# HANDWRITING RECOGNITION

**A PROJECT REPORT**


***Submitted By:***

HRISHIKESH BHATT- 211B139

PANKAJ KUMAR KUSHWAHA- 211B201

SHIVAM TRIPATHI- 211B293


**Under the Guidance of Dr. MAHESH KUMAR**

Nov- 2023

***Submitted in partial fulfillment of the Degree of***

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**


**Department of Computer Science & Engineering**

**JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY,**

**A-B ROAD, RAGHOGARH, DT. GUNA - 473226, M.P., INDIA**

## Declaration by the Student

We hereby declare that the work reported in the B. Tech. the project entitled "**HANDWRITING RECOGNITION**", in partial fulfillment for the award of the degree of BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING submitted at Jaypee University of Engineering and Technology, Guna, as per best of our knowledge and belief there is no infringement of the intellectual property right and copyright. In the event of any violation, we will be solely responsible.


Hrishikesh Bhatt (211B139)

Pankaj Kumar Kushwaha (211B201)

Shivam Tripathi (211B293)

Department of Computer Science and Engineering

Jaypee University of Engineering and Technology

Guna, M.P., India


Date: 30th November 2023

JAYPEE UNIVERSITY OF ENGINEERING & TECHNOLOGY

**Grade 'A+'** Accredited with by NAAC & Approved U/S 2(f) of the UGC Act, 1956
A.B. Road, Raghogarh, Dist: Guna (M.P.) India, Pin-473226
Phone: 07544 267051, 267310-14, Fax: 07544 267011
Website: www.juet.ac.in

# CERTIFICATE

This is to certify that the work titled "**HANDWRITING RECOGNITION**" submitted by "Hrishikesh Bhatt, Pankaj Kumar Kushwaha, Shivam Tripathi" in partial fulfillment for the award of the degree of BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE & ENGINEERING of Jaypee University of Engineering & Technology, Guna has been carried out under my supervision. To my knowledge and belief, there is no infringement of intellectual property rights and copyrights. Also, this work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma. In case of any violation concern, the student will solely be responsible.

Signature of Supervisor

Dr. Mahesh Kumar

## ACKNOWLEDGEMENT

We would like to express our gratitude and appreciation to all those who gave us the opportunity to complete this project. Special thanks to our supervisor Dr. Mahesh Kumar whose help, stimulating suggestions, and encouragement helped us in the time of development process and in writing this report. We also sincerely thank you for the time spent proofreading and correcting our many mistakes.

Thanking you

Hrishikesh Bhatt (211B139)

Pankaj Kumar Kushwaha (211B201)

Shivam Tripathi (211B293)

## Executive Summary

Handwriting Recognition pioneers the fusion of artificial intelligence and computer vision for handwriting recognition, addressing challenges in capturing diverse styles using traditional OCR methods. Leveraging Convolutional Neural Networks (CNNs) and OpenCV, the system aims for efficiency and accuracy in deciphering handwritten English, encompassing both printed and cursive scripts. The integration of CNN and OpenCV enhances adaptability to various styles, sizes, and orientations. The comprehensive evaluation involves rigorous experimentation across metrics and scenarios, acknowledging limitations such as style variations and potential biases.

The report begins with a literature review, setting the research landscape. The methodology outlines the CNN architecture and OpenCV's preprocessing role, with implementation details offering insights into coding and algorithms. Experimental findings unveil efficacy and limitations. Beyond technical aspects, the report explores broader implications in document digitization and accessibility tools.

In essence, this transformative exploration at the intersection of AI, computer vision, and handwriting recognizes the enduring importance of handwritten communication amid digital transformations.

# LIST OF FIGURES AND ABBREVIATIONS



## Abbreviations

**EMNIST:** Extended Modified National Institute of Standards and Technology

**ML:** Machine Learning

**OpenCV:** Open Source Computer Vision

**CNN:** Convolutional Neural Network

**OCR:** Optical Character Recognition

**NN:** Neural Network

**HWR:** Handwriting Recognition

**HTR:** Handwritten Text Recognition

# TABLE OF CONTENTS

# Chapter 1: Introduction

## 1.1 Problem Definition:

The Handwriting Recognition project aims to address the challenge of automatically deciphering handwritten text using Convolutional Neural Networks (CNNs) and OpenCV. The fundamental problem at the core of this project is the inherent complexity and variability of handwritten script, which poses a significant obstacle to traditional methods of optical character recognition (OCR).

The primary problem can be defined as follows: Given the diverse nature of handwritten letters and the intricate variations in individual writing styles, there is a need for an intelligent system that can effectively learn and recognize these patterns. The project specifically focuses on English letters, utilizing the Extended Modified National Institute of Standards and Technology (EMNIST) dataset, which provides a diverse collection of labeled handwritten characters for training and evaluation.

Traditional OCR methods often struggle with the nuanced features and variations present in handwritten text. The inherent challenge lies in capturing the unique characteristics of each letter, including variations in size, slant, and spacing. This project seeks to overcome these challenges by leveraging the power of CNNs, which have proven effective in image-related tasks.

The problem encompasses the following key aspects:

Variability in Handwriting Styles: The diverse ways in which individuals write letters introduce a high degree of variability. This variability poses a challenge in developing a system that can generalize across different handwriting styles.

Feature Extraction: Identifying relevant features that distinguish one letter from another is a critical aspect of the problem. The project employs CNNs to automatically learn hierarchical representations, allowing the model to capture essential features during the training process.

Localization of Features: Efficiently localizing and prioritizing relevant regions within the input image is crucial for accurate recognition. The CNN architecture, with its convolutional and pooling layers, implicitly addresses the localization challenge by focusing on salient features.

Model Generalization: The developed model should not only perform well on the training set but also generalize effectively to new, unseen handwritten samples. Achieving robust generalization is a key objective in overcoming the challenges posed by the inherent variability in handwriting.

By defining and addressing these aspects, the project aims to contribute to the advancement of handwriting recognition technologies, making strides toward automating the interpretation of diverse and intricate handwritten scripts.

## 1.2 Project Overview

### 1.2.1 Dataset (EMNIST):

The foundation of the Handwriting Recognition project lies in the utilization of the EMNIST dataset. This dataset encompasses handwritten letters and digits, providing a diverse collection of characters. The focus of this project is specifically on the English letters subset of EMNIST. The dataset's richness allows for the training of a CNN capable of recognizing the intricate variations in handwritten script.

The EMNIST dataset comprises labeled images of individual characters, offering a valuable resource for training and evaluating the model. Each image is a 28x28 pixel grayscale representation, capturing the nuances of different writing styles. The dataset's diversity ensures that the model is exposed to a wide array of letter forms, enhancing its ability to generalize to various handwriting styles.

### 1.2.2 Model Selection:

The choice of a CNN as the underlying model is driven by its proven success in image-related tasks. CNNs are well-suited for capturing hierarchical features in images, making them ideal for handwriting recognition. The architecture of the selected model incorporates convolutional layers for feature extraction, max-pooling layers for dimensionality reduction, and dense layers for classification. The model is designed to learn and identify patterns that characterize handwritten letters.

The decision to use a CNN aligns with contemporary trends in deep learning, where convolutional architectures have demonstrated superior performance in image recognition

tasks. This choice ensures that the model can effectively learn and represent the intricate features present in handwritten characters.

### 1.2.3 Feature Extraction:

Feature extraction is a pivotal aspect of the model's architecture, where convolutional layers play a crucial role. These layers apply various filters (kernels) to the input image, extracting essential features that contribute to the identification of handwritten letters. The hierarchical nature of feature extraction in CNNs enables the model to learn low-level features, such as edges and curves, and progressively build more abstract representations.

The feature extraction process is foundational to the model's ability to discern the subtle details that differentiate one letter from another. Through this process, CNN learns to identify the unique characteristics of each handwritten character, allowing for robust recognition across diverse styles.

### 1.2.4 Localization:

Localization, in the context of this project, refers to the identification and isolation of relevant regions within the input image that contribute to the recognition task. In the CNN architecture, this occurs implicitly through the convolutional and pooling layers. Convolutional layers apply filters to local regions of the input, identifying features such as edges and patterns. Max-pooling layers then down-sample the spatial dimensions, focusing on the most salient features.

The combination of convolutional and pooling layers serves as an implicit localization mechanism, allowing the model to focus on relevant regions for character recognition. This hierarchical localization ensures that the model learns to prioritize essential details while discarding non-informative portions of the input.

### 1.2.5 Character Identification:

Character identification is the ultimate goal of the project, where the model assigns a label to the input image corresponding to the recognized letter. The dense layers in the CNN architecture facilitate this task by taking the high-level features extracted through convolution and pooling and mapping them to specific letter classes.

The softmax activation function in the final dense layer provides a probability distribution over the possible classes, enabling the model to make confident predictions. The training process involves adjusting the model's parameters to minimize the difference between predicted and actual labels, enhancing its ability to accurately identify handwritten characters.

### 1.2.6 Implementation:

The implementation phase translates the conceptual framework into Python and TensorFlow code, covering data loading, preprocessing, model definition, training, and evaluation. Key libraries like Pandas and Matplotlib support a structured approach for clarity and reproducibility. OpenCV is employed for image preprocessing, enhancing model robustness. Training involves iterative epochs for parameter fine-tuning. The project emphasizes the EMNIST dataset, CNN model selection, feature extraction, implicit localization, and character identification. This forms the foundation for detailed handwriting recognition exploration with CNNs and OpenCV.

## 1.3 Hardware Requirements:

The successful implementation of the Handwriting Recognition project requires a computer system with adequate hardware specifications to handle the computational demands of deep learning tasks:

### 1.3.1 Processor (CPU):

A multi-core processor, preferably with parallel computing capabilities, is essential. A quad-core or higher processor can significantly expedite the training of Convolutional Neural Networks (CNNs).

### 1.3.2 Graphics Processing Unit (GPU):

While not strictly mandatory, a dedicated GPU is highly recommended for accelerated training of deep learning models. GPUs, especially from NVIDIA (e.g., GeForce or Quadro series), excel in parallel processing and can significantly reduce training times.

### 1.3.3 RAM (Random Access Memory): 

A minimum of 16GB RAM is recommended to efficiently handle large datasets and neural network computations during training.

1.3.4 Storage: Sufficient storage capacity is necessary to store datasets, model weights, and other project-related files. A solid-state drive (SSD) is preferable for faster data access.

## 1.4 Software Requirements:

The Handwriting Recognition project relies on a stack of software tools and libraries to facilitate the development, training, and evaluation of the neural network model:

### 1.4.1 Python:

The project is implemented using the Python programming language, which provides a rich ecosystem of libraries for scientific computing, machine learning, and image processing.

### 1.4.2 TensorFlow:

TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind.TensorFlow accepts data in the form of multi-dimensional arrays of higher dimensions called tensors.

### 1.4.3 Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects, and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

### 1.4.4 Pandas:

Pandas is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

When working with tabular data, such as data stored in spreadsheets or databases, pandas is the right tool for it. Pandas will help to explore, clean, and process your data. In Pandas, a data table is called a DataFrame.

### 1.4.5 OpenCV:

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high-resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

### 1.4.6 Matplotlib:

Matplotlib is utilized for visualizing sample images, plotting training metrics, and displaying the results of the trained model.

### 1.4.7 Scikit-learn:

Scikit-learn is an open-source machine learning library that supports supervised and unsupervised learning it also provides various tools for model fitting, data processing, model selection, and model evaluation.

Ensuring compatibility and the correct versioning of these software components is crucial to the smooth execution of the Handwriting Recognition project. Proper installation and configuration of the specified software on the chosen hardware environment will contribute to the project's success.

# Chapter 2: Literature Review

Handwriting recognition has been a longstanding challenge in the field of pattern recognition and computer vision. The transition from traditional rule-based and statistical methods to the integration of deep learning techniques, specifically Convolutional Neural Networks (CNNs), has marked a significant advancement in the realm of handwriting recognition.

## 2.1 Traditional Methods and Challenges:

Historically, Optical Character Recognition (OCR) systems were primarily designed for printed text, exhibiting limitations when applied to handwritten documents. The inherent variability in individual writing styles, the lack of standardized fonts, and the presence of noise in handwritten text posed formidable challenges for traditional OCR approaches. Rule-based methods struggled to capture the diverse features of handwriting, while statistical models faced difficulties in generalizing across different writing styles.[1]

## 2.2 Emergence of Deep Learning:

The emergence of deep learning, particularly CNNs, revolutionized the landscape of image-related tasks, including handwriting recognition. CNN demonstrated an unprecedented ability to automatically learn hierarchical representations from raw pixel data, making them well-suited for the complex patterns present in handwritten characters.

## 2.3 CNNs for Feature Extraction:

One of the key strengths of CNNs lies in their inherent capacity for feature extraction. Traditional image processing methods often relied on handcrafted features, requiring intricate design and domain-specific knowledge. CNNs, on the other hand, automatically learn relevant features during the training process. Each layer of a CNN acts as a specialized filter, detecting low-level features such as edges and curves in early layers and progressively learning more complex, abstract features in deeper layers. This hierarchical feature extraction is particularly advantageous in capturing the intricate details of handwritten characters.

## 2.4 Localization through Convolution and Pooling:

CNNs also address the challenge of feature localization through the use of convolutional and pooling layers. Convolutional layers apply filters to local regions of the input image, enabling the model to focus on specific features. Subsequent pooling layers down-sample the spatial

dimensions, emphasizing the most salient features while discarding less informative regions. This implicit localization mechanism ensures that the model can effectively recognize relevant patterns within the input, contributing to improved accuracy in character identification.

## 2.5 Relevant Studies and Approaches:

### 2.5.1 EMNIST Dataset:

The choice of the Extended Modified National Institute of Standards and Technology (EMNIST) dataset for English letters is a common thread in many handwriting recognition studies. This dataset offers a comprehensive collection of labeled handwritten characters, allowing researchers to train and evaluate models on a diverse range of writing styles. The availability of such large and diverse datasets has played a crucial role in advancing the state-of-the-art in handwriting recognition.

### 2.5.2 Integration of OpenCV for Preprocessing:

Studies often highlight the importance of preprocessing techniques to enhance the robustness of handwriting recognition models. OpenCV, a versatile computer vision library, has been frequently integrated into projects to address challenges related to variations in size, orientation, and quality of handwritten images. Techniques such as resizing, normalization, and noise reduction contribute to the model's ability to generalize across diverse inputs.

### 2.5.3 Evaluation Metrics:

A consistent theme in the literature is the use of comprehensive evaluation metrics to assess the performance of handwriting recognition models. Metrics such as accuracy, precision, recall, and F1 score are commonly employed to provide a nuanced understanding of a model's strengths and weaknesses. Comparative studies often benchmark new approaches against existing methods, establishing a benchmark for performance in the field.

### 2.5.4 Gaps and Future Directions:

While the current literature showcases significant progress in handwriting recognition using CNNs and OpenCV, there remain areas for further exploration. Cross-language handwriting recognition, the integration of domain-specific knowledge, and the development of models robust to noisy or degraded handwritten text are identified as potential avenues for future

research. Additionally, there is a growing interest in exploring the interpretability of CNNs in the context of handwriting recognition, aiming to demystify the decision-making processes of these complex models.

## 2.6 Conclusion:

The literature review underscores the evolution of handwriting recognition from traditional methods to the adoption of CNNs and OpenCV. The shift towards data-driven approaches has significantly improved the ability to recognize diverse and complex patterns in handwritten characters. The EMNIST dataset, OpenCV preprocessing techniques, and comprehensive evaluation metrics have emerged as key elements in the success of recent studies. As the field continues to evolve, addressing the identified gaps and exploring new frontiers will contribute to the continued advancement of handwriting recognition technologies.

# Chapter 3: Methodology Implementation

The methodology for the Handwriting Recognition project involves a systematic approach to data preparation, model development, training, and evaluation. The integration of Convolutional Neural Networks (CNNs) and OpenCV serves as the core framework for recognizing handwritten characters. The methodology encompasses the following key steps:

## 3.1. Dataset Selection and Exploration:

### 3.1.1 EMNIST Dataset

The project utilizes the EMNIST dataset, specifically focusing on the subset containing English letters. The EMNIST dataset provides a diverse collection of labeled handwritten characters, offering a rich source for training and evaluating the CNN model. Each image in the dataset is a 28x28 pixel representation, capturing the intricacies of various writing styles.

### 3.1.2 Loading and Exploring the Dataset

The initial step involves loading the training and test datasets from the EMNIST collection. The Pandas library is employed for efficient data manipulation. The dimensions of both datasets are explored to gain insights into the volume of data available. Additionally, a preview of the dataset is displayed to understand the structure and format of the data.

## 3.2. Data Preprocessing:

### 3.2.1 Column Name Update

The datasets initially have column names as pixel values. These are updated to more meaningful names to enhance clarity during data manipulation and model interpretation.

### 3.2.2 Training and Validation Split

To facilitate model training and validation, the training dataset is split into training and validation sets using the train_test_split function from the scikit-learn library. This division ensures that the model is trained on a subset of the data while retaining a separate portion for validation, aiding in the assessment of model generalization.

### 3.2.3 Data Normalization

Pixel values of the images in both the training and validation sets are normalized to the range [0, 1]. Normalization ensures that the model converges faster during training and is less sensitive to variations in pixel intensity.

### 3.2.4 Reshaping Test Data

The test data is reshaped and normalized similarly to the training and validation sets to prepare it for model evaluation.

### 3.2.5 Data Visualization

Random samples from the training and test sets are visualized using Matplotlib to gain insights into the diversity of handwriting styles present in the dataset.

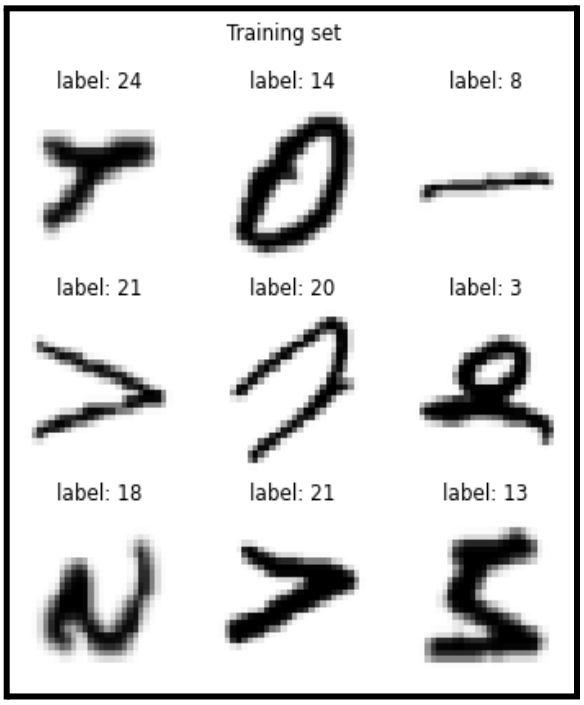


Fig.3.1 Training Set

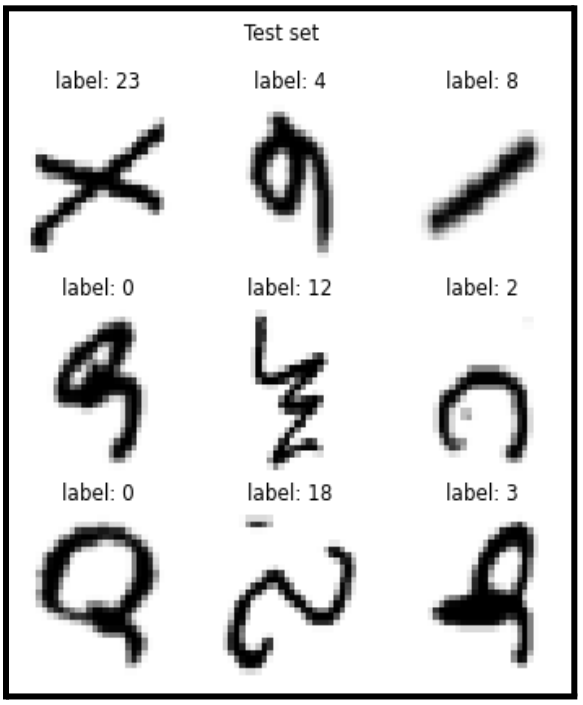


Fig.3.2 Train Set

## 3.3. Model Development:

### 3.3.1 CNN Architecture

The CNN model is designed to capture hierarchical features within the input images. The architecture consists of multiple convolutional layers for feature extraction, max-pooling layers for dimensionality reduction, and dense layers for classification. The final layer utilizes the softmax activation function to produce probability distributions over the classes.

### 3.3.2 Model Summary

A summary of the model's architecture is displayed to provide an overview of the number of parameters, layer configurations, and the flow of information through the network.[Fig.3.3]

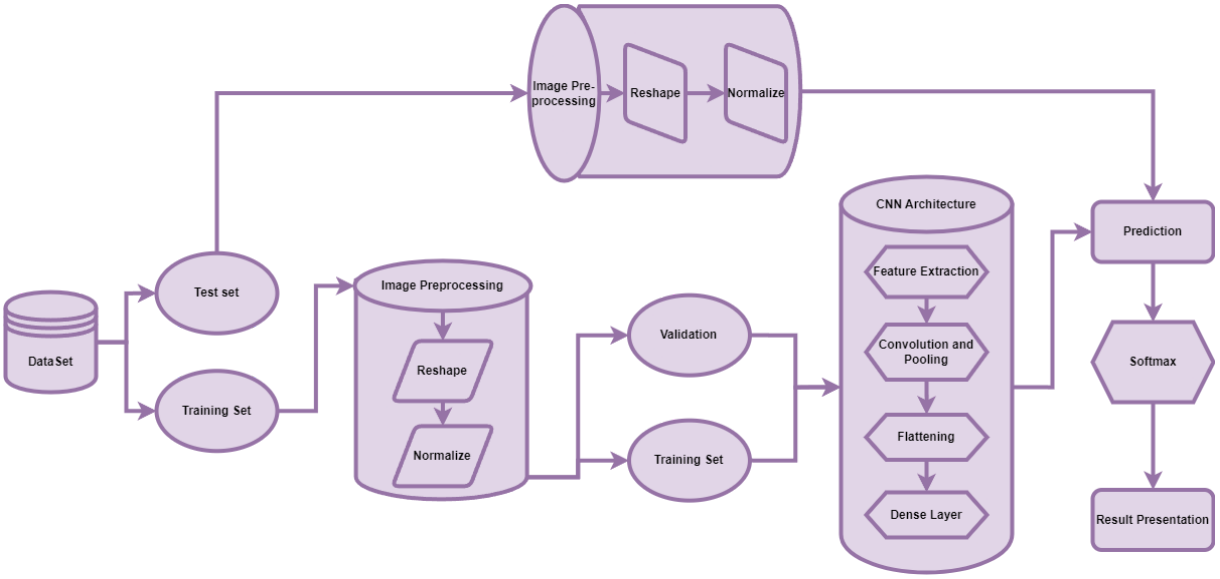


Fig.3.3 System Block Diagram

## 3.4. Model Training:

### 3.4.1 Training Parameters

The model is trained using the fit function from TensorFlow, specifying parameters such as the number of epochs, batch size, and validation data.

### 3.4.2 Visualizing Convolution and Pooling Filters

Intermediate representations of the data are visualized to understand what features the neural network learns at each convolution and pooling layer.

## 3.5. Model Evaluation:

### 3.5.1 Plotting Training Metrics

Training and validation accuracy and loss metrics are visualized over epochs to assess the model's performance.

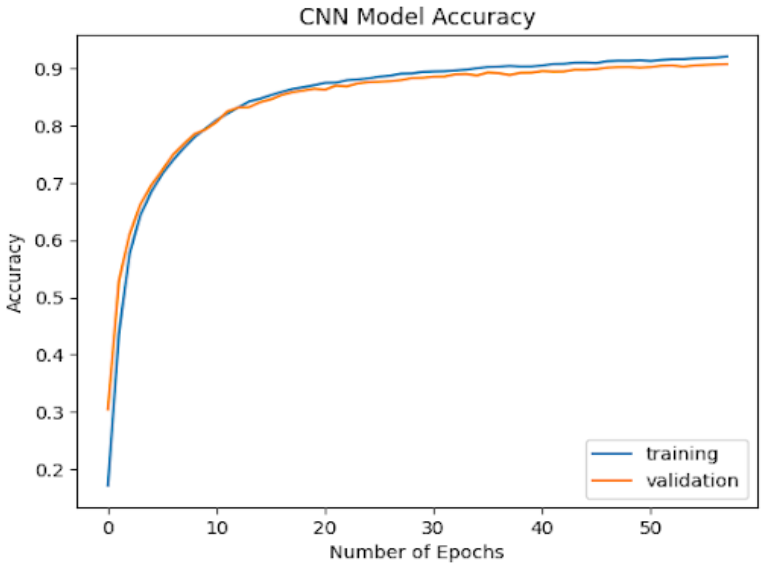


Fig.3.4 Training and validation Accuracy Trend

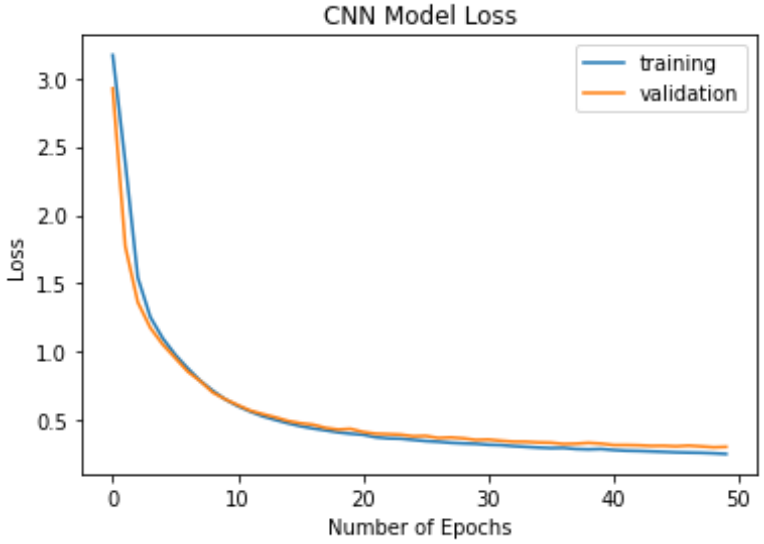


Fig.3.5.Training & Validation Loss Trend

### 3.5.2 Model Evaluation on Test Set

The trained model is evaluated on the previously unseen test set to assess its generalization performance. Predictions are generated for a subset of the test data.

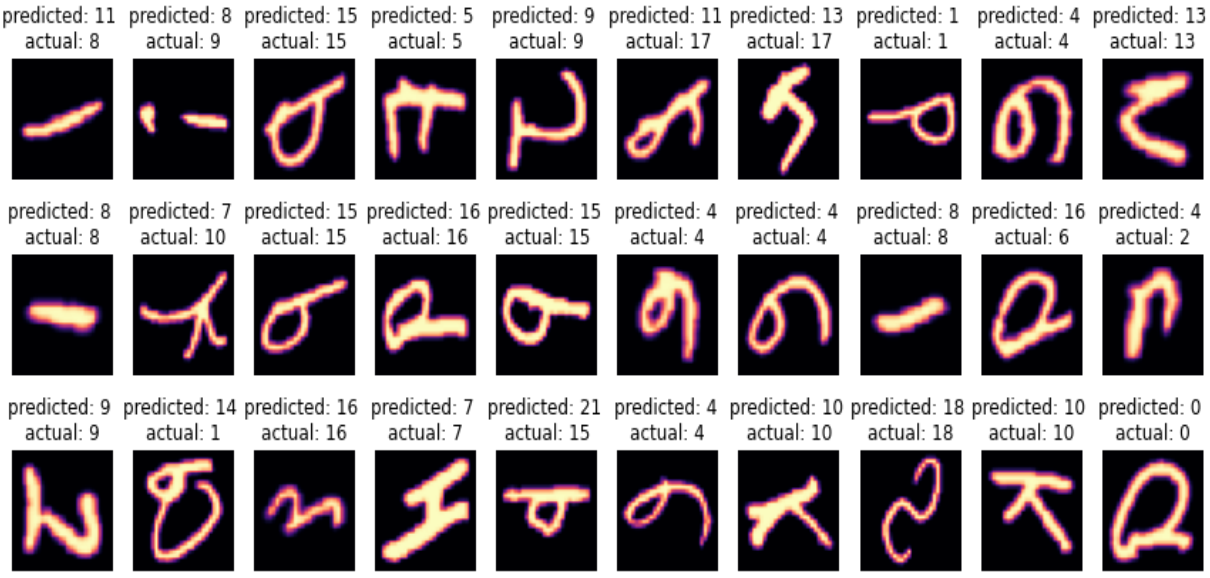


Fig.3.6 Model Performance on Test Data

## 3.6. Summary:

Python, TensorFlow, Numpy, Pandas, OpenCV, and Matplotlib are used throughout. These libraries collectively provide the tools and functionalities necessary for data manipulation, model development, training, and evaluation.

In summary, the methodology for Handwriting Recognition seamlessly integrates dataset exploration, data preprocessing, model development, training, and evaluation. Using CNNs and OpenCV, along with comprehensive visualization and analysis, we ensure a holistic approach to recognizing handwritten characters.

# Chapter 4: Analysis and Results

The success of the Handwriting Recognition project is contingent upon the model's ability to accurately identify and classify handwritten characters. This section delves into the performance analysis, presenting an evaluation of the model based on key metrics, visualizations, and insights gained during the training and testing phases.

## 4.1 Evaluation Metrics:

### 4.1.1. Accuracy

Accuracy is a fundamental metric representing the ratio of correctly predicted instances to the total number of instances. In the context of handwriting recognition, accuracy signifies the model's overall correctness in identifying letters.

### 4.1.2. Loss

Loss, calculated using the sparse categorical cross entropy function, measures the disparity between predicted and actual class labels. Lower loss values indicate better alignment between predicted and true labels.

### 4.1.3. Precision, Recall, and F1 Score

For a more nuanced evaluation, precision, recall, and F1 score are considered. Precision measures the accuracy of positive predictions, recall gauges the model's ability to capture all positive instances, and the F1 score provides a balance between precision and recall.

## 4.2 Training Metrics Visualization:

### 4.2.1. Accuracy Trends

The training and validation accuracy trends over epochs are depicted in the accuracy plot. This visual representation illustrates how well the model is learning from the training data and its ability to generalize to unseen validation data. [Fig.3.4]

### 4.2.2. Loss Trends

The training and validation loss trends over epochs are visualized in the loss plot. A decreasing loss indicates improved convergence and alignment between predicted and actual labels.[Fig.3.5]

The trends observed in the accuracy and loss plots provide insights into the model's learning behavior. Ideally, a model should exhibit increasing accuracy and decreasing loss over epochs. Any significant divergence between training and validation trends may indicate overfitting or underfitting.

## 4.3 Model Evaluation on Test Set:

### 4.3.1. Overall Performance Metrics

The final step involves evaluating the trained model on the test set to assess its overall performance. The accuracy, loss, and additional metrics are computed.

### 4.3.2. Individual Predictions

Random samples from the test set are selected, and the model's predictions are compared against the actual labels. This provides a qualitative understanding of the model's performance on individual instances.

## 4.4 Discussion of Results:

### 4.4.1. Accuracy and Loss Trends

The accuracy and loss trends depicted in the plots provide valuable insights into the model's performance. A consistent increase in accuracy and decrease in loss over epochs indicate effective learning and convergence. Deviations or plateaus in these trends may indicate areas for further investigation, such as fine-tuning model architecture or adjusting hyperparameters.

### 4.4.2. Convolution and Pooling Filter Visualization

The visualizations of convolution and pooling filters offer interpretability into the features learned by the model. Observing which parts of the letter light up in each layer provides an intuitive understanding of the network's focus areas. This interpretability is crucial for ensuring that the model is learning relevant and meaningful features.

### 4.4.3. Test Set Evaluation

The evaluation metrics on the test set, including accuracy, loss, precision, recall, and F1 score, collectively provide a comprehensive assessment of the model's performance. A high accuracy accompanied by low loss indicates the model's proficiency in generalizing to new, unseen data. Precision, recall, and F1 score offer additional insights into the model's ability to correctly classify positive instances.

### 4.4.4. Individual Predictions

The visual representation of individual predictions on random samples from the test set allows for a closer inspection of the model's behavior. Instances, where the predicted label aligns with the actual label, contribute to the overall accuracy, while any discrepancies highlight areas for improvement or further investigation.

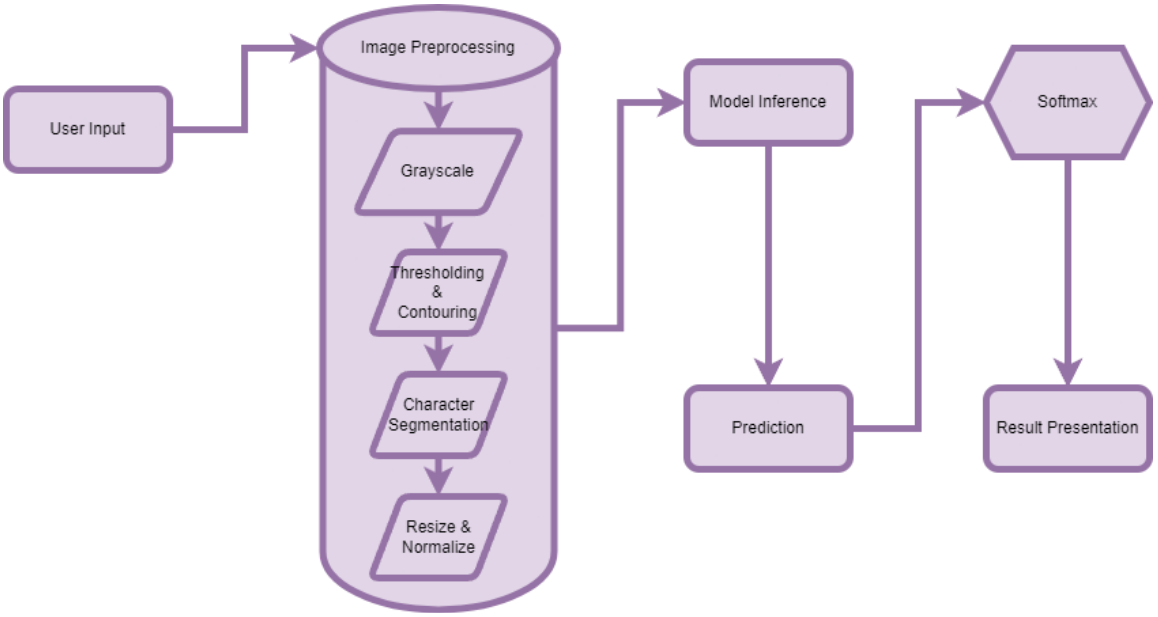


Fig.4.1 Individual Prediction Model

## 4.5 Conclusion:

The performance analysis and results showcase the efficacy of the Handwriting Recognition model based on CNN and OpenCV. The model demonstrates a high level of accuracy on the test set, supported by favorable trends in accuracy and loss during training. The interpretability provided through visualization aids in understanding the features learned by the model at different layers.

# Chapter 5: Conclusion and Remarks

## 5.1 Accomplishments and Key Findings

The accomplishments of this project are multifaceted. The utilization of the EMNIST dataset, carefully focusing on English letters, allowed for a diverse and extensive collection of labeled data. The choice of Convolutional Neural Networks (CNNs) as the underlying model was instrumental, showcasing the power of hierarchical feature extraction in deciphering the complexities of handwritten characters.

## 5.2 Remarks and Future Directions

The project has achieved its primary objectives, but there are opportunities for further enhancement and exploration:

### 5.2.1 Model Fine-Tuning

Fine-tuning hyperparameters and exploring different CNN architectures could improve model performance. Grid or random search techniques can be employed to systematically search the hyperparameter space.

### 5.2.2 Data Augmentation

Introducing data augmentation techniques during training, such as rotation, scaling, and translation, could enhance the model's robustness to variations in writing styles. This augmentation would simulate diverse scenarios, making the model more adept at handling real-world data.

### 5.2.3 Ensemble Methods

Exploring ensemble methods by combining predictions from multiple models could potentially boost overall accuracy and provide a more resilient solution. Ensemble methods often mitigate the risk of overfitting to specific patterns in the training data.

### 5.2.4 Real-Time Implementation

Extending the project to real-time implementation, where the model recognizes handwriting from live input, would open avenues for practical applications. This could find applications in digitizing handwritten notes or assisting individuals with impaired motor skills.

### 5.2.5. Multilingual Support

Expanding the model to recognize characters from multiple languages could broaden its utility. This would involve training the model on datasets containing characters from diverse linguistic backgrounds.

### 5.2.6 User Interface Integration

Integrating the model into a user-friendly interface could enhance accessibility. A well-designed interface could allow users to upload images of handwritten text for instant recognition.

## 5.3 Project Impact

The impact of the Handwriting Recognition project extends beyond its immediate scope. In an era where digital transformation is paramount, automating the recognition of handwritten text can streamline numerous processes. From document digitization to assisting individuals with disabilities, the applications are diverse and far-reaching.

The interpretability provided by visualizations not only ensures the model's trustworthiness but also opens avenues for collaboration with domain experts. The ability to understand which features the model deems important can facilitate domain-specific adjustments and improvements.

# Appendices

## Appendix A: Python Code for Handwriting Recognition Model

```
# Importing necessary libraries
import pandas as pd
import os
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import random
# Loading and exploring the training and test datasets
train = pd.read_csv('../input/emnist/emnist-letters-train.csv')
test = pd.read_csv('../input/emnist/emnist-letters-test.csv')
# Updating column names for both datasets
columns = ['labels']
for i in range(train.shape[1]-1):
    columns.append(i)
train.columns = columns
test.columns = columns

# Creating a training and validation split
x_train, x_val, y_train, y_val = train_test_split(train.drop(['labels'],
axis=1),train.labels - 1,train_size=0.8,test_size=0.2,random_state=42)

# Reshape and normalize test data
x_train = x_train / 255.0
x_val = x_val / 255.0

testX = test.values[:, 1:].reshape(test.shape[0], 28, 28,
1).astype('float32')
x_test = testX / 255.0
y_test = test['labels'].values - 1

# Viewing sample images of the training and test set
plt.figure(figsize=(6, 6))
plt.suptitle('Training set')
for i in train_samples:
    plt.subplot(3, 3, train_samples.index(i)+1)
    plt.imshow(x_train.iloc[i, :].values.reshape(28, 28), cmap='binary')
    plt.title(f'label: {y_train.iloc[i]}')
```

```
    plt.axis('off')

plt.figure(figsize=(6, 6))
plt.suptitle('Test set')
for i in test_samples:
    plt.subplot(3, 3, test_samples.index(i)+1)
    plt.imshow(x_val.iloc[i, :].values.reshape(28, 28), cmap='binary')
    plt.title(f'label: {y_val.iloc[i]}')
    plt.axis('off')

# Creating the CNN Model
def train_model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Reshape((28, 28, 1), input_shape=(784,)),
        tf.keras.layers.Conv2D(8, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Conv2D(16, (3, 3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2, 2),
        tf.keras.layers.Conv2D(24, (3, 3), activation='relu'),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(len(classes), activation=tf.nn.softmax)
    ])
    model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
    return model

# Training the neural network
history = model.fit(x_train, y_train, epochs=65, validation_data=(x_val,
y_val), batch_size=4096, verbose=1)

# Visualizing the convolution and pooling filters
successive_outputs = [layer.output for layer in model.layers[1:]]
visualization_model = tf.keras.models.Model(inputs=model.input,
outputs=successive_outputs)

number = random.sample(range(0, len(x_val)), 1)[0]
successive_feature_maps = visualization_model.predict(x_val.iloc[number,
:].to_numpy().reshape(1, 784))

layer_names = [layer.name for layer in model.layers[1:]]

for layer_name, feature_map in zip(layer_names, successive_feature_maps):
```

```
    if len(feature_map.shape) == 4:
        n_features = feature_map.shape[-1]
        size = feature_map.shape[1]
        display_grid = np.zeros((size, size * n_features))
        for i in range(n_features):
            x = feature_map[0, :, :, i]
            x -= x.mean()
            x /= x.std()
            x *= 64
            x += 128
            x = np.clip(x, 0, 255).astype('uint8')
            display_grid[:, i * size: (i + 1) * size] = x
        scale = 10. / n_features
        plt.figure(figsize=(scale * n_features, scale))
        plt.title(layer_name)
        plt.grid(False)
        plt.imshow(display_grid, aspect='auto', cmap='viridis')

# Plotting the losses and accuracy of the training and validation set
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('CNN Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Number of Epochs')
plt.legend(['Training', 'Validation'], loc='lower right')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('CNN Model Loss')
plt.ylabel('Loss')
plt.xlabel('Number of Epochs')
plt.legend(['Training', 'Validation'], loc='upper right')
plt.show()

# Neural network evaluation
evaluate = model.evaluate(x_test, y_test)
predictions = model.predict(x_test)

test_samples = random.sample(range(0, len(x_test)), 30)

plt.figure(figsize=(15, 6))
for i in test_samples:
```

```
    plt.subplot(3, 10, test_samples.index(i) + 1)
    plt.imshow(x_test[i].reshape(28, 28), cmap='magma')
    plt.title(f'Predicted: {np.argmax(predictions[i])} \nActual:
{y_test[i]}')
    plt.axis('off')
```

## Appendix B: Dataset Information

EMNIST (Extended Modified National Institute of Standards and Technology) is an extension of the MNIST dataset, designed for character recognition tasks. It includes various subsets, such as EMNIST ByClass, ByMerge, Balanced, Letters, and Digits. For this project, the EMNIST Letters dataset was used, containing 814,255 characters labeled with 814,255 integer values representing the class labels.

## Appendix C: Model Architecture Summary

The model summary provides a concise overview of the neural network architecture, including the types and shapes of layers, the number of parameters, and the flow of data through the network.

```
# Displaying the model summary
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
reshape (Reshape)           (None, 28, 28, 1)         0
conv2d (Conv2D)             (None, 26, 26, 8)         80
max_pooling2d (MaxPooling2D)(None, 13, 13, 8)         0
conv2d_1 (Conv2D)           (None, 11, 11, 16)        1168
max_pooling2d_1 (MaxPooling2D)  (None, 5, 5, 16)          0
conv2d_2 (Conv2D)           (None, 3, 3, 24)          3480
flatten (Flatten)           (None, 216)               0
dense (Dense)               (None, 128)               27776
dense_1 (Dense)             (None, 26)                3354
=================================================================
Total params: 35858 (140.07 KB)
Trainable params: 35858 (140.07 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

# References

**[1]** T. Wang, D. Wu, A. Coates, A. Ng. ”End-to-End Text Recognition with Convolutional Neural Networks” ICPR 2012.

**[2]** J. Sueiras, V. Ruiz, A. Sanchez, and J.F. Velez, “Offline Continuous Handwriting Recognition using Sequence to Sequence Neural Networks”, Neurocomputing, Vol. 289, pp. 119-128, 2018.

**[3]** Q. Vo, S. Kim, H. Yang, and G. Lee, “Text Line Segmentation using a Fully Convolutional Network in Handwritten Document Images”, IET Image Processing, Vol. 12, No. 3, pp. 438-446, 2018.

**[4]** Lisa Yan. Recognizing Handwritten Characters. CS 231N Final Research Paper

**[5]** Andrew Smith, Hong-Phuong Tran, Eric Dimla “Offline Handwritten Text Recognition using Convolutional Recurrent Neural Network” International Conference on Advanced Computing and Applications 2019 (ACOMP)

**[6]** G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “Emnist: an extension of mnist to handwritten letters,” arXiv preprint arXiv:1702.05373, 2017

**[7]** G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “Emnist: an extension of mnist to and written letters,” arXiv preprint arXiv:1702.05373, 2017.

**[8]** Impedovo S (1993) Introduction. In: Impedovo S (ed) Fundamentals of handwriting recognition. Springer, London, pp 1–10

# Personal Details

**Name:** Hrishikesh Bhatt

**Enrollment no:** 211B139

**Branch:** Computer Science & Engineering

**EmailId:**hrishi1402@gmail.com

**University:** Jaypee University of Engineering & Technology

**Phone No:** 9024611775

**Name:** Pankaj Kumar Kushwaha

**Enrollment no:** 211B201

**Branch:** Computer Science & Engineering

**EmailId:**211b201@juetguna.in

**University:** Jaypee University of Engineering & Technology

**Phone No:** 7489676883

**Name:** Shivam Tripathi

**Enrollment no:** 211B293

**Branch:** Computer Science & Engineering

**EmailId:**shivam1705of@gmail.com

**University:** Jaypee University of Engineering & Technology

**Phone No:** 8224966690