International Journal of Computer Techniques – Volume 12 Issue 3, May - June - 2025

Handwritten Character Recognition Using Deep Learning Techniques

Yerramshetty Dharvik
ComputerScience Engineering
Institute of Aeronautical Engineerinig
Hyderabad, India
21951a0536@iare.ac.in

Mukkerla Dathrish
ComputerScience Engineering
Institute of Aeronautical Engineerinig
Hyderabad, India
21951a0537@iare.ac.in

B Aditya Sriram
ComputerScience Engineering
Institute of Aeronautical Engineerinig
Hyderabad, India
21951a0503@iare.ac.in

S Kavitha
ComputerScience Engineering
Institute of Aeronautical Engineerinig
Hyderabad, India
kavithachejarla@gmail.com

Abstract— The development of an Android application for character recognition from images is a significant area of research. With the increasing need to digitize information from handwritten documents for future reference, capturing images of these documents and storing them as images has become a common practice. Optical Character Recognition (OCR) technology is used to convert handwritten text into an electronic format, a process that includes several stages such as pre-processing, segmentation, feature extraction, and postprocessing. Numerous researchers have explored the use of OCR for character recognition. This system leverages an Android smartphone to capture document images, with subsequent processing handled by OCR technology. The primary challenge lies in accurately recognizing characters from various handwriting styles. Therefore, we have developed a system designed to recognize handwritten text and convert it into editable format. The accuracy of this system largely depends on the clarity of the handwritten input. Our system achieves a 90% accuracy rate for handwritten documents, providing a straightforward method for editing or sharing the recognized text.

Keywords— Android Studio; OCR; handwritten character recognition

I. INTRODUCTION

The increasing demand for a paperless environment has driven advancements in technology aimed at digitizing handwritten documents. While recognizing handwritten text is an intuitive task for humans, it poses significant challenges for computer systems. Despite extensive research in this area, achieving 100% accuracy in handwritten character recognition remains elusive. Human eyes can easily discern different handwriting styles, but computers struggle to replicate this capability. Optical Character Recognition (OCR) offers a solution to this challenge. OCR technology is designed to convert scanned or printed image documents into editable text formats, making it a critical tool for digital text processing. The objective of this project is to harness the power of OCR through an Android application, aligning with the growing interest in mobile application development within the software industry. The Android app developed in this project provides users with the ability to recognize text from images stored in their gallery or captured directly through the device's camera. This functionality enables users to easily edit and store their cognized text in a digital format, such as a text file. The app leverages the camera of an Android device to capture binary images, which are then processed by the OCR engine to convert the visual data into editable text. The underlying technology of OCR involves scanning a document or image and converting it into a binary (black and white) version. The OCR software then analyses the image, distinguishing dark areas, which represent characters, from light areas, which constitute the background. The dark areas are subsequently processed to extract the text from the image, allowing the user to work with the digitized content.

II. EXISTING SYSTEM

Optical Character Recognition (OCR) technology has been widely researched and developed over the years, with systems focusing on converting scanned images of handwritten, printed, or typed text into machine-readable formats. These systems typically follow a multi-stage pipeline that includes image acquisition, preprocessing, segmentation, feature extraction, and text recognition.

Commercial OCR solutions like ABBYY FineReader and Google Cloud Vision are known for their accuracy in recognizing printed text. These solutions, however, often rely on proprietary algorithms and large training datasets, which allow them to deliver high recognition rates for printed documents but often face challenges with handwritten text due to its variability in style, size, and orientation [1][2]. Such systems are optimized for handling high-quality scanned images and tend to perform well under controlled conditions. However, when it comes to noisy images or text with complex structures, especially in

handwritten form, these systems show a significant drop inaccuracy [3].

On the other hand, open-source OCR solutions like Tesseract have gained traction due to their flexibility and adaptability. Tesseract, originally developed by Hewlett-Packard and now maintained by Google, has emerged as one of the most widely used OCR engines for both printed and handwritten text recognition. It is capable of handling multiple languages and works efficiently when integrated with preprocessing techniques like binarization, noise removal, and contrast enhancement [4]. However, Tesseract's performance is still affected by challenges such as poor image quality, variations in handwriting styles, and the presence of complex backgrounds [5].

Several research studies have explored improving Tesseract's accuracy for handwritten text recognition. These studies involve using additional preprocessing techniques to clean the input images and employing machine learning models to enhance character recognition accuracy [6]. Other opensource projects have also experimented with neural network-based models to achieve better results in handwritten OCR, yet many of these solutions still require further refinement to handle a broad range of writing styles [7].

Despite the advancements in OCR technologies, existing systems often struggle with two primary challenges: accurately recognizing handwritten text and ensuring reliable performance in resource-constrained environments such as mobile devices. Current OCR solutions, particularly those available on Android, tend to either depend on cloud-based processing, which requires an internet connection, or face limitations in handling complex handwriting offline [8]. These gaps highlight the need for a more accurate and efficient OCR system tailored to handwritten character recognition, particularly one that can function offline on mobile platforms like Android.

III. Literature Review

Optical Character Recognition (OCR) has been a subject of extensive research, with a particular focus on improving the accuracy and efficiency of recognizing both printed and handwritten text. Historically, OCR technology was primarily used for printed text, but with advancements in machine learning and computer vision, it has expanded to include handwritten text recognition. Memon et al. (2020) conducted a comprehensive review of OCR systems, focusing on the challenges and developments in handwritten OCR technology. They identified the complexity of handwriting styles, varying quality of input images, and the need for robust feature extraction algorithms as primary obstacles to achieving high recognition accuracy. Similarly, Islam et al.(2017) highlighted the growing need for accurate OCR systems in applications like digitizing historical documents, which often contain handwritten annotations. Their survey emphasized the role of preprocessing techniques such as noise reduction and binarization in improving recognition accuracy. In more specialized studies, Robby et al. (2019) explored the use of Tesseract OCR for recognizing Javanese script in Android applications, demonstrating the potential of integrating OCR technology into mobile platforms. Their findings showed that, while Tesseract is highly effective for recognizing printed text, additional challenges arise when applied to complex scripts or hand written documents.

Moreover, Vasudeva et al. (2012) proposed an OCR system based on artificial neural networks (ANN) for offline character recognition, achieving improved performance in recognizing characters from scanned documents.

Hand written character recognition, especially from non-English languages, has also garnered attention. For example, Pareek etal. (2020) developed a system for Gujarati handwritten character recognition, employing feature extraction techniques such as zoning and projections. Their system showed promise, although achieving high accuracy across various handwriting styles remains a challenge. Similarly, Chaudhuri et al. (2017) provided a detailed overview of different OCR systems, focusing on both rule-based and machine learning-based approaches for text recognition.

Recent studies have explored the integration of deep learning techniques in OCR. Bora et al. (2020) applied convolutional neural networks (CNN) for handwritten character recognition, using a technique known as Error-Correcting Output Codes (ECOC) to improve recognition accuracy. This approach has shown significant potential in recognizing characters from noisy or low-quality images, making it highly applicable for real world OCR applications. Additionally, Ahmed and Abidi (2019) reviewed various OCR methods, stressing the importance of balancing computational efficiency and accuracy, particularly for mobile-based OCR systems.

The use of recurrent neural networks (RNN) for OCR has also been explored. Parthiban et al. (2020) developed a system using an RNN for recognizing handwritten English text. Their research demonstrated that RNNs, due to their sequential processing capability, are highly suited for recognizing connected characters in handwriting.

In conclusion, the literature reflects a significant focus on improving the accuracy and efficiency of OCR systems, particularly for handwritten text recognition. While traditional methods such as feature extraction and ANN have provided a solid foundation, modern techniques such as CNNs and RNNs are paving the way for more accurate and scalable OCR solutions. However, challenges remain, particularly in handling diverse handwriting styles and low-quality inputs, which continue to be areas of active research.

IV. PROPOSED METHODOLOGY

The primary goal of this project is to leverage Optical Character Recognition (OCR) technology to convert captured images into machine-editable text. OCR technology operates in three fundamental stages: the scanning of the document, the recognition of characters, and the storage of the recognized text in a desired format.

For this project, we utilize the open-source OCR software Tesseract, renowned for its accuracy in converting scanned images or printed text into machine-editable text. Tesseract is
integrated into our Android application, ensuring reliable performance throughout the text recognition process.

The methodology for implementing OCR in this project involves several key steps, which are outlined in the following algorithm:

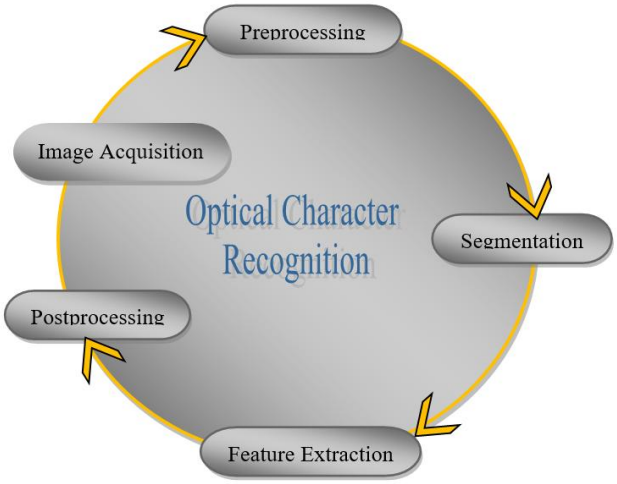


Fig.1. OCR Algorithm

A. Image Acquisition

The process begins with capturing an image using the Android device's camera. This step is crucial, as the quality of the captured image directly impacts the accuracy of the OCR.

B. Loading the Image into the GUI

Once the image is captured, it is loaded into the application's GUI, developed in Android Studio. This interface allows users to interact with the image and initiate the OCR process.

C. Preprocessing

Before text recognition, the image undergoes preprocessing to enhance its quality. This step may involve converting the image to grayscale, binarization, and noise reduction, all of which improve the OCR engine's ability to accurately detect characters.

D. Segmentation

The preprocessed image is then segmented to isolate individual characters or lines of text. This step is critical for recognizing the structure of the text within the image.

E. Feature Extraction

In this step, relevant features from the segmented image are extracted to facilitate the recognition process. These features help the OCR engine distinguish between different characters.

F. Text Recognition and Conversion

Using the Tesseract OCR engine, the extracted features are processed to recognize the text within the image. The recognized text is then converted into a machine-editable format, allowing the user to edit and store the text as needed.

E. Post-processing

After the text is recognized, post-processing techniques are applied to correct any errors or misinterpretations, ensuring the final output is as accurate as possible.

Steps involved in this work are: -

1. Image acquisition by the android camera
2. Loading the image into created Graphical User Interface (GUI) in android studio.
3. Preprocessing of the image.
4. Extraction of features from the input image.
5. Recognized data converted into text format using OCR[13].

V. ARCHITECTURE OF THE PROPOSED SYSTEM

The architecture of the proposed Optical Character Recognition (OCR) system for handwritten character recognition is designed to be modular and efficient, leveraging the capabilities of Android smartphones and the open-source Tesseract OCR engine. The system integrates several key components, including image acquisition, preprocessing, character segmentation, feature extraction, text recognition, and post-processing, each contributing to the accurate recognition of handwritten text.

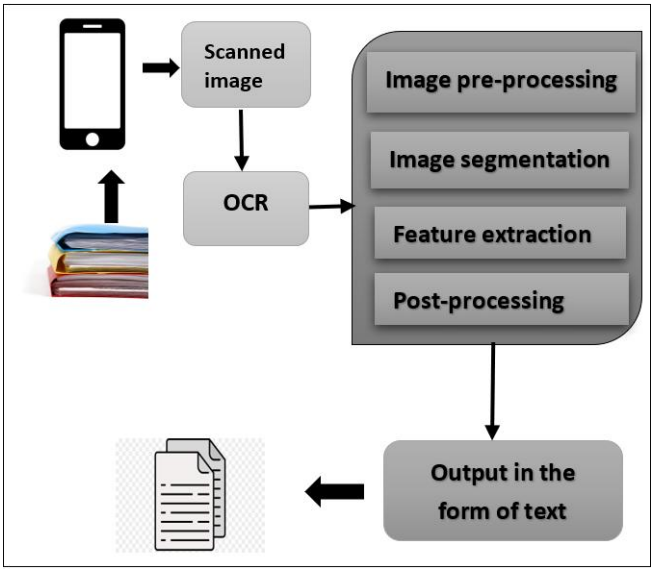


Fig.2. Architecture of the proposed system

1. Image Acquisition:

The system begins with image acquisition, where the Android device's camera captures images of handwritten documents. Users can either capture images in real time or select existing images from the device's gallery. This flexibility allows users to work with documents stored on their devices or quickly scan physical documents. The quality of the captured image is a crucial factor in determining the accuracy of the OCR process. High-resolution images enhance the ability of the OCR engine to recognize handwritten characters effectively.

2. Preprocessing

Once an image is acquired, it undergoes several preprocessing steps to enhance its suitability for OCR. Preprocessing includes:

- Grayscale Conversion: The image is converted to grayscale, simplifying the process by reducing the color information, thus making it easier to isolate characters from the background.
- Binarization: This step converts the grayscale image into a binary (black and white) image, where the foreground (text) is black, and the background is white. The Tesseract engine performs better on binary images, as it relies on high-contrast text regions.
- Noise Reduction: Preprocessing algorithms remove noise from the image, which could otherwise interfere with character recognition. This step ensures that only the text remains in focus for the subsequent stages.

3. Character Segmentation

Segmentation is a crucial stage in the OCR process, where the preprocessed image is divided into individual characters or lines of text. This step involves:

- Line Segmentation: The image is first segmented into lines of text to separate each row for further processing.
- Word and Character Segmentation: After line segmentation, the system identifies individual words and characters within each line, allowing the OCR engine to process one character at a time. This step ensures that characters are correctly isolated, improving the accuracy of recognition.

4. Feature Extraction

Feature extraction involves analyzing the segmented characters and identifying key patterns or features that distinguish one character from another. In this system:

- Shape Features: The system focuses on extracting shape-based features, such as edges, curves, and corners, which are critical in distinguishing different characters. These features are used to compare the input characters with a trained dataset, improving recognition accuracy.

5. Text Recognition Using Tesseract OCR

The core of the architecture is the Tesseract OCR engine, an open-source solution capable of recognizing both printed and handwritten text. Tesseract processes the features extracted from the segmented characters and compares them against its internal models to recognize the characters. The engine uses machine learning models trained on large datasets to improve its accuracy in recognizing various handwriting styles.

6. Post-processing

Once the text has been recognized, the system applies post-processing techniques to correct any errors or ambiguities that may have arisen during recognition. These techniques include:

- Spell Checking: The recognized text is passed through a spell-checker to identify and correct common recognition errors, particularly for handwritten text.
- Formatting: The recognized text is formatted to maintain the structure of the original document, preserving elements such as line breaks and paragraph spacing.

7. Output and Storage

After processing, the recognized text is displayed in the application's user interface, where users can edit or save the text in multiple formats, such as a .txt or .pdf file. The system also provides an option to share the recognized text through other applications or platforms. The output is stored in a dedicated folder on the device's internal storage, making it easily accessible for future use.

VI. IMPLEMENTATION

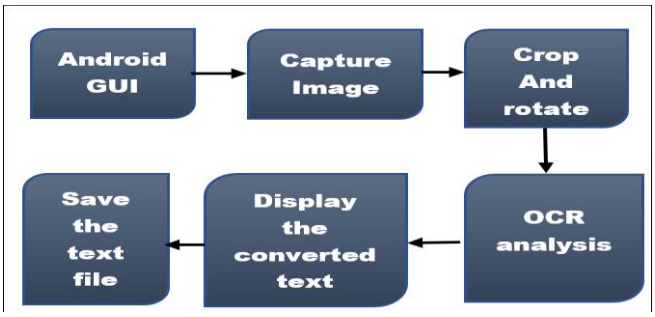


Fig.3. Implementation system

Firstly, an android app is being developed using android studio. The GUI provides the option to the user to capture the
image [14]. The user can scan the images at the first stage. The user will scan the document by the camera. Once there is an image in front of the screen the system functionality comes into the picture. The user has a choice to select the text to be
recognized. Then the binary image is sent to the OCR engine
for the further process [15]. The system is designed such that
it can easily recognize handwritten characters as input data in a proper manner and can edit the recognized output and save
the output in text format. Also, an option to share the output
of the system which is in text format and save as text format.

Fig.4. Home screen of the app

The user can use the GUI for capturing the handwritten document.[10] The user can load the image from the gallery or capture by camera. On clicking the camera icon, a sub-window open. The sub-window shows three icons. First showing the camera second showing the gallery and third showing photo applications. On clicking any of these three icons, the image is selected to process.

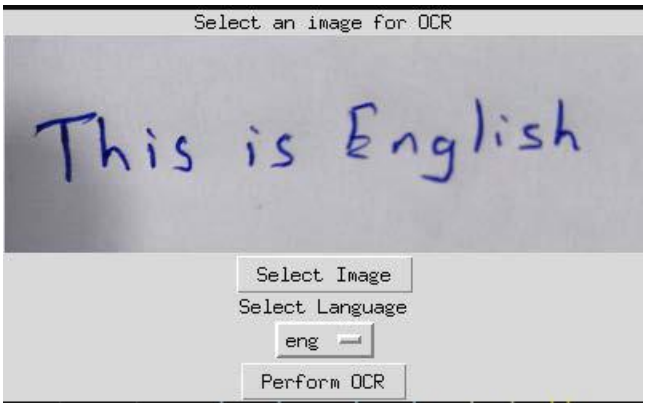


Fig.5. Selection of an image

After opening the image, changes can be done like cropping and rotating and clicking on the done button, the cropped image is sent to the OCR engine. The cropped image is shown
in fig.6

Fig.6. Displaying the output location

The text converted from the image is displayed on the screen. This text is in editable form. Next, an option to save this converted text as text file(.txt) as shown in fig.6.

Fig.8. Displaying converted text

This text file which consists of converted text will be saved in the OCR folder which would be created in the internal storage of the device. After finishing all the process, can open, edit and save that text file any time in future. This text file can also be saved as a pdf file.

VII. Results and Discussion

As shown in the result below, the image is captured using a camera. The captured image is converted into text using developed android application. The converted text is saved as a text file, which can be edited in future whenever required. The advantage of our system is that no internet connectivity is required for character recognition.

A. Handwritten Text

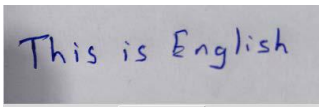


Fig.9. Displaying converted handwritten text

B. Printed Text

Fig.10. Displaying converted printed text

VIII. Limitations and Future Work

Limitations

Handwriting Variability: The system struggles with highly cursive or stylized handwriting, leading to inconsistent accuracy.

Limited Language Support: Current language recognition is limited, particularly for complex scripts and multilingual documents.

Noise Sensitivity: Low-quality images with noise or blur reduce recognition accuracy, despite preprocessing efforts.

Complex Layout Handling: The system is optimized for simple text and struggles with forms, tables, and complex document layouts.

No Real-time Feedback: Users receive feedback only after the OCR process is complete, lacking real-time correction.

Future Work

Improved Handwriting Recognition: Use deep learning to better handle diverse handwriting styles and demographics.

Expanded Language Support: Enhance recognition of complex scripts and support mixed-language documents.

Better Image Processing: Implement advanced algorithms to handle noisy or low-resolution images.

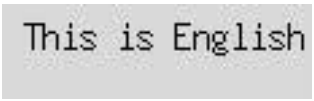
Complex Layout Recognition: Extend capabilities to accurately process forms, tables, and non-linear text.

Real-time Feedback: Add real-time suggestions and error correction during text recognition.

Cloud Integration: Explore cloud-based OCR for improved scalability and processing power.

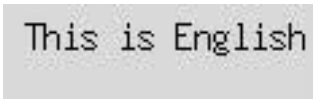
Application Integration: Enable integration with other apps through APIs for broader use cases.

Personalized Training: Allow users to train the system on their handwriting for more personalized accuracy.



IX. Conclusion

This implementation gives conversion of Handwritten Character Recognition into editable text using the android app. The image is captured by the camera and loaded into the
android app and choice is provided to the user to select a part
of an image which is to be converted. Further processing is done by OCR engine and produces the converted text on the screen. The recognized text is saved in text format. To edit the recognized text a choice is given and save them in a proper location. More accuracy is achieved when text is in printed form rather than in handwritten.



References

[1] Memon, J., Sami, M., Khan, R.A. and Uddin, M., 2020. Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR). IEEE access, 8, pp.142642-142668.

[2] Islam, N., Islam, Z. and Noor, N., 2017. A survey on optical character recognition system. arXiv preprint arXiv:1710.05703.

[3] Robby, G. & Tandra, Antonia & Susanto, Imelda & Harefa, Jeklin & Chowanda, Andry. (2019). Implementation of Optical Character Recognition using Tesseract with the Javanese Script Target in Android Application. Procedia Computer Science. 157. 499-505. 10.1016/j.procs.2019.09.006.

[4] Pareek, J., Singhania, D., Kumari, R.R. and Purohit, S., 2020. Gujarati handwritten character recognition from text images. Procedia Computer Science, 171, pp.514-523.

[5] Bora, M.B., Daimary, D., Amitab, K. and Kandar, D., 2020. Handwritten character recognition from images using CNN-ECOC. Procedia Computer Science, 167, pp.2403-2409.

[6] Chaudhuri, Arindam & Mandaviya, Krupa & Badelia, Pratixa & Ghosh, Soumya. (2017). Optical Character Recognition Systems. 10.1007/978- 3-319-50252-6_2.

[7] Ahmed, Muna & Abidi, Ali. (2019). REVIEW ON OPTICAL CHARACTER RECOGNITION.

[8] Sharma, Ankit & Chaudhary, Dipti & Ahemedabad, & India,. (2013). Character Recognition Using Neural Network.

[9] Vasudeva, Nisha & Parashar, Hem & Vijendra, Singh. (2012). Offline Character Recognition System Using Artificial Neural Network. International Journal of Machine Learning and Computing. 449-452.10.7763/IJMLC.2012.V2.165.

[10] N. Venkata Rao, Dr. A.S.C.S.Sastry, A.S.N.Chakravarthy, Kalyan Chakravart hi " opt ical charact er recognit ion t echnique algorit hms", Journal of Theoretical and Applied Information Technology , Vol.83. No.2,20th January 2016.

[11] Manoharan, Samuel. (2019). A SMART IMAGE PROCESSING ALGORITHM FOR TEXT RECOGNITION, INFORMATION EXTRACTION AND VOCALIZATION FOR THE VISUALLY CHALLENGED. Journal of Innovative Image Processing. 1. 31 38. 10.36548/jiip.2019.1.004.

[12] Chirag Patel, Atul Patel, Dhamendra Patel, "Optical Character Recognition by Open Source OCR Tool Tesseract A Case Study ",

International Journal of Computer Applications Volume 55– No.10, October 2012.

[13] Parthiban, R., Ezhilarasi, R. and Saravanan, D., 2020, July. Optical character recognition for English handwritten text using recurrent neural network. In 2020 International Conference on System, Computation, Automation and Networking (ICSCAN) (pp. 1-5). IEEE.

[14] Tiwari, Usha & Jain, Monika & Mehfuz, Shabana. (2019). Handwritten Character Recognition—An Analysis. 10.1007/978-981-13-0665-5_18.

[15] Kaur, Dalvir & Sharma, Sukesha. (2019). Various Feature Extraction and Classification Techniques. 10.1007/978-981-10-8234-4_51.