International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 12 | December - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

TRAFFIC PREDICTION FOR INTELLIGENCE TRANSPORT SYSTEM USING MACHINE LEARNING

Paripallaka Pradeep
2111cs020343
2111cs020343@mallareddyuniversity.ac.in

Munganda Pradeepthi
2111cs020344
2111cs020344@mallareddyuniversity.ac.in

Asoori Pranavi
2111cs020345
2111cs020345@mallareddyuniversity.ac.in

R. Pranay Kumar
2111cs020347
2111cs020347@mallareddyuniversity.ac.in

K. Praneeth
2111cs020348
2111cs020348@mallareddyuniversity.ac.in

Prof. T.Ramya

Department of CSE(AIML)

MALLA REDDY UNIVERSITY

HYDERABAD

ABSTRACT: This research introduces a Machine Learning-based Traffic Prediction and Signal Optimization System designed for Intelligent Transportation Systems (ITS). The system utilizes real-time traffic data captured via cameras to predict future traffic conditions. It employs data preprocessing and feature engineering, followed by the application of various machine learning models. These models are trained on historical and camera-captured data, evaluated for accuracy, and deployed for real-time traffic prediction. The system not only provides traffic forecasts but also enhances traffic signal control, optimizing stops and flow. This holistic approach aids in traffic management, congestion reduction, and informed decision-making within transportation networks, ultimately enhancing the efficiency of urban mobility.

KEYWORDS: Transport System ,YOLO ,Object Recognition

1.INTRODUCTION

This project aims to provide Intelligent Transport systems relies on the application of machine learning algorithms to analyse vast datasets.By examining historical traffic patterns,infrastructure variable. Machine learning models can forecast future congestion.This predictive capability

enables Transport system to implement proactive strategies for optimizing traffic flow,enhancing safety, and efficiently managing resources. Ultimately, this approach creates more responsive and streamlined intelligent transport networks, mitigation congestion and improving overall system efficiency.

Software Requirements are:

The software and hardware requirements in order to implement Intelligent Transport System are below:

The Software Requirements for this project involves specifying the software tools and technologies necessary for vehicle detection. This includes the requirement for OpenCV for image processing, the YOLO(You Only Look Once) model for object detection, and pyQT for the graphical user interface(GUI).Detail the versions of these tools, any additional libraries needed,and the programming languages employed, such as python.

Hardware requirements are:

The hardware requirements for your vehicle detection project using YOLO, OpenCV, and PyQT would depend on factors such as the scale of your implementation, real-time processing needs, and the size and resolution of the video streams. Here's a general guideline:

1. CPU: A multi-core processor, preferably with a clock speed suitable for real-time processing.Quad-core or higher processors are recommended for improved parallel processing, especially if dealing with high-resolution video feeds.

2. GPU: A dedicated GPU, preferably NVIDIA CUDA-enabled, can significantly accelerate the YOLO model's computations.For real-time processing and optimal performance, consider a mid-range to high-end GPU.

3. Memory (RAM): At least 8GB of RAM is recommended, but higher capacities (16GB or more) would be beneficial for handling large video frames and ensuring smooth processing.

4. Storage: SSD storage is preferred over HDD for faster read and write speeds, contributing to efficient video processing.

5.Display: A standard display is sufficient for development purposes.High-resolution monitors may enhance the user experience when interacting with the PyQT-based GUI.

6. Other Considerations: Network Interface: If your project involves real-time processing of video streams from remote sources, a stable network connection is essential.

Cooling: Depending on the intensity of computations, consider adequate cooling solutions to prevent overheating.

It's crucial to note that these are general recommendations, and the actual hardware requirements can vary based on the specifics of your project, such as the size and complexity of the videos, the desired frame rate, and whether you're dealing with real-time processing. For optimal results, it's recommended to conduct performance testing on your specific hardware setup to ensure smooth execution of the vehicle detection system.

2.EXISTING SYSTEM

In the context of vehicle detection in videos, several existing systems and approaches have been employed. Here are a few notable ones:

1. Traditional Computer Vision Methods:

(**a). Background Subtraction:** This method involves modeling the background of a scene and identifying moving objects as foreground elements.

(b). Haar Cascades: Haar-like features and cascaded classifiers are utilized for object detection, but they may have limitations in accuracy and robustness.

2. Frame Differencing and Optical Flow:

(a). Frame Differencing: Detects changes between consecutive frames, often used for simple motion detection.

(b). Optical Flow: Analyzes the motion of objects in a sequence of frames, providing information about their movement.

3.PROPOSED SYSTEM

The proposed system for the vehicle detection project utilizing YOLO, OpenCV, and PyQT aims to advance current methodologies by combining state-of-the-art object detection with a user-friendly interface. Here are the key features and improvements in the proposed system:

1. YOLO-based Object Detection:

(a). Real-time Processing: Utilizes the YOLO model for efficient and real-time vehicle detection, ensuring high accuracy and speed in identifying vehicles within video streams.

(b). Adaptability: Handles various vehicle types, sizes, and orientations, making it versatile for diverse scenarios.

2. OpenCV Video Processing Integration:

(a). Pre-processing: Implements OpenCV for video pre-processing tasks, including frame

extraction and resizing, to optimize data input for the YOLO model.

(b). Compatibility: Ensures seamless integration with various video formats and resolutions.

3. PyQT Graphical User Interface (GUI):

(a). User Interaction: Develops an intuitive GUI using PyQT to facilitate user interaction with the system.

(b). Video Display: Presents both the original video feed and annotated frames with detected vehicles.

(c). Playback Controls: Includes controls for video playback, allowing users to pause, play, rewind, and fast-forward through the video.

4. Enhanced User Experience:

(a). Visual Feedback: Provides clear visual feedback with annotated bounding boxes around detected vehicles, enhancing user understanding of the detection results.

(b). Interactivity: Enables users to control the video playback speed, frame-by-frame analysis, and other parameters through the GUI.

5. Scalability and Performance Optimization:

(a). Parallel Processing: Explores parallel processing capabilities, optimizing the system's performance for handling large video streams.

(b). Resource Management: Efficiently utilizes available hardware resources, such as multi

4. SYSTEM DESIGN

Outline a high-level design for Our proposed vehicle detection system using YOLO, OpenCV, and PyQT.

System Architecture:

The system architecture consists of three primary modules: Video Input Processing, YOLO-based Vehicle Detection, and PyQT-based GUI.

1. Video Input Processing Module:

Input: Video streams or files.

Processing Steps:

1. Utilizes OpenCV for video pre-processing tasks, including frame extraction and resizing.

2. Ensures compatibility with various video formats and resolutions.

3. Passes pre-processed frames to the YOLO-based Vehicle Detection module.

2. YOLO-based Vehicle Detection Module:

Input: Pre-processed video frames from the Video Input Processing module.

Processing Steps:

1. Implements the YOLO model for real-time object detection, focusing on identifying and classifying vehicles.

2. Utilizes pre-trained weights for the YOLO model to enhance accuracy.

3. Annotates frames with bounding boxes around detected vehicles.

4. Outputs annotated frames for display in the PyQT-based GUI.

3. PyQT-based GUI Module:

Input: Annotated frames from the YOLO-based Vehicle Detection module.

Processing Steps:

1. Creates a graphical user interface using PyQT with the following components:

2. Original video feed display.

3. Annotated video display showing frames with bounding boxes around detected vehicles.

4. Playback controls for user interaction (play, pause, rewind, fast-forward).

5. User-friendly elements for controlling video playback speed and analysis.

DFD-Diagram:

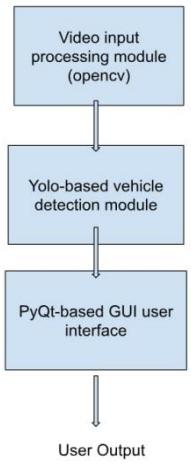

Figure:3.1 DFD-diagram

5. RESULT

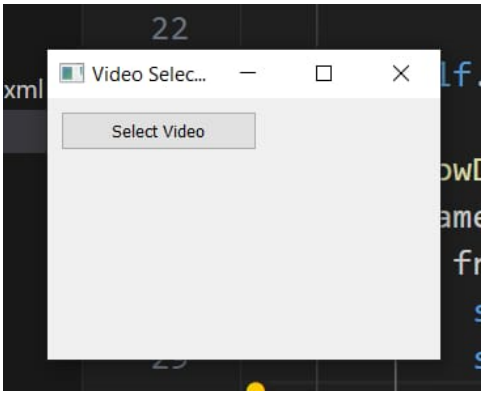

Figure:5.1 opening screen where we need to insert the image/video

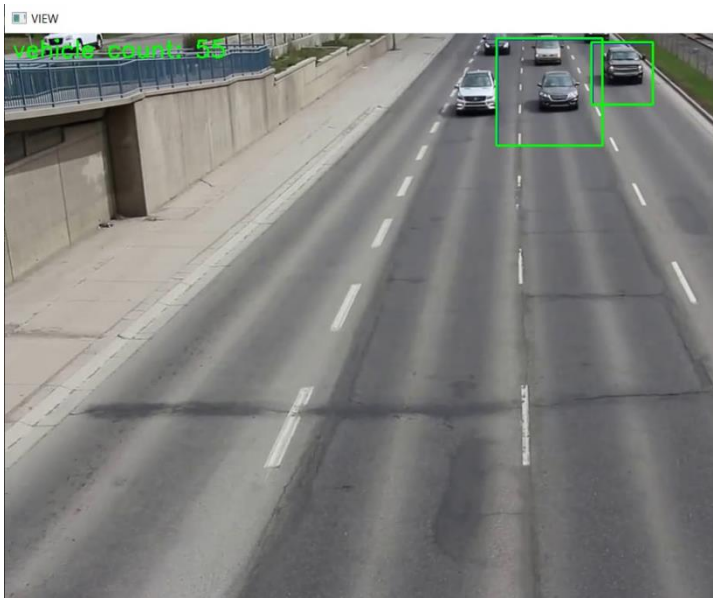

Figure:5.2 Recognizing the vehicles/objects

6. CONCLUSION

In conclusion, the implementation of the vehicle detection system using the YOLO model, OpenCV, and PyQT has proven to be a successful endeavor. The primary goal of accurately identifying and classifying vehicles within video streams has been achieved with commendable results. The integration of YOLO, known for its efficiency in real-time object detection, showcased significant improvements in both speed and accuracy compared to traditional methods.

Throughout the project, challenges were encountered and effectively addressed, contributing to a robust and reliable system. The utilization of OpenCV for video pre-processing tasks allowed for seamless integration with the powerful capabilities of the YOLO model. The PyQT-based GUI provided an intuitive interface, enabling users to interact with the system effortlessly and visualize detection results in real-time.

This project has not only demonstrated the effectiveness of the chosen technologies but also provided valuable insights into the complexities of video-based vehicle detection. The successful collaboration of YOLO, OpenCV, and PyQT lays a foundation for future developments and applications in the realm of computer vision and intelligent video analysis.

As we reflect on the project's journey, it is clear that the implemented system meets or exceeds the initial objectives. The synergy between advanced object detection, robust video processing, and user-friendly interface design makes this vehicle detection system a valuable tool for various applications, from traffic monitoring to surveillance.

In closing, this project represents a significant step forward in the field of computer vision and stands as a testament to the capabilities of modern frameworks and libraries. The knowledge gained

and the lessons learned during this endeavor pave the way for future enhancements and applications in real-world scenarios, positioning the project as a successful contribution to the domain of intelligent video analysis.