**Assignment No: 5**

**Text Identification Using OpenCV, Tesseract (OCR), and Deep Neural Network**

**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.

**Objective:**

1. **Image Processing with OpenCV**:
   * Explore techniques like filtering, resizing, and edge detection.
   * Gain experience in manipulating image pixels for preprocessing.
2. **Using Tesseract for OCR**:
   * Learn to install and configure Tesseract for text extraction.
   * Understand different modes and configurations of Tesseract.
3. **Enhancing Text Recognition with Deep Learning**:
   * Investigate how deep learning models (e.g., CNNs) can improve text recognition accuracy.

**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 a minimum of 4GB RAM; GPU (optional for faster processing)

**Libraries and Packages:**

* **OpenCV**: For image processing and manipulation.
* **Tesseract**: For OCR to extract text from images.
* **NumPy**: For numerical computations.
* **Matplotlib**: For visualizing images and results.
* **TensorFlow/Keras**: (Optional) For building deep learning models.

**Theory:**

1. **OpenCV**:
   * An open-source library for computer vision, useful for image manipulation, transformations, and feature extraction.
2. **Tesseract**:
   * An open-source OCR engine that uses machine learning to extract text from images.
3. **Deep Learning**:
   * Neural networks, especially Convolutional Neural Networks (CNNs), can be used to enhance OCR results by refining the detection of characters and text in images.

**Methodology:**

1. **Image Acquisition**:
   * Load images using OpenCV (e.g., scanned documents or images with text).
2. **Preprocessing**:
   * Convert the images to grayscale.
   * Apply thresholding to binarize the image.
   * Perform morphological operations (e.g., dilation, erosion) to improve text visibility and reduce noise.
3. **Text Extraction**:
   * Use Tesseract to extract text from preprocessed images. Adjust Tesseract’s parameters for optimal text recognition.
4. **Deep Learning Enhancement** (Optional):
   * Train a CNN model on images of text to enhance OCR performance.
   * Use a labeled dataset to teach the model how to recognize characters and words more accurately than standard OCR.
5. **Evaluation**:
   * Compare the extracted text with the original text using accuracy metrics like precision, recall, and F1-score.
6. **Visualization**:
   * Visualize the images, processed results, and extracted text using Matplotlib.

**Working/Algorithm:**

1. **Import Necessary Libraries**:

import cv2

import pytesseract

import numpy as np

from matplotlib import pyplot as plt

from tensorflow import keras

1. **Image Acquisition**:
   * Load the input image using OpenCV:

img = cv2.imread('image\_path.jpg')

1. **Preprocessing**:
   * Convert the image to grayscale:

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

* + Apply thresholding:

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

* + Perform morphological operations:

kernel = np.ones((5,5), np.uint8)

morph = cv2.morphologyEx(thresh, cv2.MORPH\_CLOSE, kernel)

1. **Text Extraction**:
   * Extract text from the preprocessed image using Tesseract:

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

print(text)

1. **Deep Learning Enhancement:**
   * Define a CNN architecture for improving OCR accuracy:

model = keras.Sequential([

keras.layers.Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(image\_height, image\_width, 1)),

keras.layers.MaxPooling2D(pool\_size=(2, 2)),

keras.layers.Flatten(),

keras.layers.Dense(128, activation='relu'),

keras.layers.Dense(num\_classes, activation='softmax')

])

1. **Evaluation**:
   * Compare predicted and actual text using metrics such as Character Error Rate (CER).
2. **Result Visualization**:
   * Plot the original and processed images using Matplotlib:

plt.imshow(morph, cmap='gray')

plt.title('Processed Image')

plt.show()

**Advantages:**

* **Accurate Text Recognition**: Works for images with different fonts and layouts.
* **Improved Performance with Deep Learning**: CNNs can further enhance OCR results.

**Limitations:**

* **Image Quality Dependency**: Text recognition performance varies with image quality and preprocessing techniques.
* **Preprocessing Required**: Effective OCR requires robust preprocessing.

**Applications:**

* **Document Scanning**
* **License Plate Recognition**
* **Data Extraction from Images**

**Diagram:**



**Conclusion:**

By combining OpenCV, Tesseract, and deep learning, this system can efficiently extract and identify text from various images, making it suitable for applications like document scanning and license plate recognition. Performance can be significantly enhanced using deep learning models, allowing for accurate text identification even in challenging conditions like noisy or low-resolution images.