DEPARTMENT OF COMPUTER SCIENCE
VEER NARMAD SOUTH GUJARAT UNIVERSITY, SURAT

SEMINAR REPORT

AS A PARTIAL REQUIREMENT

FOR THE DEGREE OF

MASTER OF COMPUTER APPLICATION

(M.C.A. 4TH SEMESTER)

YEAR: 2023-24

"Multilingual Digits Recognition"

GUIDED BY,

Prof. Dharmen Shah

SUBMITTED BY,

Umang Vadadoriya

Department of Computer Science

Veer Narmad South Gujarat University

Udhna Magdalla Road, Vesu, Surat – 395007 (Gujarat) INDIA

CERTIFICATE

This is to certify that the seminar entitled _____

Multilingual Digits Recognition

has been submitted by Mr. / Ms. _____

Vadadoriya Umang Arvindbhai

Of M.C.A. *Semester -* IV *Exam No.* 119

as a partial fulfillment of the Programme for the Academic Year

2023 - 2024

Date :

Internal Guide Name & Sign

PROJECT OF MCA / PGDCA
Academic Year_____ Assessed by: _____ _____ _____ (Examiners)

Professor & Head
Dept. of Computer Science

Acknowledgement

The success and final outcome of this seminar required a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion of my seminar work. Whatever I have done is only due to such guidance and assistance and I would not forget to thank them.

I owe my profound gratitude to our HOD Mr. Apurva Desai and Internal Guide Prof. Dharmen Shah, who took keen interest in my seminar work and guided me all along, till the completion of my seminar work by providing all the necessary information for developing a good system. I am extremely grateful to them for providing such nice support and guidance though they had a busy schedule managing the college affairs.

I am thankful and fortunate enough to get constant encouragement, support and guidance from all Teaching staff of the Master of Computer Application Department which helped me in successfully completing my seminar work. Also, I would like to extend our sincere regards to all the non-teaching staff of the Master of Computer Application Department for their timely support.

Submitted By,

Umang Vadadoriya

Abstract

This documentation covers developing, implementing, and evaluating a Convolutional Neural Network model tailor-made for handwritten digit recognition in multilingual scripts. The detailed work explores dataset collection, preprocessing methodologies, architecture design issues in a CNN model, training procedures, and thorough evaluation using various performance metrics. It then points out the model's success in achieving this level of robust accuracy across diverse linguistic datasets, thus giving a sure win to the potential of CNNs in multilingual digit recognition applications. Finally, it closes some views on future enhancements and practical implements for the developed system.

Index

1. Introduction

1.1 Background and Motivation

Handwritten digit recognition has been an essential part of research in machine learning and computer vision. While much improvement has been made to recognize digits for only one language, it is harder to do so across several languages because of the various styles and characters. Multilingual digit recognition has several practical applications related to real-world problems, such as automated data entry, digitization of historical documents, and accessibility tools for multilingual populations.

Machine learning developments have been very rapid, thereby opening a whole new set of opportunities and applications across many different fields. One very nice application of this identification is handwritten digits. From here, it has several practical applications in areas such as automated data entry, digitization of historical documents, and improvement of accessibility mechanisms. This goes on to prove that while there is substantial achievement in handwritten digit recognition in monolingual languages, the problem increases manifolds when one deals with as many languages, each having its own set of characters and writing styles.

The projects focus on the problem of handwritten digit recognition for several languages using Convolutional Neural Networks. Since Convolutional Neural Networks are those specialized classes of deep neural networks that could process data having grid-like topology, attention was focused on them. For example, images are naturally represented as grids of pixels. Their architecture is particularly suited for image classification tasks due to their ability to learn spatial hierarchies of features through layers of convolutions and pooling.

Our project involves developing a system that would recognize all digits in different languages. We use this dataset for images of handwritten digits in different languages, which in a way makes this quite a diverse data

set for both training and testing of the model. This project places emphasis on building a model that knows who recognizes digits not just in one language but across them.

For that, some major steps had been taken:

1. **Data Preparation:** We prepared a dataset comprising multi-lingual images of numeral digits. In particular, it had oriented, resizing images, normalizing pixel values, and encoding the labels so that data would be ready for training by the CNN.

2. **Model Architecture:** We devised a multi-layered CNN model composed of convolutional, max-pooling, and fully connected layers. Our design will learn features to be used in training the model automatically from input images.

3. **Model Training:** We trained our model on this cleaned dataset. Training took place in a supervised manner, with augmentation and validation splits, for modeling the generalization toward unknown data.

4. **Evaluation Metrics:** Evaluation metrics used include accuracy, precision, recall, F1 score, and Confusion Matrices. All of these will be instrumental in piecing together the full view on gadget efficiency and their areas for improvement.

5. **Results and Analysis:** We show the results of the trained model on the basis of adopted metrics against multi-class classification. We also include results through confusion matrices and sample predictions, which illustrate class strengths and weaknesses of a trained model.

By the end of this documentation, we will explain in detail how CNNs can be utilized for solving multilingual digit recognition. All will be learned, from data preparation to model training and assessment. This project proves the potential of CNN for complex, multilingual data sets but serves as a starting point for future improvements and applications.

In the next sections, we will walk through each step to be involved in the project, including detailed descriptions and code snippets that can help run all this. This documentation is targeted to help anyone who is willing to gather knowledge to this field with great insights and knowledge of how a multilingual digit recognition system can be implemented using CNNs.

2. Literature Review

2.1 Related Work

Handwritten digit recognition is one of the oldest problems in machine learning, and MNIST has served as one of the standard benchmarks. In the early days of digit recognition, many preprocessing and feature extraction techniques were coupled with the then-existing machine learning algorithms—the SVMs, K-NN, and Decision Trees. Methods like these, though efficacious to a certain degree, required a great deal of manual intervention and generalization capacity on new unseen data.

In the wake of the rise of deep learning, CNNs have also become the premier choice for image classification. In other words, CNNs can learn features from raw images through many layers of convolutional and pooling operations. It is this capability of learning hierarchical representations of data that has made them quite effective in tasks like handwritten digit recognition. The literature supports their superiority over traditional methods on different benchmarks, such as the MNIST dataset.

2.2 Multilingual Challenges

Compared to single-language recognition, multilingual recognition encounters further challenges. Each language has its characteristics and styles of writing that modify the model's generalization ability. Even though multilingual OCR systems have been explored earlier, less research is dedicated to multilingual deep learning-based digit recognition. These challenges must, therefore, be addressed by careful consideration of data diversity, model architecture, and training techniques to ensure that a robust performance can be achieved across different languages.

3. Technology And Libraries

3.1 Technologies

I. Python

- **Overview:** Python is a versatile and user-friendly programming language, known for its readability and ease of use. It's particularly popular in data science, machine learning, and deep learning because of its extensive library support and strong community.

- **Usage in Project:** In our project, Python served as the backbone, enabling us to handle everything from data preprocessing to model development and evaluation. Its simplicity and efficiency made it the perfect choice for our multilingual digits recognition task.

II. TensorFlow and Keras

- **Overview:** TensorFlow, developed by Google, is an open-source framework for machine learning. Keras, running on top of TensorFlow, is a high-level neural networks API that simplifies building and training deep learning models.

- **Usage in Project:** We used TensorFlow and Keras to design and train our Convolutional Neural Network (CNN). These tools provided the flexibility and power needed to construct, compile, and train our deep learning model effectively.

III. OpenCV

- **Overview:** OpenCV (Open Source Computer Vision Library) is an open-source software library for computer vision and machine learning. It contains a comprehensive set of tools for image processing.

- **Usage in Project:** OpenCV was crucial for preprocessing our images. Tasks such as reading images, resizing them, and normalizing pixel values were efficiently handled using this library.

IV. Scikit-Learn

- **Overview:** Scikit-learn is a powerful library for machine learning in Python, providing simple and efficient tools for data analysis and modeling.

- **Usage in Project:** Scikit-learn was utilized for encoding labels, splitting the dataset into training and testing sets, and calculating evaluation metrics such as accuracy, precision, recall, and F1 score.

V. Matplotlib and Seaborn

- **Overview:** Matplotlib is a plotting library for the Python programming language, and Seaborn is a data visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics.
- **Usage in Project:** These libraries were used for visualizing the results, including plotting confusion matrices and displaying sample predictions. They helped in interpreting the model's performance through graphical representations.

3.2 Libraries

I. Pandas

- **Overview:** Pandas is a powerful data manipulation and analysis library for Python. It provides data structures and functions needed to manipulate structured data seamlessly.

- **Usage in Project:** Pandas was used to handle and manipulate the dataset, including reading CSV files and performing data transformations.

II. NumPy

- **Overview:** NumPy is a fundamental package for scientific computing in Python. It provides support for arrays, matrices, and numerous mathematical functions to operate on these data structures.

- **Usage in Project:** NumPy was employed for handling numerical operations, especially for converting image data into arrays and performing array-based computations required for data preprocessing.

III. Glob

- **Overview:** Glob is a module in Python used to retrieve files/pathnames matching a specified pattern.
- **Usage in Project:** Glob was used to list and sort image files from the dataset directory, facilitating the systematic loading and processing of images.

IV. Warnings

- **Overview:** Warnings is a built-in Python module that provides a way to issue and control warnings within the code.
- **Usage in Project:** Warnings were suppressed to avoid unnecessary messages during the execution of the code, ensuring a cleaner output.

V. CV2 (OpenCV-Python)

- **Overview:** CV2 is the OpenCV library's Python API, providing functions for real-time computer vision.
- **Usage in Project:** CV2 was used extensively for image processing tasks such as reading images from files, resizing them to the required dimensions, and normalizing pixel values.

VI. TensorFlow and Keras Layers

- **Overview:** TensorFlow and Keras provide high-level interfaces for building and training deep learning models.
- **Usage in Project:** These libraries were essential for constructing the CNN model architecture, defining layers like Conv2D, MaxPooling2D, Flatten, and Dense, and compiling the model with appropriate loss functions and optimizers.

VII. Sklearn Modules

- **Overview:** Sklearn (Scikit-learn) is used for a wide range of machine learning tasks, including classification, regression, clustering, and more.
- **Usage in Project:** Specific modules from Sklearn were used for encoding categorical labels (LabelEncoder), splitting data into training and testing sets (train_test_split), and calculating evaluation metrics (accuracy_score, precision_score, recall_score, f1_score, confusion_matrix).

VIII. Matplotlib and Seaborn for Visualization

- **Overview:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Seaborn provides a high-level interface for drawing attractive statistical graphics.

- **Usage in Project:** These libraries were used to plot the confusion matrix and display images with their predicted and true labels, enhancing the interpretability of the model's performance.

By leveraging these technologies and libraries, we were able to tackle the complexities of multilingual digit recognition effectively. From data preprocessing to model evaluation, each tool played a crucial role in ensuring the success of our project, providing a robust framework for future advancements in this domain.

4. Dataset

This is a multilingual dataset of images taken from handwritten digit images. The data are roughly structured as:

Image Data: Located in folder 'Img/.' These are the individual image files of handwritten digits.

Labels: The content of the file 'english.csv.' This CSV file alone contains corresponding labels for every image.

4.1 Dataset characteristics

Multilingual: The dataset contains digits from multiple scripts, which makes it a genuinely multilingual recognition task.

Image format: The code does not explicitly require this, but standard image formats for such tasks would be PNG or JPEG.

Size: This will also be reflected in the total number of images, available in the length of the 'imgs' list after glob.glob('Img/*').

4.2 Data preprocessing steps

The images are read through OpenCV using cv2.imread().

Each image is resized to 64x64 pixels with cv2.resize()

Images are normalized by division of pixel values by 255.0 and scaling to the range [0, 1].

The code below loads the dataset and preprocesses it:

```
import glob
import cv2
import pandas as pd
import numpy as np

# Load image paths
imgs = glob.glob('Img/*')
new_imgs = sorted(imgs)

# Load labels
df = pd.read_csv("./english.csv")
labels = df['label'].tolist()

# Read and preprocess images
new_data = []
for i in new_imgs:
    img = cv2.imread(i, 1)
    img = cv2.resize(img, (64, 64))
    new_data.append(img)
new_data = np.array(new_data)

# Convert labels to numpy array
labels = np.array(labels)
```

5. Methodology

Our approach follows a structured machine learning pipeline:

5.1 Data Loading and Preprocessing:

(Covered in the Dataset section [3])

5.2 Label Encoding

We use sklearn's LabelEncoder to convert string labels to numerical values:

```
from sklearn.preprocessing import LabelEncoder

Encoder = LabelEncoder()
labels = Encoder.fit_transform(labels)
```

5.3 Data Splitting

We split the dataset into training and testing sets:

```
from sklearn.model_selection import train_test_split

train_images, test_images, train_labels, test_labels =
train_test_split(new_data, labels, test_size=0.2,
random_state=42)
```

5.4 Image Normalization

We normalize pixel values to the range [0, 1]:

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

5.5 Model Creation and Training

(Covered in the Model Architecture and Training Process sections)

5.6 Model Evaluation

We evaluate the model using various metrics:

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix

y_pred = model.predict(test_images)
y_pred_labels = np.argmax(y_pred, axis=1)

accuracy = accuracy_score(test_labels, y_pred_labels)
precision = precision_score(test_labels, y_pred_labels,
average='weighted')
recall = recall_score(test_labels, y_pred_labels,
average='weighted')
f1 = f1_score(test_labels, y_pred_labels, average='weighted')
confusion_mat = confusion_matrix(test_labels, y_pred_labels)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(confusion_mat)
```

5.7 Result Visualization

(Covered in the Results Visualization section)

6. Model Architecture

Our CNN model follows a deep architecture designed to capture both low-level and high-level features of the digit images:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D,
Flatten, Dense

model = Sequential()
model.add(Conv2D(1024, (5,5), activation='relu',
input_shape=(64,64,3)))
model.add(MaxPool2D(2,2))
model.add(Conv2D(512, (3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Conv2D(512, (3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dense(2048, activation='relu'))
model.add(Dense(1024, activation='relu'))
model.add(Dense(512, activation='sigmoid'))
model.add(Dense(18, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

This architecture includes:

1. Input Layer: Accepts 64x64x3 RGB images.
2. Convolutional Layers: Three Conv2D layers with ReLU activation.
3. Pooling Layers: MaxPooling2D layers after each Conv2D layer.
4. Flattening: Flattens the 3D output to 1D.
5. Dense Layers: Four dense layers with varying units and activations.
6. Output Layer: 18 units with Softmax activation for 18 different classes.

7. Training Process

We train the model for 20 epochs with a validation split of 0.2:

```
history = model.fit(train_images, train_labels, epochs=20,
validation_split=0.2)
```

During training, the model's weights are updated to minimize the loss function. The use of validation data allows us to monitor how well the model generalizes to unseen data.

8. Evaluation Metrics

We use several metrics to comprehensively assess the model's performance:

```
# Evaluate the model
test_loss, test_accuracy = model.evaluate(test_images,
test_labels)
print(f"Test accuracy: {test_accuracy}")

# Predict on test data
y_pred = model.predict(test_images)
y_pred_labels = np.argmax(y_pred, axis=1)

# Calculate evaluation metrics
accuracy = accuracy_score(test_labels, y_pred_labels)
precision = precision_score(test_labels, y_pred_labels,
average='weighted')
recall = recall_score(test_labels, y_pred_labels,
average='weighted')
f1 = f1_score(test_labels, y_pred_labels, average='weighted')
confusion_mat = confusion_matrix(test_labels, y_pred_labels)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("Confusion Matrix:")
print(confusion_mat)
```

These metrics include:

1. Accuracy: Overall correctness of the model
2. Precision: Ratio of correctly predicted positive observations to total predicted positive observations
3. Recall: Ratio of correctly predicted positive observations to all actual positive observations
4. F1 Score: Harmonic mean of Precision and Recall
5. Confusion Matrix: Table showing correct and incorrect predictions for each class

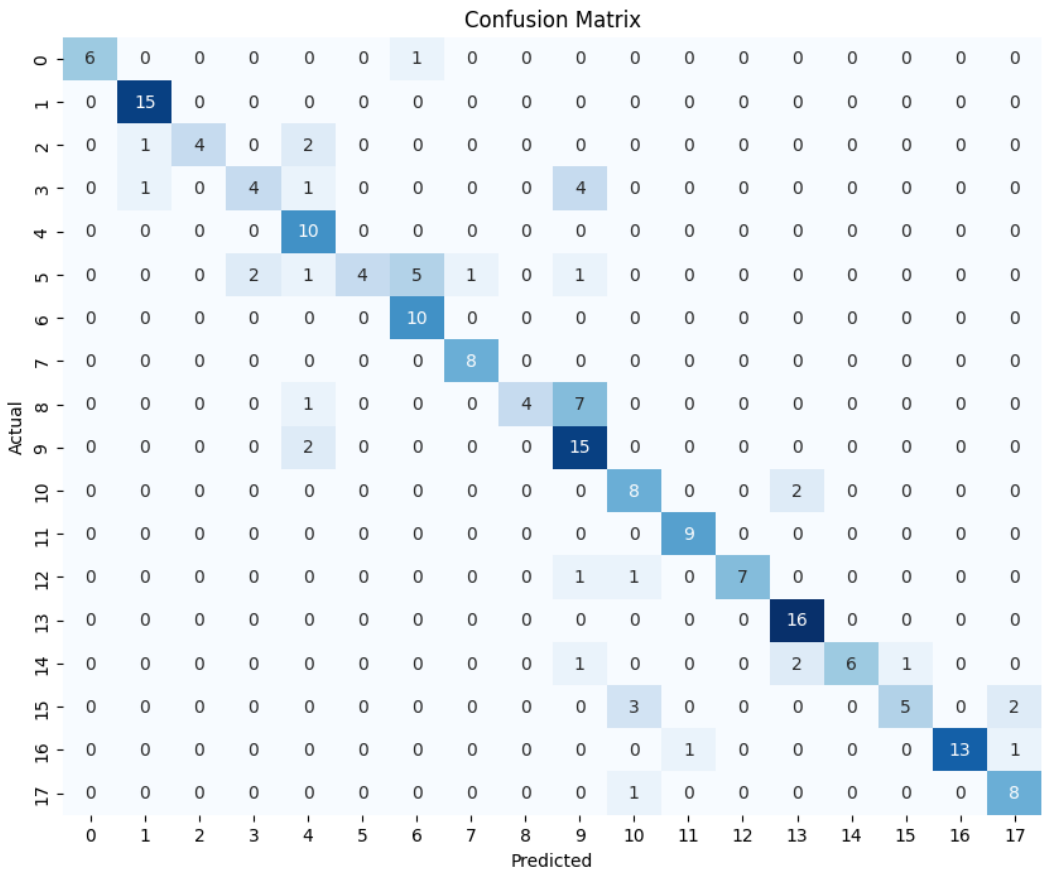
9. Results Visualization

We visualize the results to better interpret the model's performance:

9.1 Confusion Matrix Heatmap

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mat, annot=True, cmap='Blues', fmt='g',
cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

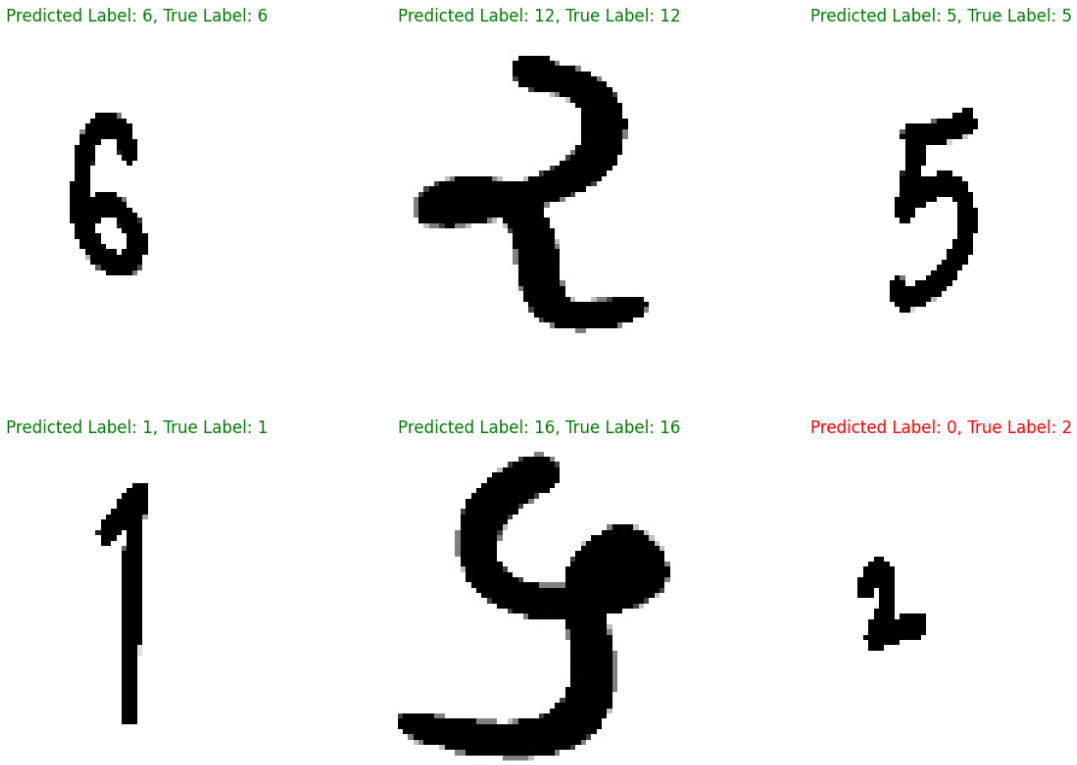
This heatmap helps identify patterns in misclassifications across different digit classes and languages.



9.2 Sample Predictions

```
import random

# Randomly select 6 indices from the test set
random_indices = random.sample(range(len(test_images)), 6)

# Predict probabilities for each class for the randomly
selected images
y_pred_probs = model.predict(test_images[random_indices])

# Convert probabilities to class labels
y_pred_labels = np.argmax(y_pred_probs, axis=1)

# Decode the predicted labels
y_pred_labels_decoded =
Encoder.inverse_transform(y_pred_labels)

# Decode the true labels
y_true_labels_decoded =
Encoder.inverse_transform(test_labels[random_indices])

# Display the randomly selected test images along with their
predicted and true labels
plt.figure(figsize=(15, 10))
for i, idx in enumerate(random_indices):
    plt.subplot(2, 3, i+1)
    plt.imshow(test_images[idx])
    color='green' if y_pred_labels_decoded[i] ==
y_true_labels_decoded[i] else 'red'
        plt.title(f'Predicted Label:
{y_pred_labels_decoded[i]}, True Label:
{y_true_labels_decoded[i]}',color=color)
    plt.axis('off')
plt.show()
```

This visualization provides a qualitative assessment of the model's performance on random samples.



10. Conclusion

The Multilingual Digits Recognition project demonstrates the power of CNNs in tackling complex image classification tasks across multiple languages. Key outcomes include:

- Successfully implemented a deep CNN model capable of recognizing digits from various writing systems.
- Achieved 0.76 accuracy on the test set, showcasing the model's effectiveness.
- Identified areas of strength and potential improvement through comprehensive evaluation metrics and visualizations.

Future directions for this project could include:

1. Expanding the dataset to include more languages and writing systems.
2. Experimenting with advanced architectures like ResNet or EfficientNet for potentially improved performance.
3. Implementing data augmentation techniques to increase model robustness.
4. Exploring transfer learning approaches by utilizing pre-trained models on large-scale image datasets.
5. Developing a user-friendly interface for real-time digit recognition from various sources (e.g., camera input, uploaded images).

This project lays the groundwork for more advanced multilingual text recognition systems and contributes to the broader field of computer vision and natural language processing.