**Assignment No: - 5**

**Name :- Isha Ghorpade**

**Enrollnment :- 22420020**

**Roll :- 381066**

**Problem Statement:**

To implement a system that identifies and extracts text from images using OpenCV for image processing, Tesseract for Optical Character Recognition (OCR), and a deep neural network for further enhancement of text extraction.

**Objectives**

* Understand image preprocessing with OpenCV
  + Explore techniques like filtering, resizing, grayscale conversion, thresholding, and edge detection.
  + Gain practical experience with pixel-level image manipulation.
* Learn and implement Tesseract for OCR
  + Install and configure Tesseract for text recognition.
  + Use different Tesseract modes for extracting text from various document layouts.
* Enhance OCR using Deep Learning
  + Train deep neural networks (e.g., CNNs) on annotated datasets to improve accuracy.
  + Explore hybrid approaches integrating traditional OCR with deep models.

**S/W Packages and H/W 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 training and inference

**Libraries and Packages Used**

* OpenCV → Image processing and preprocessing
* Tesseract → OCR engine for text extraction
* NumPy → Numerical computations and array manipulation
* Matplotlib → Visualization of preprocessing and results
* TensorFlow / Keras → Deep learning framework for model training

**Theory**

1. OpenCV

A powerful library for computer vision, capable of reading, transforming, and enhancing images. Techniques like grayscale conversion, thresholding, and morphological operations are crucial for preparing images for OCR.

2. Tesseract OCR

An open-source OCR engine developed by Google. It recognizes text in scanned documents, images, and natural scenes. Tesseract can handle multiple languages, fonts, and layouts and outputs text in machine-readable formats.

3. Deep Learning Enhancement

Deep neural networks, particularly Convolutional Neural Networks (CNNs), can be used to enhance OCR results. They are effective in recognizing complex patterns, removing noise, and handling challenging inputs like distorted or handwritten text.

**Methodology**

1. Image Acquisition
   * Load images in different formats (JPG, PNG, TIFF) using OpenCV.
   * Collect diverse datasets including scanned documents, street signs, and handwritten notes.
2. Preprocessing
   * Convert to grayscale to reduce complexity.
   * Apply thresholding (binary images for better contrast).
   * Use morphological operations (erosion/dilation) to enhance text regions and remove noise.
3. Text Extraction (OCR)
   * Use pytesseract.image\_to\_string() to recognize text.
   * Experiment with page segmentation modes and OCR engine modes.
4. Deep Learning Enhancement (Optional)
   * Train a CNN on annotated datasets for improved recognition accuracy.
   * Use data augmentation (rotation, noise, brightness adjustments) to generalize across image types.
5. Evaluation
   * Measure OCR accuracy using Character Error Rate (CER) and Word Error Rate (WER).
   * Compare results before and after deep learning enhancement.
6. Visualization
   * Display original image, preprocessed version, and extracted text.
   * Plot bar charts for accuracy improvement.

**Advantages**

* High accuracy in extracting text from complex images.
* Supports multiple languages and text layouts.
* Open-source and widely supported across platforms.
* Extensible with deep learning for improved performance.

**Limitations**

* OCR accuracy decreases with poor-quality or noisy images.
* Preprocessing is critical; weak preprocessing reduces accuracy.
* Deep learning requires large labeled datasets and computational resources.
* Struggles with artistic fonts or extreme distortions.

**Applications**

* Document Digitization → Converting scanned books, records, and reports into searchable text.
* License Plate Recognition → Automated vehicle identification in traffic monitoring systems.
* Data Extraction → Reading invoices, forms, and bills for automation.
* Assistive Technology → Helping visually impaired individuals by reading text aloud.

**Working / Algorithm**

Step 1: Import Libraries

import cv2

import pytesseract

import numpy as np

import matplotlib.pyplot as plt

from tensorflow import keras

Step 2: Image Acquisition

image = cv2.imread("sample.png")

Step 3: Preprocessing

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

\_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH\_BINARY)

processed = cv2.morphologyEx(thresh, cv2.MORPH\_OPEN, np.ones((2,2), np.uint8))

Step 4: OCR Extraction

text = pytesseract.image\_to\_string(processed)

print("Extracted Text:", text)

Step 5: Deep Learning Enhancement (Optional)

* Train CNN on image-text pairs.
* Use the trained model to refine OCR outputs.

Step 6: Evaluation

* Compare extracted text with ground truth.
* Calculate CER and WER.

Step 7: Result Display

* Show processed image and extracted text side by side.

**Conclusion**

This project demonstrates a hybrid approach to text recognition using OpenCV, Tesseract, and Deep Learning. OpenCV ensures robust preprocessing, Tesseract extracts textual content, and deep learning enhances recognition accuracy, particularly in noisy or complex scenarios.

The system finds applications in document scanning, license plate detection, assistive technologies, and automated data entry. While performance depends on image quality and preprocessing, integrating deep learning provides significant improvements, making it a practical and scalable solution for real-world OCR tasks.