

Nexthikes IT solutions

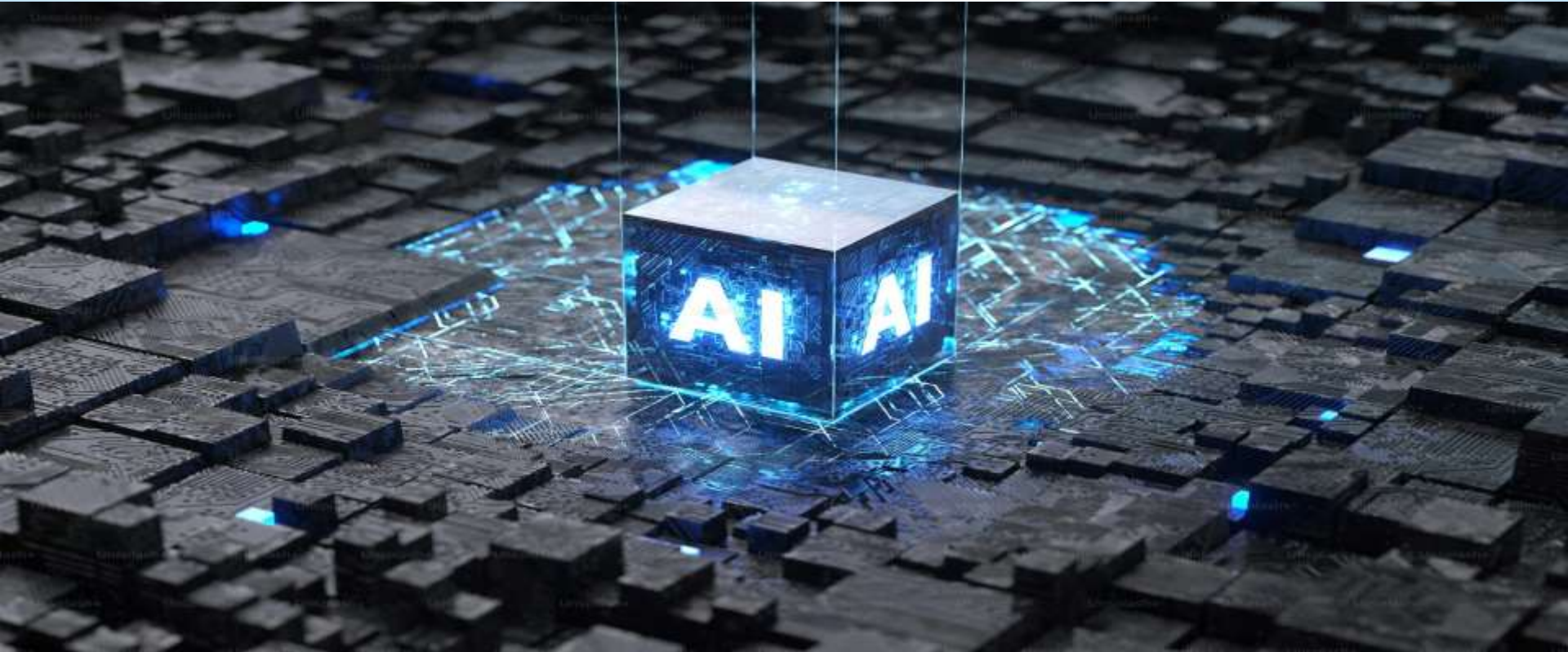
PROJECT 10

Priyanshu Kumar

11 Nov 2025

TITLE:

Custom Object Character Recognition (OCR) using YOLO v3 and Tesseract



- Manually entering data takes too much time and often leads to mistakes.
- Lab reports come in many different layouts with no standardized structure.
- Traditional OCR tools struggle to correctly extract key details like patient names, test types, and result values.
- Variations in fonts, formatting, and scan quality further reduce recognition accuracy.
- There is a need for a smart, automated system that can reliably identify and extract specific fields from lab documents.



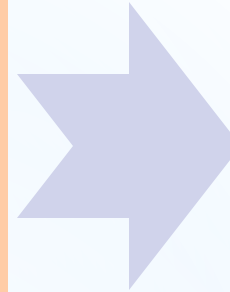
Problem Statement — Custom OCR

OBJECTIVE

▣ **Develop a Custom OCR Pipeline**
Combine YOLOv3 for region detection and Tesseract OCR for text extraction.



⚙️▣ **Convert Image Data into Editable Formats**
Generate CSV / PDF reports for easy digital use and analysis.



✅ **Ensure Accuracy & Efficiency**
Optimize preprocessing (grayscale, thresholding, noise removal) to enhance OCR accuracy.



📄▣ **Detect Key Text Regions Automatically**
Identify structured elements – *Patient Name, Test Names, and Results* – regardless of layout.



☁️▣ **Deploy on Cloud Platforms**
Implement and test using AWS SageMaker, Google Drive, or Streamlit Cloud for real-time accessibility.

OCR Pipeline

1 Data Collection & Annotation

Gather sample lab reports (scanned/PDF/images).

Label regions (Name, Age, Test Name, T3, T4, TSH) using LabelImg.

2 YOLOv3 Model Training

Train on labeled data for object detection.

Output: bounding boxes around text regions.

3 Region Extraction

Crop detected areas using YOLO bounding box coordinates.

4 Text Recognition (Tesseract OCR)

Extract text from cropped regions.

Apply preprocessing: grayscale, thresholding, denoising.

5 Post-Processing & Data Structuring

Clean text, map to fields (Patient Name, Test Name, etc.).

Store structured info in JSON/CSV.

6 Report Generation & Deployment

Generate downloadable PDF reports.

Deploy app using **Streamlit** for real-time use.

Data Preparation

Dataset Collection:
Gathered lab report images
containing patient and test
information.

Data Split:

- 80% of images used for training
- 20% reserved for testing and validation

Preprocessing:

- Resized all images to 416×416 pixels (YOLO input size).
- Enhanced clarity using brightness/contrast adjustments.
- Ensured proper image–annotation matching.

Annotation Process:

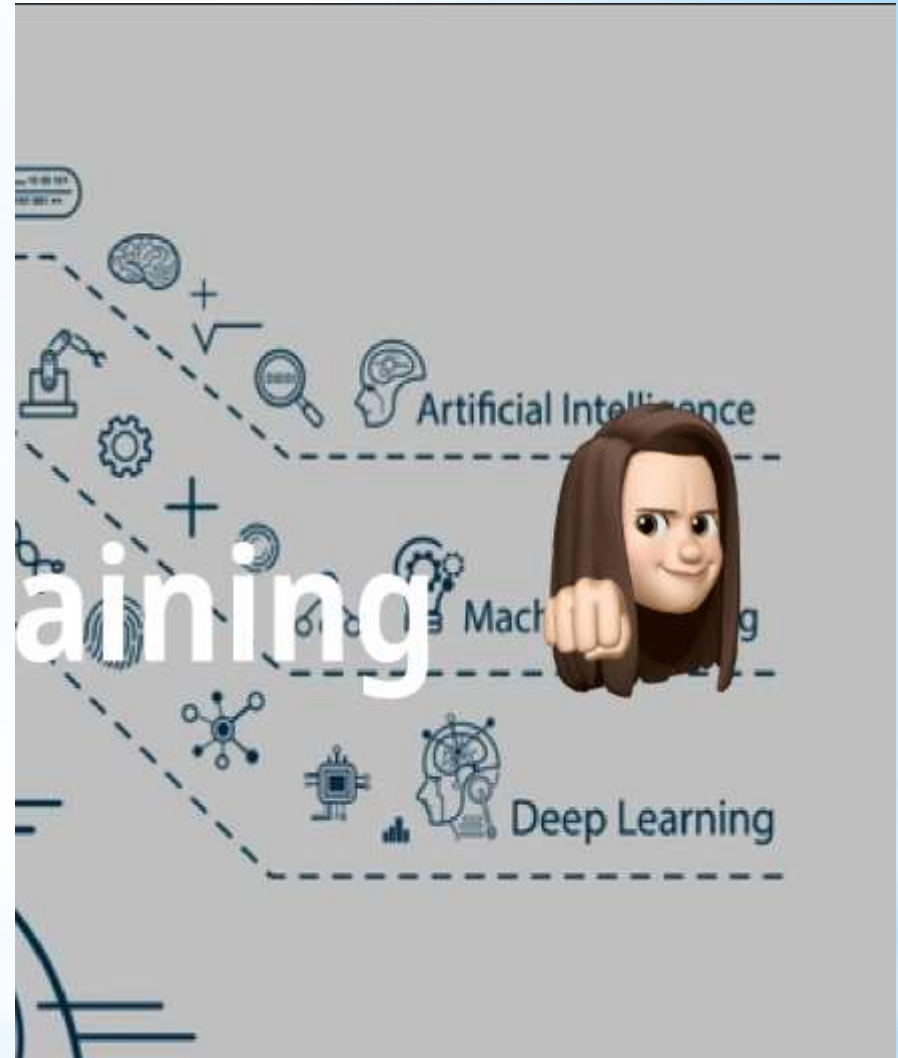
Used *Labellmg* to annotate key fields such as:

- Patient Name
- Age/Sex
- Sample ID
- Test Name
- T3, T4, and TSH values

Annotation Format:
Exported annotations in YOLO format:

class_id x_center y_center
width height

- * **Model Used:**
YOLOv3 (You Only Look Once v3) — a real-time object detection model.
- * **Training Environment:**
- * Google Colab GPU for accelerated training
- * Dependencies: OpenCV, PyTorch, and Darknet
- * **Configuration:**
- * Input size: 416 × 416 pixels
- * Batch size: 16
- * Learning rate: 0.001
- * Epochs: 100
- * **Training Steps:**
- * Loaded annotated dataset (images + YOLO .txt files)
- * Modified YOLO config and .data files
- * Trained the model using Darknet
- * Saved best weights to models/yolov3_custom.weights
- * **Output:**
- * Bounding boxes for each target field (Patient Name, T3, T4, TSH, etc.)
- * High detection accuracy on unseen reports



Model Training YOLOv3

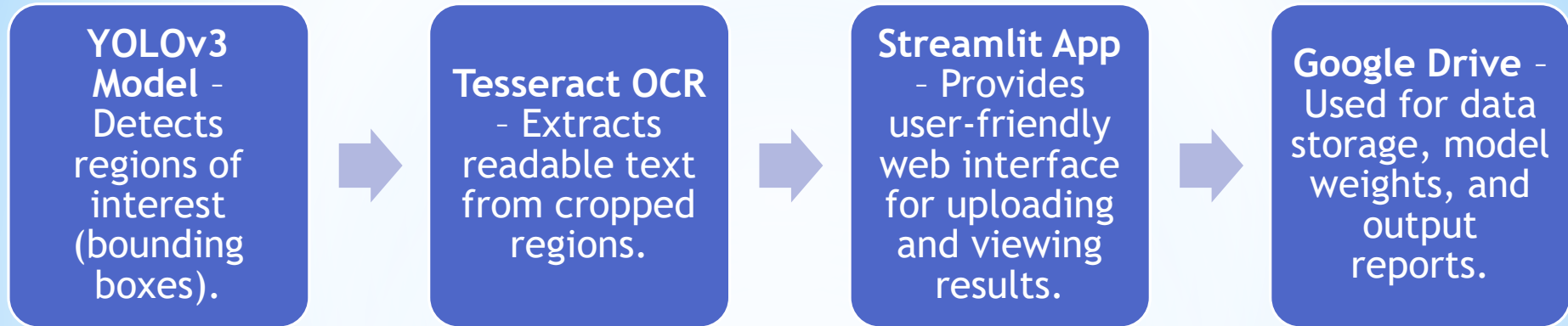
OCR & Text Extraction (Tesseract)

Image → Preprocess → OCR → Structured Output

- **Input:**
Cropped regions (e.g., Patient Name, T3, T4, TSH).
- **Preprocessing:**
 - Resize and convert to **grayscale**.
 - Apply **Gaussian blur** to remove noise.
 - Use **Otsu's thresholding** for text clarity.
- **Binary Conversion:**
Convert to **black text on white background** for better OCR accuracy.
- **Text Extraction:**
Pass preprocessed images to **Tesseract OCR** for recognition.
- **Output:**
Structured text data (JSON/CSV) containing recognized fields.

System Architecture & Integration Flow

Key Components



Workflow Overview

- User uploads a lab report through the **Streamlit UI**.
- Image is passed to the **YOLOv3 model** to detect text fields.
- Detected fields are cropped and sent to **Tesseract OCR** for text extraction.
- Extracted text is structured (JSON, CSV, or PDF).
- Results are automatically stored and downloadable via **cloud integration**

Local Deployment in VS Code

Development Setup

IDE Used: Visual Studio Code (VS Code)

Programming Language: Python 3.9

Environment: Virtual Environment (.venv) for package isolation

Frameworks & Libraries:

- **OpenCV** – image processing
- **pytesseract** – OCR text extraction
- **streamlit** – web app interface
- **reportlab** – PDF generation

Results and Output Visualization

OCR Processing Summary

Input: Lab report images uploaded through Streamlit

YOLOv3 detected key regions (Patient Name, Test Name, T3, T4, TSH)

Cropped regions passed to Tesseract for text extraction

Results displayed instantly and saved as CSV/PDF

INSIGHT AND CONCLUSION

Summary

Successfully developed a Custom Object Character Recognition (OCR) system using YOLOv3 and Tesseract.

The model efficiently detects and reads key information from lab reports such as: Patient Name, Test Name, T3, T4, and TSH values.

Fully implemented and tested locally in VS Code with a Streamlit-based interface for real-time results and PDF generation.

THANK YOU