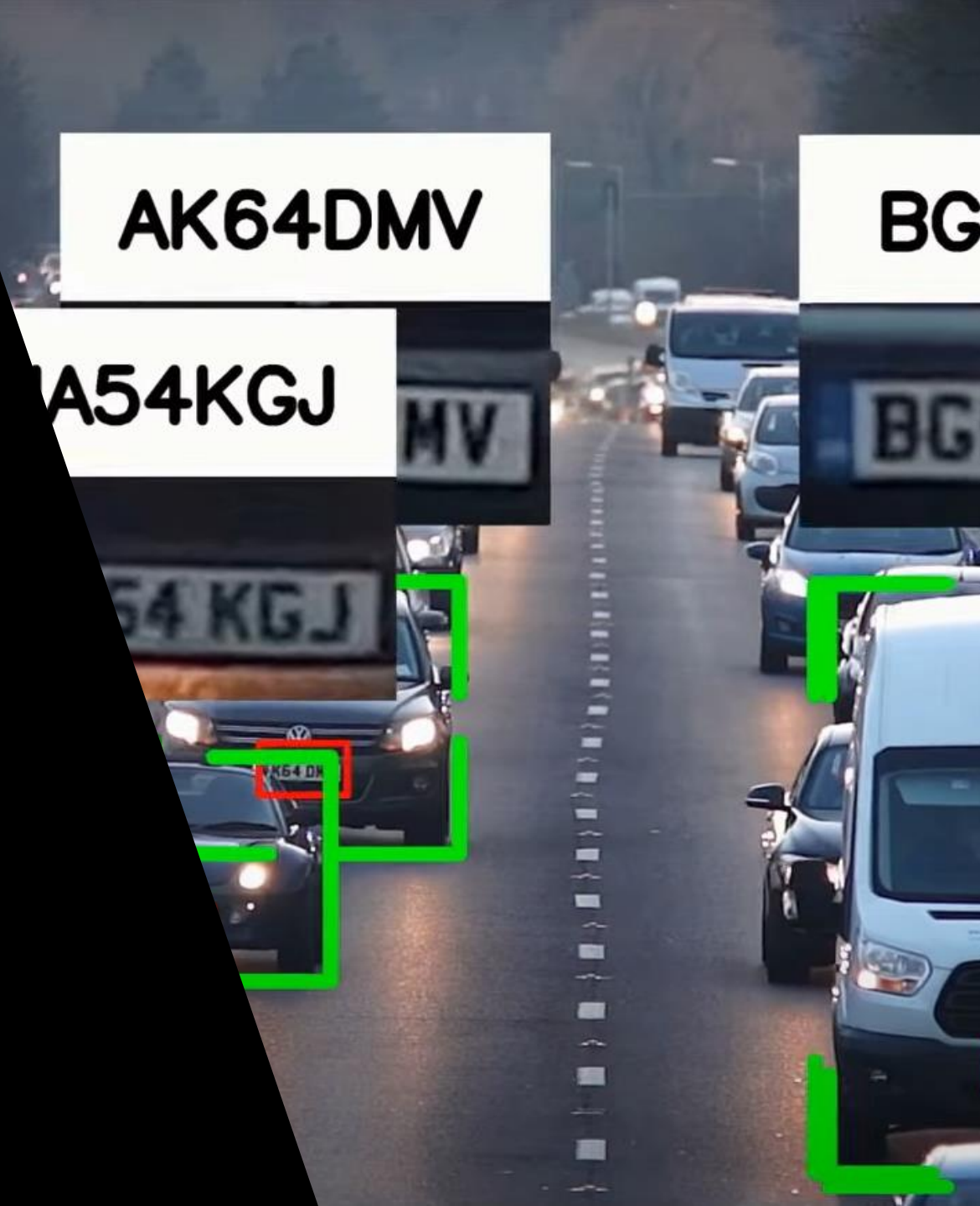
AUTOMATIC NUMBER PLATE RECOGNITION USING DEEP LEARNING

Leveraging Python, YOLOv8, and EasyOCR for Detection and Reading



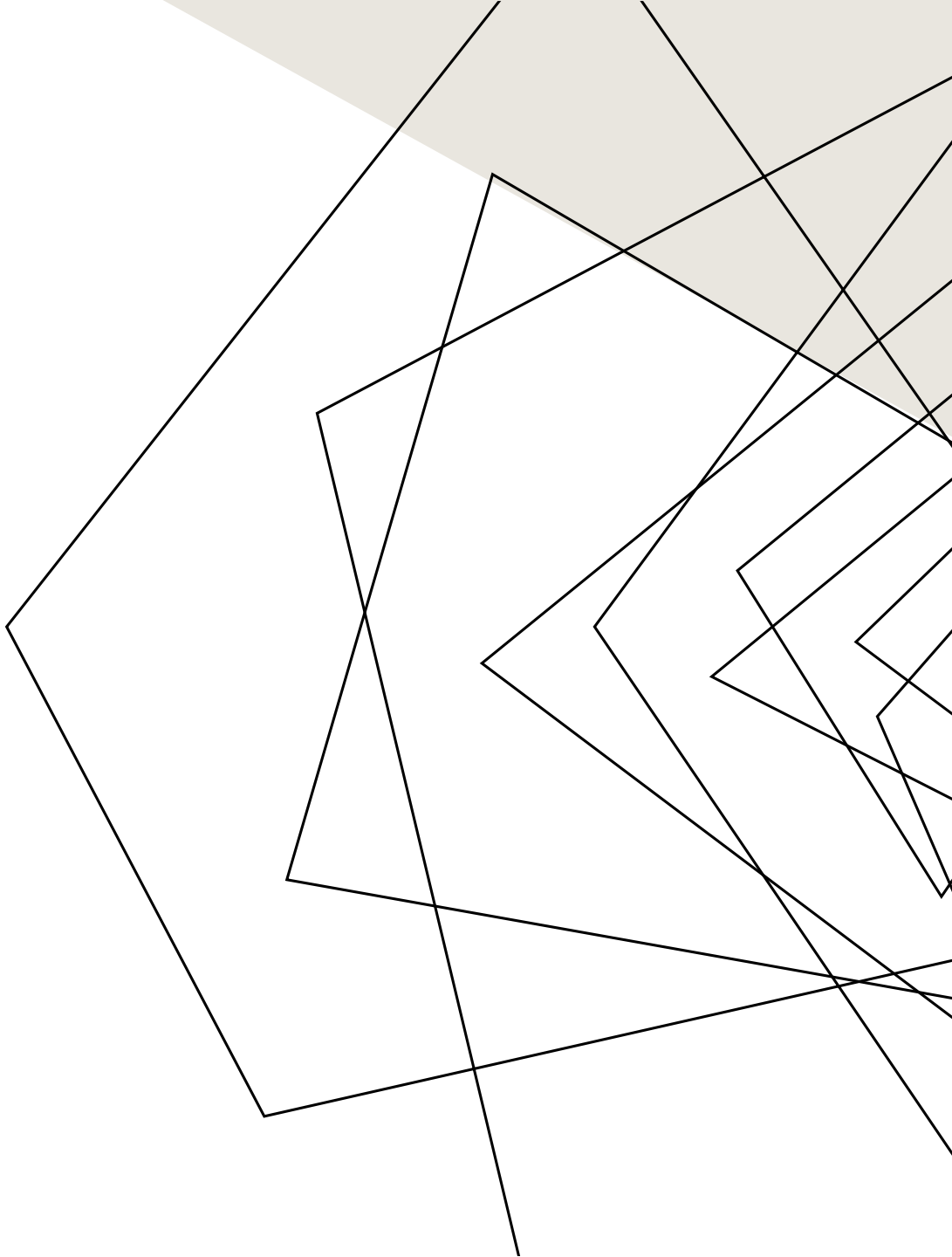
TEAM CONTRIBUTIONS:

Ahmed Omar Ali

- **Dataset Preparation:** Managed test.csv and test_interpolated.csv, ensuring proper annotations.
- **YOLOv8 Implementation:** Trained YOLOv8, tuned parameters, and analyzed model accuracy.
- **System Design:** Developed core detection logic in main.py and contributed to the system workflow.
- **Presentation Preparation:** Documented findings and prepared the slides.

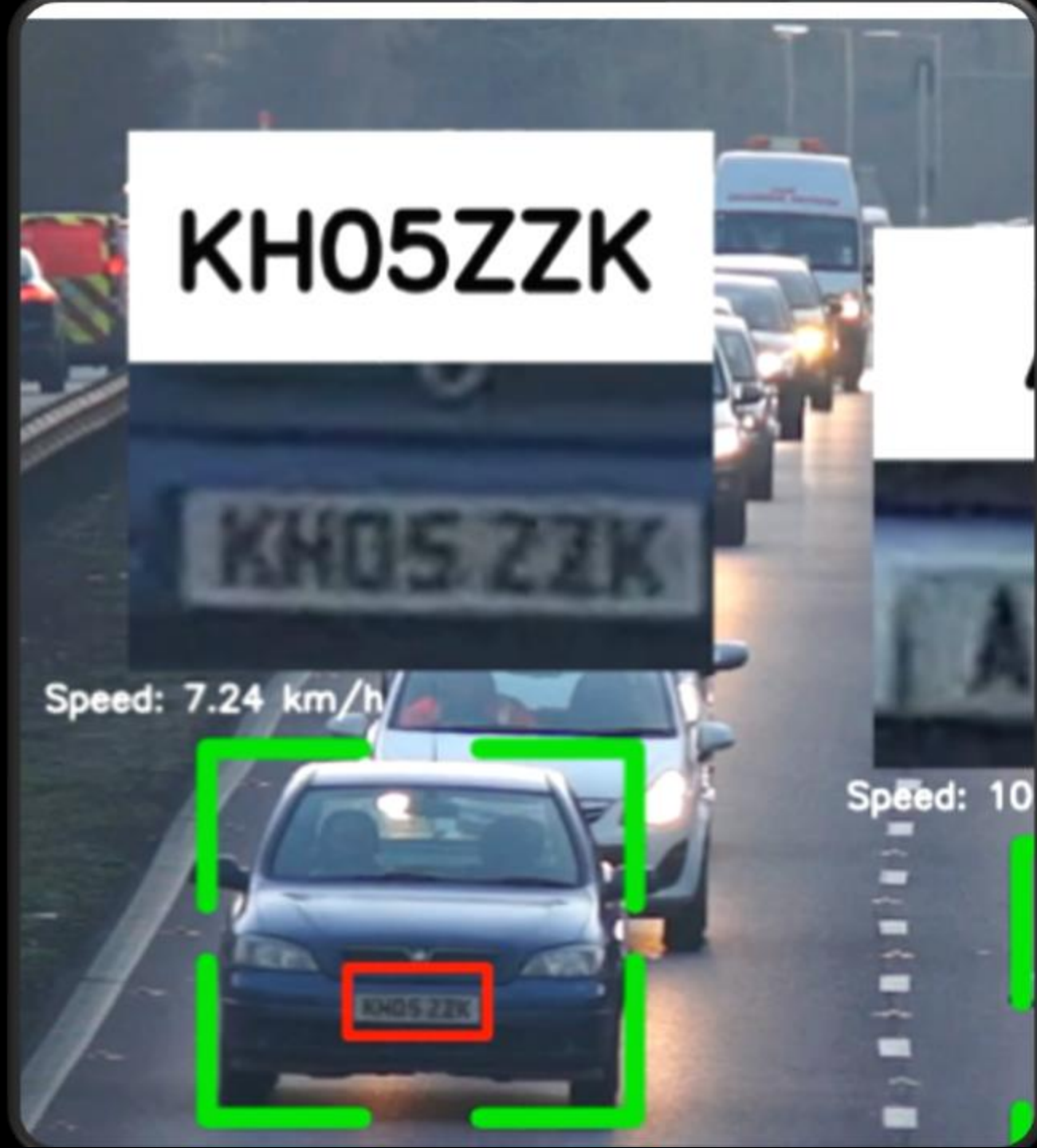
Sharmarke Mohamed Ibrahim

- **OCR Integration:** Integrated EasyOCR for license plate text recognition, addressing accuracy challenges.
- **Visualization and Tracking:** Used `visualize.py` for results and implemented `sort.py` for tracking multiple plates.
- **Presentation Preparation:** Documented findings, focusing on challenges and conclusions.



Content Overview:

- **INTRODUCTION**
- **OBJECTIVES**.
- **TOOLS AND LIBRARIES:** HIGHLIGHT TECHNOLOGIES USED (PYTHON, YOLOV8, EASYOCR).
- **DATASET OVERVIEW:** DESCRIPTION OF DATA PREPARATION AND EXAMPLES.
- **SYSTEM ARCHITECTURE:** WORKFLOW OF THE DETECTION AND OCR INTEGRATION.
- **YOLOV8 DETECTION:** KEY DETAILS ABOUT THE DETECTION PROCESS AND RESULTS.
- **EASYOCR EXTRACTION:** OCR PROCESS, CHALLENGES, AND TEXT RECOGNITION OUTPUTS.
- **IMPLEMENTATION DETAILS:** OVERVIEW OF MAJOR CODE FILES AND THEIR FUNCTIONS.
- **VISUALIZATION OF RESULTS:** SHOWCASE DETECTED PLATES AND OCR OUTPUTS.
- **EVALUATION AND PERFORMANCE:** ACCURACY METRICS, CHALLENGES, AND IMPROVEMENTS.
- **APPLICATIONS:** REAL-WORLD USES AND POTENTIAL FUTURE ENHANCEMENTS.
- **CONCLUSION**



INTRODUCTION

The project focuses on detecting vehicles, extracting license plates, and reading numbers. Real-world applications include traffic management, toll automation, and law enforcement.

OBJECTIVES

- Detect and locate number plates in images/videos using YOLOv8.
- Recognize and extract text using EasyOCR.
- Ensure high accuracy and real-time processing.

TOOLS AND LIBRARIES

- Technologies Used:

1-Python for programming

2-YOLOv8 for object detection

3-EasyOCR for text recognition

4-OpenCV for image processing

Advantages: YOLOv8’s speed and EasyOCR’s multilingual capabilities.

DATASET OVERVIEW

test.csv (Raw Data):

frame_nmr: Frame number.

car_id: Unique car ID.

car_bbox: Car bounding box.

license_plate_bbox: License plate bounding box.

Dataset Split:

Total Images: 24242
Train set: 21174 images
Valid set: 2048 imsges
Test set: 1020 images

`test_interpolated.csv` (Processed Data):

- Interpolated bounding boxes for missing frames. Includes:

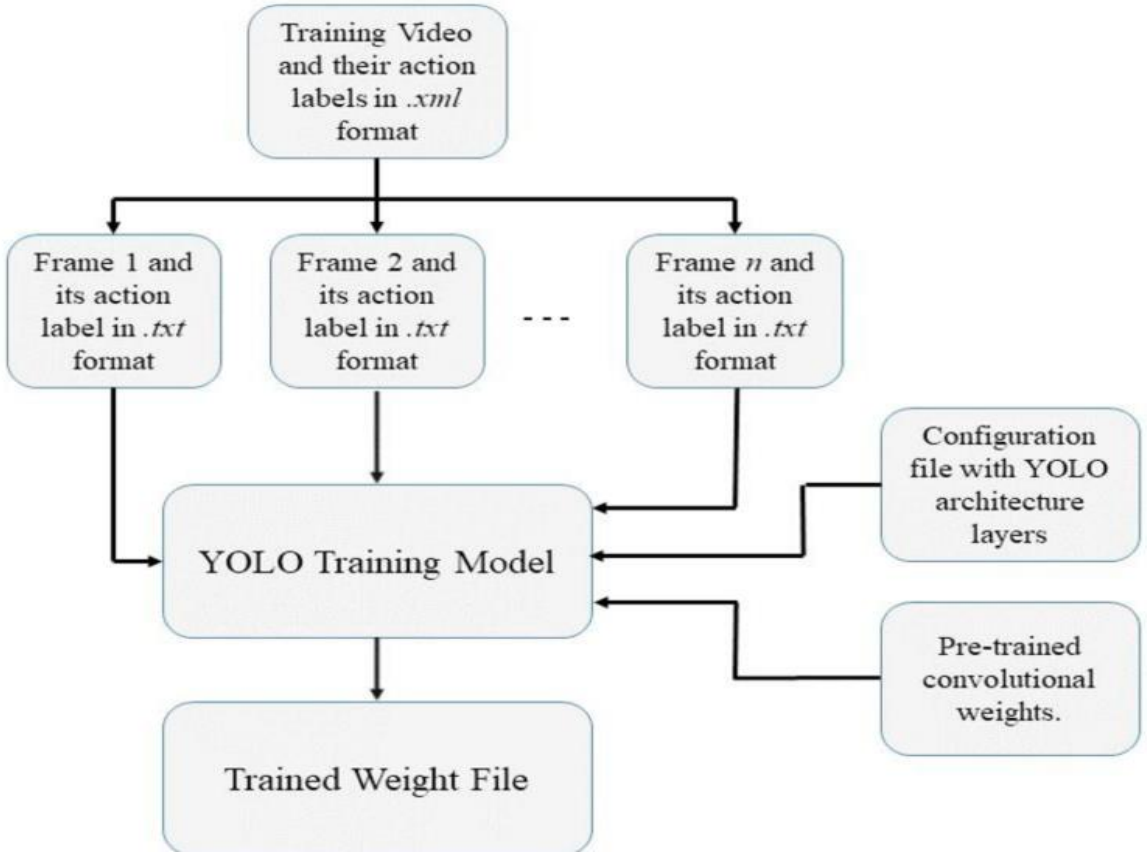
`license_plate_bbox_score`: Detection confidence.`license_number`: Extracted text.
`license_number_score`: OCR confidence.

Key Metrics:

- Total Frames: 702
- Total Cars Detected: *32*
- Missing Frames Filled: **(Interpolated):** 1,179

Dataset source: https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/4

SYSTEM ARCHITECTURE



- Workflow:
  1. Input image or video
  2. YOLOv8 for vehicle and plate detection
  3. Extract plate regions
  4. OCR for reading license numbers.

CODE IMPLEMENTATION

The project is implemented across multiple scripts:

- Detection (`main.py`)
- Data Processing (`add_missing_data.py`)
- Visualization (`visualize.py`)
- Tracking (`sort.py`).

Key snippets are highlighted.

YOLOV8 FOR OBJECT DETECTION

- YOLOv8 is used for its speed and accuracy. Training data and performance metrics are highlighted. Sample images include bounding boxes for detected cars and plates.

YOLO (FOR VEHICLE TRACKING)

This part of the code identifies which vehicle corresponds to the detected license plate using YOLO by comparing the bounding box coordinates (x1, y1, x2, y2) of the license plate with the vehicle's bounding box (xcar1, ycar1, xcar2, ycar2). If a match is found, it assigns the corresponding car_id.

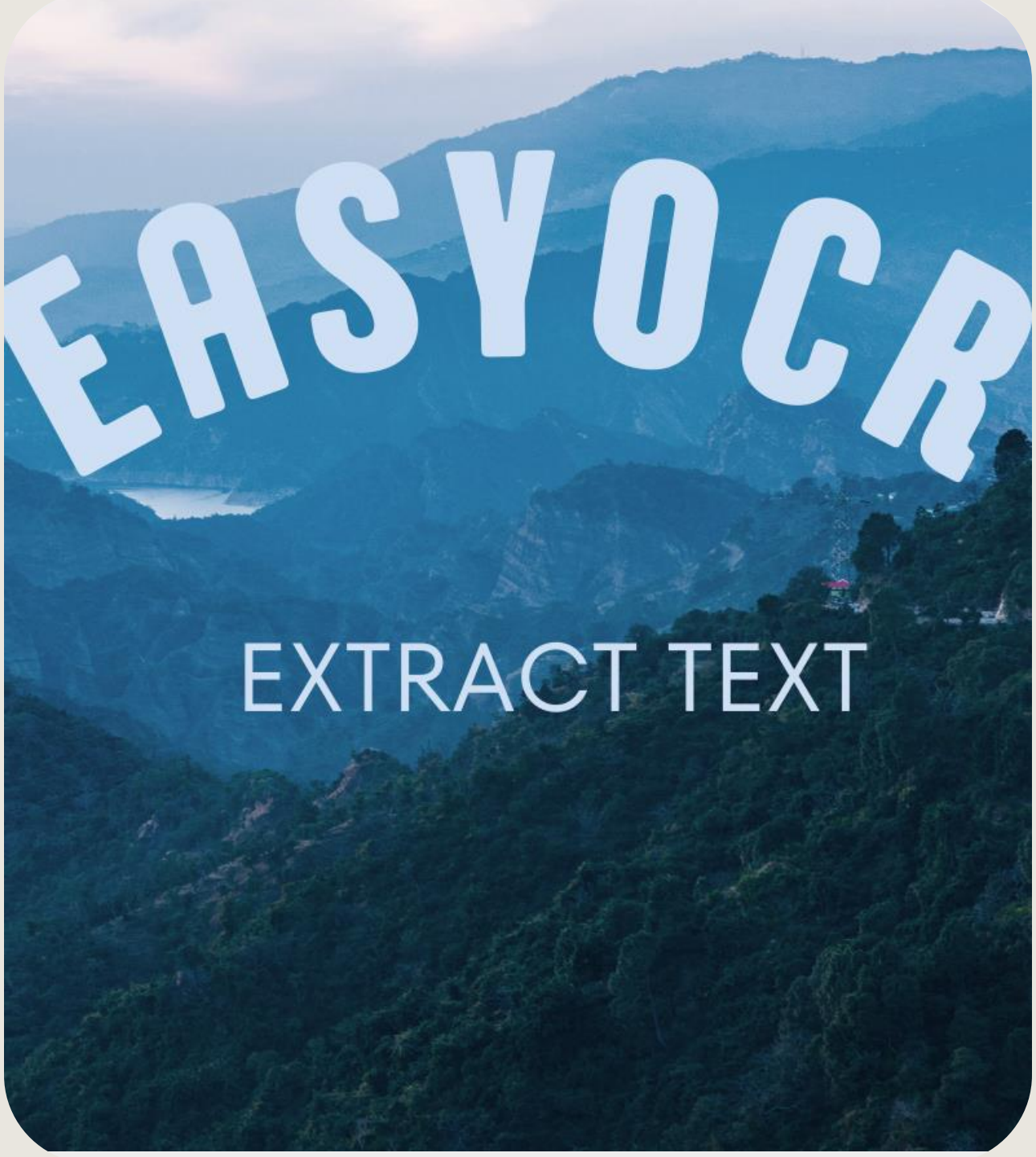
```
def get_car(license_plate, vehicle_track_ids):
    x1, y1, x2, y2, score, class_id = license_plate

    foundIt = False
    for j in range(len(vehicle_track_ids)):
        xcar1, ycar1, xcar2, ycar2, car_id = vehicle_track_ids[j]

        if x1 > xcar1 and y1 > ycar1 and x2 < xcar2 and y2 < ycar2:
            car_indx = j
            foundIt = True
            break

    if foundIt:
        return vehicle_track_ids[car_indx]

    return -1, -1, -1, -1, -1
```

Content Overview:

- **INTRODUCTION**
- **OBJECTIVES**.
- **TOOLS AND LIBRARIES:** HIGHLIGHT TECHNOLOGIES USED (PYTHON, YOLOV8, EASYOCR).
- **DATASET OVERVIEW:** DESCRIPTION OF DATA PREPARATION AND EXAMPLES.
- **SYSTEM ARCHITECTURE:** WORKFLOW OF THE DETECTION AND OCR INTEGRATION.
- **YOLOV8 DETECTION:** KEY DETAILS ABOUT THE DETECTION PROCESS AND RESULTS.
- **EASYOCR EXTRACTION:** OCR PROCESS, CHALLENGES, AND TEXT RECOGNITION OUTPUTS.
- **IMPLEMENTATION DETAILS:** OVERVIEW OF MAJOR CODE FILES AND THEIR FUNCTIONS.
- **VISUALIZATION OF RESULTS:** SHOWCASE DETECTED PLATES AND OCR OUTPUTS.
- **EVALUATION AND PERFORMANCE:** ACCURACY METRICS, CHALLENGES, AND IMPROVEMENTS.
- **APPLICATIONS:** REAL-WORLD USES AND POTENTIAL FUTURE ENHANCEMENTS.
- **CONCLUSION**

EASYOCR FOR TEXT EXTRACTION

- EasyOCR reads text from detected plate regions. Challenges include poor image quality and obstructed plates. Examples show successful and failed OCR readings.



EASYOCR (FOR LICENSE PLATE TEXT RECOGNITION)

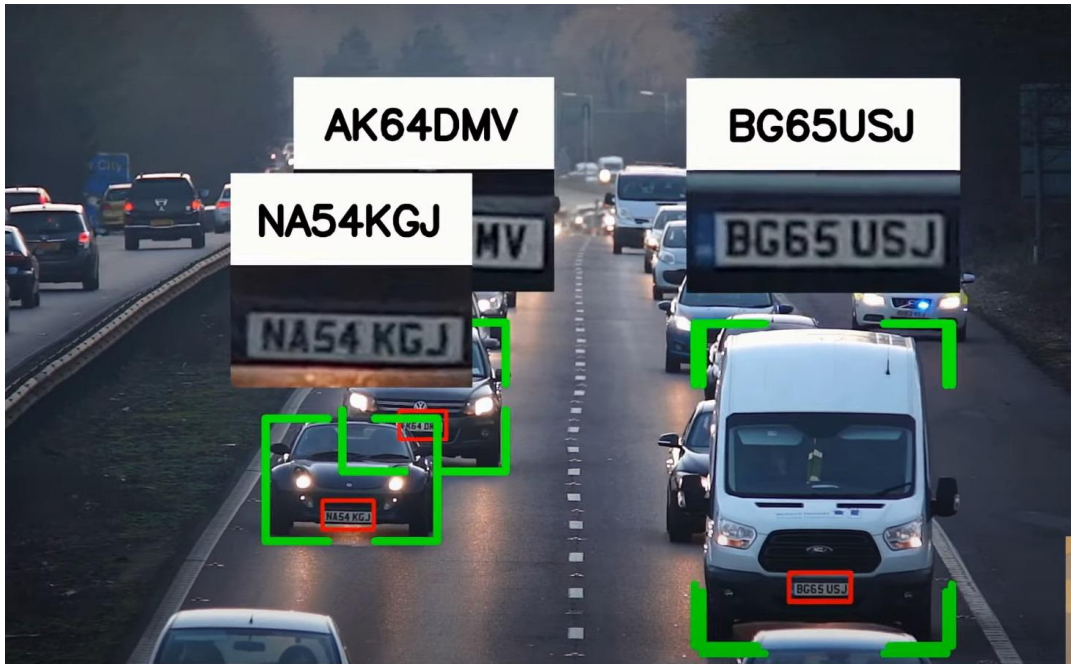
This code snippet uses EasyOCR to read and recognize text from the cropped license plate image (license_plate_crop). It checks if the detected text complies with the required format and returns the formatted license plate text along with its confidence score.

```python
reader = easyocr.Reader(['en'], gpu=False)
def read_license_plate(license_plate_crop):
    detections = reader.readtext(license_plate_crop)

    for detection in detections:
        bbox, text, score = detection

        text = text.upper().replace(' ', '')

        if license_complies_format(text):
            return format_license(text), score

    return None, None
```

VISUALIZATION OF RESULTS

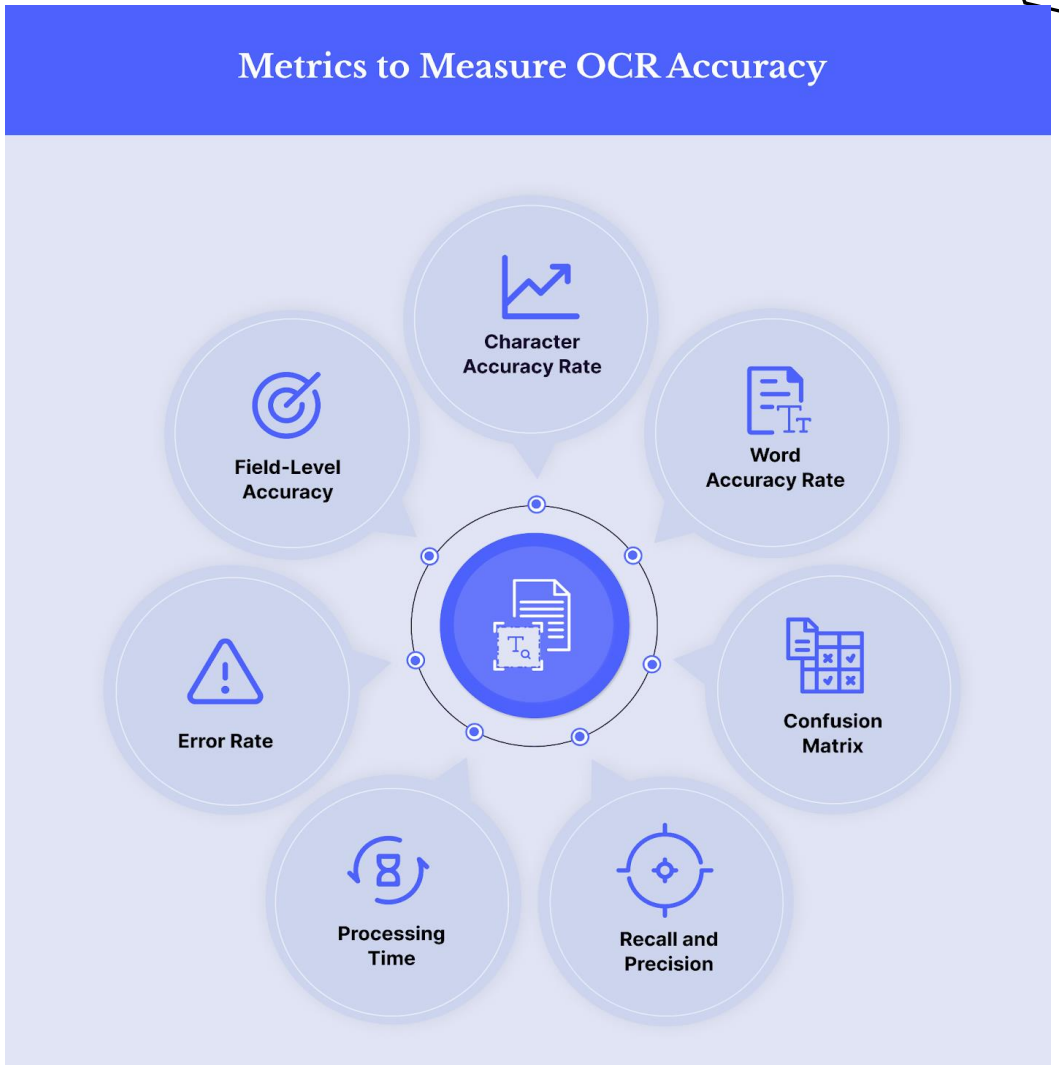
Before

After



EVALUATION AND PERFORMANCE

- Metrics include detection accuracy, OCR accuracy, and system performance. Challenges include false positives and dataset limitations.



APPLICATIONS

- Use cases:

- Traffic enforcement
- Toll automation systems
- Parking management

- Future Enhancements:

- Multilingual plates
- Better low-light detection.

CONCLUSION

- Successfully integrated YOLOv8 and EasyOCR for real-time recognition.
- Challenges and potential improvements identified.

- Takeaway: Importance of this technology in various industries.