# Assignment 5 – Text identification using OpenCV, Tesseract (OCR) and deep neural network

Name : Aayush Bokde

Roll No: 382016

PRN: 22310316

## Problem Statement

Implement an Optical Character Recognition (OCR) system in Python using OpenCV and Tesseract to extract text from images.

## Objectives

* To understand the working of OCR using Tesseract.
* To preprocess images for improved text detection.
* To implement text recognition from images using OpenCV and Pytesseract.
* To evaluate OCR accuracy on sample images.

## Requirements

* Operating System: Windows/Linux/MacOS
* Python Version: 3.x
* Tools: Jupyter Notebook / Anaconda / Google Colab / Python Script
* Libraries Used: OpenCV, Pytesseract, NumPy
* Tesseract-OCR software installed on the system

## Theory

Optical Character Recognition (OCR) is the process of extracting text from images or scanned documents. OpenCV is used for image preprocessing (grayscale conversion, thresholding, noise removal) while Tesseract OCR is a popular open-source text recognition engine.  
  
Working steps:  
1. Convert image to grayscale.  
2. Apply thresholding to make text distinct.  
3. Pass the processed image to Tesseract for text extraction.

## Methodology

1. Import Libraries: OpenCV, Pytesseract.
2. Image Input: Read the input image using cv2.imread().
3. Preprocessing: Convert image to grayscale and apply thresholding.
4. OCR Extraction: Use pytesseract.image\_to\_string() with proper configuration.
5. Output: Display detected text.

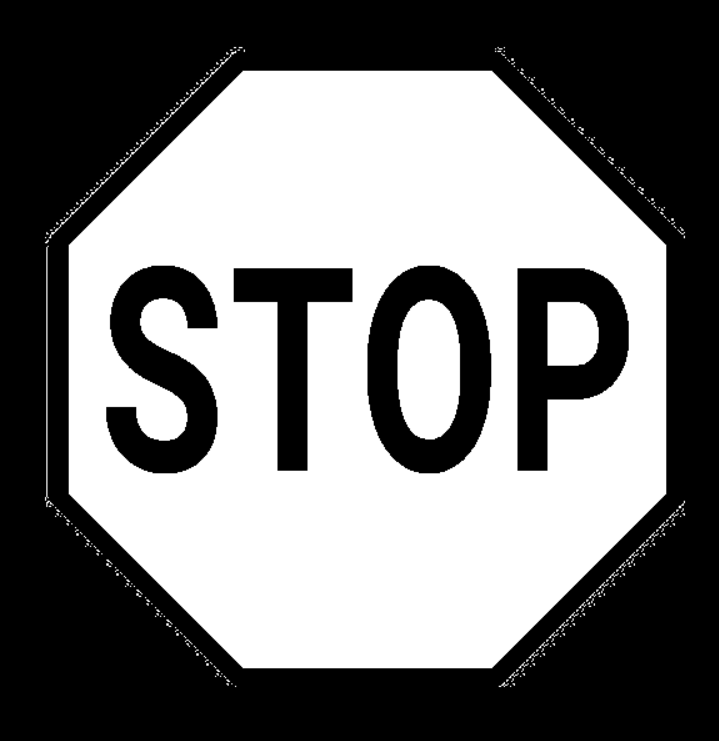
## Code Explanation

import cv2  
import pytesseract  
  
# Path to Tesseract executable (Windows specific)  
pytesseract.pytesseract.tesseract\_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"  
  
# Load the image  
image = cv2.imread("image.png")  
  
# Convert to grayscale  
gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
  
# Apply thresholding (white text on black background)  
gray = cv2.threshold(gray, 150, 255, cv2.THRESH\_BINARY\_INV)[1]  
  
# OCR extraction  
text = pytesseract.image\_to\_string(gray, config="--oem 3 --psm 7")  
print("Detected Text:", text)

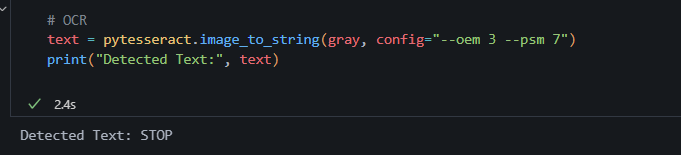
## Graphs and Visualizations

• Input Image (Original)

  
• Preprocessed Image (Grayscale + Thresholded)



• Detected Text Output printed on console.



## Advantages

* Extracts text from images efficiently.
* Works on noisy and scanned documents with preprocessing.
* Open-source and easy to integrate in Python.

## Limitations

* Accuracy depends on image quality and font style.
* May require heavy preprocessing for complex images.
* Struggles with handwritten text.

## Applications

* Digitization of printed documents.
* Automatic number plate recognition.
* Data extraction from receipts, invoices, and ID cards.
* Assistive technology for visually impaired users.

## Working / Algorithm

1. Import OpenCV and Pytesseract libraries.
2. Read input image.
3. Convert image to grayscale.
4. Apply thresholding for text clarity.
5. Perform OCR using Tesseract.
6. Print detected text.

## Conclusion

An OCR system was implemented using OpenCV for image preprocessing and Tesseract for text recognition. The system successfully extracted text from images, demonstrating the effectiveness of OCR in real-world applications like document scanning and automated data entry.