PROJECT REPORT

Multi-OCR Image to Text Comparison System

---

1. Introduction

Optical Character Recognition (OCR) is a key technology used to convert text present in images into machine-readable format. OCR plays a crucial role in document digitization, automation, data extraction, and accessibility applications.

Different OCR engines use different algorithms, ranging from traditional rule-based systems to deep learning models, resulting in variations in accuracy and performance.

This project focuses on building a Multi-OCR Image to Text Comparison System that evaluates and compares the performance of three popular OCR engines:

- Tesseract OCR
- EasyOCR
- PaddleOCR

The system processes multiple images automatically, extracts text using each OCR engine, measures execution time, and generates both tabular and graphical performance comparisons.

---

2. Problem Statement

Different OCR engines produce different results depending on:

- Image quality
- Font style
- Layout complexity
- Processing speed

There is no single OCR engine that performs best in all scenarios. Hence, there is a need for a system that:

- Runs multiple OCR engines on the same image
- Compares their outputs
- Evaluates performance objectively

3. Objectives

The main objectives of this project are:

1. To extract text from images using multiple OCR engines
2. To preprocess images for better OCR accuracy
3. To compare OCR engines based on execution time
4. To process images in batch mode
5. To generate structured output files, CSV tables, and charts

---

4. Tools and Technologies Used

Tool / Library	Purpose
Python	Core programming language
OpenCV	Image preprocessing
Tesseract OCR	Traditional OCR engine
EasyOCR	Deep learning-based OCR
PaddleOCR	Advanced OCR framework
Pandas	CSV data processing
Matplotlib	Data visualization
PyCharm	Development environment
CSV	Performance comparison output

---

5. System Architecture

The system follows a modular architecture:

1. Images are read from the images/ folder
2. Preprocessing is applied using OpenCV

3. OCR engines are executed independently
4. Text outputs are saved into files
5. Execution times are recorded in a CSV file
6. Charts are generated for performance analysis

This modular design improves:

- Readability
- Maintainability
- Scalability

6. Project Structure

The project is organized using a modular and extensible folder structure to separate concerns such as image input, OCR logic, output storage, and performance visualization.

MULTI-OCR IMAGE TO TEXT COMPARISON SYSTEM/

```
│
├── images/
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
│
├── output/
│   ├── tesseract_output.txt
│   ├── easyocr_output.txt
│   ├── paddleocr_output.txt
│   ├── ocr_comparison.csv
│   └── charts/
│       ├── avg_execution_time.png
│       └── execution_time_per_image_easyocr.png
```

└── execution_time_per_image_paddleocr.png

└── execution_time_per_image_tesseract.png

│

├── ocr/

│ ├── __init__.py

│ ├── tesseract_ocr.py

│ ├── easyocr_ocr.py

│ └── paddleocr_ocr.py

│

├── multi_ocr.py

└── plot_charts.py

7. Image Preprocessing

Before applying OCR, images are preprocessed to improve accuracy.

Steps:

1. Image is loaded using OpenCV
2. Converted to grayscale
3. Binary thresholding is applied

This improves:

- Text clarity
- Contrast
- OCR recognition quality

---

8. OCR Engines Used

8.1 Tesseract OCR

- Open-source OCR engine
- Fast execution

- Moderate accuracy
- Best suited for clean printed text

8.2 EasyOCR

- Deep learning-based OCR
- High text coverage
- Slower execution
- Handles complex fonts and layouts

8.3 PaddleOCR

- Advanced OCR framework
- Strong for structured documents
- Higher processing time
- Less effective on artistic layouts

---

9. Implementation

The system processes all images present in the images/ directory.
For each image:

- Text is extracted using all OCR engines
- Execution time is measured
- Outputs are saved automatically

Performance data is collected and stored in a CSV file for further analysis.

---

10. Results

The system successfully processed all images and generated OCR outputs.

Sample Performance Comparison

OCR Engine	Accuracy	Speed
Tesseract	Medium	Fast
EasyOCR	High	Slow
PaddleOCR	Medium	Slow

The CSV file ocr_comparison.csv contains detailed execution time information for all images.

---

11. Performance Visualization and Analysis

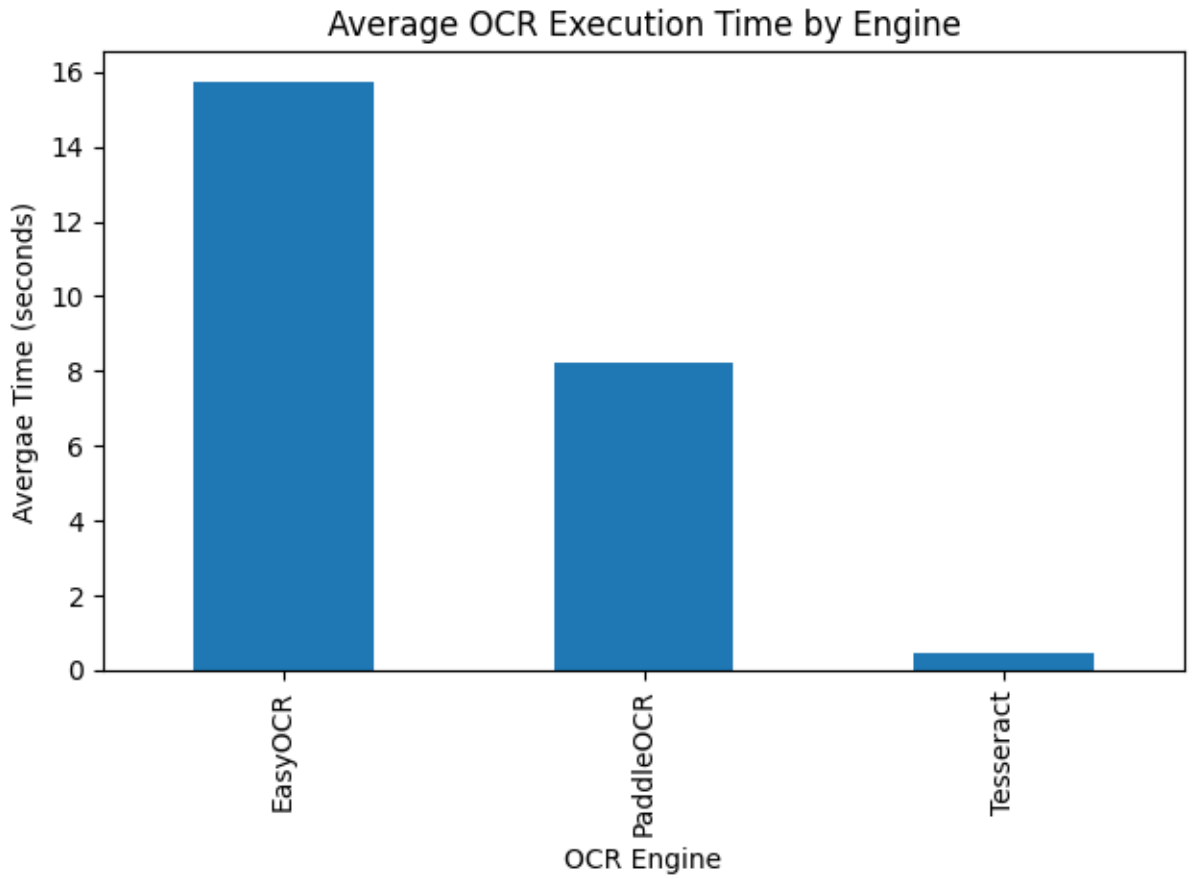
To enhance result interpretation, execution-time data stored in the CSV file was visualized using charts created with Pandas and Matplotlib.

11.1 Average Execution Time Comparison

A bar chart was generated to compare the **average execution time** of each OCR engine.

Observations:

- Tesseract is the fastest OCR engine
- EasyOCR has the highest execution time due to deep learning models
- PaddleOCR performs slower than Tesseract but comparable to EasyOCR

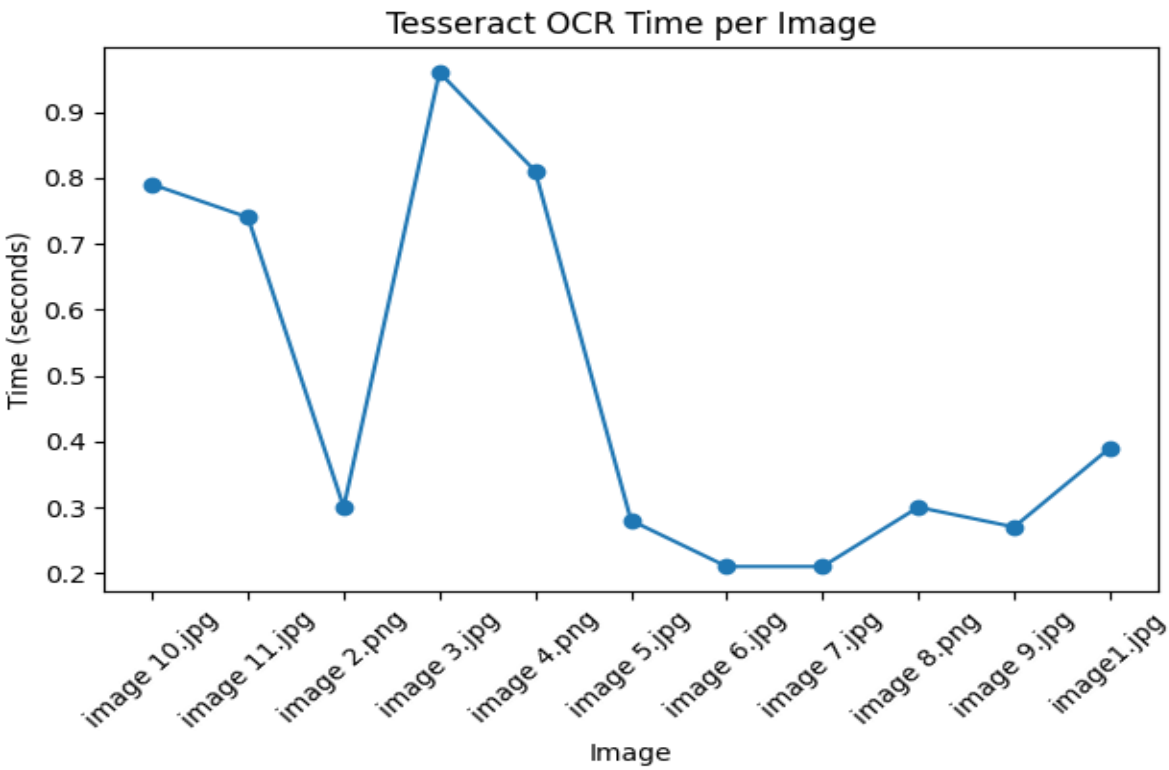


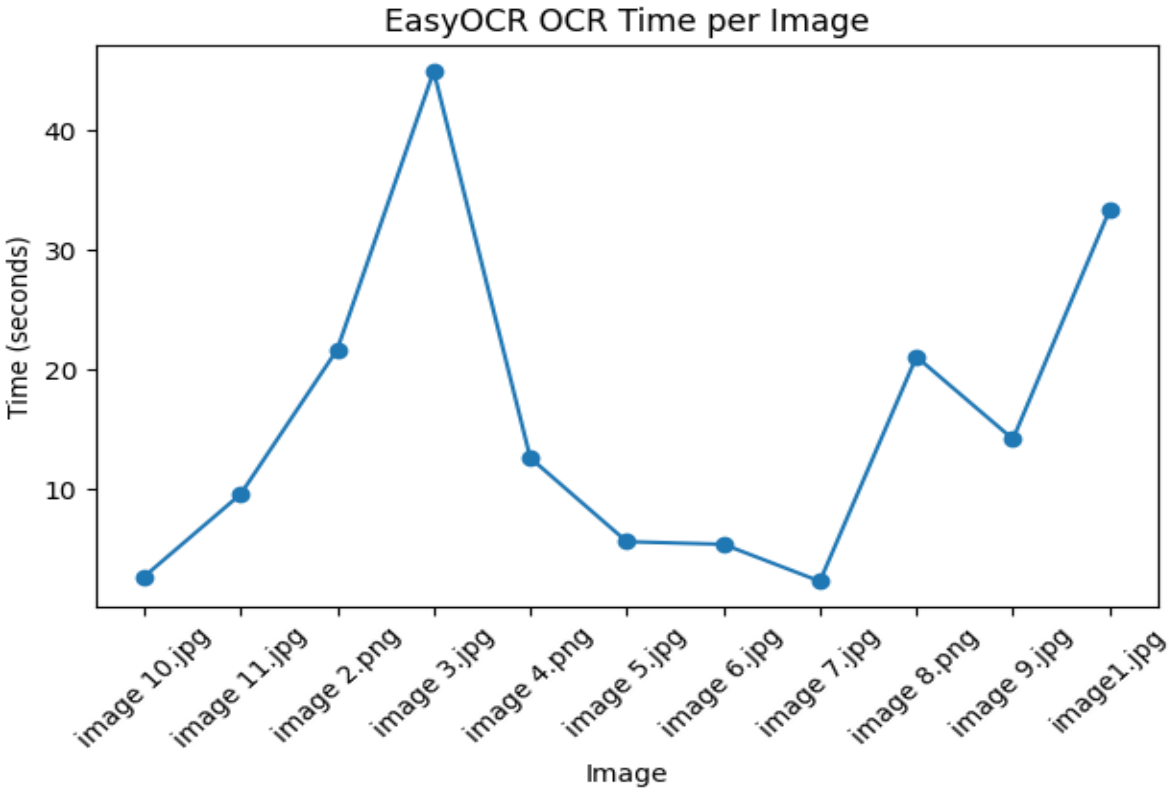
11.2 Execution Time per Image

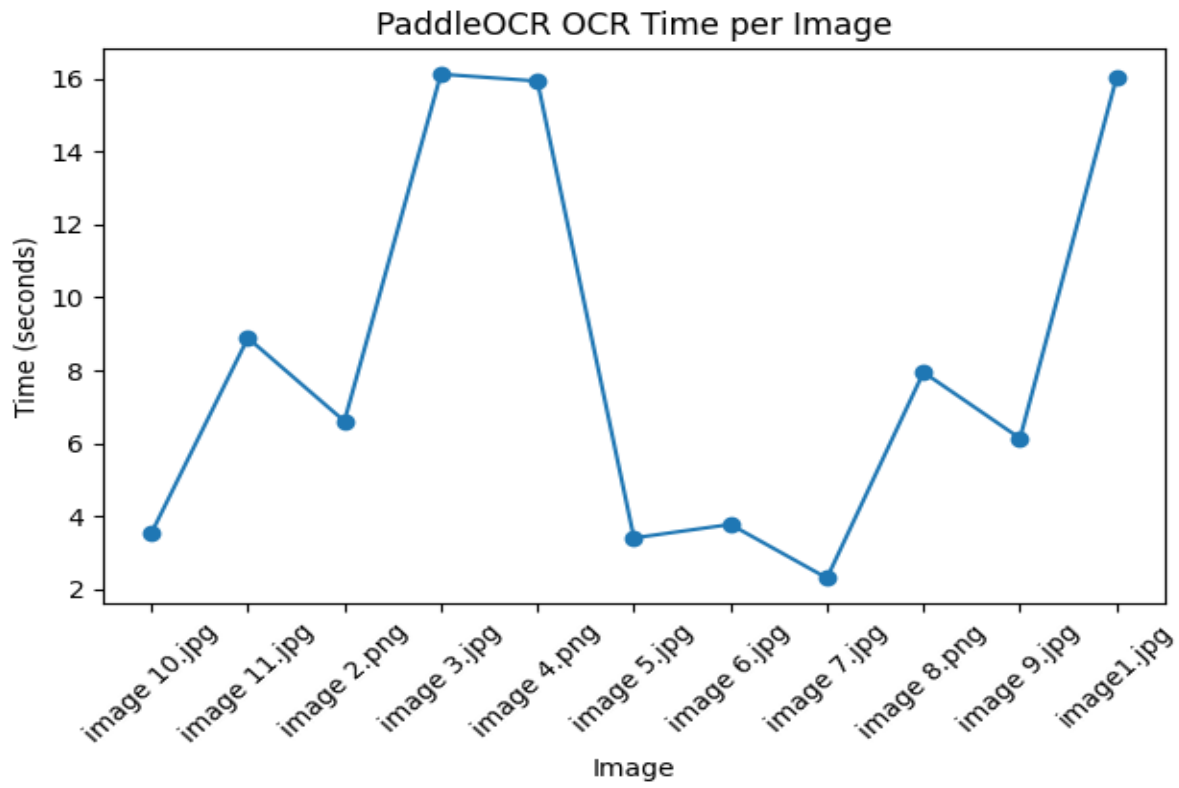
Line charts were generated to analyze execution time variation across different images.

Observations:

- Execution time varies based on image complexity
- Images with stylized or dense text take longer
- EasyOCR shows higher variability compared to Tesseract


Tesseract OCR Time per Image
Time (seconds)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
image 10.jpg
image 11.jpg
image 2.png
image 3.jpg
image 4.png
image 5.jpg
image 6.jpg
image 7.jpg
image 8.png
image 9.jpg
image1.jpg
Image


EasyOCR OCR Time per Image
Time (seconds)
40
30
20
10
image 10.jpg
image 11.jpg
image 2.png
image 3.jpg
image 4.png
image 5.jpg
image 6.jpg
image 7.jpg
image 8.png
image 9.jpg
image1.jpg
Image



11.3 Overall Insights

- Tesseract is best for fast processing
- EasyOCR provides better text coverage at the cost of speed
- PaddleOCR works well for structured documents

Visualization confirms that OCR engine selection should be use-case driven.

---

12. Challenges Faced

- Dependency management on Windows
- Handling deprecated library APIs
- Performance differences across OCR engines
- Batch processing multiple images

All challenges were resolved through modular design and systematic debugging.

13. Conclusion

This project successfully demonstrates a Multi-OCR Image to Text Comparison System capable of batch processing images, extracting text using multiple OCR engines, and evaluating their performance using both tabular and graphical analysis.

The system highlights that no single OCR engine is universally optimal, and selecting the right OCR solution depends on accuracy requirements and performance constraints.

---

14. Future Enhancements

- GPU acceleration using Linux or WSL
- Accuracy metrics such as Word Error Rate (WER)
- Automatic chart export as image files
- Multilingual OCR comparison
- Web-based OCR comparison dashboard

---

15. References

- Tesseract OCR Documentation - Tesseract User Manual | tessdoc
- EasyOCR GitHub Repository - GitHub - JaidedAI/EasyOCR: Ready-to-use OCR with 80+ supported languages and all popular writing scripts including Latin, Chinese, Arabic, Devanagari, Cyrillic and etc.
- PaddleOCR Official Documentation - Installation - PaddleOCR Documentation
- OpenCV Python Documentation - OpenCV: OpenCV-Python Tutorials

16.Appendix A: Diagrams

All system-related diagrams have been created using draw.io and are submitted as a separate document titled:

“System architecture and logical flow diagrams.drawio_D.Bhargav.pdf”

This appendix includes:

- System Architecture Diagram
- Block Diagram
- Logical Flow Diagram
- Data Flow Diagram (Level 0)
- Module Interaction Diagram