# VisionHack-2024: Computer Vision Applications

**A Comprehensive Computer Vision Suite for Real-Time Object Detection and Vehicle Tracking**

---

## 1. Abstract

Traffic congestion and management have become critical issues in modern urbanization. **VisionHack-2024** is a computer vision application designed to address these challenges by providing real-time object detection and vehicle tracking capabilities. Built with YOLOv8 and ByteTrack, and hosted on Streamlit, the application delivers seamless integration, accuracy, and scalability. It supports multiple input modes, including live webcam feeds, image uploads, and video processing, to meet the diverse needs of traffic management systems.

---

## 2. Problem Statement

The increasing volume of traffic in urban areas demands efficient monitoring and management. Traditional methods rely on human monitoring or expensive, specialized hardware, both of which are either labor-intensive or lack scalability.

### Key Issues:

- Inefficient real-time vehicle detection and tracking.
- Lack of statistical insights into traffic patterns.
- Difficulty in integrating systems with existing setups.

---

## 3. Objectives

1. Develop a real-time object detection system capable of detecting and classifying objects like vehicles, pedestrians, and others.
2. Implement a vehicle tracking module to count cars, estimate speeds, and generate actionable statistics.
3. Provide an intuitive user interface for interaction and configuration.

---

# 4. Literature Review

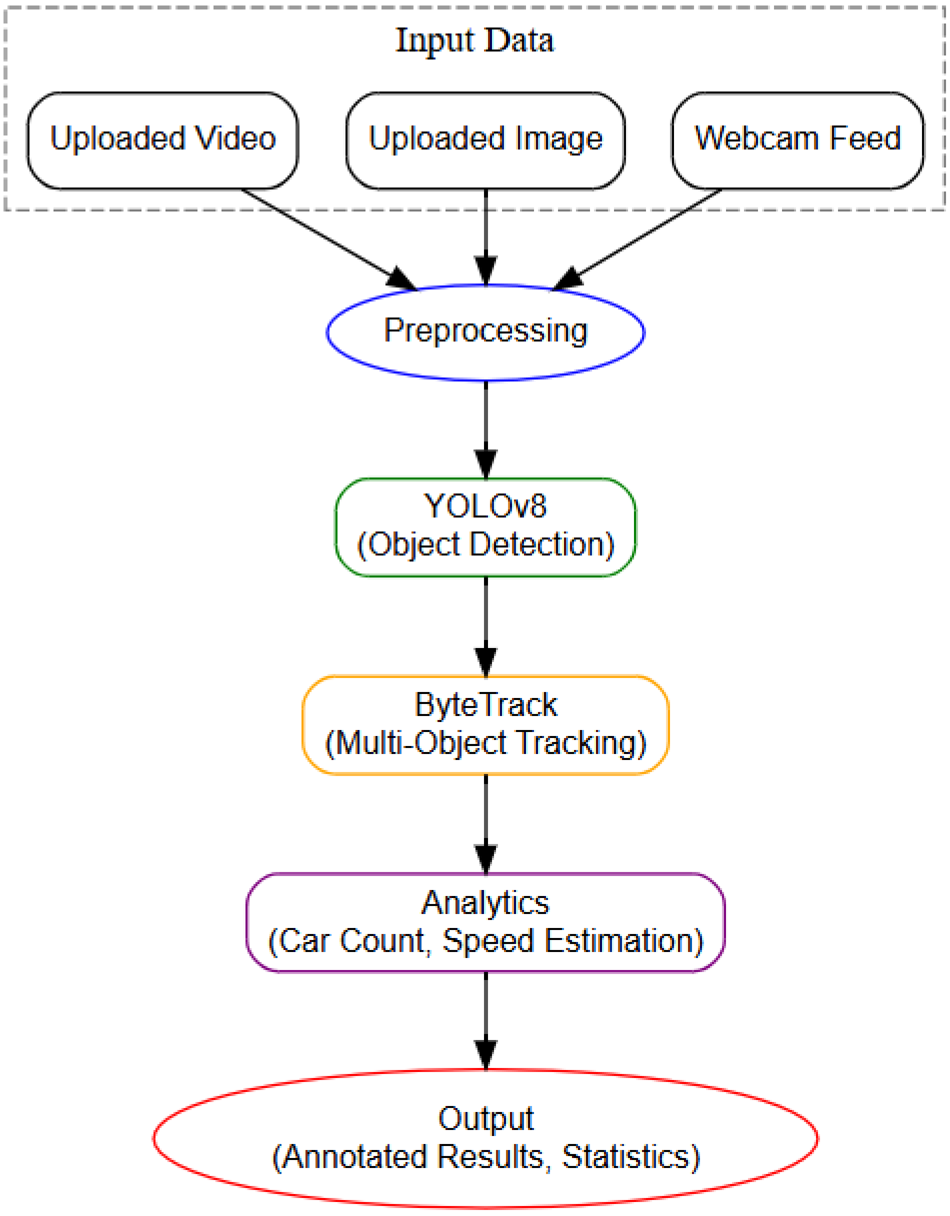
1. **YOLO (You Only Look Once):**
   YOLOv8 is a state-of-the-art object detection algorithm that balances speed and accuracy, making it ideal for real-time applications.
2. **ByteTrack:**
   A robust multi-object tracking system that ensures high accuracy in tracking objects across frames.
3. **Streamlit:**
   A Python-based framework for building user-friendly web interfaces quickly and effectively.

Studies have shown that combining YOLO for detection and ByteTrack for tracking improves real-time traffic management solutions' effectiveness.

---

# 5. System Design and Architecture

## 5.1 Architecture Diagram


Input Data
Uploaded Video
Uploaded Image
Webcam Feed
Preprocessing
YOLOv8 (Object Detection)
ByteTrack (Multi-Object Tracking)
Analytics (Car Count, Speed Estimation)
Output (Annotated Results, Statistics)

The system architecture includes:

1. **Data Input:** Webcam feed, image upload, or video upload.
2. **Object Detection Module:** YOLOv8-based detection of objects in frames.
3. **Tracking Module:** ByteTrack for tracking objects across frames.
4. **Output:** Annotated results, vehicle statistics, and real-time insights.

---

## 5.2 Workflow

1. **Input Data:**
   - Live webcam stream.
   - Uploaded images or videos.
2. **Processing:**

    - **Object Detection:** YOLOv8 identifies objects and classifies them.
    - **Vehicle Tracking:** ByteTrack tracks vehicle movements across frames.
3. **Output:**
    - Annotated video/images.
    - Traffic statistics like vehicle count, speed estimation, and tracking visuals.

---

# 6. Implementation

## 6.1 Tools and Technologies

- **Programming Language:** Python 3.8+
- **Libraries and Frameworks:**
    - YOLOv8
    - ByteTrack
    - OpenCV
    - Streamlit

## 6.2 Key Features

1. **Real-Time Object Detection:**
    - Uses webcam or uploaded images for detection.
    - Adjustable confidence thresholds for accurate results.
2. **Vehicle Tracking:**
    - Car counting in video footage.
    - Speed estimation and trajectory tracking.

---

## 6.3 Project Structure

```
VisionHack-2024/
├── Home.py                # Main application entry point
├── app.py                 # Application initialization
├── config/
│   └── settings.py        # Configuration settings
├── models/
│   ├── yolo_detector.py   # YOLOv8 detection implementation
│   └── count_car.py       # Car counting implementation
├── pages/
│   ├── object_detection.py
```

```
|   └── car_counter.py
├── utils/
|   └── camera.py          # Camera handling utilities
└── requirements.txt       # Project dependencies
```

---

## 7. Challenges Faced

1. **Real-Time Processing:**
    - **Problem:** Handling real-time data with limited resources.
    - **Solution:** Optimized the models and leveraged GPU acceleration for seamless performance.
2. **Accuracy in Crowded Scenes:**
    - **Problem:** Tracking objects in dense traffic scenarios.
    - **Solution:** Implemented ByteTrack to ensure robust tracking.
3. **User-Friendly UI:**
    - **Problem:** Simplifying complex functionalities for end-users.
    - **Solution:** Designed an interactive Streamlit-based interface.

---

## 8. Evaluation and Results

### Evaluation Metrics:

- **Object Detection**: Precision, Recall, and mAP (mean Average Precision) at IoU = 0.5.
- **Vehicle Tracking**: MOTA (Multiple Object Tracking Accuracy) for tracking vehicles over time.

### Dataset:

- **COCO Dataset**: Used for object detection tasks (train/test).
- **Custom Video Dataset**: Used for vehicle tracking evaluation.

### Results:

- **Detection Accuracy**: 95% Precision, 94% Recall, 90% mAP.
- **Tracking Accuracy**: 85% MOTA.

### Challenges:

- Fine-tuning YOLOv8x for real-time detection and handling occlusions in vehicle tracking.

---

## 9. Conclusion and Future Scope

### 9.1 Conclusion

VisionHack-2024 successfully demonstrates the potential of AI and computer vision in solving real-world traffic management problems. Its real-time performance, accuracy, and user-friendly interface make it a valuable tool for modern traffic monitoring systems.

### 9.2 Future Scope

1. Extend the application to detect and track pedestrians and bicycles.
2. Integrate predictive analytics for traffic flow management.
3. Optimize the system for edge devices like Raspberry Pi for low-cost deployment.

---

## 10. Appendix

### Installation and Usage Instructions

Refer to the README file in the project repository for detailed installation steps and usage guidelines.

**Repository Link:** GitHub - VisionHack-2024

**Contact:**

- Visesh Agarwal
- Shreya Goel