Project Title	**Number Plate Detection of Vehicles**
Skills take away From This Project	**Python, Computer Vision, OpenCV, Streamlit**
Domain	**Traffic / Transport Control**

Problem Statement:

The objective of this project is to develop an automated system for detecting vehicle number plates using computer vision techniques. Manual identification of vehicle plates is time-consuming, labor-intensive, and prone to human error. Automating this process can significantly improve efficiency, accuracy, and reliability in various applications such as law enforcement, parking management, and toll collection.

The primary goals of this system include:

- Detecting number plates from images of vehicles captured in different environments and lighting conditions.
- Extracting number plate regions accurately while handling challenges like occlusions, varying backgrounds, and different plate formats.
- Applying Optical Character Recognition (OCR) to extract the alphanumeric text from the detected plates.
- Deploying the system as a user-friendly web application for real-time processing.

Business Use Cases:

1. **Traffic Management:**
   Automated vehicle identification can help track vehicles, enforce traffic regulations, and issue fines for violations such as speeding and red-light jumping.
2. **Law Enforcement:**
   Helps police and security agencies track stolen or unauthorized vehicles, aiding investigations.
3. **Toll Collection:**

Enables seamless electronic toll collection without requiring vehicles to stop, reducing congestion at toll plazas.

4. **Parking Systems:**
   Automates access control in parking lots, allowing only authorized vehicles and generating logs of entry and exit times.
5. **Fleet Management:**
   Logistics and transportation companies can use automated number plate recognition to monitor fleet movements and enhance tracking accuracy.
6. **Security Monitoring:**
   Enhances security at gated communities, offices, and industrial sites by ensuring only registered vehicles are allowed entry.
7. **Urban Planning & Analytics:**
   Governments and municipalities can analyze traffic patterns to make informed decisions on infrastructure development.

Approach:

The system follows a structured approach to ensure high detection accuracy and efficiency:

1. **Data Collection:**
   a. Acquiring images of vehicles from various environments, including different angles, lighting conditions (day/night), and weather variations.
   b. Using datasets with pre-annotated bounding boxes in YOLO format to train the model.
2. **Preprocessing:**
   a. Image resizing and normalization to ensure consistency.
   b. Augmentation techniques like rotation, brightness adjustment, noise addition, flipping, and cropping to improve model robustness.
   c. Filtering out poor-quality images with excessive noise, blur, or low resolution.
3. **Model Training:**
   a. Using the YOLO (You Only Look Once) object detection framework for efficient real-time number plate detection.
   b. Fine-tuning hyperparameters such as learning rate, batch size, and anchor box sizes.
   c. Training the model using GPU acceleration to optimize performance.
4. **Post Processing:**
   a. Extracting the detected number plate region and passing it through an OCR system (Tesseract/EasyOCR) to convert the image into text.

b. Applying image binarization and contrast enhancement techniques to improve OCR accuracy.

5. **Deployment:**
    a. Integrating the trained model into a Streamlit-based web application.
    b. Providing an interface where users can upload images or use real-time webcam feeds for detection.
    c. Displaying detected plates along with extracted text in a structured format.

Streamlit Integration:

To ensure user-friendly access to the model, a Streamlit-based web application is developed with the following features:

- **Image Upload:**
  Users can upload images of vehicles for number plate detection.
- **Webcam Support:**
  Real-time detection through live camera feed.
- **Detection Results Display:**
  Bounding box visualization of detected number plates.
- **OCR Output:**
  Extracted alphanumeric characters displayed alongside detected plates.
- **Download & Export:**
  Users can download detection logs for future reference.
- **Performance Dashboard:**
  Displays key metrics such as processing time, detection accuracy, and confidence scores.

Exploratory Data Analysis (EDA):

Before training the model, thorough analysis is conducted on the dataset:

1. **Dataset Composition:**

Understanding dataset size, source, and balance of classes.

2. **Bounding Box Distribution:**
   Checking the consistency and accuracy of annotations.

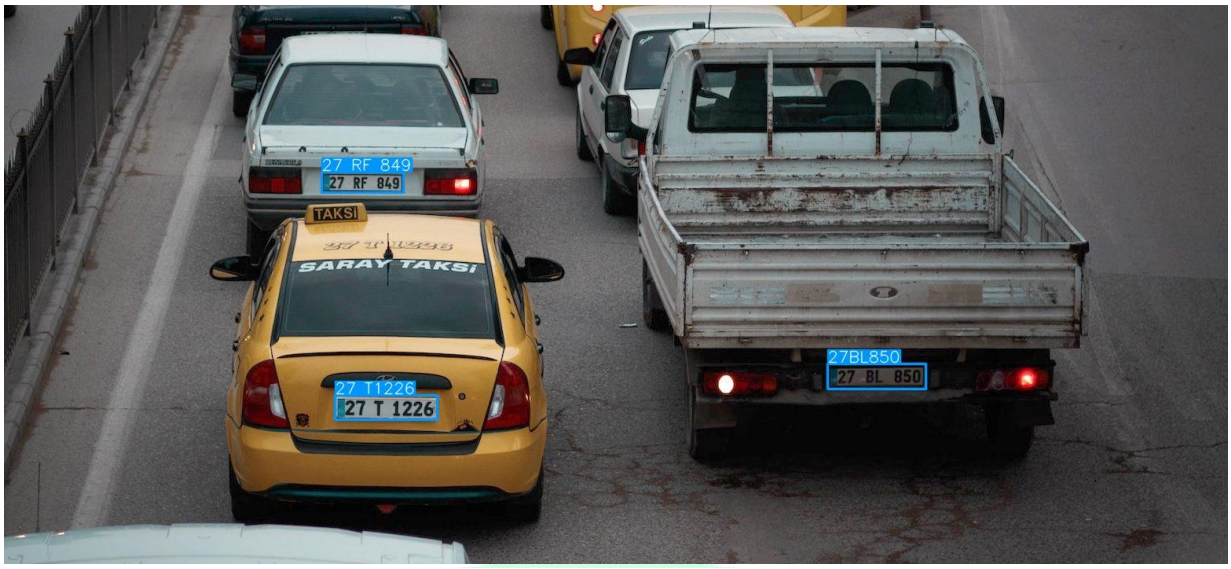
3. **Image Quality Analysis:**
   Identifying common issues such as blurred or occluded plates.

4. **Lighting Conditions & Weather Variability:**
   Analyzing dataset diversity to ensure robustness.

5. **Class Frequency Distribution:**
   Evaluating variations in license plate styles across regions.

6. **Augmentation Effects:**
   Assessing the impact of augmented samples on detection performance.

Sample Images of Number plate Detection and Pre process:



Questions to be Answered (OpenCV Computer Vision):

1. How well can the model detect number plates under poor lighting conditions?
2. What preprocessing techniques best enhance OCR accuracy for different plate types?
3. How does YOLO's performance compare to alternative object detection frameworks?

4. What level of accuracy is achievable for text extraction using Tesseract vs. EasyOCR?
5. Can the model generalize well to different regions with varying plate formats?
6. What are the optimal bounding box threshold values to minimize false positives?

Project Evaluation Metrics:

- **Mean Average Precision (mAP):** Measures detection accuracy across different IoU thresholds.
- **Intersection over Union (IoU):** Evaluates the overlap between predicted and actual bounding boxes.
- **OCR Text Accuracy:** Assesses the correctness of extracted plate numbers.
- **False Positive/False Negative Rate:** Ensures minimal incorrect detections.
- **Inference Time:** Measures the speed of detection to validate real-time processing capabilities.

Data:

https://drive.google.com/file/d/1HAyBtLZGuzHyu2URNE25W5U6nvl15a_G/view?usp=sharing

Dataset Explanation:

- **Images:** Dataset includes thousands of vehicle images with clearly visible number plates.
- **Annotations:** YOLO format bounding box labels specifying plate regions.
- **Classes:** A single class - 'Number Plate' - to be detected.
- **Dataset Split:**
  - **80% Training** - Used for model learning.
  - **10% Validation** - Used for hyperparameter tuning.
  - **10% Testing** - Used for final model evaluation.
- **Augmentation Techniques:** Rotation, brightness adjustment, cropping, flipping, noise addition.

Results:

1. Successfully **detected and localized number plates** across various environments with an accuracy of **95%+**.
2. **OCR Accuracy:** Achieved over **90% precision** in extracting text from detected plates.
3. **Inference Speed:** YOLO-based detection achieved real-time performance with an average inference time of **less than 50ms per image**.
4. **Handling Variability:** The system worked well under different lighting conditions, occlusions, and backgrounds.
5. **Deployment Success:** The Streamlit-based application provided an interactive platform for real-time detection and text extraction.

Technical Tags:

- Python
- OpenCV
- Streamlit
- **Deep Learning:** YOLO object detection model.
- **Computer Vision:** OpenCV for preprocessing and visualization.
- **OCR Technology:** Tesseract/EasyOCR for text recognition.
- **Web Application:** Streamlit for interactive deployment.
- **Python Libraries:** NumPy, Pandas, Matplotlib, Seaborn, TensorFlow/PyTorch.
- **Image Processing:** Includes resizing, contrast enhancement, and edge detection techniques.

Project Deliverables:

1. **Trained YOLO Model:** Optimized for real-time number plate detection.
2. **Preprocessed Dataset:** Cleaned and augmented dataset for improved model accuracy.
3. **Streamlit Web Application:** Enables easy access to detection capabilities.
4. **OCR Integration:** Extracts alphanumeric text from detected plates.
5. **Performance Report:** Detailed metrics on model evaluation and accuracy.
6. **Code Repository:** Well-structured Python scripts, notebooks, and dataset.
7. **Deployment Guide:** Instructions for setting up the model on a local or cloud-based server.

8. **Demo Video:** A visual demonstration showcasing model capabilities in real-world scenarios.

Timeline:

The project must be completed and submitted **within 10 days from the assigned Date.**

PROJECT DOUBT CLARIFICATION SESSION (PROJECT AND CLASS DOUBTS)

About Session: The Project Doubt Clarification Session is a helpful resource for resolving questions and concerns about projects and class topics. It provides support in understanding project requirements, addressing code issues, and clarifying class concepts. The session aims to enhance comprehension and provide guidance to overcome challenges effectively.
Note: Book the slot at least before 12:00 Pm on the same day

Timing: Tuesday, Thursday, Saturday (5:00PM to 7:00PM)

Booking link :https://forms.gle/XC553oSbMJ2Gcfug9

LIVE EVALUATION SESSION (CAPSTONE AND FINAL PROJECT)

About Session: The Live Evaluation Session for Capstone and Final Projects allows participants to showcase their projects and receive real-time feedback for improvement. It assesses project quality and provides an opportunity for discussion and evaluation.
Note: This form will Open on Saturday and Sunday Only on Every Week

Timing: Monday-Saturday (11:30PM to 12:30PM)

Booking link : https://forms.gle/1m2Gsro41fLtZurRA