AUTOMATED OPTICAL CHARACTER RECOGNITION (OCR) FOR IMAGE PROCESSING

Authors :

Tanishq Pandey, School of computer science, KIIT, Bhubneshwar, India, 2206305@kiit.ac.in

Yusuf Imtiyaz, School of computer science, KIIT, Bhubneshwar, India, 2206394@kiit.ac.in

Abstract :

An approach to optical character recognition (OCR) utilizing deep learning and image processing techniques is presented in this research. Preprocessing photos, using OCR methods, and checking the accuracy of the retrieved text are all part of the implementation. Using a variety of datasets, the study offers a thorough analysis of OCR performance and identifies the main issues and solutions in text extraction

Keywords :*Optical Character Recognition, Image Processing, Deep Learning, Text Extraction, OCR Performance*

I. INTRODUCTION

One essential technology for automatic text extraction and digital document processing is optical character recognition, or OCR. In order to extract text from photos, this research investigates an OCR model based on deep learning. The goal is to assess different OCR strategies, such as model topologies, performance measures, and preprocessing methodologies

EASE OF USE AND MAINTAINING THE INTEGRITY OF THE SPECIFICATIONS

The implementation of OCR follows a structured approach using OpenCV and Tesseract OCR. Image preprocessing techniques, including grayscale conversion, thresholding, and morphological operations, enhance text readability before recognition. The developed pipeline ensures automation in text extraction while preserving image quality and structural integrity. The system is designed for ease of integration, requiring minimal user intervention while maintaining high accuracy across varied datasets. All modules adhere to predefined configurations to ensure consistency in results and reproducibility of experiments.

Prepare Your Data Before Processing

Before performing OCR, it is essential to preprocess and organize image data properly. All raw images should be stored in a structured format, ensuring clarity and high resolution. Text regions should be enhanced through preprocessing techniques like binarization, denoising, and contrast adjustment to maximize recognition accuracy.

Keeping image files separate from textual output helps maintain system efficiency. Avoid unnecessary formatting within image files, such as excessive compression, which may degrade quality. Additionally, OCR models should be trained or fine-tuned on cleaned datasets to prevent misclassification and improve extraction accuracy.

Abbreviations and Acronyms

Define abbreviations and acronyms the first time they appear in the text, even if they have already been introduced in the abstract. Common abbreviations related to OCR, such as Optical Character

Recognition (OCR), Convolutional Recurrent Neural Network (CRNN), and Open Source Computer Vision Library (OpenCV), do not need redefinition. Avoid using abbreviations in section titles unless necessary to maintain clarity and readability.

A. ***Some Common Mistakes***

Using low-resolution images, which reduces OCR accuracy.

Incorrect binarization leading to loss of text information.

Poorly segmented text regions causing misclassification.

Failure to apply noise reduction techniques before OCR processing.

Not tuning OCR model parameters for different font styles and sizes.

Ignoring language-specific preprocessing requirements.

Incorrect selection of OCR engines for different types of documents

Methodology

A. Data Preprocessing

Image noise reduction through grayscale conversion and thresholding.

Application of morphological operations for text enhancement.

Segmentation of characters using edge detection techniques.

B. OCR Model Implementation

Use of Tesseract OCR for text extraction.

Application of deep learning-based OCR techniques such as CRNN.

Evaluation using standard OCR benchmark datasets.

C. Performance Evaluation

Metrics: Accuracy, Precision, Recall, and F1-Score.

Comparison with traditional OCR methods.

Error analysis and refinement strategies.

ACKNOWLEDGMENT

My personal appreciation goes out to the Tesseract OCR team for creating a robust open-source text recognition engine that serves as the project's cornerstone. The successful implementation of OCR functionality was made possible by the substantial documentation and vibrant community support.

I am appreciative to the OpenCV and Matplotlib developers for offering powerful image processing capabilities that improved OCR accuracy through picture improvement, binarization, noise reduction, and inversion.

REFERENCES

1. Tesseract OCR GitHub Repository. Available at: https://github.com/tesseract-ocr/tesseract
2. Soni, B. (n.d.). *Binarization: A Powerful Technique in Image Processing*. Medium. Available at: https://medium.com/@brijesh_soni/topic-13-binarization-ca1059d8c1ce
3. Stack Overflow. *Binarization of a Grayscale Image*. Available at: https://stackoverflow.com/questions/65927513/binarization-an-image-grayscale
4. TPointTech. *OpenCV Erosion and Dilation*. Available at: https://www.tpointtech.com/opencv-erosion-and-dilation
5. GeeksforGeeks. *How to Install PIL on Windows*. Available at: https://www.geeksforgeeks.org/how-to-install-pil-on-windows/
6. Anaconda Documentation. *Pandas Tutorial*. Available at: https://docs.anaconda.com/navigator/tutorials/pandas/

7 Smith, R. (2007). *An Overview of the Tesseract OCR Engine*. Available at: https://github.com/tesseract-ocr/docs/blob/main/tesseract_overview.pdf

(A detailed explanation of how Tesseract OCR works, its architecture, and its evolution.)

8 OpenCV Documentation. *Image Processing (e.g., Thresholding, Morphological Operations)*. Available at: https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html

(Covers image binarization techniques like Otsu's thresholding and adaptive thresholding.)

9 Python Tesseract (pytesseract) Documentation. Available at: https://pypi.org/project/pytesseract/

(Python wrapper for Tesseract OCR, essential for integrating OCR functionality in Python projects.)