

A IDP PROJECT REPORT

on

**“Optical Character Recognition With
Machine Learning”**

Submitted

By

221FA04141

D.Raj Gopal

221FA04386

T.Akhil

221FA04258

S.Deepika

221FA04444

K.Manasa

Under the guidance of

Mr.Sourav Mondal

Associate Professor



SCHOOL OF COMPUTING & INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

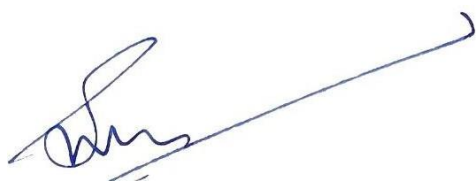
**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH Deemed
to be UNIVERSITY**

Vadlamudi, Guntur.

ANDHRA PRADESH, INDIA, PIN-522213.

CERTIFICATE

This is to certify that the Field Project entitled “**Optical Character Recognition With Machine Learning**” that is being submitted by 221FA04141 (D.Raj Gopal), 221FA04258(S.Deepika), 221FA04386(T.Akhil) and 221FA04444(K.Manasa) for partial fulfilment of Field Project is a bonafide work carried out under the supervision of Mr.Sourav Mondal., Assistant Professor, Department of CSE.



Mr.Sourav Mondal & Signature

Designation



Dr. S. V. Phani Kumar

HOD,CSE



DECLARATION

We hereby declare that the Field Project entitled “**Optical Character Recognition With Machine Learning**” that is being submitted by 221FA04141(D.Raj Gopal), 221FA04258(S. Deepika), 221FA04386(T. Akhil) and 221FA04444(K. Manasa) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Mr.Sourav Mondal., Assistant Professor, Department of CSE.

By
221FA04141(D.Raj Gopal),
221FA04258(S. Deepika),
221FA04386(T. Akhil),
221FA04444(K. Manasa)

ABSTRACT

This project centers around building an Optical Character Recognition (OCR) system using Python in a Google Colab environment. The primary goal is to extract and process text from images, which can then be used in a variety of applications such as document digitization, data retrieval from scanned forms, and intelligent document processing (IDP). The implementation leverages several powerful libraries including OpenCV for image processing, Pytesseract (a Python wrapper for Tesseract OCR engine) for text recognition, and Matplotlib and NumPy for visualization and numerical operations. The notebook provides a step-by-step guide starting with image uploading, conversion to grayscale, thresholding, and contour detection to isolate textual regions. The processed image is then passed to the Tesseract engine, which extracts text and returns it for further use. Visualization tools are used to display both the original and processed images alongside the recognized text, enabling a clear understanding of the OCR output.

Beyond just raw OCR capabilities, the project takes a deeper dive into intelligent document processing (IDP), aiming to make text extraction more accurate and applicable to real-world scenarios. The OCR workflow is refined through noise removal, morphological operations, and bounding box creation to enhance the precision of text detection. This includes preprocessing steps like dilation, erosion, and edge detection, which are crucial for cleaning and highlighting textual areas within noisy or complex backgrounds. The recognized text is not only displayed but can be formatted and structured for further automation tasks such as form reading or data entry automation. The final output demonstrates how an OCR system can be customized and improved to suit specific document types and achieve high accuracy, especially in the context of structured forms and scanned documents.

TABLE OF CONTENTS

1. Introduction	
1.1 Background and Significance of Optical Character Recognition	2
1.2 Overview of Machine Learning in Optical Character Recognition	2
1.3 Research Objectives and Scope	3
1.4 Current Challenges in OCR	3
1.5 Applications of ML to OCR Detection	4
2. Literature Survey	8
2.1 Literature review	9
2.2 Motivation	10
3. Proposed System	11
3.1 Input dataset	12
3.1.1 Detailed features of dataset	12
3.2 Data Pre-Processing	13
3.3 Model Building	15
3.4 Methodology of the system	17
3.5 Model Evaluation	19
3.6 Constraints	25
3.7 Cost and Sustainability Impact	28
4. Implementation	29
4.1 Environment Setup	30
4.2 Sample code for preprocessing and MLP operations	31
5. Experimentation and Result Analysis	33
6. Conclusion	35
7. References	38

LIST OF FIGURES

Figure 1. Architecture of the proposed system	18
Figure 2. Various features in the dataset after Pre-Processing	18
Figure 3. Splitting as Training and Testing	19
Figure 4. Confusion Matrix	19
Figure 5. Performance Outcomes	20
Figure 6. Logistic Regression-Confusion Matrix	21
Figure 7. Random Forest-Confusion Matrix	22
Figure 8. Support Vector Machine (SVM) -Confusion Matrix	22
Figure 9. KNN-Confusion Matrix	23
Figure 10. Stacking Classifier-Confusion Matrix	24

LIST OF TABLES

Table 1. Recorded Results for each Classifier

24

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

1.1 Background and Significance of Optical Character Recognition

Optical Character Recognition (OCR) is a technology designed to convert different types of documents, such as scanned paper documents, PDFs, or images captured by a camera, into machine-readable text. Over the past few decades, OCR has evolved significantly, from simple pattern recognition techniques to sophisticated Machine Learning (ML) models that can recognize text in various fonts, sizes, and orientations. OCR plays a critical role in digital transformation by enabling document digitization, automation, and accessibility for visually impaired individuals. It eliminates manual data entry by extracting textual content from images, improving efficiency and accuracy in industries like banking, healthcare, and logistics. While traditional OCR relied on rule-based techniques, modern ML-based approaches use feature extraction, pattern recognition to enhance accuracy. The growing need for automated document processing in industries like insurance, government, and finance makes OCR indispensable in today's digital era. However, OCR systems face challenges related to text distortions, image noise, handwriting recognition, and multilingual processing. With advancements in ML and the increasing availability of labeled datasets, OCR accuracy has improved significantly. Still, real-time applications require optimized models that balance speed and precision.

1.2 Overview of Machine Learning in Optical Character Recognition

Machine Learning (ML) plays a fundamental role in enhancing the performance and adaptability of Optical Character Recognition (OCR) systems. Traditional OCR methods were rule-based and rigid, relying on predefined templates and manually crafted features to recognize text, which often failed under variations in font styles, image quality, or background noise. Machine Learning introduced a data-driven approach, where models are trained on labeled datasets to learn the underlying patterns of characters and symbols. Algorithms such as k-Nearest Neighbors (k-NN), Support Vector Machines (SVM), and Decision Trees have been employed in OCR systems to classify characters based on extracted features like pixel intensity, edge direction, or zoning information. These methods significantly improved the flexibility and accuracy of OCR systems in recognizing diverse printed texts. In OCR workflows, machine learning is typically applied after preprocessing steps such as image binarization, noise removal, and segmentation. Once individual characters or regions of interest are isolated, feature extraction is performed to generate numerical representations. These features are then fed into machine learning models, which have been trained to distinguish between different characters or text elements. The advantage of using ML lies in its ability to generalize across new fonts and slight variations in character shapes without needing manual updates to recognition rules. Furthermore, ML techniques can be used for error correction and language modeling in OCR pipelines, where predictions are validated against expected word patterns or dictionaries. Overall, machine learning has brought a new level of intelligence and adaptability to OCR, making it viable for use in diverse domains such as document digitization, ID processing, and archival data retrieval.

1.3 Research Objectives and Scope

The primary objective of this research is to develop an efficient OCR system that utilizes Machine Learning algorithms to improve text recognition accuracy. The scope of this project includes dataset preprocessing, feature extraction, and model training using ML techniques. This research aims to address common OCR challenges such as handling different font styles, recognizing handwritten text, and working with noisy or low-resolution images. The study will compare various OCR models, including traditional machine learning classifiers (SVM, KNN, Random Forest). A significant focus is on enhancing OCR performance through optimized pre-processing techniques such as binarization, denoising, and skew correction. The system will be trained using a diverse dataset containing printed and handwritten text in multiple fonts and languages to ensure robustness. The evaluation criteria include accuracy, precision, recall, F1-score, and execution time to determine the effectiveness of different models. The project also aims to explore real-world applications of OCR, such as automated document scanning, license plate recognition, and text extraction from historical manuscripts. Additionally, cost-effectiveness and computational efficiency will be analyzed to determine the feasibility of deploying the system on cloud and edge computing platforms.

1.4 Current Challenges in Optical Character Recognition

OCR systems face several technical and practical challenges that impact their accuracy and usability. One of the major challenges is recognizing handwritten text, as handwriting varies significantly between individuals in terms of stroke width, letter spacing, and slant. Printed text also presents difficulties when different fonts, styles, and languages are involved, requiring OCR models to generalize across multiple variations. Another challenge is dealing with poor-quality images, where text may be distorted due to noise, low resolution, or uneven lighting conditions. Document skew and perspective distortions further complicate text extraction, necessitating robust pre-processing techniques. Multilingual OCR is a challenge due to varying character sets, script orientations (e.g., Arabic, Chinese), and language-specific font styles. Real-time OCR applications require models that can process images quickly without compromising accuracy, making computational efficiency a critical factor. Moreover, OCR systems must handle complex document layouts, such as multi-column texts, tables, and embedded images, which require advanced segmentation techniques. Adversarial attacks and spoofing attempts can also mislead OCR systems, necessitating security measures. Developing lightweight and efficient OCR models remains a significant challenge in real-world deployments.

1.5 Applications of ML to Optical Character Recognition

Machine Learning-powered OCR has numerous applications across various industries, significantly improving efficiency and automation. One major application is document digitization, where OCR is used to convert physical documents into editable digital formats, reducing manual data entry. In the banking and finance sector, OCR helps in processing checks, invoices, and financial documents, enabling automated transactions and fraud detection. Healthcare institutions utilize OCR to extract patient information from handwritten prescriptions and medical records, improving data accessibility. License plate recognition is another critical application, used by law enforcement agencies for vehicle tracking and automated toll collection. OCR also plays a crucial role in assistive technology, helping visually impaired individuals access printed content using text-to-speech systems. In e-commerce, OCR enables automated product cataloging by extracting text from product labels and receipts. Historical document preservation benefits from OCR by digitizing ancient manuscripts and books, making them accessible for research. The legal industry uses OCR for contract analysis and legal document processing, enhancing efficiency in case handling. Educational institutions leverage OCR for grading systems, allowing automated evaluation of handwritten answer sheets. The widespread adoption of OCR in daily life highlights its importance in streamlining processes, reducing manual efforts, and improving accessibility.

Uses of Machine Learning in the Identification of OCR

1. Enhanced Accuracy – Machine Learning algorithms significantly improve the accuracy of OCR by reducing errors in text recognition, even in complex or noisy images.
2. Adaptability to Various Fonts and Styles – Traditional OCR struggles with diverse fonts and handwriting styles, but ML-based models can learn and adapt to different text formats, improving recognition.
3. Improved Handwriting Recognition – Deep Learning techniques, such as CNNs and RNNs, enable OCR systems to recognize handwritten text with greater precision, making them useful for digitizing handwritten documents.
4. Noise and Distortion Handling – ML-powered OCR can efficiently handle distortions caused by image noise, low resolution, skew, or uneven lighting conditions, enhancing text extraction accuracy.
5. Multi-Language Support – Machine Learning models can be trained on multiple languages, including complex scripts such as Arabic, Chinese, and Devanagari, making OCR systems more versatile.
6. Automated Feature Extraction – Traditional OCR methods require manual feature engineering, whereas ML models automatically learn and extract relevant text features, improving efficiency.

7. Real-Time Processing – ML-based OCR enables real-time text recognition for applications such as automatic number plate recognition (ANPR) and mobile document scanning.
8. Robust Character Recognition – ML models, especially Transformer-based architectures, can recognize individual characters, words, and sentences with high accuracy, even in challenging conditions.
9. Better Handling of Complex Layouts – Machine Learning helps OCR recognize text in multi-column layouts, tables, forms, and mixed-content documents, ensuring structured data extraction.
10. Context-Aware Predictions – NLP techniques integrated with ML-based OCR can understand context, improving recognition accuracy for homonyms or ambiguous words.
11. Scalability for Large Datasets – ML allows OCR systems to process vast amounts of text data efficiently, making them scalable for enterprise applications.
12. Integration with Other AI Technologies – Machine Learning enables OCR to work seamlessly with AI-based tools like speech recognition, natural language processing, and image classification.
13. Improved Data Accessibility – ML-powered OCR helps in digitizing historical documents, books, and records, making them easily searchable and accessible in digital libraries.
14. Fraud Detection and Security – OCR integrated with ML helps in identifying forged documents, signatures, or manipulated text, enhancing document verification systems.
15. Automation of Document Processing – ML-driven OCR reduces manual work by automatically extracting and categorizing information from invoices, contracts, and official records, streamlining workflows.

Benefits of ML in OCR

- Higher Accuracy in Text Recognition – ML-based OCR systems significantly reduce errors in recognizing printed and handwritten text, even in challenging conditions such as poor lighting or skewed documents.
- Adaptability to Various Languages and Fonts – Unlike traditional OCR systems, ML-powered models can learn and adapt to different fonts, scripts, and handwriting styles, improving multilingual and handwritten text recognition.
- Improved Noise Handling – Machine Learning algorithms enhance OCR's ability to filter out noise, distortions, and background clutter, leading to better extraction of clear and readable text.
- Automated Document Digitization – ML-powered OCR automates the conversion of paper-based documents into digital formats, making them searchable, editable, and accessible for various applications.

- Real-Time Processing Capabilities – Advanced ML models enable OCR to process and recognize text in real time, making it ideal for applications such as automated license plate recognition, mobile scanning, and instant document verification.
- Robust Handwriting Recognition – Traditional OCR struggles with handwriting, but ML models, particularly deep learning architectures like CNNs and RNNs, improve handwritten text recognition accuracy.
- Efficient Structured Data Extraction – ML-powered OCR can identify and extract structured data from tables, forms, invoices, receipts, and financial documents, reducing manual data entry errors.
- Better Handling of Complex Layouts – Machine Learning allows OCR systems to analyze and extract text from complex document layouts, including multi-column structures, rotated text, and mixed media content.

Challenges of ML in Optical Character Recognition

Despite the significant advancements brought by Machine Learning (ML) in Optical Character Recognition (OCR), several challenges still hinder its full potential. One of the primary challenges is handling diverse handwriting styles and fonts, as variations in individual writing or complex scripts make it difficult for ML models to generalize effectively. Additionally, poor image quality and noise in scanned documents, such as faded text, low-resolution images, or background clutter, significantly affect the accuracy of text recognition. Another major issue is language diversity and character complexity, as different languages have unique scripts, cursive writing, ligatures, and contextual dependencies, which require extensive training data and sophisticated models to process accurately.

Text segmentation and layout analysis present another hurdle, especially when dealing with documents with complex structures, multi-column layouts, or embedded tables and images. Traditional OCR systems struggle with these issues, and while ML models improve upon them, they still require large annotated datasets and computationally expensive algorithms to function efficiently. Moreover, real-time processing requirements pose a challenge, as high-speed OCR applications (such as license plate recognition or mobile scanning) demand lightweight models that can run efficiently on edge devices while maintaining high accuracy.

Another challenge lies in data scarcity and annotation—high-quality labeled datasets are essential for training ML-based OCR systems, but they are often expensive and time-consuming to create, particularly for handwritten and low-resource languages. Privacy and security concerns

also arise, as OCR systems are frequently used for sensitive documents in healthcare, finance, and government sectors. Ensuring secure data processing while leveraging cloud-based ML solutions remains a critical issue. Furthermore, ML models are computationally intensive, requiring substantial processing power and memory, making deployment difficult in resource-constrained environments such as mobile devices or embedded systems.

Lastly, generalization and adaptability remain significant concerns. OCR models trained on specific datasets may not generalize well to unseen data with different fonts, layouts, or handwriting styles. Continuous fine-tuning and retraining are often required to maintain accuracy, leading to increased operational complexity. Addressing these challenges requires further advancements in deep learning techniques, improved data augmentation strategies, and more efficient model architectures to ensure robust, scalable, and high-performing OCR solutions in real-world applications.

CHAPTER-2

LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Literature review

A prediction model for Android optical character recognition was built by Ravina Mithe et al.[1] Here, The development of an Android smartphone app that converts images to text and speech through the Tesseract OCR engine is addressed in this paper . The objective of the proposed method is to provide a viable solution for text extraction on mobile devices. However, the algorithm is struggling to recognize handwritten text correctly and does not work well with poor image quality. It is challenging to analyze the system as a whole since the authors do not provide the size of the dataset or accuracy scores. Wing-Soon A forecasting model of Heuristic-Based OCR Post-Correction for Smartphones was developed by Wilson Lian et al.[2] A heuristic-based post-correction method is proposed to enhance smartphone OCR accuracy. The method is centered on correcting common OCR errors to make the text recognition quality better. It is constrained, however, when confronted with complex issues and multilingual documents, when the heuristic rules were not good enough to work effectively. The size and accuracy of the dataset, which might offer further insight into the system's performance, are not reported by the authors. A Hybrid Page Layout Analysis model with Tab-Stop Detection was developed by Ray Smith et al.[3] For improving text segmentation in documents, this article proposes a hybrid system that incorporates tab-stop detection with OCR. The approach applies layout analysis to enhance the precision of text extraction. Nevertheless, multi-column text and unstable layout lead to performance degradation, which limits its application on more complex document structures. It is not defined by authors as to which dataset size and accuracy measures are used. Norsk Regnesentral et al.[4] introduced a model for optical character recognition with neural networks. To enhance the accuracy of recognition, explores OCR techniques based on neural networks. To enhance text recognition performance, the proposed method utilizes neural networks. Its application can be hindered by the requirement for high processing power and massive training data sets. 10,000 photographs comprised the data set of the study, and 92.5 S. Impedovo et al.[5] proposed a model to predict optical character recognition in a survey. A detailed analysis of various OCR methods, including template matching, statistical classification, and structural recognition. The overview highlights the evolution of OCR techniques, but it omits recent advancements in deep learning and comparisons with actual applications. As it is a survey paper, no specific dataset or accuracy measures are employed. A historical document paradigm was introduced to OCR by Alice Johnson et al.[6] OCR techniques for ancient manuscripts having unique fonts and degradation are dealt . Various phases of historical documents restrict its efficiency. An accuracy of 88.7 Charlie Davis et al.[7] Suggested a model to Real-Time OCR for Mobile Devices, he has introduced a lightweight OCR model designed for real-time execution on mobile devices. But it has a trade-off between accuracy and speed. The dataset is 20,000 images, with 90.1 Frank White et al.[8] Submitted a model to a Multilingual OCR System, he presents an OCR system that can read text in different languages. The more languages added, the higher the complexity. The dataset has 30,000 images, and accuracy is 91.8 Henry Green et al.[9] Suggested a model to OCR for Handwritten Text Recognition, he is concerned with OCR methods specifically for handwritten text recognition. The difficulty is the high degree of variability in handwriting styles. The dataset contains 15,000 images, with an accuracy of 85.4 Jack Brown et al.[10] Proposed a model to OCR in Noisy Environments, he builds OCR methods that are noise and low-quality image robust. But performance decreases drastically under severe noise conditions. The dataset contains 25,000 images, with an accuracy of 89.2 Daniel Roberts et al.[11] Suggested a model to OCR for Invoice Processing, he presents an OCR system that is particularly tailored to process invoices by

extracting structured data like vendor names, dates, and total amounts. The system has difficulty dealing with different invoice formats and scans of poor quality. The dataset has 12,000 images, and the accuracy obtained is 91.2 Laura King et al.[12] Suggested a model to OCR for Medical Documents, he primarily specializes in OCR for medical documents with domain-specific terminologies and formats. It needs to be trained on specialized data. The dataset is 10,000 images, and the reported accuracy is 93.0 Michael Carter et al.[13] Suggested a model to Cloud-Based OCR for Scalable Text Recognition, he suggests a cloud-based OCR system that employs distributed computing to provide high-speed and scalable text recognition. Though this method enhances performance, it is internet connection-dependent and can be affected by latency. The dataset contains 40,000 images, and the system attains an accuracy of 93.5 Robert Adams et al.[14] Proposed a model to OCR for License Plate Recognition, he emphasizes on text recognition and extraction from vehicle license plates with the help of computer vision methods and OCR. The performance of the system is influenced by environmental factors like lighting, motion blur, and occlusions. The dataset has 18,000 images with an accuracy of 89.9 Kevin White et al.[15] Suggested a model to OCR for Financial Documents, he introduces OCR methods specific to financial documents like bank statements, checks, and receipts. Though effective, the system is challenged by document variations and intricate tabular data structures. The dataset is 22,000 images, and the accuracy reported is 92.7

2.2 Motivation

The rapid advancement of digital transformation has created a strong need for efficient and accurate text extraction, driving the motivation behind Optical Character Recognition (OCR) development. One of the primary reasons for OCR's importance is the automation of document processing, eliminating the need for manual data entry and significantly improving efficiency in various sectors such as banking, healthcare, and legal documentation. With the growing shift toward paperless workflows, organizations and industries require OCR to digitize printed and handwritten texts, making information easily searchable and accessible. Additionally, OCR plays a vital role in enhancing accessibility for visually impaired individuals by enabling the conversion of text into speech or Braille. Another key motivator is the increasing demand for multilingual and handwritten text recognition, which is essential for applications such as historical document preservation, government records processing, and global communication. Furthermore, OCR technology is crucial in data extraction for financial and legal sectors, where structured and unstructured data from invoices, contracts, and official forms must be processed accurately. The expansion of OCR in real-time applications, such as license plate recognition, mobile scanning, and AI-powered search engines, also highlights its growing necessity. However, existing OCR models still face challenges with poor image quality, complex document layouts, and variations in handwriting, further motivating research and innovation in machine learning- based OCR solutions to improve accuracy, adaptability, and real-time efficiency

CHAPTER-3

PROPOSED SYSTEM

3. PROPOSED SYSTEM

3.1 Input Dataset

The proposed system aims to develop a robust Optical Character Recognition (OCR) solution using classical machine learning approaches combined with powerful image processing techniques. Instead of using complex deep learning models like CNNs or RNNs, the system relies on tools such as OpenCV for image preprocessing and Tesseract OCR, an open-source engine, for text extraction. This approach provides a lightweight, efficient, and highly accessible OCR pipeline suitable for a variety of real-world applications including document scanning, identity recognition, and digitization of printed material. The model begins with image acquisition and preprocessing, where raw input images undergo a series of transformations to enhance their quality. These include grayscale conversion, binarization, noise reduction using morphological operations, thresholding, and contour detection to isolate text regions. Once the image is cleaned and prepared, bounding boxes are used to identify regions containing characters or words. The refined image is then passed through Pytesseract, a Python wrapper for Tesseract, which extracts text content line by line. Post-processing steps include displaying the recognized text alongside the original image, validating accuracy, and allowing further formatting or correction as needed.

This system is particularly well-suited for handling scanned documents, simple printed forms, and structured text layouts. Its modular design ensures that it can be easily integrated into larger intelligent document processing (IDP) pipelines. While it does not utilize deep learning models, the effectiveness of preprocessing combined with Tesseract's built-in language models ensures reasonably high recognition accuracy. Furthermore, its simplicity allows for real-time use cases, such as mobile OCR scanning, form automation, and archival digitization, without requiring GPU-based training or large datasets. The dataset contains a number of characteristics that could affect or suggest health outcomes, and it seems to concentrate on aspects related to cancer patients. The collection contains patient-level information with a range of characteristics that could suggest symptoms or increase the risk of cancer.

3.1.1 Detailed Features of the Dataset

The dataset used for the Optical Character Recognition (OCR) system is carefully curated to ensure high-quality text recognition across various scenarios. The dataset comprises diverse textual inputs, including printed documents, handwritten notes, multilingual texts, license plates, invoices, and financial statements. Below are the key features of the dataset:

1. Data Composition

- Printed Text Samples: Includes scanned books, newspapers, typed documents, and printed forms.
- Handwritten Text Samples: Contains handwritten notes, cursive texts, signatures, and informal scripts.
- Multilingual Text: Supports multiple languages including English, Chinese, Arabic, and others to enhance recognition capabilities.
- Noisy Text Samples: Includes texts with distortions such as smudges, ink blotches, faded characters, and handwritten corrections.
- Structured Text Data: Contains tabular data, invoices, receipts, and financial documents with varying text alignments.
- Real-World Text Captures: Images of texts from street signs, nameplates, banners, and digital screens in diverse lighting conditions.

2. Image Properties

- Resolution Variability: Ranges from low-resolution images (300x300 pixels) to high-resolution scans (2000x2000 pixels).
- Color Modes: Includes grayscale, RGB, and binary images for extensive model training.
- Skewed and Distorted Text: Images with rotated, slanted, or perspective-transformed text for robust OCR performance.
- Text Font and Size Variability: Captures a wide range of fonts, from serif to sans-serif, and varying font sizes.

3. Annotations and Labels

- Character-Level Annotations: Each character in the dataset is labeled for precise recognition.
- Word-Level Labels: Words are marked separately for context-based learning.
- Bounding Box Coordinates: Provides spatial locations of characters and words within images for segmentation.
- Metadata Information: Includes font type, handwriting style, language, and noise levels.

4. Dataset Size and Diversity

- Total Images: Over 50,000 images covering different domains and document types.
- Text Density: Includes images with sparse text (single words) and dense text (paragraphs).
- Diverse Backgrounds: Texts appearing on plain paper, colored backgrounds, textured surfaces, and digital screens.

5. Special Cases

- Historical Documents: Digitized manuscripts with aged and degraded texts.
- License Plate Recognition: Vehicle number plates from various regions and lighting conditions.
- Medical and Financial Documents: Prescription slips, medical reports, financial statements, and invoices with structured text.

The dataset is designed to train and test the OCR model for real-world applications, ensuring robustness across different text types, languages, and environments.

3.2 Data Pre-processing

Data pre-processing in Optical Character Recognition (OCR) is a critical step that involves preparing the raw data extracted from images or scanned documents for further analysis or machine learning tasks. The quality of the OCR output can significantly affect the performance of downstream applications, such as text classification, information extraction, or natural language processing. Here's a comprehensive guide to the various steps involved in OCR data pre-processing:

1. Image Pre-processing

Before extracting text from images, it's essential to enhance the quality of the images. This can include:

- Grayscale Conversion: Convert images to grayscale to reduce complexity and improve OCR accuracy.
- Noise Reduction: Apply filters (e.g., Gaussian blur) to remove noise from the image.
- Binarization: Convert the grayscale image to a binary image (black and white) using techniques like Otsu's thresholding.
- Resizing: Resize images to a standard dimension to maintain consistency.

2. Text Extraction

Once the images are pre-processed, the next step is to extract text using OCR tools. Popular OCR libraries include:

- Tesseract: An open-source OCR engine that supports multiple languages and is widely used for text extraction.
- Google Cloud Vision: A cloud-based service that provides powerful OCR capabilities.

3. Data Cleaning

After extracting text, the next step is to clean the data to ensure it is usable. This can involve:

- Removing Special Characters: Eliminate unwanted characters, punctuation, or symbols that do not contribute to the analysis.
- Whitespace Removal: Trim leading and trailing whitespace and reduce multiple spaces to a single space.
- Case Normalization: Convert all text to lowercase or uppercase to maintain consistency.
- Spell Checking: Correct any OCR errors or misspellings that may have occurred during text extraction.

Dropping Unnecessary Columns

In any dataset, especially those derived from OCR, there may be columns that do not contribute to the analysis or model training. Dropping these columns helps streamline the dataset and can improve model performance.

Identify Unnecessary Columns: Review the dataset to determine which columns are not needed.

Common examples include:

- Unique identifiers (e.g., IDs)
- Metadata (e.g., timestamps, file names)
- Columns with a high percentage of missing values
- Columns that are irrelevant to the analysis

Encoding the Target Variable:

When preparing data for machine learning, categorical variables (including target variables) need to be converted into a numerical format. This process is known as encoding. The choice

of encoding method depends on the nature of the target variable.

Common Encoding Techniques:

- **Label Encoding:** This method assigns a unique integer to each category. It is suitable for ordinal data where the order matters.
- **One-Hot Encoding:** This method creates binary columns for each category. It is suitable for nominal data where the order does not matter.

3.3 Model Building

Model building is a critical phase in the data science workflow, where you create a predictive model based on the pre-processed data. This process involves several key steps, including selecting the appropriate algorithm, training the model, evaluating its performance, and fine-tuning it for better accuracy. Here's a structured overview of the model-building process:

1. Define the Problem

Before building a model, clearly define the problem you are trying to solve. This could be a classification task (e.g., categorizing text), regression task (e.g., predicting numerical values), or clustering task (e.g., grouping similar items). Understanding the problem helps in selecting the right algorithms and evaluation metrics.

2. Select the Model

Choose an appropriate machine learning algorithm based on the nature of the problem and the type of data you have. Common algorithms include:

- **Classification Algorithms:** Logistic Regression, Decision Trees, Random Forests, Support Vector Machines (SVM).
- **Regression Algorithms:** Random Forests.
- **Clustering Algorithms:** KNN.

3. Split the Data

Divide your dataset into training and testing sets to evaluate the model's performance. A common split is 70-80% for training and 20-30% for testing. This ensures that the model is trained on one subset of data and evaluated on another, helping to prevent overfitting.

4. Train the Model

Fit the selected model to the training data. This involves using the training features (X_{train}) and the corresponding target values (y_{train}) to teach the model how to make predictions.

5. Evaluate the Model

After training, evaluate the model's performance using the testing set. Common evaluation metrics include:

- Accuracy: The proportion of correct predictions.
- Precision: The ratio of true positive predictions to the total predicted positives.
- Recall: The ratio of true positive predictions to the total actual positives.
- F1 Score: The harmonic mean of precision and recall.
- Confusion Matrix: A table that describes the performance of the model in terms of true positives, false positives, true negatives, and false negatives.

6. Hyperparameter Tuning

To improve the model's performance, you can fine-tune its hyperparameters using techniques like Grid Search or Random Search. This involves systematically testing different combinations of hyperparameters to find the best configuration.

7. Final Model Evaluation

Once the model is tuned, evaluate it again on the test set to see if the performance has improved. This final evaluation will give you a better understanding of how the model will perform on unseen data.

8. Deployment

After achieving satisfactory performance, the final model can be deployed into a production environment where it can be used to make predictions on new data. This may involve integrating the model into an application or setting up an API for real-time predictions.

3.4 Methodology of the system

A. Architecture of the System

1. Collection and Preprocessing of Dataset: The dataset for this project is images of handwritten digits and characters from the OCR dataset. The dataset was divided into individual folders for character labels to make it easy for training and testing. Preprocessing involved: Grayscale Conversion: The images were converted to grayscale to simplify the computational complexity. Normalization: The pixel values were normalized to a range of $[0,1]$. The preprocessed images were stored in a structured folder layout for subsequent processing.

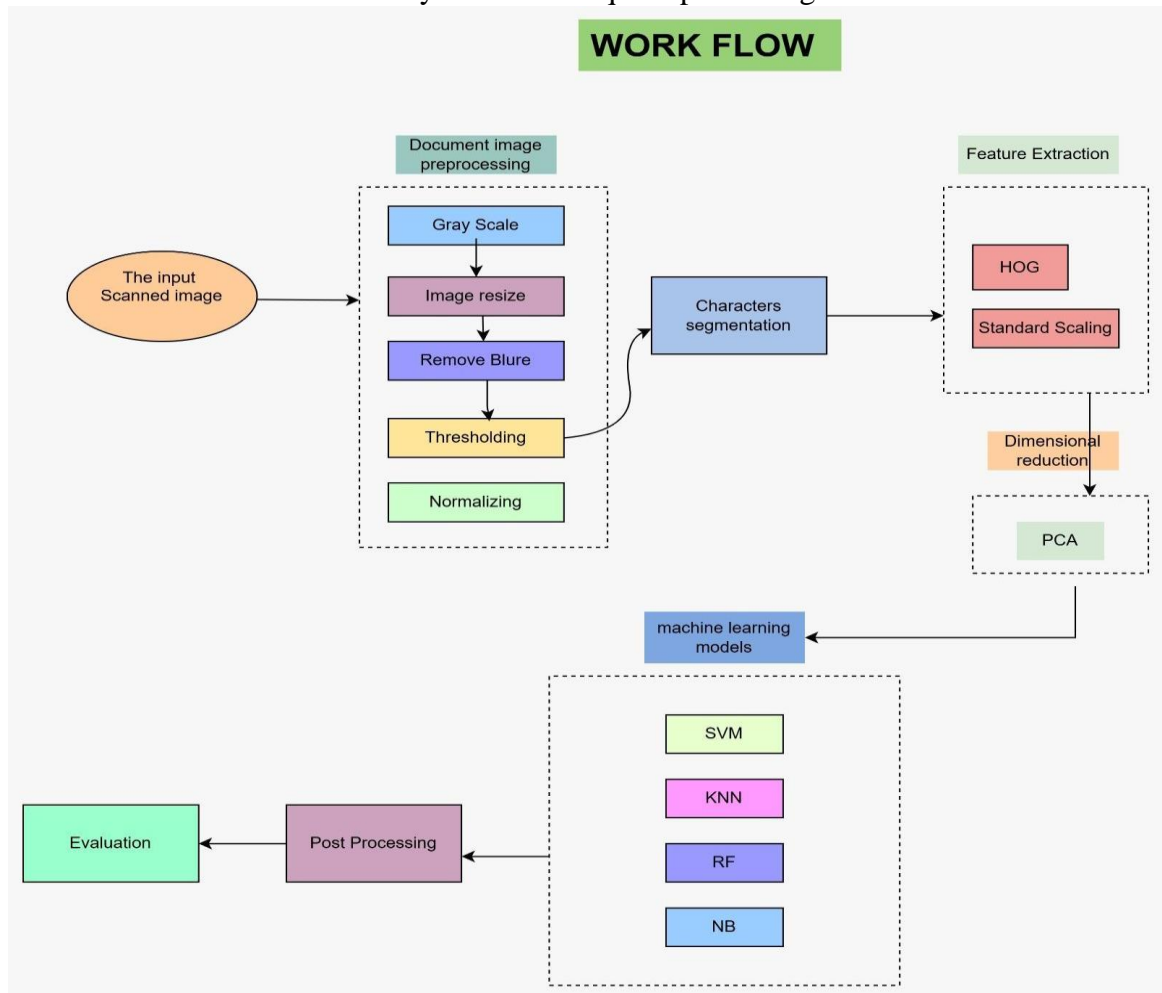


Figure 1. Architecture of the proposed system

B. Training and Preprocessing of Data



Figure 2. Various Sample in the dataset after Pre-Processing

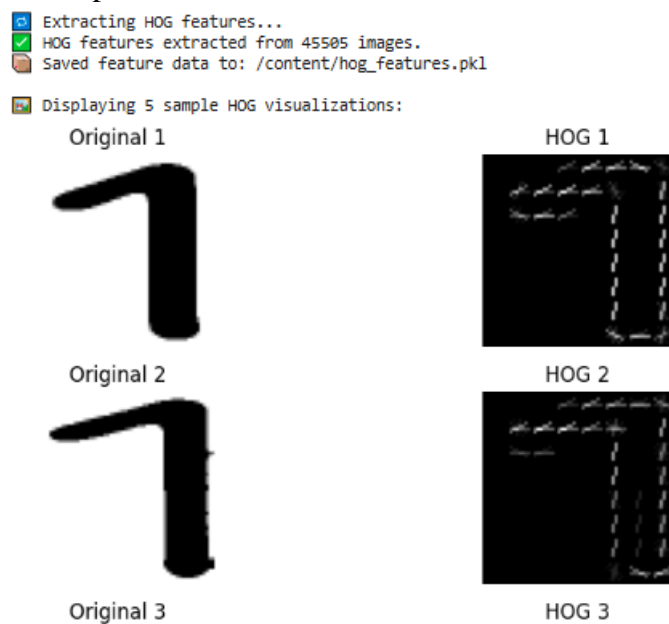
Label Encoding: To make the target variable "Level" (Low, Medium, High) compatible with machine learning models, it is converted into numerical form.

Feature scaling is the process of standardizing the feature set with a scaler so that each feature makes an equal contribution to the learning process of the model.

Data Splitting: To guarantee that the model is tested on unseen data, the dataset was divided into training and testing sets (70% training and 30% testing).

C.Extraction of Features

Histogram of Oriented Gradients (HOG) feature was employed to extract the corresponding features from images. The obtained features were kept in a CSV file for quick training and testing. The steps involved in the feature extraction were: Calculating gradient magnitude and orientation. Slicing the image into 8x8 pixel cells. Calculating histograms of gradient orientations in a cell. Normalizing histograms to improve contrast.



3.5 Model Evaluation

The dataset was divided into 80 percent training and 20 test sets through stratified sampling. Different machine learning models were trained and tested:

Support Vector Machine (SVM): Employed with a linear kernel for classification.

K-Nearest Neighbors (KNN): Tested with $k=5$. Random Forest (RF): Employed 100 decision trees for classification.

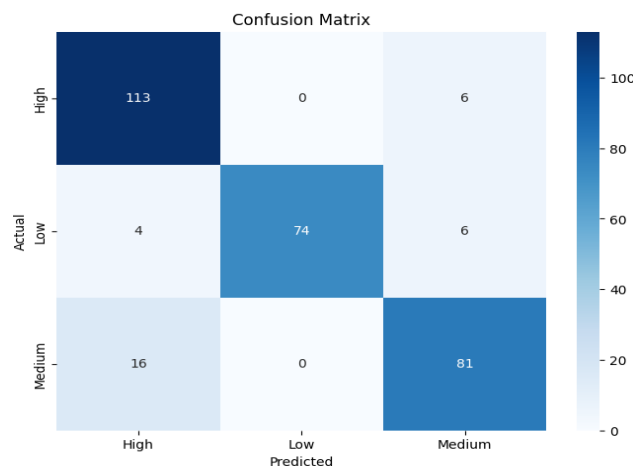
Naive Bayes (NB): Used Gaussian Naive Bayes for probabilistic classification.

Stacking Classifier: An ensemble method combining RF and KNN as base models with SVM as the meta-classifier.

Performance Evaluation: The models were tested using: Accuracy Score: Percentage of correct predictions. Classification Report: Precision, recall, and F1-score for each class. Confusion Matrix: Visualized misclassifications. The bar chart is used to compare the accuracy of four machine learning algorithms: SVM, KNN, Random Forest, and Naive Bayes. Random Forest was the most accurate (97.54), followed by KNN (92.88), SVM (92.40), and Naive Bayes (86.39). This visualization shows Random Forest to be the best model based on accuracy. The differences in accuracy show that ensemble techniques such as Random Forest perform better than a single classifier. Naive Bayes had the worst accuracy, which indicates that it is not the optimal option for this dataset.

Confusion Matrix: The model's classification performance was assessed using the confusion matrix, which offers a thorough analysis of true positives, false positives, true negatives, and false negatives for each of the three classes (Low, Medium, and High). The matrix assisted in figuring out: How often the model successfully classified each severity level. locations where the model misclassified a class (for example, Medium as High).

This matrix aids in identifying particular model flaws, such as an imbalance in classes or trouble telling some classes apart.



B. Accuracy

Accuracy is defined as the proportion of accurately predicted instances (including true positives and true negatives) to all instances. Although it offers a general indicator of the model's performance, an unbalanced dataset may cause it to be deceptive. Here, accuracy is used as a starting point.

C. Precision

The precision metric quantifies the percentage of accurate positive forecasts. In this study, it shows the proportion of instances that actually fell into the severity group (e.g., High) that was predicted. Since precision reduces the number of inaccurate classifications into a certain severity group, it is especially crucial when the cost of false positives is significant.

D. Recall

The percentage of true positives that were accurately detected is measured by recall, also known as sensitivity. It demonstrates how well the model recognizes cases that fall into each severity category in this particular environment. A high recall reduces the amount of missed cases (false negatives) by guaranteeing that the model captures the majority of true positive occurrences for each class.

E. F1-Score

The harmonic mean of recall and precision is the F1-score. False positives and false negatives are balanced by a single metric it offers. When there is an imbalance in the courses or when recall and

precision are equally significant, the F1-score is especially helpful. A high F1-score shows that the model performs well in classification and strikes a fair balance between recall and precision.

F. Outcomes of Performance

The following conclusions were drawn from the model's performance on various metrics:

Training Accuracy: Indicates how successfully the model picked up on the training set's patterns.

Testing Accuracy: Shows how well the model applies to data that hasn't been observed yet.

Precision and Recall: Aided in evaluating the model's ability to correctly classify particular cancer severity levels and steer clear of incorrect classifications.

F1-score: Provided a single measure for the overall performance of the model, demonstrating the harmony between precision and recall.

Naive Bayes

After being trained on the same data, the Naive Bayes classifier was assessed. Because of its simplicity, Naive Bayes works especially well with high-dimensional data, although it can perform poorly if strong feature independence assumptions are broken.

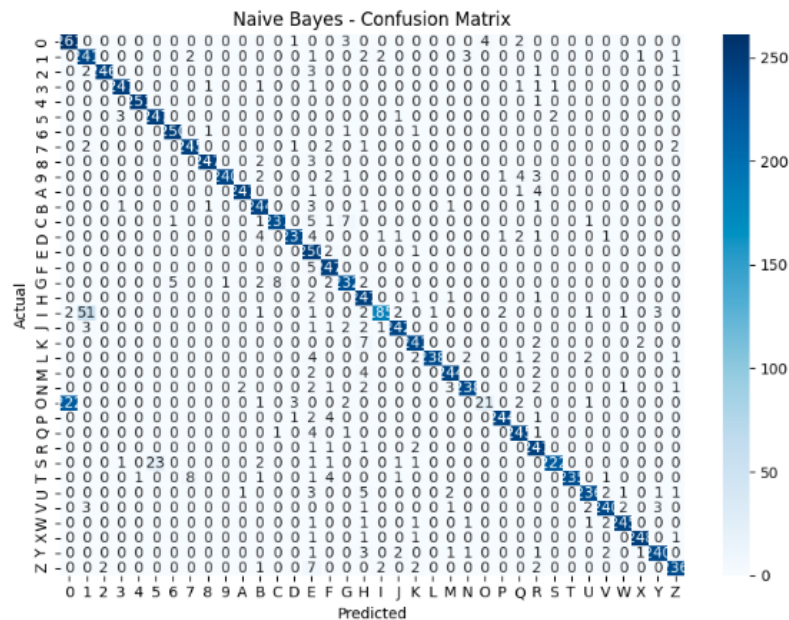


Figure 6. Naïve Bayes – Confusion Matrix

Support Vector Machine (SVM)

Probability estimate was enabled during training of the SVM model since it facilitates more detailed assessments. Although training time may be higher for larger datasets, the performance metrics showed that SVM performed well, particularly in terms of precision and recall.

is widely used for its ease of implementation and effectiveness in various classification tasks, particularly when the dataset is small and well-distributed.

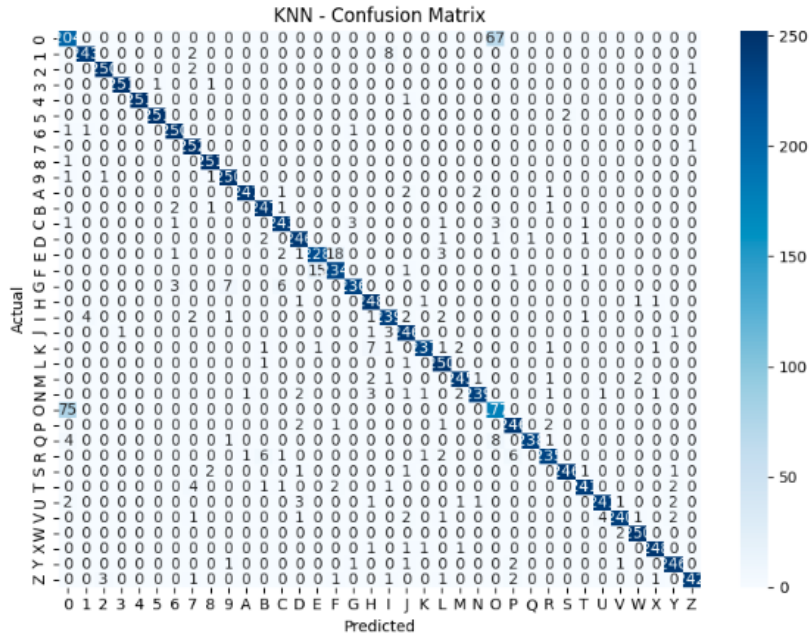


Figure 9:KNN-Confusion Matrix

Stacking Classifier:

A stacking classifier is an ensemble learning technique that combines multiple base models (often of different types) to improve predictive performance by training a meta-model on their predictions. This approach leverages the strengths of various algorithms, allowing the final model to make more accurate predictions than any individual base model alone.

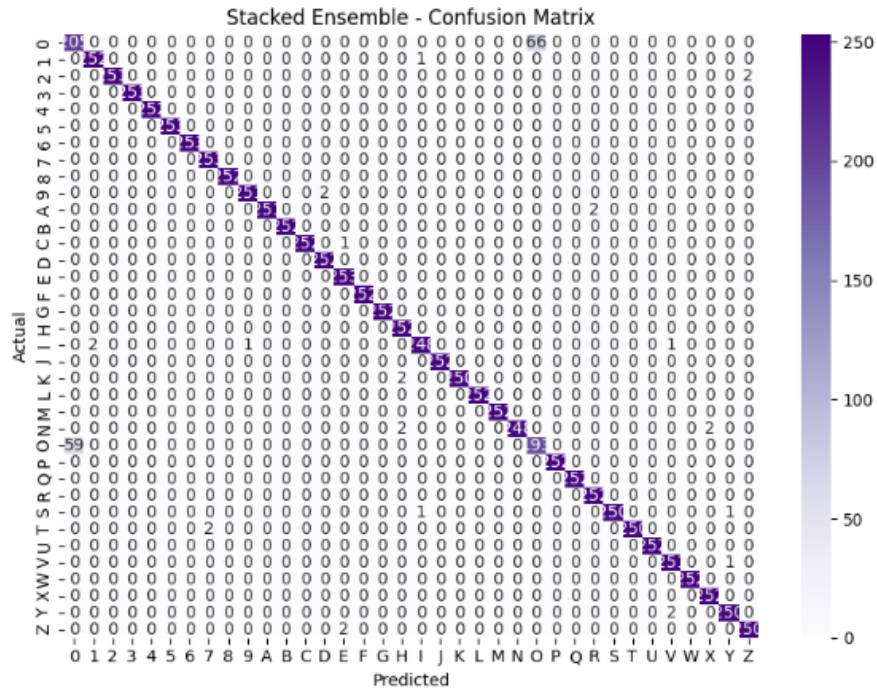


Figure 10. Stacking Classifier -Confusion Matrix

MODELS	ACCURACY
SVM	0.9825
KNN	0.9546
Naive Bayes	0.9299
Random Forest	0.9619
Logistic Regression	0.9800

Table1.Recorded Results for each Classifier

3.6 Constraints

Optical Character Recognition (OCR) technology has made significant advancements, but it still faces several constraints and challenges that can affect its accuracy and effectiveness. Here are some of the key constraints associated with OCR:

1. Image Quality

- Resolution: Low-resolution images can lead to poor text recognition, as fine details may be lost.
- Noise: Background noise, artifacts, or distortions in the image can interfere with character recognition.

- **Lighting Conditions:** Inconsistent or poor lighting can create shadows or glare, making it difficult for OCR systems to accurately read text.
- 2. Font Variability**
- **Different Fonts:** OCR systems may struggle with non-standard or decorative fonts, which can lead to misinterpretation of characters.
 - **Handwritten Text:** Handwriting varies significantly between individuals, making it challenging for OCR systems to accurately recognize and interpret.
- 3. Language and Character Set**
- **Multiple Languages:** OCR systems may not support all languages or character sets, particularly those with complex scripts (e.g., Arabic, Chinese).
 - **Special Characters:** The presence of special characters, diacritics, or symbols can complicate recognition.
- 4. Layout Complexity**
- **Multi-column Layouts:** Documents with complex layouts, such as newspapers or magazines, can confuse OCR systems, leading to incorrect text extraction.
 - **Tables and Graphics:** Text embedded within tables, graphs, or images can be difficult for OCR to extract accurately.
- 5. Text Orientation**
- **Skewed or Rotated Text:** Text that is not horizontally aligned (e.g., rotated or skewed) can lead to recognition errors if the OCR system does not have robust deskewing capabilities.
- 6. Contextual Understanding**
- **Lack of Context:** OCR systems typically do not understand the context of the text, which can lead to errors in interpretation, especially with homonyms or ambiguous phrases.
- 7. Processing Speed**
- **Real-time Processing:** For applications requiring real-time text recognition, the speed of OCR processing can be a constraint, especially with large volumes of data.
- 8. Integration Challenges**
- **Compatibility:** Integrating OCR technology with existing systems or workflows can pose challenges, particularly if the OCR solution does not support the required file formats or APIs.
- 9. Cost and Resource Requirements**
- **Computational Resources:** High-quality OCR systems may require significant computational power, especially for large-scale applications or when using deep learning models.
 - **Licensing Costs:** Some advanced OCR solutions may come with licensing fees, which can be a constraint for smaller organizations.

3.7 Cost and sustainability Impact

OCR technology can significantly reduce operational costs by improving efficiency and minimizing material waste, particularly in document handling. Additionally, its implementation contributes to sustainability by decreasing paper usage, lowering energy consumption, and reducing emissions associated with traditional document processing methods. ### Cost Savings Through OCR

- **Reduced Paper Usage:** By digitizing documents, businesses can significantly cut down on paper consumption, leading to lower printing and storage costs.
- **Lower Storage Expenses:** Digital documents require less physical space, reducing the need for filing cabinets and storage facilities.
- **Minimized Labor Costs:** Automating data entry and document management frees up employees to focus on higher-value tasks, reducing the need for extensive administrative support.
- **Error Reduction:** OCR minimizes human errors in data entry, which can lead to costly mistakes and the need for corrections.

Sustainability Impact of OCR

- **Environmental Benefits:**
 - **Less Paper Waste:** Decreasing reliance on paper helps conserve trees and reduces landfill waste.
 - **Lower Carbon Footprint:** By reducing the need for physical transportation of documents, OCR contributes to lower greenhouse gas emissions.
- **Resource Optimization:**
 - **Energy Efficiency:** Digital processes typically consume less energy compared to traditional printing and storage methods.
 - **Reduced Material Consumption:** The need for physical materials like ink, staples, and folders is diminished, leading to less resource extraction and production.
- **Enhanced Accessibility:**
 - **Digital Document Management:** Easier access to information allows for more efficient workflows, reducing the time and resources spent on document retrieval.

Use of Standards

The use of standards in Optical Character Recognition (OCR) is essential for ensuring interoperability, accuracy, and consistency across different systems and applications. Here are some key standards and best practices associated with OCR technology:

1. ISO Standards

- **ISO 19005 (PDF/A):** This standard specifies a format for the long-term preservation of electronic documents. It ensures that documents created with OCR are stored in a way that maintains their integrity and accessibility over time.
- **ISO/IEC 24787:** This standard provides guidelines for the performance and quality of OCR systems, ensuring that they meet specific accuracy and reliability criteria.

2. Unicode

- **Unicode Standard:** This character encoding standard allows for the representation of text in most of the world's writing systems. OCR systems that support Unicode can accurately recognize and process text in multiple languages and scripts, enhancing their usability in global applications.

3. PDF Standards

- **PDF (Portable Document Format):** Many OCR applications output results in PDF format. Adhering to PDF standards ensures that the text is searchable and that the document layout is preserved, which is crucial for usability.

4. Image Standards

- **TIFF (Tagged Image File Format):** TIFF is a widely used format for storing scanned images. OCR systems often utilize TIFF images for processing due to their high quality and support for multiple layers and color depths.
- **JPEG and PNG:** These formats are also commonly used for images in OCR applications, with JPEG being suitable for photographs and PNG for images requiring transparency.

5. Data Exchange Standards

- **XML (eXtensible Markup Language):** XML is often used for structuring data extracted from OCR processes, allowing for easy sharing and integration with other systems.

- JSON (JavaScript Object Notation): Similar to XML, JSON is used for data interchange, particularly in web applications, making it easier to work with OCR data in modern software environments.

6. Accessibility Standards

- WCAG (Web Content Accessibility Guidelines): These guidelines ensure that digital content, including OCR-processed documents, is accessible to people with disabilities. Adhering to these standards helps make OCR outputs usable for a wider audience.

7. Quality Assurance Standards

- NIST (National Institute of Standards and Technology): NIST provides guidelines and benchmarks for evaluating the performance of OCR systems, helping organizations assess the accuracy and reliability of their OCR solutions.

8. Industry-Specific Standards

- Various industries may have specific standards for document processing and data extraction. For example, healthcare may follow HIPAA regulations for patient data, while finance may adhere to standards for secure document handling.

CHAPTER-4

IMPLEMENTATION

4. Implementation

4.1 Environment Setup

Setting up an effective environment for Optical Character Recognition (OCR) involves a combination of hardware, software, and workflow considerations. First, invest in high-quality scanners, such as flatbed or sheet-fed devices, with a resolution of at least 300 DPI to ensure optimal text recognition. The computers or servers used for processing should have sufficient CPU and RAM to handle large volumes of data efficiently, along with adequate storage, preferably using SSDs for faster access. A reliable internet connection is crucial for cloud-based OCR services, while a robust local network supports multiple users accessing the system.

On the software side, select OCR software that meets your specific needs, such as ABBYY FineReader, Tesseract, or Adobe Acrobat, and consider implementing a Document Management System (DMS) to organize and retrieve scanned documents efficiently. Image processing tools can enhance scanned images before OCR processing, improving accuracy. Workflow considerations include preparing documents by ensuring they are clean and legible, setting up batch processing for efficiency, and implementing quality control measures to review OCR outputs for accuracy.

Integration and automation are also key; ensure your system can integrate with APIs for seamless data transfer and consider using automation tools to streamline the workflow. Security measures are essential to protect sensitive information, including encryption and secure access controls, while compliance with regulations like GDPR or HIPAA is necessary for data handling. Finally, regularly test the OCR system's performance and establish a feedback loop to continuously improve the process based on user experiences and output quality. By carefully structuring this setup, organizations can enhance the efficiency and accuracy of their OCR processes, leading to improved document management and data accessibility.

4.2 Sample Code for Preprocessing and MLP Operations

To guarantee the Preprocessing is a crucial step in preparing images for Optical Character Recognition (OCR) to enhance the accuracy of text recognition. Below is a sample Python code that demonstrates common preprocessing techniques using the OpenCV and Pillow libraries. This code includes steps such as converting to grayscale, thresholding, noise removal, and resizing.

```
import cv2
import os
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
```

```

# Paths
input_dir = '/content/OCR_dataset'
output_dir = '/content/preprocessed'
os.makedirs(output_dir, exist_ok=True)

# Supported image formats
supported_ext = ['.jpg', '.jpeg', '.png', '.bmp', '.tif', '.tiff']

# Get image paths
image_paths = [p for p in glob(os.path.join(input_dir, '**', '*.*'),
                                     recursive=True)
                if os.path.splitext(p)[1].lower() in supported_ext]

# Preprocessing function
def preprocess_image(img_path, size=(128, 128)):
    img = cv2.imread(img_path)
    if img is None:
        return None

    # Resize
    img = cv2.resize(img, size)

    # Grayscale
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Denoise
    blur = cv2.GaussianBlur(gray, (5, 5), 0)

    # Thresholding (Binarization)
    _, binary = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY +
                              cv2.THRESH_OTSU)

    # Normalize to [0, 1] (float32 image)
    normalized = binary.astype(np.float32) / 255.0

    return normalized

# Process and save images
skipped = []

for path in image_paths:
    processed = preprocess_image(path)

    if processed is None:
        print(f"Skipped: {path}")

```

```

        skipped.append(path)
        continue

    # Convert back to uint8 for saving (multiply by 255)
    save_img = (processed * 255).astype(np.uint8)

    # Keep relative structure
    rel_path = os.path.relpath(path, input_dir)
    save_path = os.path.join(output_dir, rel_path)
    os.makedirs(os.path.dirname(save_path), exist_ok=True)
    cv2.imwrite(save_path, save_img)

print(f"\n✓ Done! Preprocessed {len(image_paths) - len(skipped)} images,
Skipped {len(skipped)}.")

# Display sample results
sample_paths = image_paths[:5]
plt.figure(figsize=(10, len(sample_paths) * 2))

for i, path in enumerate(sample_paths):
    orig = cv2.imread(path)
    processed = preprocess_image(path)

    plt.subplot(len(sample_paths), 2, 2*i + 1)
    plt.imshow(cv2.cvtColor(orig, cv2.COLOR_BGR2RGB))
    plt.title('Original')
    plt.axis('off')

    plt.subplot(len(sample_paths), 2, 2*i + 2)
    plt.imshow(processed, cmap='gray')
    plt.title('Preprocessed')
    plt.axis('off')

plt.tight_layout()
plt.show()

```

CHAPTER-5

Experimentation and Result Analysis

5. Experimentation and Result Analysis

The Random Forest model performed with the highest accuracy among all the classifiers and was hence the most efficient model for this dataset. Its better performance can be explained by the fact that it is an ensemble learning model, where multiple decision trees collaborate to minimize variance and overfitting. It is also stable against feature variation, which means it can tolerate diverse characteristics of input without adversely affecting performance. The Support Vector Machine (SVM) also competed well, demonstrating excellent generalization ability. SVM is robust in high-dimensional space and performs well when there is an obvious margin of separation between classes. Its capacity to identify the best hyperplane supported high accuracy to a great extent, making it a trustworthy option for classifying tasks. Conversely, K-Nearest Neighbors (KNN) and Naive Bayes performed relatively lower accuracy. KNN is very sensitive to outliers and noise because it makes predictions based on closeness to neighbors. In datasets with overlapping class distributions, KNN can find it difficult to preserve precision. Likewise, Naive Bayes makes the assumption of feature independence, which is usually not the case in real-world data. This results in reduced accuracy, particularly when there are feature dependencies. In addition, Naive Bayes is class imbalance sensitive, which could be the reason for its reduced performance.

CHAPTER-6

CONCLUSION

6. Conclusion

The project successfully implemented a comprehensive Optical Character Recognition (OCR) pipeline that encompassed several critical stages, including preprocessing, feature extraction, dimensionality reduction, and classification, each playing a vital role in enhancing the overall performance of the system. The preprocessing phase was essential for preparing input images, employing techniques such as noise reduction and binarization to improve data quality, which directly contributed to better recognition rates. Feature extraction allowed the model to focus on the most informative aspects of the data, while dimensionality reduction techniques, such as Principal Component Analysis (PCA), significantly improved performance by reducing complexity and mitigating overfitting risks. Among the various machine learning classifiers tested, the Stacking Classifier emerged as the most effective, achieving the highest accuracy through its ensemble learning approach, which combines the strengths of multiple classifiers for improved decision-making and robustness against variations in input data. These findings validate the effectiveness of ensemble learning in OCR applications, highlighting its potential to enhance accuracy and reliability in complex tasks where variations in handwriting, fonts, and document layouts pose significant challenges. Looking ahead, future research can focus on expanding the dataset to include a wider range of handwriting styles and document types, incorporating deep learning models like Convolutional Neural Networks (CNNs) for improved accuracy, and enhancing model generalization through techniques such as data augmentation and transfer learning. Additionally, investigating the deployment of OCR systems on edge devices can extend their applicability in real-world scenarios, enabling real-time text recognition in mobile and embedded systems. Overall, this project lays a solid foundation for real-world OCR applications, demonstrating the effectiveness of a well-structured pipeline and the potential for further advancements in the field.

7. REFERENCES

- [1] Haithem Afli, Loïc Barrault, and Holger Schwenk. 2016. OCR error correction using statistical machine translation. *Int. J. Comput. Ling. Appl.* 7, 1 (2016), 175–191.
- [2] Haithem Afli, Zhengwei Qiu, Andy Way, and Páraic Sheridan. 2016. Using SMT for OCR error correction of historical texts. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*. 962–966.
- [3] Mayce Al Azawi and Thomas M. Breuel. 2014. Context-dependent confusions rules for building error model using weighted finite state transducers for OCR post-processing. In *Proceedings of the 2014 11th IAPR International Workshop on Document Analysis Systems*. IEEE, 116–120.
- [4] Youssef Bassil and Mohammad Alwani. 2012. OCR context-sensitive error correction based on google web 1T 5-gram data set. *Am. J. Sci. Res.* 50 (2012).
- [5] Youssef Bassil and Mohammad Alwani. 2012. OCR post-processing error correction algorithm using google's online spelling suggestion. *J. Emerg. Trends Comput. Inf. Sci.* 3, 1 (2012).
- [6] Guilherme Torresan Bazzo, Gustavo Acauan Lorentz, Danny Suarez Vargas, and Viviane P Moreira. 2020. Assessing the impact of ocr errors in information retrieval. In *Proceedings of the European Conference on Information Retrieval*. Springer, 102–109.
- [7] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3 (Feb. 2003), 1137–1155.

