Name: Aziz Sayyad
Roll No: 381069
Batch: A3

Assignment No: - 5

Problem Statement:
Implement a text identification system using OpenCV for image preprocessing, Tesseract OCR for text recognition, and Deep Neural Networks for enhancing recognition accuracy.

Objectives

- To understand the concepts of **Optical Character Recognition (OCR)** and its importance in real-world applications.
- To preprocess text images using **OpenCV** for improved feature clarity.
- To apply **Tesseract OCR** for automatic extraction of textual information.
- To integrate **Deep Neural Networks (DNNs/CNNs/RNNs)** for boosting recognition accuracy.
- To evaluate system accuracy using datasets of printed and handwritten text.
- To compare the baseline accuracy of Tesseract with the enhanced performance after DNN integration.

Software and Hardware Requirements

Software Packages:

- Operating System: Windows/Linux/MacOS
- Kernel: Python 3.x
- Tools: Jupyter Notebook, Anaconda, Google Colab
- Libraries: OpenCV, Tesseract OCR, TensorFlow/Keras, NumPy, Pandas, Matplotlib, Pytesseract, Scikit-learn

Hardware:

- CPU with minimum 8GB RAM
- GPU (NVIDIA CUDA-enabled) recommended for DNN training
- High-resolution camera or scanner for capturing input images

Theory

Optical Character Recognition (OCR):
OCR is a technology that converts scanned images, photos, or PDFs containing text into machine-readable digital text.

Components:

1. **OpenCV**
    - Performs image preprocessing: grayscale conversion, thresholding, denoising, deskewing, contour detection, and edge enhancement.

- Improves the quality of text regions before passing to the OCR engine.

2. **Tesseract OCR**
   - Open-source OCR engine developed by Google.
   - Extracts text from preprocessed images.
   - Provides bounding boxes and confidence scores for recognized characters.
3. **Deep Neural Networks (DNNs)**
   - Handle challenging cases where Tesseract alone struggles.
   - CNNs (Convolutional Neural Networks) are effective for character recognition.
   - RNNs/LSTMs (Recurrent Neural Networks) handle sequential dependencies in handwritten text.
   - Pretrained models (e.g., CRNN – Convolutional Recurrent Neural Network) can be fine-tuned.

Working Principle

1. **Image Preprocessing (OpenCV):**
   - Convert image → grayscale
   - Apply **adaptive thresholding** for binarization
   - Remove noise using **morphological operations**
   - Apply **deskewing** if text is tilted
   - Segment text regions using contours
2. **Text Extraction (Tesseract OCR):**
   - Use Pytesseract wrapper
   - Extract raw text and confidence values
3. **Deep Neural Network Enhancement:**
   - Train CNN/RNN/CRNN model on datasets like IAM (handwriting) or MNIST (characters).
   - Use the model to re-check or correct Tesseract outputs.
   - Handle challenging cases (blurred, handwritten, noisy images).

Methodology

1. **Data Acquisition:**
   - Collect a dataset of printed and handwritten images.
   - Use IAM, MNIST, or custom scanned documents.
2. **Preprocessing (OpenCV):**
   - Grayscale conversion
   - Adaptive Thresholding / Otsu Binarization
   - Morphological filtering for noise removal
   - Deskewing using Hough Line Transform
3. **Text Detection & Extraction (Tesseract OCR):**
   - Apply Pytesseract
   - Extract text regions with bounding boxes
4. **Deep Learning Enhancement:**
   - Train CNN model for character recognition
   - Use RNN/LSTM for sequence-based recognition (handwriting/words)
   - Integrate predictions with OCR output to refine accuracy
5. **Evaluation Metrics:**
   - **Accuracy** = (Correctly recognized characters / Total characters)

- **Precision & Recall** for text segments
- **Word Error Rate (WER)**

Advantages

- Works for **both printed and handwritten** text
- Preprocessing ensures cleaner input for OCR
- Tesseract is **open-source, efficient, and multilingual**
- DNNs improve recognition for noisy, distorted, or handwritten samples
- Can be scaled for **large-scale document automation**

Limitations

- Requires high-quality images for best performance
- Handwritten text recognition remains challenging
- Deep learning models are **computationally expensive**
- Sensitive to skewed or rotated text
- Fonts, stylized writing, and low-resolution images may reduce accuracy

Applications

- **Document digitization**: Books, records, receipts
- **Automatic number plate recognition** (ANPR)
- **Banking & Education**: Handwritten forms, exam sheets
- **Healthcare**: Digitizing prescriptions & patient records
- **Real-time translation apps** (e.g., Google Translate camera)
- **Assistive technology** for visually impaired users

Algorithm / Workflow

1. Import libraries (OpenCV, Pytesseract, TensorFlow/Keras).
2. Load input text image.
3. Preprocess image (grayscale → threshold → denoise → deskew).
4. Apply Tesseract OCR to extract text.
5. Train and integrate DNN for character/word recognition.
6. Compare results with ground truth labels.
7. Display and save recognized text.

Future Enhancements

- Use **Transformer-based models (Vision Transformers, OCR-BERT)** for improved performance.
- Deploy the model as a **real-time mobile/desktop application**.
- Implement **multilingual OCR with language detection**.
- Integrate with **cloud-based APIs** (Google Vision, AWS Textract) for scalability.

Conclusion

Text identification using **OpenCV, Tesseract, and Deep Neural Networks** is a powerful approach to extract text from printed and handwritten documents. OpenCV ensures clean

preprocessing, Tesseract provides reliable OCR, and DNNs boost accuracy for complex scenarios like noisy, distorted, or handwritten inputs. This combined system is widely applicable in **document digitization, healthcare, banking, license plate recognition, and assistive technologies**, making it a valuable solution in today's data-driven world.