# **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.

### **Objectives**

1. **To understand the basics of image processing using OpenCV**
   * Explore different image processing techniques such as filtering, resizing, and edge detection.
   * Gain hands-on experience with manipulating image pixels for preprocessing.
2. **To learn how to use Tesseract for OCR**
   * Understand the installation and configuration of Tesseract.
   * Familiarize with different Tesseract modes for various text extraction tasks.
3. **To explore deep learning techniques for improving text recognition accuracy**
   * Investigate training models on annotated datasets for improved OCR performance.
   * Explore CNN-based architectures effective for image-to-text recognition.

### **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 processing

### **Libraries and Packages Used**

* **OpenCV** – Computer vision and image processing.
* **Tesseract** – Open-source OCR engine.
* **NumPy** – Numerical computation.
* **Matplotlib** – Visualization.
* **TensorFlow/Keras** – Deep learning model building (optional).

### **Theory**

1. **OpenCV**
   * A powerful computer vision library.
   * Enables reading, writing, and processing of images.
   * Provides operations like filtering, transformations, and feature detection.
2. **Tesseract**
   * A leading OCR engine that extracts text from images.
   * Can be trained for different languages and fonts.
   * Highly adaptable for diverse applications.
3. **Deep Learning**
   * A branch of ML using neural networks with multiple layers.
   * Learns high-level features automatically.
   * CNN-based models significantly improve OCR results by refining text recognition.

### **Methodology**

1. **Image Acquisition**
   * Load images with OpenCV in various formats.
   * Ensure dataset diversity for effective training.
2. **Preprocessing**
   * Convert images to grayscale.
   * Apply thresholding for binary representation.
   * Perform morphological operations (erosion, dilation) to enhance visibility and reduce noise.
3. **Text Extraction**
   * Apply Tesseract OCR to extract text.
   * Experiment with configuration options for optimized results.
4. **Deep Learning Enhancement (Optional)**
   * Train a CNN-based model on labeled text-image datasets.
   * Use data augmentation for robustness.
   * Integrate with OCR results for improved accuracy.
5. **Evaluation**
   * Metrics: Precision, Recall, F1-score, CER (Character Error Rate), WER (Word Error Rate).
   * Visualize improvements with charts/graphs.
6. **Display Results**
   * Output recognized text to console or save to a file.

### **Advantages**

* High accuracy in extracting text from diverse image formats.
* Adaptable to different fonts, sizes, and layouts.
* Integration of deep learning enhances recognition capabilities.

### **Limitations**

* Performance depends on image quality and preprocessing.
* Requires significant preprocessing for optimal results.
* Training deep learning models requires annotated datasets.

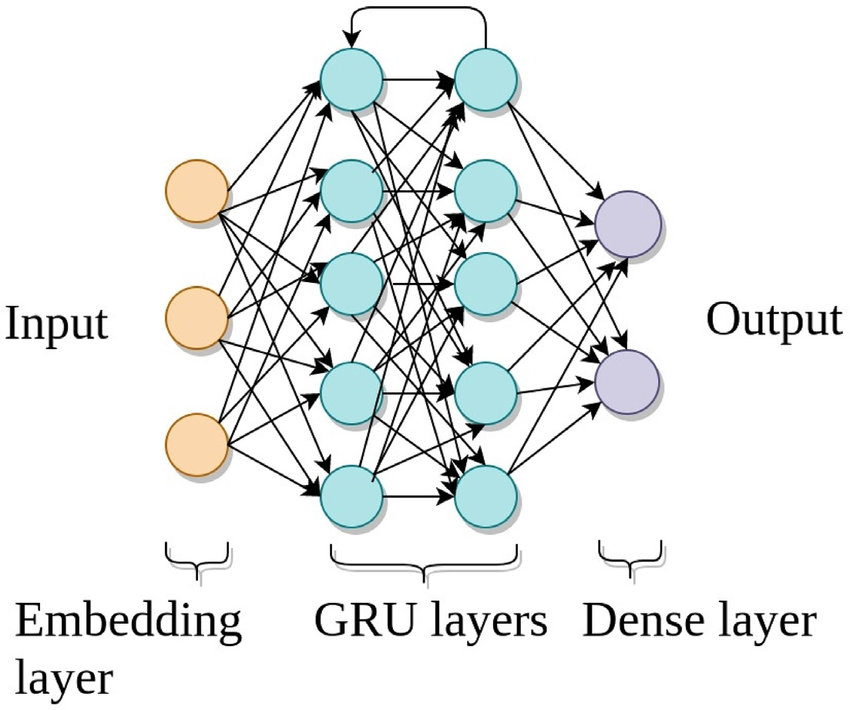
### **Applications**

* **Document Scanning** – Digitizing printed text.
* **License Plate Recognition** – Traffic monitoring & security systems.
* **Data Extraction** – Automating information retrieval from forms, invoices, and records.

### **Working / Algorithm**

1. **Import Libraries**
   * OpenCV, Tesseract, NumPy, Matplotlib, TensorFlow/Keras.
2. **Image Acquisition**
   * Load input images using cv2.imread().
3. **Preprocessing**
   * Convert to grayscale (cv2.cvtColor).
   * Apply thresholding (cv2.threshold).
   * Perform morphological operations (cv2.morphologyEx).
4. **Text Extraction**
   * Extract text using pytesseract.image\_to\_string().
5. **Deep Learning Enhancement (Optional)**
   * Prepare dataset (train/test split).
   * Build CNN model using Keras.
   * Train and validate for improved recognition.
6. **Evaluation**
   * Use metrics: CER, WER.
   * Visualize improvements with Matplotlib.
7. **Display Result**
   * Print or save extracted text.

### **Diagram**

**

### 

### **Conclusion**

The integration of **OpenCV, Tesseract, and deep learning** provides a comprehensive system for **text identification in images**.

* OpenCV ensures effective preprocessing.
* Tesseract performs robust OCR.
* Deep learning models (CNNs) enhance recognition by handling variations in fonts, sizes, and layouts.

This combination achieves **high accuracy** in real-world scenarios like **document digitization, license plate recognition, and automated data extraction**. However, results still depend heavily on image quality and preprocessing. Proper dataset preparation and fine-tuning are crucial for achieving optimal outcomes.