**Name : Janhvi Katakdhond**

**PRN: 22420013**

**Roll No : 382065**

**Batch: B3**

**Assignment No 5**

**Text identification using OpenCV, Tesseract (OCR) and deep neural network**

**Problem Statement:**

Implement text identification in images using OpenCV for preprocessing, Tesseract for OCR, and a deep learning pipeline for improved accuracy.

**Objective:**

* To understand how Optical Character Recognition (OCR) works with Tesseract.
* To preprocess images using OpenCV for better OCR performance.
* To implement text detection, region marking, and text extraction.
* To evaluate the accuracy of extracted text from noisy images.

**Technical Apparatus used:**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Jupyter Notebook, Anaconda, or Google Colab
* **Hardware:** CPU with minimum 4GB RAM; optional GPU for faster processing

**Libraries and Packages used:**

* OpenCV
* NumPy
* Tesseract OCR
* Pytesseract
* Google Colab utilities

**Theory:**

Optical Character Recognition (OCR) is the process of converting text in images into machine-readable text. Tesseract OCR is one of the most widely used open-source engines for this purpose. However, raw images may contain noise, distortion, or low contrast, which reduces OCR accuracy. To improve performance, image preprocessing techniques such as grayscale conversion, noise removal, and thresholding are applied using OpenCV. Bounding boxes can be drawn around detected text regions, and deep neural networks can further enhance text recognition in complex scenarios.

**Methodology:**

1. **Install and Import Libraries:**

* Install Tesseract OCR engine and pytesseract Python wrapper **Data Preparation:**

1. **Image Processing:**

A **Sequential** model is built using Keras:

* Convert image to grayscale.
* Apply noise reduction using Non-Local Means Denoising.
* Perform binary thresholding with Otsu’s method.

1. **Text Detection:**

* Use pytesseract.image\_to\_data() to detect text regions.
* Extract bounding box coordinates and confidence scores..

1. **Mark Text Regions:**

* Draw bounding boxes around detected text with confidence > 60%.

1. **Text Extraction:**

* Extract recognized text strings from detected regions.
* Combine them into the final output.

1. **Display Results:**

* Show original image, processed image, detected regions, and extracted text..

**Advantages:**

* Open-source and widely supported OCR engine.
* Works with multiple languages and fonts.
* Preprocessing improves recognition accuracy.
* Can detect and extract text from noisy or scanned documents.

**Limitations:**

* Sensitive to image quality and resolution.
* Accuracy decreases with handwritten or stylized fonts.
* May require additional deep learning models for complex documents.

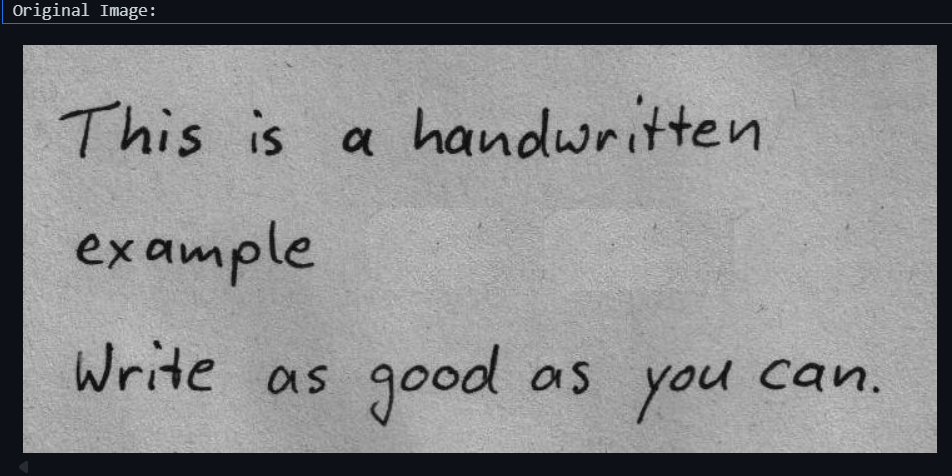
**Applications:**

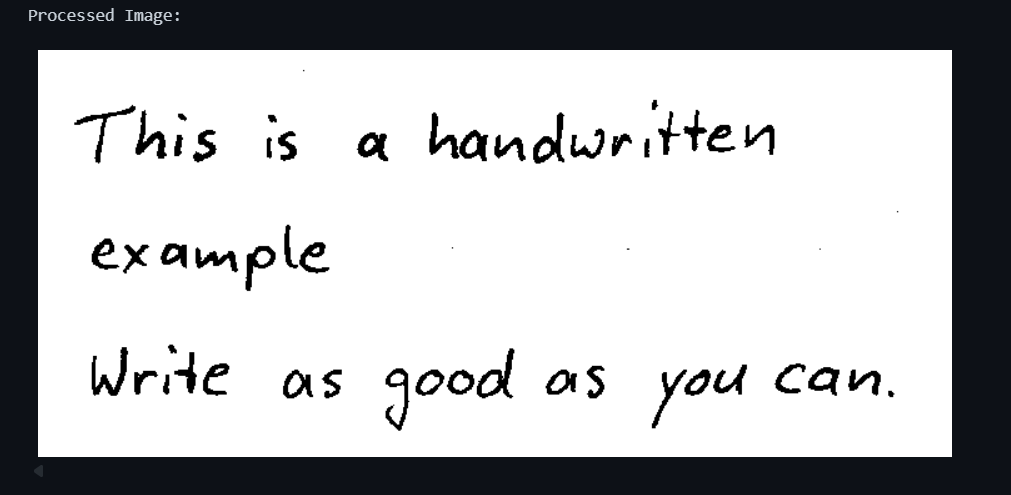
* Digitization of printed documents
* Digitization of printed documents
* Automated form processing
* Assisting visually impaired individuals through text-to-speech conversion

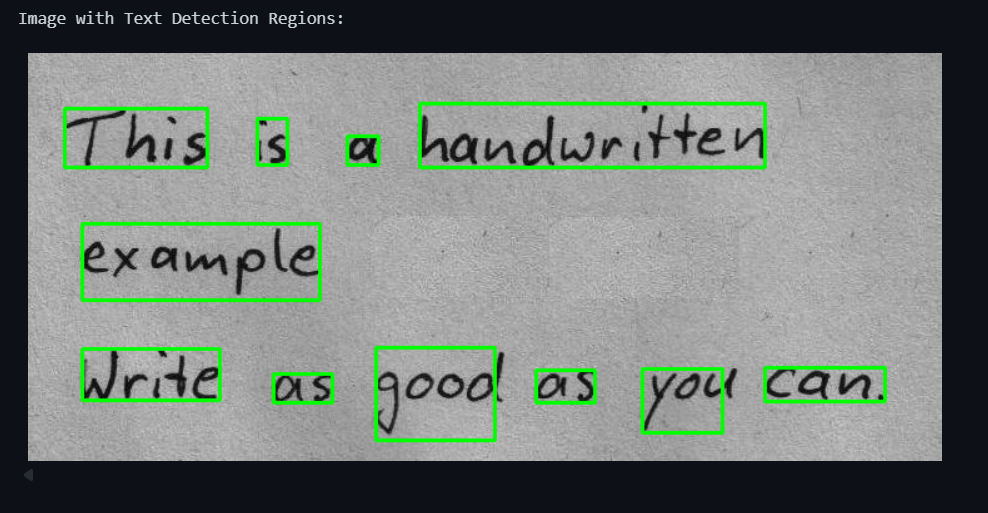
**Conclusion:**

In this assignment, text identification was implemented using OpenCV for preprocessing and Tesseract OCR for recognition. The preprocessing pipeline (grayscale, denoising, thresholding) significantly improved OCR accuracy. Bounding boxes were drawn to highlight detected text, and final extracted text was displayed. This demonstrates the integration of computer vision and OCR techniques for text extraction tasks.

**Implementation & Results:**

****

****

****