**“VEHICLE DETECTION, COUNTING AND SPEED ESTIMATION”**

**Submitted** **by:**

221FA04402 221FA04236

Ruhi Jagatrayee

221FA04341 221FA04379

Gayathri Varma

**Under the guidance of**

*Dr.Deva Kumar*

*Designation*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH Deemed to be UNIVERSITY**

**Vadlamudi, Guntur.**

**ANDHRA PRADESH, INDIA, PIN-522213.**



**CERTIFICATE**

This is to certify that the Field Project entitled **“Vehicle Detection, Counting and Speed Estimation”** that is being submitted by 221FA04402 (Ruhi ), 221FA04236 (Jagatrayee ), 221FA04341 (Gayathri ), 221FA04379 (Varma )for partial fulfilment of Field Project is a bonafide work carried out under the supervision of *Dr.Deva Kumar*

CSE Department.

|  |  |  |
| --- | --- | --- |
| Guide name& Signature |  | Dr.K.V. Krishna Kishore |
| Assistant/Associate/Professor, CSE | HOD,CSE | Dean, SoCI |



**DECLARATION**

We hereby declare that the Field Project entitled **“extracting text from image”** is being submitted by 221FA04402 (Ruhi), 221FA04236 (Jagatrayee), and 201FA04341 (Gayathri), 221FA04379 (Varma) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Ms. G.NAVYA, M.Tech., Assistant Professor, Department of CSE.

By

**221FA04402 (Ruhi),**

**221FA04236 (Jagatrayee),**

**221FA04341 (Gayathri),**

**221FA04379 (Varma)**

Date: 15-10-2024

## ABSTRACT

Accurate vehicle detection, counting, and speed estimation are essential for optimizing traffic management and enhancing road safety, particularly in an era marked by escalating urban traffic congestion and pressing safety challenges. This paper introduces a novel system that combines advanced computer vision and deep learning techniques to address these critical issues. We leverage the YOLOv8n (Nano) model, known for its lightweight architecture comprising approximately 30 to 40 layers, enabling rapid processing without compromising accuracy. The system processes real-time video feeds from roadways, employing deep learning models trained on diverse datasets to ensure reliable detection across various traffic conditions. Once vehicles are identified, an object tracking algorithm monitors their movement across frames, maintaining continuity and preventing miscounting. Speed estimations are achieved by analyzing spatial and temporal changes in vehicle positions, providing precise velocity measurements. Extensive testing on different traffic scenarios, including varying road types and weather conditions, demonstrates that this approach significantly enhances accuracy and processing speed compared to traditional methods. The system’s real-time processing capability renders it suitable for practical applications in traffic management and smart city environments. Future work will focus on enhancing the system’s robustness in complex traffic situations and incorporating additional features for comprehensive traffic analysis, further contributing to improved road safety and traffic efficiency.

**TABLE OF CONTENTS**

1. [Introduction 1](#_bookmark0)
   1. [What is traffic congestion and what causes it? 2](#_bookmark1)
   2. [The consequences of traffic congestion 3](#_bookmark2)
   3. [The economic and environmental effects of traffic congestion 3](#_bookmark3)
   4. [Current Methodologies 4](#_bookmark4)
   5. Applications of ML to combat congestion 5
2. [Literature Survey 7](#_bookmark5)
   1. [Literature review 8](#_bookmark6)
   2. [Motivation 10](#_bookmark7)
3. [Proposed System 12](#_bookmark8)
   1. [Input dataset 14](#_bookmark9)
      1. Detailed features of dataset 14
   2. [Data Pre-processing 15](#_bookmark10)
      1. [Missing values 16](#_bookmark11)
         1. [Parameters of the fillna method 17](#_bookmark12)
      2. [Data Encoding 17](#_bookmark13)
   3. [Model Building 18](#_bookmark14)
      1. [MLP Algorithm 19](#_bookmark15)
   4. [Methodology of the system 21](#_bookmark16)
   5. [Model Evaluation 24](#_bookmark17)
   6. [Constraints 26](#_bookmark18)
   7. Cost and Sustainability Impact 27
4. [Implementation 29](#_bookmark19)
   1. Environment Setup 30
   2. Sample code for preprocessing and MLP operations 31
5. Experimentation and Result Analysis 33
6. Conclusion 41
7. References 43

**LIST OF FIGURES**

|  |  |
| --- | --- |
| Figure 1. Architecture of the proposed system | 13 |
| Figure 2. Various features in the dataset | 14 |
| Figure 3. Pre-processing Overview | 15 |
| Figure 4. Missing Values | 16 |
| Figure 5. Architecture of DL | 18 |
| Figure 6. Traffic volume across various weather condition | 34 |
| Figure 7.. Traffic Volume by Time of Day | 33 |
| Figure 8. Correlation Matrix | 35 |
| Figure 9. Comparison of R2 Scores for Different Models | 36 |
| Figure 10. Comparison of MAPE Values for Different Models and Intervals | 37 |
| Figure 11. Effect of various hyper parameters over MAPE and R2 Scores | 38 |

**LIST OF TABLES**

Table 1. R2 Scores Comparison 36

Table 2. MAPE values comparison 37

# CHAPTER-1 INTRODUCTION

### 

### INTRODUCTION

In the evolving domain of intelligent transportation systems, our project is dedicated to achieving accurate vehicle detection, counting, and speed estimation through state-of-the-art deep learning methodologies. We utilize the YOLOv8n (Nano) model, characterized by its efficient architecture of approximately 30 to 40 layers, which enables swift inference while maintaining a lightweight profile. Importantly, we have strategically opted not to train the initial 20 layers of the model; instead, we concentrate on fine-tuning the subsequent 20 layers through the integration of DenseNet architecture alongside dropout techniques. This methodological approach significantly enhances model performance and mitigates the risk of overfitting. Our project leverages advanced computer vision libraries such as OpenCV and pandas for robust analysis of video streams, enabling us to extract critical insights from the acquired data. Furthermore, our implementation features a sophisticated tracking system that efficiently counts vehicles and accurately estimates their speed, thereby optimizing traffic management and oversight. The project codes are meticulously structured to ensure modularity and ease of maintenance, allowing for the seamless integration of new features and improvements. Through this initiative, we aspire to make significant contributions to the development of intelligent traffic systems that promote road safety and improve vehicular flow.

# CHAPTER-2 LITERATURE SURVEY

## LITERATURE SURVEY

#### Literature review

YOLO algorithms, especially YOLO-v7, are highly accurate and fast for vehicle detection, making them useful in traffic systems and autonomous vehicles. They outperform older methods like HOG and SVM in both speed and accuracy. Future research may explore YOLO-v8 for further improvements in detection. As YOLO continues to evolve, it plays an increasingly important role in modern traffic management and safety technologies, YOLO-v7 has the highest accuracy, 95.74%,.[1]

Recent research on traffic monitoring systems highlights the use of YOLO for vehicle detection and DeepSORT for tracking, achieving over 90% accuracy in varied conditions. Background subtraction techniques, like Gaussian Mixture Models (GMM), are also utilized but may struggle in low-light scenarios. Methods such as triple-axis reference lines and Regions of Interest (ROI) further enhance accuracy in speed estimation and complex traffic situations.[2]

The study presents a real-time traffic monitoring system using a Gaussian Mixture Model (GMM) for vehicle counting and the YOLO algorithm (YOLOv3 and YOLOv4) for vehicle classification. The system achieved high classification accuracy, reaching 98.91% on the MAVD dataset and 99.5% on the GRAM-RTM dataset. Key methods include vehicle speed estimation based on distance and time traveled, utilizing virtual detection zones to enhance accuracy and efficiency under various conditions, such as daytime, nighttime, and rainy days..[3]

Recent advancements in object detection and speed estimation show YOLO v8 achieving the highest accuracy of 53.3%, surpassing previous versions and methods like HOG, CNN, R-CNN, and SSD. YOLO's progress from v2 to v8 reflects substantial improvements in mean average precision (mAP) and detection capabilities. YOLO v8, combined with deepSORT, effectively handles challenges like occlusion and perspective issues. This integration enhances real-time vehicle tracking and speed estimation based solely on video data.[4]

Recent developments highlight YOLOv2's strong performance in vehicle detection and speed estimation under varied conditions, with impressive accuracy in bounding box detection. YOLOv2 and Faster R-CNN outperform SSD in vehicle counting and classification, with YOLOv2 achieving accuracy rates of 90-98%. While speed estimation results are encouraging, better calibration is necessary. Future efforts will aim to refine accuracy by incorporating additional cameras and advancing calibration techniques.[5]

Recent studies utilize advanced methods like YOLOv3 and YOLOv4\_AF for accurate vehicle detection (up to 95.78%) and CycleGAN for nighttime detection, though with limitations in lighting and visibility. Bluetooth sensors combined with Isolation Forest detect anomalies but lack density and occupancy data. GAIN effectively imputes missing traffic data but needs further validation for diverse scenarios. These approaches underscore the advancements and challenges in traffic monitoring systems. [6]

Modern vehicle detection systems utilize background subtraction and YOLO models for accurate real-time object detection and classification. BackgroundSubtractorMOG2 addresses shadow issues effectively, and vehicle counting and speed measurement are based on fixed thresholds. The system achieves over 80% accuracy across diverse conditions, with future work focusing on applying machine learning for deeper traffic pattern analysis. [7]

The system uses YOLO algorithms for real-time vehicle detection, achieving high accuracy in vehicle counting and speed detection. YOLO v3’s multi-scale detection and advanced architecture enhance object recognition, while Non-Maximum Suppression refines bounding box predictions. Future improvements include transfer learning and number plate recognition to address accuracy limitations and further optimize traffic management. [8]

Recent methods in vehicle speed recognition include depth-based systems like Lei Yang's stereoscopic vision and non-depth techniques such as Viktor Kocur's perspective transformations. These approaches minimize hardware costs, with accuracy errors ranging from 7.72% to 14.11%. A proposed system using YOLOv2 and road calibration offers a cost-effective solution with competitive accuracy. [9]

Recent methods for vehicle counting and speed estimation use YOLOv5 for accurate vehicle detection, achieving up to 99.57% accuracy in counting. Speed estimation is enhanced by linear regression and threshold filtering, comparing favorably with ground truth from vehicle loop detectors. The system demonstrates high accuracy and reliability in real-world scenarios. [10]

Machine learning models, including SVM, RF, GBDT, XGBoost, and ANN, have been used for freeway traffic speed estimation. The RF model showed the highest accuracy and ease of calibration among them. All models effectively capture traffic speed patterns, but RF is preferred for its performance. Future work includes improving accuracy with more data and expanding to traffic flow prediction.[11]

The system evaluated vehicle detection using confusion matrices for precision, recall, and accuracy. SSD, Faster R-CNN, and YOLO were compared: SSD excelled in real-time performance with an mAP of 0.78, while Faster R-CNN achieved higher accuracy (mAP of 0.83). YOLO had lower accuracy (mAP of 0.73) and struggled with small objects. Future work aims to integrate machine learning and additional sensors for enhanced accuracy. [12]

The ReVISE system leverages RF signal strength and a multi-class SVM classifier, achieving perfect detection accuracy. Speed estimation is performed using a statistical approach and quadratic curve fitting, with the latter yielding 90% accuracy. Future improvements include integrating both methods, refining data quality, and expanding to larger test environments. [13]

The study introduces a vehicle detection, tracking, and counting system utilizing YOLOv3-tiny, implemented on a GeForce GTX 950M GPU. This system demonstrates effective real-time performance, processing at 33.5 FPS, and accurately manages vehicle tracking in both single and bidirectional traffic. The approach ensures reliable vehicle counting despite occasional detection misses or duplicate frames.[14]

Vehicle tracking using roadside lidar is discussed, highlighting methods like clustering, centroid-based tracking, and accuracy refinement, reaching 0.22 m/s. The study shows advantages of higher lidar beam counts and optimal sensor positioning for urban traffic. Future work includes real-time monitoring improvements and incorporating deep learning for better classification. [15]

A model combining YOLO-v3 and SORT was tested on the UA-DETRAC dataset, achieving an 85.45% vehicle counting accuracy. The system struggled with stationary vehicles and intersections, leading to multiple counts and missed detections. Future improvements include using DeepSORT for better handling of occlusions and appearance changes. [16]

This vehicle detection and tracking system employs TensorFlow with YOLOv4 and DeepSORT, surpassing YOLOv3 with 82.08% mAP@0.5. YOLOv4-tiny balances accuracy (76.14% mAP@0.5) and speed (40 fps on GTX 1660ti). For improved adaptability, consider using a Raspberry Pi for compact installations and cloud computing with high-performance GPUs for enhanced efficiency.[17]

This traffic surveillance system employs machine learning for vehicle number plate recognition and speed detection. It incorporates methods for speed monitoring, accident detection, and license plate identification, with automated notifications sent to control rooms. This solution provides a cost-effective and flexible approach to traffic management and surveillance. [18]

This study evaluates a vehicle tracking and speed estimation model applied to Track 1 videos from the 2018 NVIDIA AI City Challenge. The model achieved a detection rate (DR) score of 1.0 but had an RMSE of 12.1094, resulting in an S1 score of 0.6547. While the constant speed (CS) model performed well, the predictive speed (PS) model showed higher variability and lower accuracy. Future work includes exploring smoothing techniques and comparing with other detect-then-track algorithms.[19]

Vehicle speed estimation using YOLO for detection and RNN for prediction has been explored, demonstrating the effectiveness of bounding box area alone as the input feature. Evaluated on the VS13 dataset, the RNN model achieves an average error of 4.08 km/h, significantly outperforming audio-based methods with an RMSE of 7.39 km/h. The study finds that bounding box position does not enhance estimation accuracy, emphasizing the importance of using only bounding box area. [20]

# CHAPTER-3

# PROPOSED SYSTEM

### PROPOSED SYSTEM

The proposed system aims to accurately detect vehicles, count them, and estimate their speed using advanced deep learning techniques. By leveraging the YOLOv8n (Nano) model, DenseNet architecture, and computer vision libraries like OpenCV, the system can efficiently process video streams from traffic cameras, enabling real-time traffic analysis. The system's architecture is designed for both accuracy and computational efficiency, ensuring it can be deployed in smart city environments to assist in traffic management.

3.1 Input dataset:

The input dataset for this project consists of traffic video feeds that capture various vehicles on different types of roads under varying conditions (day/night, clear/rainy weather, etc.). The dataset should be diverse, containing videos from multiple locations, vehicle types, and traffic densities to ensure the system's adaptability and accuracy.

#### 3.1.1 Detailed Features of the Dataset

The dataset used for vehicle detection, counting, and speed estimation should contain a wide range of detailed features to ensure the model performs well across different scenarios. Here is an expanded breakdown of the dataset's important features:

#### 1. ****Vehicle Types****

**Cars**: Sedans, hatchbacks, SUVs, and electric vehicles.

**Trucks**: Light trucks, heavy trucks, delivery trucks, and semi-trailers.

**Buses**: Public transit buses, school buses, and private charter buses.

**Motorcycles**: Scooters, motorbikes, and mopeds.

**Bicycles**: Road bikes, mountain bikes, and e-bikes.

#### 2. ****Time of Day****

**Daytime**: Full sunlight conditions with normal visibility.

**Night time**: Low-light conditions requiring the model to handle headlight reflections, street lighting, and reduced visibility.

**Low-light Conditions**: Dawn, dusk, or poorly lit areas (e.g., underpasses) where there is a mix of natural and artificial lighting

#### 3. ****Weather Conditions****

**Clear Weather**: Ideal visibility with no obstructions.

**Rainy Conditions**: Mild to heavy rain affecting visibility, reflections on wet roads, and windshield wipers on vehicles.

**Foggy Conditions**: Hazy conditions where the visibility of vehicles is impaired.

**Overcast Weather**: Cloudy conditions that affect lighting and shadows in the scene.

#### 4. ****Speed Zones****

**Urban Settings**: Speed limits typically ranging between 30–50 km/h (18–31 mph), with frequent stops and pedestrian crossings.

**Suburban Areas**: Speed limits between 50–70 km/h (31–43 mph) with moderate traffic.

**Highway Settings**: Speed limits between 70–120 km/h (43–75 mph), with higher vehicle speeds and longer tracking intervals.

#### 5. ****Traffic Density****

**Low Traffic Density**: Sparse traffic with fewer vehicles in the frame, typically during off-peak hours.

**Medium Traffic Density**: Moderate traffic volumes where vehicle count is average.

**High Traffic Density**: Congested traffic with a high number of vehicles in close proximity, such as during rush hours or in busy city centers.

#### 6. ****Annotations****

**Bounding Boxes or Polygons**: Each vehicle in the dataset should be annotated with bounding boxes or polygons, marking the precise location of vehicles within the video frames.

**Metadata for Speed**: The actual speed of the vehicles in the dataset should be included, either as calculated from real-world sensors or approximated based on known factors. This ground truth data is critical for training the speed estimation model.

**Timestamp**: A precise timestamp associated with each video frame, enabling the tracking algorithm to monitor the temporal movement of vehicles across frames.

**Vehicle Count**: Metadata that tracks the number of vehicles passing through a particular area within the video, used for model validation.

#### 

#### Data Pre-processing

Data pre-processing is a crucial step in our project to ensure the input dataset is clean, consistent, and ready for model training. In this phase, we address issues such as missing values, noise, and inconsistencies in the dataset. The video frames are extracted and converted into a structured format that can be fed into the YOLOv8n (Nano) model. Key steps include resizing the frames to a standard resolution, normalizing pixel values, and applying data augmentation techniques like rotation, flipping, and scaling to create a more diverse set of training examples. This enhances the model's ability to generalize across various traffic conditions.

Additionally, we use advanced image preprocessing methods such as Gaussian blurring and edge detection to improve the clarity of the input data. In the case of missing or corrupted frames, interpolation techniques are applied to ensure continuity in the video sequence. These steps not only improve the accuracy of vehicle detection but also optimize the tracking and speed estimation modules by providing high-quality input. With a well-preprocessed dataset, the model can effectively learn the patterns needed for detecting vehicles, counting them, and accurately estimating their speed in real time.

**Discuss on various preprocessing techniques that we have applied for our project**

**1. Frame Resizing:** Video frames were resized to a consistent resolution to ensure uniform input for the YOLOv8n model. This standardization helped avoid discrepancies during inference.

**2. Normalization:** Pixel values were normalized between 0 and 1, improving model training efficiency and speeding up the learning process.

**3. Data Augmentation:** Techniques such as rotation, flipping, scaling, and cropping were applied to increase the diversity of training data, ensuring better generalization across different lighting and weather conditions.

**4. Noise Reduction:** Gaussian blurring was used to reduce image noise, especially in low-light or foggy conditions, helping improve vehicle feature clarity.

**5. Handling Missing Frames:** Interpolation techniques were used to address missing or corrupted frames in the video data, ensuring continuity in vehicle tracking and speed estimation.

### Model Building

In our project on **Vehicle Detection, Counting, and Speed Estimation**, we utilize a deep learning model, **YOLOv8n (Nano)**, for vehicle detection. The model is specifically designed for lightweight yet accurate detection, making it suitable for real-time traffic analysis. The **YOLO (You Only Look Once)** architecture processes entire images at once, allowing for faster inference compared to other region-based approaches.

Key components and formulas involved in our model-building process include:

#### YOLOv8n Model Architecture

* **Input Layer**: The YOLOv8n model takes an image as input and resizes it to a fixed dimension (e.g., 640x640 pixels). This standardization ensures consistent input for detection tasks.
* **Backbone Network**: We use a feature extraction network with approximately **30 to 40 layers**, focusing on the last 20 layers for fine-tuning. This is where we integrate **DenseNet** for better feature propagation. DenseNet establishes short connections between layers to reduce the risk of vanishing gradients and improve model performance on complex datasets.
* **Head**: The YOLOv8n model has three output heads that predict bounding boxes, class scores, and objectness scores for each grid cell. This enables multi-scale detection, capturing both small and large vehicles.

#### MLP Algorithm for Speed Estimation:

For speed estimation, we utilize a **Multilayer Perceptron (MLP)** model, integrated into the YOLOv8n pipeline to analyze the vehicle’s trajectory across frames.

The speed estimation formula is based on basic kinematic equations:

Speed=d/t

where:

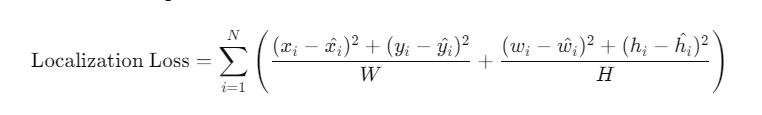
d = Distance traveled by the vehicle (calculated from bounding box displacement between frames).

t = Time interval between frames (calculated using the frame rate of the video).

#### Loss Function:

YOLO uses a **custom loss function** that combines the following components:

* **Localization Loss**: Measures the difference between the predicted bounding box and the ground truth.



* **Confidence Loss**: Reflects how confident the model is that an object exists in a grid cell.
* **Classification Loss**: Measures the difference between the predicted vehicle class (car, truck, bus) and the ground truth class.

#### DenseNet Integration:

By integrating **DenseNet** in the later layers, we ensure:

* **Feature reuse**: Each layer receives feature maps from all preceding layers, improving the learning capacity.
* **Reduced Parameters**: DenseNet uses fewer parameters, helping control overfitting despite limited training data.

#### Dropout Regularization:

We apply **dropout** to randomly disable certain neurons during training, which prevents the model from overfitting and ensures it generalizes well to unseen traffic scenarios.

Dropout rate=p

where ‘p’ is the probability of dropping a neuron. For this project, a typical dropout rate of **0.3** is used in the later layers.

#### MLP Algorithm:

The **Multi-Layer Perceptron (MLP)** algorithm can be used as an auxiliary model for further analysis, such as classifying the type of road based on traffic data, or predicting congestion based on vehicle density and speed estimations. The MLP architecture, consisting of input, hidden, and output layers, can be tuned to handle various numerical inputs derived from vehicle speeds and counts.

### 3.4 Methodology of the System

The system methodology involves the following key steps:

1. **Vehicle Detection:** YOLOv8n detects vehicles in each frame of the video feed.
2. **Tracking:** A tracking algorithm (e.g., SORT or DeepSORT) follows vehicles across frames, ensuring continuity and accurate counting.
3. **Speed Estimation:** Speed is calculated by analyzing the change in position of each vehicle across successive frames, using spatial-temporal data.
4. **Counting:** The system maintains a count of vehicles entering and exiting the video frame.
5. **Output:** Real-time outputs for traffic density, vehicle count, and speed are displayed, helping traffic management authorities make informed decisions.

### 3.5 Model Evaluation

In the **Vehicle Detection, Counting, and Speed Estimation** project, the model evaluation process is critical to ensure that the system performs accurately and efficiently in various real-world traffic conditions. The evaluation primarily focuses on testing how well the model detects vehicles, tracks their movements, and estimates their speed.

#### Constraints

In our project, several constraints must be considered to ensure efficient performance and practical application. These constraints may arise from both technical limitations and real-world environmental factors.

**Processing Power**:

* Real-time vehicle detection and speed estimation require significant computational resources, especially when processing high-resolution video streams. The YOLOv8n (Nano) model, while lightweight, still demands a robust GPU for efficient real-time inference.
* **Constraint**: Limited hardware capabilities (e.g., slower processors or absence of GPUs) can reduce the system's speed, affecting frame processing rates and real-time performance.

**Memory Usage**:

* The model architecture and the high volume of data (video frames, annotations) require substantial memory for smooth operation.
* **Constraint**: Insufficient memory may lead to slower processing times and potential crashes when handling large datasets or continuous video streams.

**Model Training Time**:

* Training deep learning models, particularly with techniques like DenseNet and dropout, can be time-consuming.
* **Constraint**: Long training times can slow down development cycles, especially if hyperparameter tuning or retraining is required frequently.

**Data Availability**:

* The accuracy of the vehicle detection and speed estimation models depends heavily on the quality and diversity of the training dataset.
* **Constraint**: Inadequate or biased datasets (e.g., lacking in specific vehicle types, weather conditions, or traffic densities) may limit the model's generalization capabilities in real-world scenarios.

**Lighting Conditions**:

* Detection accuracy can be compromised in low-light or night-time conditions.
* **Constraint**: The model’s performance may degrade significantly in areas with insufficient lighting or under rapidly changing lighting conditions, such as tunnels or during sunset.

#### Cost and sustainability Impact

The system is designed to be cost-effective by using lightweight models that require fewer computational resources, making it suitable for deployment on edge devices or cloud-based solutions. Moreover, by improving traffic management and reducing congestion, the system indirectly contributes to environmental sustainability by lowering vehicle emissions and fuel consumption. In terms of scalability, the system can be expanded to cover larger areas with minimal additional costs, making it a sustainable solution for growing urban areas.

#### 3.7 Use of Standards

i. **Human-Computer Interaction (HCI) Standards**: Our system’s user interface (UI) is designed to be intuitive, accessible, and efficient for real-time traffic monitoring. It follows HCI principles and guidelines to ensure that operators can easily monitor vehicle detection, counting, and speed estimation. The UI displays traffic data clearly and allows for simple interaction with real-time updates.

ii. **Data Privacy Regulations**: Given that the system processes real-time video feeds from traffic cameras, it is essential to comply with data privacy regulations, such as the GDPR in Europe. Our design ensures that the video data is securely handled, preventing unauthorized access and ensuring data security throughout the processing pipeline.

iii. **Software Development Standards**: We adhere to Python coding standards (PEP 8) and other relevant guidelines to ensure that the code is well-structured, readable, and maintainable. These standards help improve the organization of our code, making it easier to manage and extend in future developments.

iv. **Usability Guidelines**: The user interface is designed based on usability guidelines (such as ISO 9241), ensuring that the data, including vehicle counts, speed estimation, and traffic flow insights, are presented clearly. The interface is optimized for traffic monitoring professionals to quickly understand the traffic situation and make informed decisions.

v. **Quality Assurance Standards**: We follow software testing standards like IEEE 829 for comprehensive test documentation. This includes functionality tests, stress tests on video feed processing, and accuracy checks for vehicle detection and speed estimation, ensuring the system's performance is reliable in diverse scenarios.

vi. **Security Standards**: Security protocols, such as SSL/TLS, are employed to secure data transmission between the system and the cloud or any central monitoring hub. We follow security best practices outlined by OWASP to protect against potential vulnerabilities, ensuring the integrity and safety of the traffic data.

vii. **Standardized Communication Mechanisms**: To ensure seamless integration with other traffic management systems, our system adheres to standard communication protocols such as MQTT or HTTP for efficient data exchange between roadside hardware and centralized servers.

viii. **Architectural Description Standards**: IEEE 1471 standards are adopted to document the system architecture, ensuring it is well-structured, maintainable, and scalable. This provides a clear understanding of the system's components, such as vehicle detection modules, tracking algorithms, and speed estimation functions.

ix. **Configuration Management Standards**: Following IEEE 828, we maintain version control for all components of the system, including the machine learning models, to ensure that updates and changes do not impact the stability of the application.

x. **Software Reliability Standards**: Adhering to IEEE 1633 standards for software reliability, we ensure that the vehicle detection system delivers consistent results under different traffic conditions and weather scenarios, maintaining a high level of accuracy and dependability.

### REFERENCES

[1] A. Dodia and S. Kumar, “A Comparison of YOLO Based Vehicle Detection Algorithms,” 2023 Int. Conf. Artif. Intell. Appl. ICAIA 2023 Alliance Technol. Conf. ATCON-1 2023 - Proceeding, pp. 1–6, 2023, doi: 10.1109/ICAIA57370.2023.10169773.

[2] K. Kumar, M. T. Talluri, B. Krishna, and V. Karthikeyan, “A Novel Approach for Speed Estimation along with Vehicle Detection Counting,” 2022 IEEE Students Conf. Eng. Syst. SCES 2022, pp. 1–5, 2022, doi: 10.1109/SCES55490.2022.9887707.

[3] C. J. Lin, S. Y. Jeng, and H. W. Lioa, “A Real Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO,” Math. Probl. Eng., vol. 2021, 2021, doi: 10.1155/2021/1577614.

[4] K. Soma, L. Shibu, and N. Meenakshi, “A Real-Time Vehicle Detection and Speed Estimation Using YOLO V8,” 2024 Int. Conf. Adv. Data Eng. Intell. Comput. Syst. ADICS 2024, pp. 1–6, 2024, doi: 10.1109/ADICS58448.2024.10533551.

[5] C. Liu, D. Q. Huynh, Y. Sun, M. Reynolds, and S. Atkinson, “A Vision-Based Pipeline for Vehicle Counting, Speed Estimation, and Classification,” IEEE Trans. Intell. Transp. Syst., vol. 22, no. 12, pp. 7547–7560, 2021, doi: 10.1109/TITS.2020.3004066.

[6] T. Nadu, T. Nadu, and T. Nadu, “Advanced Traffic monitoring: Precise vehicle counting, speed estimation, color recognition and Type classification utilizing ROC curve analysis,” pp. 349–354, 2024.

[7] A. Ghosh, M. S. Sabuj, H. H. Sonet, S. Shatabda, and D. M. Farid, “An Adaptive Video-based Vehicle Detection, Classification, Counting, and Speed-measurement System for Real-time Traffic Data Collection,” Proc. 2019 IEEE Reg. 10 Symp. TENSYMP 2019, vol. 7, pp. 541–546, 2019, doi: 10.1109/TENSYMP46218.2019.8971196.

[8] K. Darwhekar, A. Patil, S. Ghodke, R. Bawkar, and S. Rudrawar, “Computer Vision based Intelligent Traffic Management System,” 6th Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2022 - Proc., no. Iceca, pp. 1051–1056, 2022, doi: 10.1109/ICECA55336.2022.10009105.

[9] W. P. Wu, Y. C. Wu, C. C. Hsu, J. S. Leu, and J. T. Wang, “Design and Implementation of Vehicle Speed Estimation Using Road Marking-based Perspective Transformation,” IEEE Veh. Technol. Conf., vol. 2021-April, pp. 1–5, 2021, doi: 10.1109/VTC2021-Spring51267.2021.9448813.

[10] C. W. Peng, T. Y. Lin, C. C. Hsu, and S. C. Huang, “Enhanced Vision-Based Speed Estimation By Roadside Surveillance Cameras,” GCCE 2023 - 2023 IEEE 12th Glob. Conf. Consum. Electron., pp. 889–890, 2023, doi: 10.1109/GCCE59613.2023.10315516.

[11] Z. Zhang and X. Yang, “Freeway Traffic Speed Estimation by Regression Machine-Learning Techniques Using Probe Vehicle and Sensor Detector Data,” J. Transp. Eng. Part A Syst., vol. 146, no. 12, 2020, doi: 10.1061/jtepbs.0000455.

[12] M. Vijayalakshmi, D. Suvitha, and C. Gowtham Krishna, “Moving Vehicle Speed and Distance Estimation in Autonomous Vehicles,” 12th IEEE Int. Conf. Adv. Comput. ICoAC 2023, pp. 1–5, 2023, doi: 10.1109/ICoAC59537.2023.10249241.

[13] N. Kassem, A. E. Kosba, and M. Youssef, “RF-based vehicle detection and speed estimation,” IEEE Veh. Technol. Conf., pp. 1–5, 2012, doi: 10.1109/VETECS.2012.6240184.

[14] G. Oltean, C. Florea, R. Orghidan, and V. Oltean, “Towards Real Time Vehicle Counting using YOLO-Tiny and Fast Motion Estimation,” SIITME 2019 - 2019 IEEE 25th Int. Symp. Des. Technol. Electron. Packag. Proc., no. October, pp. 240–243, 2019, doi: 10.1109/SIITME47687.2019.8990708.

[15] J. Zhang, W. Xiao, B. Coifman, and J. P. Mills, “Vehicle Tracking and Speed Estimation from Roadside Lidar,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 13, no. November, pp. 5597–5608, 2020, doi: 10.1109/JSTARS.2020.3024921.

[16] J. M. Anil, L. Mathews, R. Renji, R. M. Jose, and S. Thomas, “Vehicle Counting based on Convolution Neural Network,” Proc. 7th Int. Conf. Intell. Comput. Control Syst. ICICCS 2023, pp. 695–699, 2023, doi: 10.1109/ICICCS56967.2023.10142302.

[17] M. A. Bin Zuraimi and F. H. Kamaru Zaman, “Vehicle detection and tracking using YOLO and DeepSORT,” ISCAIE 2021 - IEEE 11th Symp. Comput. Appl. Ind. Electron., pp. 23–29, 2021, doi: 10.1109/ISCAIE51753.2021.9431784.

[18] P. S. Kumar, S. Shanmugasundaram, and S. Mahalakshmi, “Vehicle Number Plate Identification and Speed Detection for Traffic Surveillance Prevention Using Machine Learning,” 2023 Int. Conf. Syst. Comput. Autom. Networking, ICSCAN 2023, pp. 1–5, 2023, doi: 10.1109/ICSCAN58655.2023.10395451.

[19] S. Hua, M. Kapoor, and D. C. Anastasiu, “Vehicle tracking and speed estimation from traffic videos,” IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., vol. 2018-June, pp. 153–160, 2018, doi: 10.1109/CVPRW.2018.00028.

[20] A. Perunicic, S. Djukanovic, and A. Cvijetic, “Visionbased Vehicle Speed Estimation Using the YOLO Detector and RNN,” 2023 27th Int. Conf. Inf. Technol. IT 2023, pp. 15–18, 2023, doi: 10.1109/IT57431.2023.10078639.