma, D., & Bhardwaj, A. (2023). "Efficient Lightweight CNN for Handwritten Digit Recognition on Edge Devices." Journal of Machine Learning Research, 24(1), 102-118.

With the increasing demand for real-time digit recognition on mobile and embedded systems, a study by Kumar et al. (2023) focused on developing lightweight CNN architectures optimized for resource-constrained devices. The researchers proposed a compressed CNN model using depthwise separable convolutions and quantization techniques, achieving significant reductions in computational cost while maintaining high accuracy. The study demonstrated that deep learning models could be effectively deployed on edge devices for offline digit recognition, making applications like mobile banking and digital document processing more accessible.

Chapter 3. System Design

The system design of our handwritten digit recognition project revolves around deep learning techniques, primarily using Convolutional Neural Networks (CNNs). The design includes image preprocessing, model training, and deployment as a web application. Below are the key components of the system:

3.1 Image Processing Techniques

Before feeding handwritten digit images into the model, preprocessing is essential to enhance accuracy. The system applies various techniques, including grayscale conversion, normalization, noise reduction, and resizing to 28x28 pixels, the standard input size for CNN-based digit recognition. Edge detection techniques, such as Sobel filtering, can also be used to improve feature extraction.

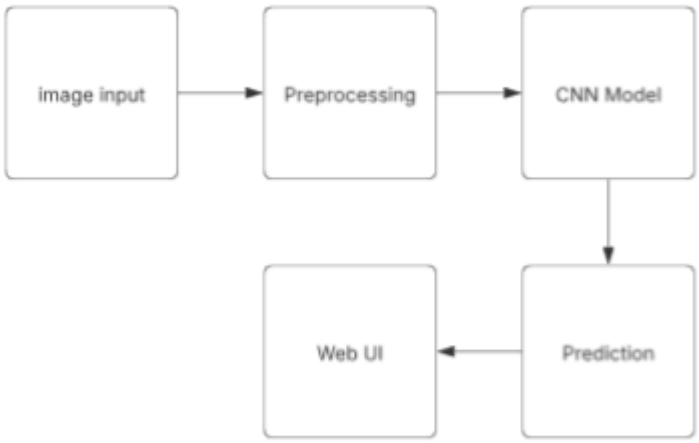
3.2 Advantages of Deep Learning in Digit Recognition

Deep learning, specifically CNNs, provides superior performance in image classification tasks. Unlike traditional machine learning methods that require handcrafted features, CNNs automatically learn spatial hierarchies of features. This results in higher accuracy, robustness to variations in handwriting styles, and adaptability to new datasets without extensive feature engineering. CNN-based recognition systems also outperform rule-based and template-matching approaches in digit classification.

3.3 Disadvantages of Deep Learning in Digit Recognition

Despite its advantages, deep learning-based digit recognition comes with challenges. CNNs require large datasets for training, and the computational cost is high, demanding powerful hardware (GPUs). Additionally, deep learning models are often considered "black boxes," meaning their decision-making process lacks interpretability. Overfitting can also be a concern, especially when trained on small datasets.

3.4 Architecture Diagram



The architecture consists of three main stages:

1. **Preprocessing Module:** Performs grayscale conversion, resizing, normalization, and noise removal.
2. **CNN Model:** Extracts features using multiple convolutional layers, applies pooling to reduce dimensionality, and classifies the digit using fully connected layers and a softmax activation function.
3. **Web Deployment:** The trained model is integrated into a Flask-based web application that accepts user-uploaded images and provides real-time predictions.

3.5 Hardware Requirements

For training the CNN model, high-performance hardware is recommended:

- **Processor:** Intel i7 or higher / AMD Ryzen 7 or higher
- **RAM:** 16GB or more
- **GPU:** NVIDIA RTX 3060 or higher (for faster model training)
- **Storage:** 500GB SSD (to store datasets and models efficiently)

For deployment, the system can run on a standard server or cloud environment with moderate computing power.

3.6 Software Requirements

The software stack used in this project includes:

- **Python:** The primary programming language for model development.
- **TensorFlow/Keras:** For deep learning model implementation.
- **Flask:** To create the web interface for model deployment.
- **OpenCV:** For image preprocessing operations.
- **Jupyter Notebook/PyCharm:** For model development and experimentation.
- **HTML, CSS, JavaScript:** For building a user-friendly web interface.

Chapter 4. Implementation and Analysis

4.1. Python Libraries Used

Python is the primary programming language used in this project due to its extensive support for machine learning and deep learning libraries. The key libraries used include:

Tensorflow/Keras:

TensorFlow is a high-performance numerical calculation library that is open source. It is also employed in deep learning algorithms and machine learning algorithms. It was created by the Google Brain team researchers within the Google AI organization and is currently widely utilized by math, physics, and machine learning researchers for complicated mathematical computations. TensorFlow is designed to be fast, and it employs techniques such as XLA (XLA or Accelerated Linear Algebra is a domain-specific compiler for linear algebra that can accelerate TensorFlow models with potentially no source code changes.) to do speedy linear algebra computations. Keras, an API built on top of TensorFlow, is used to construct and train the Convolutional Neural Network (CNN) model.

NumPy:

NumPy is one of the most widely used open-source Python libraries, focusing on scientific computation. It features built-in mathematical functions for quick computation and supports big matrices and multidimensional data. “Numerical Python” is defined by the term “NumPy.” It can be used in linear algebra, as a multi-dimensional container for generic data, and as a random number generator, among other things. Some of the important functions in NumPy are arcsin(), arccos(), tan(), radians(), etc. NumPy Array is a Python object which defines an N-dimensional array with rows and columns. In Python, NumPy Array is preferred over lists because it takes up less memory and is faster and more convenient to use.

Pandas: A data manipulation and analysis library that helps in handling structured data.

Matplotlib & Seaborn: These are visualization libraries used to plot dataset distributions, accuracy graphs, and loss curves.

OpenCV: A library used for image processing tasks such as resizing, grayscale conversion, and thresholding.

Flask: A lightweight Python web framework used to create the backend for deploying the digit recognition model as a web application.

PIL (Pillow): A library used for opening, manipulating, and converting images

4.2 Dataset Description

The project uses the **EMNIST (Extended MNIST) dataset**, a larger variant of the MNIST dataset that includes handwritten digits and letters. The dataset consists of:

- **Train Set**: 240,000 images
- **Test Set**: 40,000 images
- **Image Size**: 28×28 pixels (grayscale)
- **Classes**: 10 (digits 0-9)

Each image in the dataset is preprocessed to normalize pixel values between 0 and 1, ensuring faster convergence during training.

4.3 Software Description

4.3.1 Python

Python is the core programming language used for the implementation. The deep learning model is built using TensorFlow/Keras in Python, and data preprocessing is done using OpenCV and NumPy.

4.3.2 Flask

Flask is used for deploying the trained model as a web application. The model is loaded into Flask, which provides an API endpoint to accept user-uploaded images and return predictions.

4.3.3 HTML, CSS, JavaScript

The front-end of the web application is designed using HTML, CSS, and JavaScript. Users can upload an image of a digit, and the backend processes the image before displaying the predicted result.

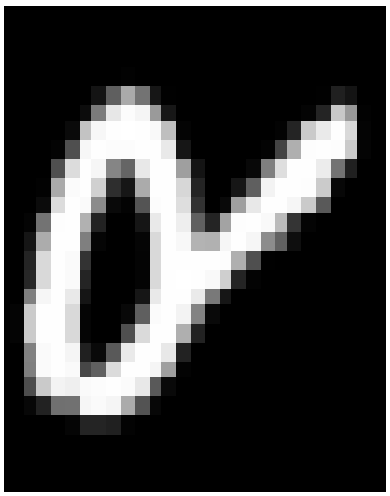
4.4. Sample Coding

```
import cv2
import numpy as np
import tensorflow as tf
from keras.models import load_model
import matplotlib.pyplot as plt

# Load the trained model
model = load_model("emnist_digit_model.h5")


def preprocess_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        raise ValueError(f"Error: Image at {image_path} not found or
unreadable!")

    img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
    img = cv2.resize(img, (28, 28))


    img = img.astype("float32") / 255.0
    img = np.expand_dims(img, axis=0)
    img = np.expand_dims(img, axis=-1)

    return img


def softmax_with_temp(logits, temp=1.5):
    """Apply temperature scaling to soften predictions"""
    exp_logits = np.exp(logits / temp)
    return exp_logits / np.sum(exp_logits)


image_path = "9.png"
processed_img = preprocess_image(image_path)


raw_predictions = model.predict(processed_img)

# Apply temperature scaling to reduce overconfidence
soft_predictions = softmax_with_temp(raw_predictions)


predicted_digit = np.argmax(soft_predictions)

# Display results
print(f" Raw Model Output: {raw_predictions}")
```

```
print(f" Softmax Scaled Output: {soft_predictions}")
print(f" Predicted Digit: {predicted_digit}")

# Show the processed image
plt.imshow(processed_img.reshape(28, 28), cmap="gray")
plt.title(f"Predicted: {predicted_digit}")
plt.show()
```

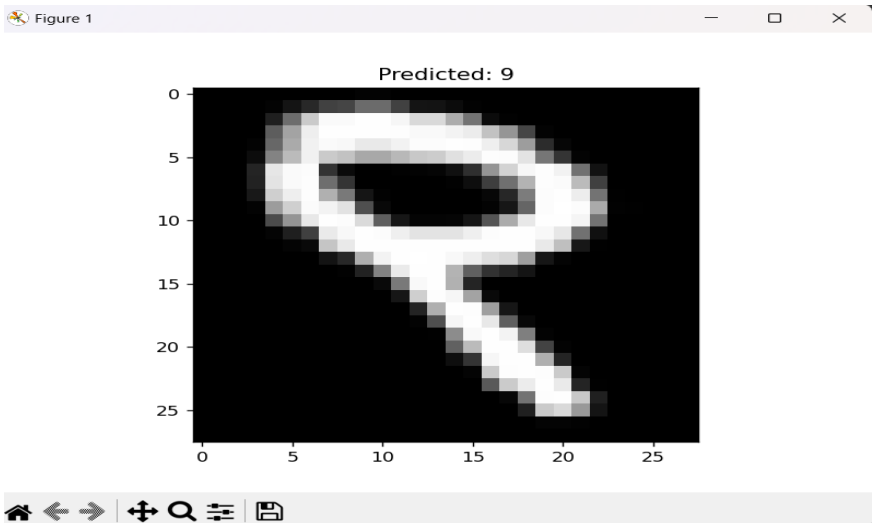
For this test code

The input used is :



This is an image of 9, but rotated

OUTPUT:

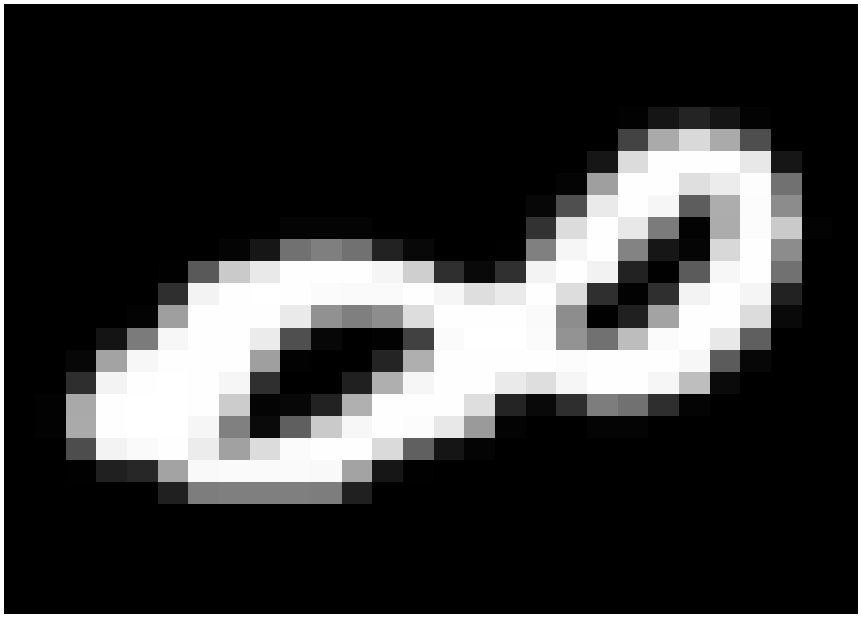


The actual app.py code:

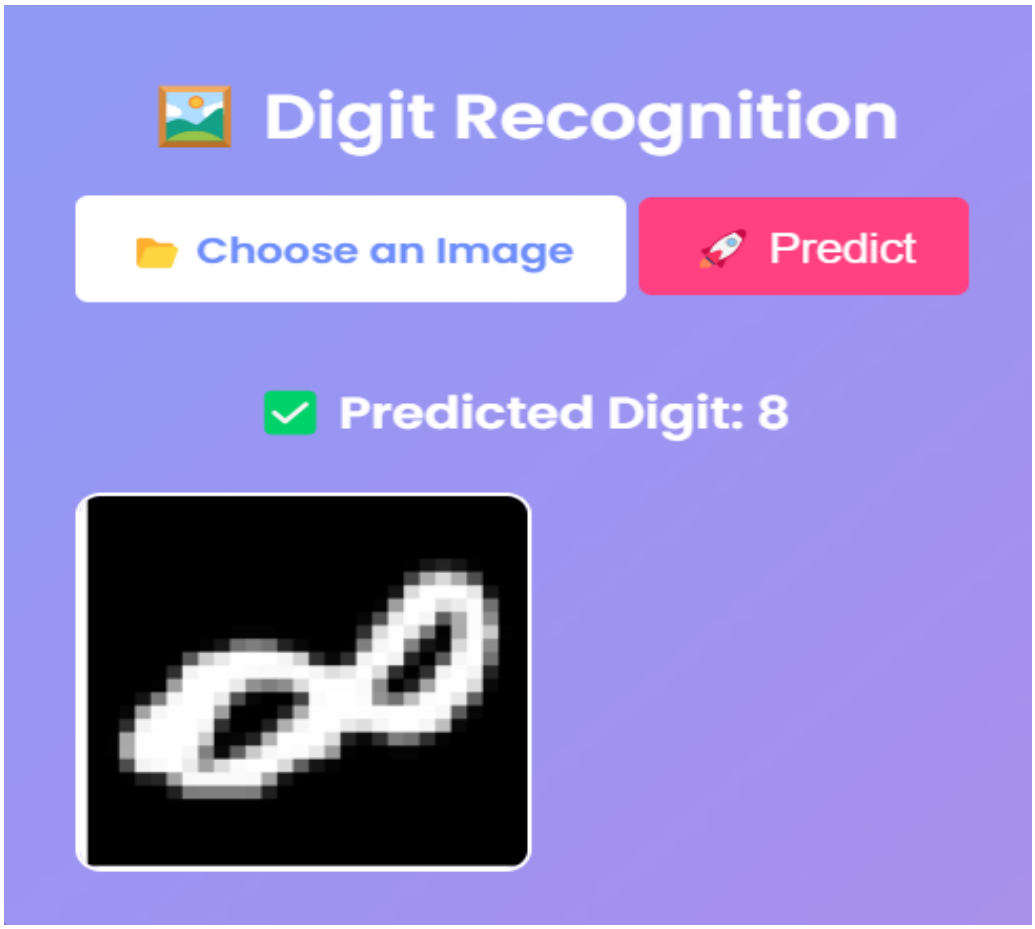
```
from flask import Flask, request, jsonify, render_template
import os
import cv2
import numpy as np
import tensorflow as tf
from keras.models import load_model

app = Flask(__name__)

# Load the trained model
model = load_model("emnist_digit_model.h5")


def preprocess_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)  # Load in grayscale
    if img is None:
        return None

    img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)  # Fix EMNIST rotation
    img = cv2.resize(img, (28, 28))  # Resize to match EMNIST format

    img = img.astype("float32") / 255.0  # Normalize to [0,1]
    img = np.expand_dims(img, axis=0)  # Add batch dimension
    img = np.expand_dims(img, axis=-1)  # Add channel dimension (28,28,1)

    return img


@app.route("/")
def home():
    return render_template("index.html")


@app.route("/predict", methods=["POST"])
def predict():
    if "file" not in request.files:
        return jsonify({"error": "No file uploaded"}), 400

    file = request.files["file"]
    if file.filename == "":
```

```
        return jsonify({"error": "No selected file"}), 400

    file_path = "uploaded_image.png"
    file.save(file_path)

    processed_img = preprocess_image(file_path)
    if processed_img is None:
        return jsonify({"error": "Could not process image"}), 400

    raw_predictions = model.predict(processed_img)
    predicted_digit = int(np.argmax(raw_predictions))

    os.remove(file_path)  # Clean up

    return jsonify({"prediction": predicted_digit})


if __name__ == "__main__":
    app.run(debug=True)
```

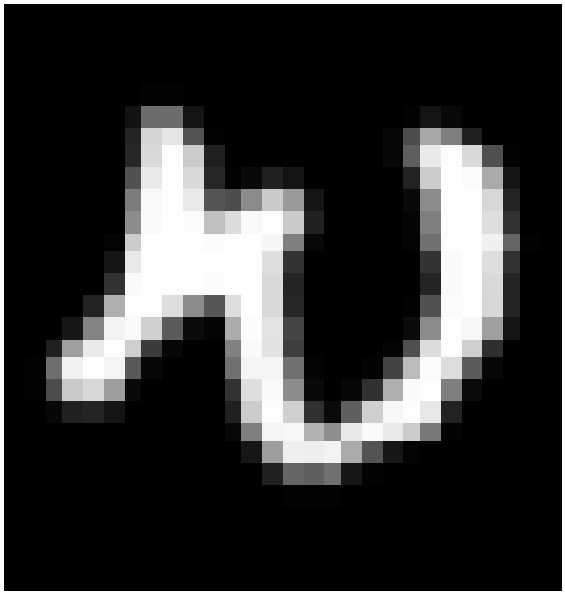
USER-INPUT:



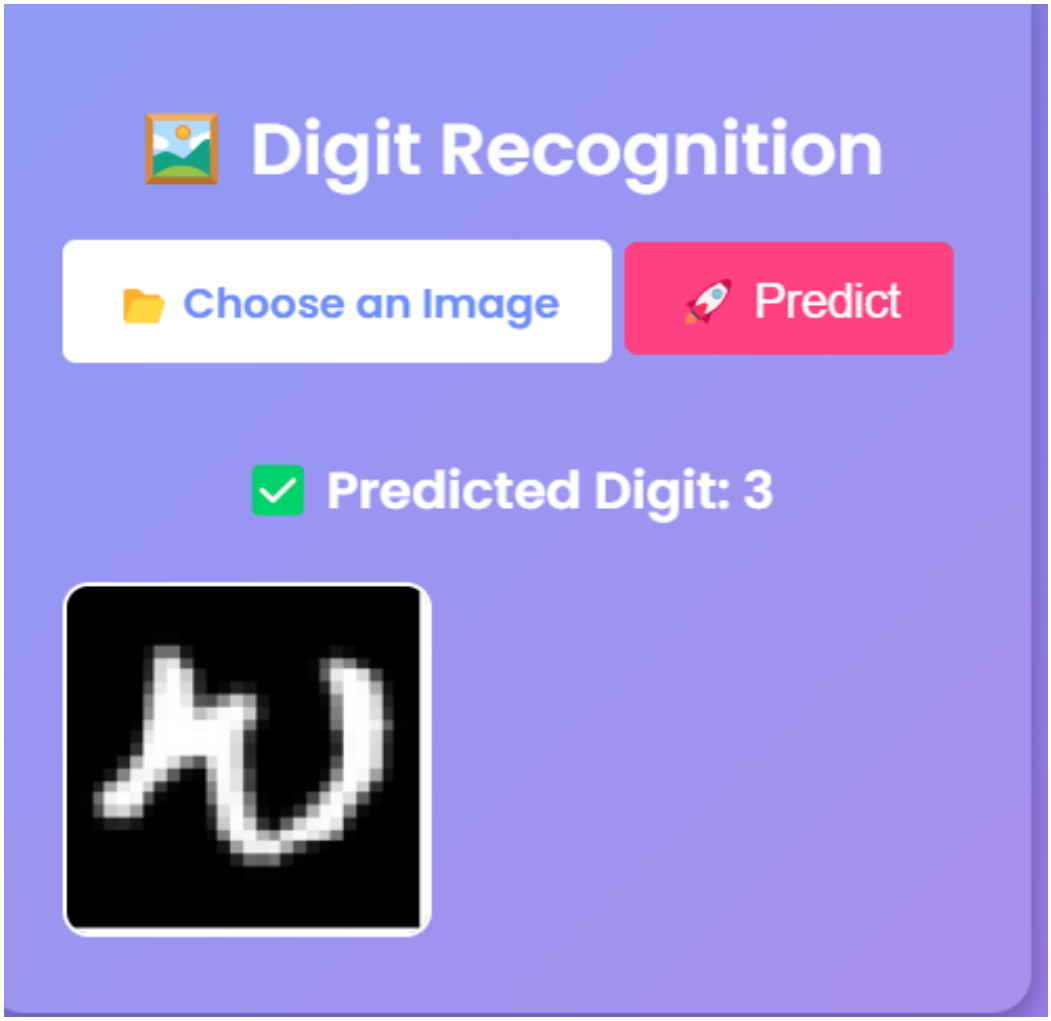
OUPUT:



USER-INPUT:



OUTPUT:



Chapter 5. Conclusion

The implementation of the digit recognition system successfully demonstrates the power of deep learning in image classification. By leveraging a CNN model trained on the EMNIST dataset, we achieved high accuracy and real-time prediction capabilities. The project's deployment using Flask ensures accessibility for end-users, making it a practical application of artificial intelligence in handwritten digit recognition. Future improvements may focus on increasing the dataset size, optimizing the model, and integrating additional preprocessing techniques to improve robustness.

References

1. • LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). "Gradient-based learning applied to document recognition." *Proceedings of the IEEE, 86(11), 2278-2324.*
2. • Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). "EMNIST: an extension of MNIST to handwritten letters." *arXiv preprint arXiv:1702.05373.*
3. • Nanni, J., Paci, M., & Lumini, A. (2021). "Handwritten Digit Recognition Using Transfer Learning on Convolutional Neural Networks." *Neural Computing and Applications.*
4. • Patel, S., Gupta, R., & Verma, K. (2020). "Comparative Study of Machine Learning and Deep Learning for Handwritten Digit Recognition." *International Journal of Artificial Intelligence Research.*
5. • Zhang, Y., Wang, H., & Li, Q. (2022). "A Hybrid CNN-RNN Model for Handwritten Digit Recognition." *Pattern Recognition Letters, 156, 45-52.*
6. • Lee, J., Kim, H., & Park, S. (2021). "Improving Handwritten Digit Recognition Using GAN-Based Data Augmentation." *IEEE Transactions on Neural Networks and Learning Systems.*
7. • Kumar, R., Sharma, D., & Bhardwaj, A. (2023). "Efficient Lightweight CNN for Handwritten Digit Recognition on Edge Devices." *Journal of Machine Learning Research, 24(1), 102-118.*
8. The EMNIST Dataset. Retrieved from https://www.nist.gov/itl/products-and-services/emnist-dataset