Enhancing Urban Surveillance with Car Dash-Cams in Smart Cities

Dhanalakshmi R
Centre for Cyber Physical Systems
School of Computer Science and Engineering
Vellore Institute of Technology (VIT)
Chennai, India
dhanalakshmi.r@vit.ac.in

Shivam Kumar Thakur
School of Computer Science and Engineering
Vellore Institute of Technology (VIT)
Chennai, India
shivamkumar.thakur2022@vitstudent.ac.in

Sahaya Beni Prathiba B
Centre for Cyber Physical Systems
School of Computer Science and Engineering
Vellore Institute of Technology (VIT)
Chennai, India
prathiba.sbb@vit.ac.in

***Abstract*—Technological developments are becoming important for efficient city security measures as metropolitan areas rapidly expand. Area without standard street cameras, this paper sug gests using YOLO v10 models mounted on dashboard cameras to improve city security. Real-time car dash cam data analysis is used, to identify and count different objects, including walkers, cars, motorcycles, and people. A server gets data and unique automobile IDs regularly. The information is kept on the server in a coordinate-indexed time series database. Cities that use dash cams to get a complete, real-time picture of their surroundings can increase security and promote proactive decision-making. In addition to enhancing the current surveillance infrastructure, this strategy provides an affordable way to increase municipal security coverage in areas where more conventional approaches are insufficient. This strategy will provide an affordable way to increase municipal security coverage in areas where more conventional approaches are inefficient.**

***Index Terms*—YOLOv10, Artificial Intelligence, Smart Cities, surveillance systems**

I. Introduction

Expanding cities requires strong surveillance systems. It is essential for effectively managing the increasing traffic volume and preserving public safety. Nevertheless, financial limitations prevent many communities from setting all-encompassing surveillance systems. For this reason, prioritizing traffic monitoring on roadways is essential to reduce traffic, guarantee effective mobility, and improve urban management in general. Every year, road accidents claim millions of lives, so it is critical to have an effective and proactive accident detection system. There are several issues with the conventional accident detection system, including low scalability, connectivity, power consumption, and sensor reliability. [1].

A smart city is created by integrating various, interconnected, and automated elements of a city into a single, networked system. A smart city is made up of several intelligent objects that are positioned around the city, collect data at certain locations, and then process data to help decision-makers. Smart cities have an impact on every aspect of daily life, including smart grids, parking, industry, transit, and employment and health care facilities. Central database management systems, client server architectures, cloud computing, and other cutting-edge Wireless Sensor Networks (WSNs) technologies are used in smart city models. Modern computing demands smart cities in order to make the environment digital, responsive, efficient, dependable, and automated. Researchers are interested in a few problems and obstacles related to smart cities, despite their many advantages. Although academics have concentrated on offering numerous solutions for smart cities, there are still several challenges that need to be addressed. There aren't many research methodologies that don't handle intelligent traffic monitoring and guidance systems. [2].

Consistent urbanization is causing significant obstacles in our day-to-day existence. Compared to rural areas, 50% of the world's population lived in cities as of 2007. According to UN Population Fund estimates, by 2030 nearly 60% of the world's population is expected to live in urban areas [3]. Cities may expand as resource management functions like transportation, energy consumption, and goods distribution become more efficient. However, it may also result in gridlock, problems with waste management, pollution, noise, traffic, and security. Reaching greater efficiency and safety will depend on properly collecting, analysing, and displaying data. [4].Based on the requirements, each country defines a smart city differently based on its geographical location, specific needs, and economic conditions [5].

Profound machine learning technologies have opened new avenues for enhancing urban surveillance and monitoring systems. By incorporating a YOLOv10 model into dash-cam data processing systems, we can achieve a more cost-effective method of monitoring the road environment. The paper aims to

TABLE I
COMPARISON OF APPROACHES IN SMART CITY TECHNOLOGIES

Aspect	Shan et al. (2024)	Rajkumar (2024)	Xiao et al. (2024)
Technologies Leveraged	Big data, IoT	Quantum computing, AI, IoT, advanced optics	Multi-Objective Optimization (MOO), real-time data
Application Focus	Smart city situational awareness	Optical IoT for real-time urban data collection	Public transportation route optimization
Key Strengths	High localization precision, extensive coverage, real-time, high-resolution data acquisition	Dynamic optimization based on real-time data	Real-time, dynamic data sources
Challenges Addressed	Urban management efficiency	Delayed decision-making due to data lag	Traffic patterns, cost, environmental impact
Future Prospects	5G, 6G integration for enhanced capabilities	Quantum-enhanced sensors for precision measurement	Continued adaptation to evolving urban environments

show how machine learning models with vehicle dash-cam can act as, an affordable alternative to traditional street cameras and to explore the viability of integrating them into automotive systems for urban surveillance. We aim to demonstrate the benefits of this approach, such as lower costs, scalability, and adaptability.

The study's key conclusions reveal a great deal of promise for improving urban surveillance with the suggested strategy. This is achieved by assisting with the monitoring of the road environment in real-time, quickly identifying and categorizing a wide range of objects on the road, and ultimately improving public safety. Using a YOLOv10 model in-vehicle system is also beneficial since it reduces overall costs and maintenance requirements by removing the demand for expensive, stationary street cameras.

II. RELATED WORKS

Recent research focuses on developing technologies that can help us gain an overall awareness of smart cities. The Chinese government plans to integrate IoT and big data into smart cities for better management, as outlined in their five-year plan. This will use GPS data to achieve wide coverage and better localization accuracy. Massive mobile data for spatiotemporal modeling and cybersecurity insurance are shown as major technologies. [6].To develop situational awareness, these technologies have shown that they are feasible and, therefore, are being used in Chinese mega-cities. With the advancement of 5G and 6G technologies, these technologies can be validated in more real-time applications. By integrating massive mobile data and artificial intelligence, we can gain analytical capabilities for proactive smart city management.

Along with artificial intelligence, quantum computing can also play an important role. In optical IoT, sensors gather real-time data from urban areas. [3]. These sensors can include LIDAR, high-resolution cameras, and others. Many conventional technologies exist, but they rely heavily on non-real-time data, which reduces the efficiency and accuracy of the current system. This problem is addressed by this study, which suggests an advanced smart city architecture that will collect data from all sensors in real-time. This data will be analyzed by AI, and quantum-enhanced sensors will help in making better decisions. With this strategy, we can manage our cities more effectively.

Xiao et al. support MOO techniques, which will enhance route planning in smart cities [7].These techniques will consider factors such as traffic, cost, environmental effects, and others, helping to better understand urban mobility. Many complex algorithms already exist that use route planning with the help of real-time traffic data from GPS. This study enhances the overall concept of a smart city monitoring system by adding route planning features, providing a more holistic solution.

Sunagawa et al. presented a system to monitor drowsiness using a camera in front of drivers in semi-autonomous driving vehicles. Their research showcases that with high-resolution data, we can detect when the driver is engaged in tasks not related to driving [8]. Simoncini et al. classify unsafe maneuvers in driving by using camera feed, GPS, and IMU sensor data. They employ a deep learning-based two-stream neural architecture. Their study showcases the efficiency of combining these two data sources [9]. Rocky et al. review different learning approaches for autonomous driving and discuss their strengths and weaknesses. They also mention various ways to detect accidents using audio-video, segment-based methods, and more [10].

For managing vehicle networks, the idea of using blockchain is also proposed. Cloud servers are mostly used in current technologies, but they are vulnerable to data forgery and privacy concerns. With blockchain, data integrity can be maintained, but the authentication of data before entry remains a problem. Park et al provide a solution to solve this issue and present an overall architecture to address it [11]. To detect traffic anomalies from dashcam videos, a study proposed an architecture consisting of accident localization and accident classification [12]. This research was tested and showed promising results; however, challenges such as data imbalance and the size of the dataset affected the accuracy of classification.

Maruyama and Ohashi propose a AI model to predict accidents. Their model primarily uses the divergence of visual attention and the focus of expansion [5].Jhala et al. focus on diverse weather and traffic conditions during object detection and classification in autonomous vehicles [13]. A reinforcement learning-based approach is also introduced to detect traffic accidents using dashcam videos and the DARC-based model showed good accuracy [14].

TABLE II
COMPARISON OF APPROACHES FOR SMART CITY SURVEILLANCE

Existing Research	**Proposed Approach**
Utilizes big data, IoT, and AI for situational awareness	Uses dash-cams with YOLOv10 for real-time surveillance
Relies on mobile data, GPS, and traffic reports	Utilizes dash-cams from cars, sending data in real-time
Focuses on dynamic data for smart city management	Integrates real-time detection and response via dash-cams
Uses complex algorithms for anomaly detection	Detects anomalies using YOLOv10, requests images as needed
Addresses cost-efficiency in route planning	Offers cost-effective surveillance with existing dash-cams
Considers environmental footprints in route optimization	Supports municipal surveillance without additional devices
Uses AI for predictive analytics and decision support	Enables predictive analytics with YOLOv10 and time series
Emphasizes integration of diverse technologies	Simplifies integration using existing dashcam infrastructure
Requires large-scale deployment for validation	Scalable solution using existing vehicle infrastructure
Enhances resource allocation through data-driven decisions	Efficiently monitors areas with limited municipal budgets

III. GENERAL METHODOLOGY

A. Introduction to Deep Learning

Deep Learning is an important subfield of machine learning and has revolutionized several sectors by enabling computers to learn from vast amounts of data andmake decisions with little ongoing human intervention. Unlike conventional machine learning methods that rely on feature engineering and shallow structures, deep learning approaches such as CNNs (Convolutional Neural Networks) and RNNs (Recurrent Neural Networks) comprehend hierarchical representations of data through multiple levels ofabstraction. Their approach allows them to recognize intricate relationships and connections from raw data, making them particularly valuable for applications such as speechand image recognition, natural language processing, and autonomous systems.

B. Introduction to Convolutional Neural Network (CNN)

Convolutionalneural networks, or CNNs, supply a body for processing as well as examining aesthetic input. Typically there are nine layers inCNN architectures: Input Layer, Convolutional Layer, Activation Layer, Pooling Layer, Batch Normalization Layer, Dropout Layer, Flatten Layer, Fully Connected (Dense) Layer and Output Layer. The convolutional layers apply filters to the input data to extract important featureslike edges and textures from it. Activation functions substitue non-linearly allows us to learnexactly the nonlinear function we wanna learn. Activation functionslike ReLU and other non-linear functions introduce non-linearity in the model, permitting it to copy more complex structures. Afterthe convolutional and pooling layers retrieve high-level features, fully connected layers produce the last outputs like class labels or continuous values. They can learn from thedata and build hierarchical representations of input CNNs.

Input Layer: The input to a CNN is typically a multi-channel image. For instance, an RGB image with dimensions Height×Width×3.

Convolutional Layer: Convolutional layers apply convolution operations on the input using a set of learnable filters, or kernels. Each filter convolved moves across the input to produce a feature map.

Convolution Operation:

$$O(i,j) = \sum_{x}\sum_{y} I(i+x, j+y) \cdot K(x,y) \tag{1}$$

where: I (input image), K (kernel), coordinates of the output are represented by i, j, and x, y are the spatial coordinates of the kernel. Feature maps, also known as activation maps, are the results of the convolution procedure.

Convolution Layer Output Size Formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1 \tag{2}$$

Here, W represents the input size (both width and height), F denotes the filter or kernel size, P refers to the amount of padding applied and S stands for the stride length. The resulting output size, W_{out}, corresponds to the width and height of the output after the convolution operation.

Activation Function (ReLU): To add non-linearity, an activation function is applied element-by-element following convolution. ReLU, or Rectified Linear Unit, is a function defined as the maximum of zero and the input value x. Essentially, it outputs x if x is positive, and zero otherwise. This activation function replaces all negative values with zero.

Pooling Layer: By lowering the spatial dimensions of the feature maps, pooling layers improve computing efficiency and offer a type of spatial invariance. Max Pooling extracts the highest value from every feature map sub-region

$$P(i,j) = \max_{x,y \in R(i,j)} O(x,y) \tag{3}$$

Pooling Layer Output Size Formula:

$$W_{out} = \frac{W - F}{S} + 1 \tag{4}$$

Here, W represents the input size, both in terms of width and height. F refers to the size of the pooling filter, while S stands for the stride. The resulting output size, denoted as W_{out}, corresponds to the width and height of the output after the pooling operation.

Flattening: To feed into a fully connected layer, the feature maps from the final convolutional or pooling layer are flattened into a one-dimensional vector.

Fully Connected Layer: A fully linked (dense) layer connects each neuron in the current layer to every other neuron in the layer below.

Output Layer with Softmax: The last layer frequently converts into probabilities using a softmax activation function for multi-class categorization.

Softmax Function:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}} \tag{5}$$

where: $\sigma(z)_i$ is the probability of the i-th class, z_i is the raw score for the i-th class, and K is the total number of classes.

Training Process: Gradient descent and backpropagation are used to train the CNN. The key steps include:

Forward Pass Propagate data through each layer to determine the network's output for a specific input.

Loss Calculation Cross-Entropy Loss (for classification):

$$L = -\sum_{i=1}^{N} t_i \log(p_i) \tag{6}$$

where t_i is the true label, p_i is the predicted probability for class i, and N is the number of classes.

Backward Pass (Compute Gradients) For a fully connected layer, the gradients are computed as:

$$\frac{\partial L}{\partial W} = \delta \cdot x^T \tag{7}$$

Here,

$$\delta = \frac{\partial L}{\partial z} \tag{8}$$

and x is the input to the layer.

For the ReLU activation function:

$$f'(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

For the softmax function:

$$\frac{\partial \sigma_i}{\partial z_j} = \sigma_i(\delta_{ij} - \sigma_j) \tag{10}$$

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \tag{11}$$

where δ_{ij} is the Kronecker delta.

Weight Update: **Momentum:** Gradient vectors are accelerated in the proper directions by momentum, which speeds up convergence.

$$v_t = \gamma v_{t-1} + \eta \nabla L_t \tag{12}$$

$$W_t = W_{t-1} - v_t \tag{13}$$

where γ is the momentum factor (typically between 0.8 and 0.9).

AdaGrad: Based on historical gradient information, it adapts the learning rate of each parameter.

$$G_t = G_{t-1} + (\nabla L_t)^2 \tag{14}$$

$$W_t = W_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla L_t \tag{15}$$

where G_t represents the cumulative sum of squared gradients up to time step t, and ϵ is a small constant added to prevent division by zero.

RMSprop: AdaGrad is modified by RMSprop; it uses a moving average of squared gradients and improves performance in non-convex environments.

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)(\nabla L_t)^2 \tag{16}$$

$$W_t = W_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \nabla L_t \tag{17}$$

Adam (Adaptive Moment Estimation): Adam combines the ideas of momentum and RMSprop.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla L_t \tag{18}$$

$$W_t = W_{t-1} - \frac{\eta\, m_t}{\sqrt{v_t} + \epsilon} \tag{19}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla L_t)^2 \tag{20}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{21}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{22}$$

where: β_1 and β_2 are hyperparameters for the moving averages (typically $\beta_1 = 0.9$ and $\beta_2 = 0.999$).

For several epochs, repeat the forward and backward passes until the model converges or performs well enough. In conclusion, CNNs are very useful for a variety of computer vision problems because they use a hierarchical, multi-layered technique to adaptively and learn spatial hierarchies of characteristics from the given input images.

C. Introduction to YOLO

You Only Look Once (YOLO) is a state-of-the-art object detection system designed for real-time object detection. As opposed to conventional methods that employ classifiers at several scales and locations, YOLOv10 increases speed and efficiency by formulating object detection as a single regression problem. It estimate bounding boxes and class probabilities from the entire image in a single assessment.

YOLO use SxS grid to divide the input image. A confidence score p, on the basis of coordinates (x, y, w, h), indicates object presence and probable class. Each grid cell predicts multiple bounding boxes.

IV. Proposed Methodology

A. Car Dash-cam Surveillance Operations

The concept is to turn dash cams, which are already present in cars, into security cameras. These dash cams continuously capture images of the surroundings while the car is moving. In order to identify various items on the road, such as vehicles, pedestrians, traffic signs, and other relevant objects, the system analyses this video feed in real-time using the rapid object detection method YOLOv10. Owing to its remarkable quickness in identifying and classifying objects, YOLOv10 is highly appropriate for dynamic scenarios such as city streets.

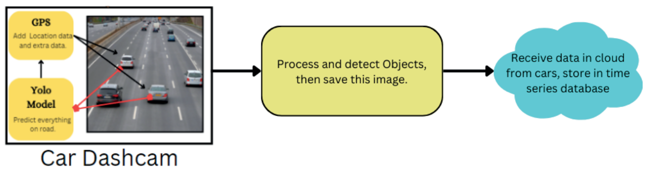
GPS
Add Location data and extra data.
Yolo Model
Predict everything on road.
Car Dashcam
Process and detect Objects, then save this image.
Receive data in cloud from cars, store in time series database

Fig. 1. Overall Architecture Diagram

B. Data Management and Extraction for Monitoring

YOLOv10 records the kind, count, and timestamped coordinates of objects of interest, which include anomalies and specific incidents (such accidents or suspicious behaviour), in addition to their coordinates. Only this metadata is regularly delivered to a central server in JSON format, not the whole video stream. The server issues a retrieval request back to the car with its registration number if more investigation is required, say to obtain a picture or a video of a certain location at a specific time. The dashcam-equipped vehicle uploads the necessary media data to the server upon request, allowing for thorough surveillance without requiring continuous high bandwidth data transfer.

By providing only metadata until more research is needed, this method guarantees the effective use of network resources, allays privacy concerns, and streamlines real-time monitoring and incident response in urban settings.

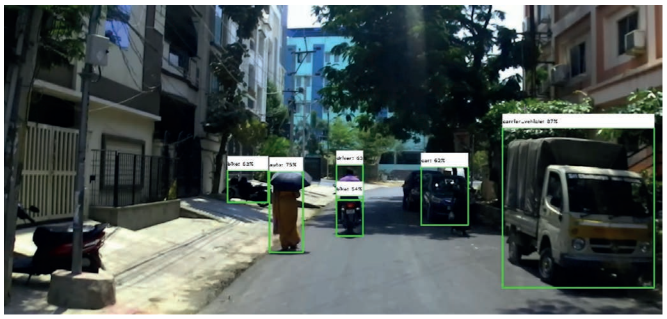

Fig. 2. Results of Car Dash-cam Algorithm Detection

The following pseudo code describes the algorithm used for real-time detection with the YOLOv10 and Server Processing for Dash-cam Data

Algorithm 1 Car Dash-cam Surveillance Algorithm

```
1: Initialization:
2: Start dashcam and system modules
3: Load pre-trained YOLOv10 model
4: while car is operational do
5:     frame ← CaptureVideoFrame()
6:     if frame = NULL then
7:         Continue to next iteration {Skip if frame capture failed}
8:     end if
9:     preprocessed_frame ← PreprocessFrame(frame)
10:    detected_objects ← YOLOv10_Detect(preprocessed_frame)
11:    metadata ← [ ]
12:    if detected_objects = ∅ then
13:        Continue to next iteration {No detections in this frame}
14:    end if
15:    for each object in detected_objects do
16:        object_type ← ClassifyObject(object)
17:        timestamp ← GetCurrentTimestamp()
18:        coordinates ← GetObjectCoordinates(object)
19:        if IsAnomaly(object) then
20:            anomaly_flag ← TRUE
21:        else
22:            anomaly_flag ← FALSE
23:        end if
24:        metadata_item ← CreateMetadataItem(object_type, timestamp, coordinates, anomaly_flag)
25:        metadata.append(metadata_item)
26:    end for
27:    formatted_metadata ← FormatToJSON(metadata)
28:    if NetworkAvailable() then
29:        SendToServer(formatted_metadata)
30:    else
31:        CacheLocally(formatted_metadata) {Store if no connection}
32:    end if
33:    if ServerRequestReceived() then
34:        request_details ← GetServerRequestDetails()
35:        requested_media ← RetrieveMedia(request_details)
36:        if requested_media ≠ NULL then
37:            UploadToServer(requested_media)
38:        else
39:            LogWarning("Requested media not found or corrupted")
40:        end if
41:    end if
42: end while
```

Algorithm 2 Server Processing for Dashcam Data

```
 1: Initialization:
 2: Start central server and database services
 3: Establish connections with all active vehicles
 4: while server is operational do
 5:   received_metadata ← ReceiveFromVehicles()
 6:   if received_metadata = ∅ then
 7:     Wait(DelayInterval) {No incoming data; pause briefly}
 8:     Continue
 9:   end if
10:   for each metadata_item in received_metadata do
11:     analyzed_event ← AnalyzeMetadata(metadata_item)
12:     if IsSignificantEvent(analyzed_event) then
13:       LogEvent(analyzed_event)
14:       TriggerAlert(analyzed_event)
15:       SendMediaRetrievalRequest(analyzed_event)
16:     end if
17:   end for
18:   if MediaDataReceived() then
19:     media_data ← ReceiveMediaData()
20:     if ValidateMediaData(media_data) then
21:       AssociateWithEvent(media_data)
22:     else
23:       LogWarning(”Received media data invalid or incomplete”)
24:     end if
25:   end if
26:   GenerateReports()
27:   UpdateRealTimeMonitoringDashboard()
28: end while
```

V. Proposed Use Cases

A. Case Study: Accident Prevention and Traffic Management

This study aims to show how dash cams on cars will help us in real-time traffic monitoring and spot accidents or other possible dangers on the road. Dash cams are mounted in automobiles and continuously record and transmit data to a central server, enabling prompt replies and proactive measures to lessen traffic and enhance road safety.

Benefits: Traffic flow can be significantly improved in cities by incorporating dashcam data into traffic management systems. With the help of real-time detection of anomalies like accidents or increased vehicle overflow, traffic problems can be solved quickly.

B. Case Study: Public Safety and Law Enforcement

Sensitive locations and high-crime areas can also be monitored using real-time data, which can be helpful for authorities to take necessary action.

Benefits: With enhanced real-time data, quicker responses can be made during emergencies or criminal activities. This will help in enhancing road safety and security. Dashcam technology will eliminate the overhead of camera installation and monitoring.

VI. Conclusion

In order to emphasize road safety, privacy, and efficiency while adhering to regulations and building public confidence, more secure and effective intelligent traffic management systems may be created with help of this technology. To improve real-time data fusion capabilities, future research should concentrate on integrating various data sources, such as traffic management systems, IoT sensors, and dash board cameras. In dynamic traffic settings, this integration will facilitate thorough situational awareness and well-informed decision-making.

References

[1] S. Kishore, R. R. Nair, and T. Babu, “Enhancing road safety: A smart accident alert system for rapid emergency response,” in *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2024, pp. 1325–1330.

[2] S. Latif, H. Afzaal, and N. A. Zafar, “Intelligent traffic monitoring and guidance system for smart city,” in *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2018, pp. 1–6.

[3] R. Petrolo, V. Loscri, and N. Mitton, “Towards a smart city based on cloud of things: A survey on the smart city vision and paradigms,” *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 1, p. e2931, 2017.

[4] G. Raja, G. Saravanan, S. B. Prathiba, Z. Akhtar, S. A. Khowaja, and K. Dev, “Smart navigation and energy management framework for autonomous electric vehicles in complex environments,” *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 18 641–18 650, 2023.

[5] Y. Maruyama and G. Ohashi, “Accident prediction model using divergence between visual attention and focus of expansion in vehicle-mounted camera images,” *IEEE Access*, vol. 11, pp. 140 116–140 125, 2023.

[6] Z. Shan, L. Shi, B. Li, Y. Zhang, X. Zhang, and W. Chen, “Empowering smart city situational awareness via big mobile data,” *Frontiers of Information Technology & Electronic Engineering*, vol. 25, no. 2, pp. 286–307, 2024.

[7] M. Xiao, L. Chen, H. Feng, Z. Peng, and Q. Long, “Smart city public transportation route planning based on multi-objective optimization: A review,” *Archives of Computational Methods in Engineering*, pp. 1–25, 2024.

[8] M. Sunagawa, K. Takahashi, S. I. Shikii, and M. Yoshioka, “Dashcam-based driver monitor for semi-autonomous driving,” in *2023 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2023, pp. 1–2.

[9] M. Simoncini, D. C. de Andrade, S. Salti, L. Taccari, F. Schoen, and F. Sambo, “Two-stream neural architecture for unsafe maneuvers classification from dashcam videos and gps/imu sensors,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[10] A. Rocky, Q. J. Wu, and W. Zhang, “Review of accident detection methods using dashcam videos for autonomous driving vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 8356–8374, August 2024.

[11] J. S. Park, A. Lee, K. W. Lee, S. W. Shin, S. S. Htun, and J. H. Han, “Ai-based modeling architecture to detect traffic anomalies from dashcam videos,” in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2022, pp. 1480–1482.

[12] B. M. Rao, S. D. Nair, and R. Dhanalakshmi, “Accident detection using densenet,” in *2022 IEEE International Conference on Data Science and Information System (ICDSIS)*. IEEE, 2022, pp. 1–4.

[13] J. S. Jhala, C. Joshi, and D. Anand, “Deep learning driven object detection and classification for autonomous vehicles in diverse traffic and weather conditions,” in *2024 1st International Conference on Emerging Technologies for Dependable Internet of Things (ICETI)*. IEEE, 2024, pp. 1–8.

[14] I. Cho, P. K. Rajendran, T. Kim, and D. Har, “Reinforcement learning for predicting traffic accidents,” in *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 2023, pp. 684–688.