OCR Assignment Report: PyTesseract and EasyOCR

1. Introduction
Optical Character Recognition (OCR) is used to convert text from images into editable text. In this assignment, two OCR methods were implemented and compared. The first method uses PyTesseract, which is a traditional OCR engine. The second method uses EasyOCR, which is a deep learning based OCR system. The main goal is to compare their performance using the same dataset.

2. Dataset Used
The dataset used for this assignment is the OCR Dataset of Multi-Type Documents from Kaggle. It contains scanned forms and real-life documents. The dataset already includes training and test folders. Due to limited computational resources, only a small subset of the training data was used for EasyOCR training.

3. PyTesseract OCR
PyTesseract is a ready-to-use OCR engine and does not require any training. The focus while using PyTesseract was on image preprocessing and configuration.

3.1 Image Preprocessing
Before applying OCR, the images were converted to grayscale. After that, thresholding was applied to remove background noise and make the text clearer. This helped improve the OCR output.

3.2 Page Segmentation Mode (PSM)
Different Page Segmentation Modes were tested, including default mode, PSM 6, and PSM 3. After comparing the outputs, PSM 3 gave the best and most readable text, so it was selected for final evaluation.

3.3 Evaluation
The OCR output from PyTesseract was compared with the ground truth text provided in the dataset annotations. Character Error Rate (CER) and Word Error Rate (WER) were calculated to measure the performance.

4. EasyOCR
EasyOCR is a deep learning based OCR system. It uses neural networks to detect and recognize text. EasyOCR was tested using both the pretrained model and a fine-tuned version.

4.1 Pretrained EasyOCR
First, the pretrained EasyOCR English model was used directly on the test images. The extracted text was cleaned and evaluated using CER and WER, similar to PyTesseract.

4.2 Transfer Learning
To apply transfer learning, the pretrained EasyOCR recognition model was adapted using a small set of training images. The detection part of the model was kept unchanged, and only the recognition part was adapted. This was done to save time and computational resources.

4.3 Evaluation After Training
After fine-tuning, EasyOCR was tested again on the same test images. The new results showed better accuracy compared to the pretrained model.

5. Results and Comparison
PyTesseract produced higher error rates, especially on complex real-life documents. EasyOCR performed better due to its deep learning approach. Fine-tuning EasyOCR further improved its performance even with a small training dataset.

6. Conclusion
In this assignment, two OCR methods were implemented and compared. PyTesseract is simple

and fast but struggles with complex documents. EasyOCR performs better and gives more accurate results. Overall, deep learning based OCR systems are more effective for real-world documents.