**Name:** Pushkar Charkha  
**Roll No:** 381002  
**PRN:** 22310023

**Assignment No: -** 5  
**Title: -** Text Identification using OpenCV, Tesseract (OCR), and Deep Neural Network

**Problem Statement:**

Implement a system that can automatically **detect and recognize text from images** using **OpenCV, Tesseract OCR, and Deep Neural Networks**. The model should preprocess the image, identify text regions, and extract textual content.

**Objective:**

* To perform **text detection and recognition** using OCR.
* To preprocess noisy images for better OCR accuracy.
* To integrate **OpenCV image processing** with **Tesseract OCR**.
* To demonstrate bounding box visualization of text regions.
* To extract readable text from an image for further applications.

**S/W Packages and H/W apparatus used:**

* **Operating System:** Windows/Linux/MacOS
* **Kernel:** Python 3.x
* **Tools:** Google Colab / Jupyter Notebook
* **Libraries:** OpenCV, NumPy, PyTesseract, Pillow
* **Hardware:** CPU / GPU (optional for DNN models)

**Theory:**

**Optical Character Recognition (OCR):**

* A technique to extract text from images.
* Uses pattern recognition and machine learning.

**Tesseract OCR:**

* An open-source OCR engine developed by Google.
* Converts scanned images and handwritten/printed text into machine-readable text.

**OpenCV:**

* Used for **image preprocessing** (grayscale, denoising, thresholding).
* Helps enhance the image quality for OCR.

**Deep Neural Networks (DNN):**

* Can improve text detection using models like EAST text detector.
* DNN-based OCR outperforms traditional rule-based methods for complex documents.

**Methodology:**

1. **Image Loading:** Load input image using OpenCV.
2. **Preprocessing:**
   * Convert to grayscale.
   * Apply denoising filter.
   * Perform binary thresholding using Otsu’s method.
3. **Text Detection:**
   * Use **pytesseract.image\_to\_data()** to get bounding boxes.
   * Apply confidence filtering (>60%).
4. **Bounding Box Drawing:** Highlight detected text with rectangles.
5. **Text Extraction:** Extract text from identified regions.
6. **Display Output:** Show original image, bounding box image, and extracted text.

**Results:**

****

**Advantages:**

* Can read printed and handwritten text.
* Works on noisy/blurred images with preprocessing.
* Easy integration with Python.

**Limitations:**

* Accuracy depends on image quality.
* Struggles with complex fonts or curved text.
* Sensitive to low-resolution images.

**Applications:**

* Document digitization (books, invoices, forms).
* License plate recognition.
* Text extraction from scanned reports.
* Assistive technology for visually impaired users.

**Conclusion:**

Text identification using **OpenCV and Tesseract OCR** was successfully implemented. The model detected and extracted text from the given image with reasonable accuracy. This approach can be extended with **deep neural network-based detectors (like EAST or CRNN)** for improved performance in real-world applications.