IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 November 2024, accepted 22 December 2024, date of publication 31 December 2024, date of current version 13 January 2025.

Digital Object Identifier 10.1109/ACCESS.2024.3524676

RESEARCH ARTICLE

SafeTrax: Smart Collision Prediction and Alert System Using IoT for Sustainable Traffic Safety

MD. FAHIM BIN AMIN, ISRAT JAHAN KHAN, FARIA AKTER, ADIB AHMED, AND MD. MOTAHARUL ISLAM, (Member, IEEE)
Department of Computer Science and Engineering, United International University, Dhaka 1212, Bangladesh
Corresponding author: Md. Motaharul Islam (motaharul@cse.uiu.ac.bd)

This work was supported by the Institute for Advanced Research Publication Grant of United International University under Grant IAR-2024-Pub-065.

ABSTRACT With the rise in traffic accidents worldwide, it has become very important to have a more effective way to ensure road safety. To address this issue, we have made a system called SafeTrax. SafeTrax uses deep learning to predict traffic accidents and car crashes. Additionally, it has been implemented with the Internet of Things (IoT) to gather real-time data from sensors, and AWS Greengrass to process that data quickly and give drivers timely warnings. Unlike old safety systems, SafeTrax can predict if an accident might happen. It warns drivers beforehand to make roads safer for cars, people, and other animals. SafeTrax uses video frames from camera sensors in cars to look for dangers. It then uses a special deep-learning program to figure out if a crash might happen. One important part of SafeTrax is the integration of AWS IoT Greengrass. It helps the system work faster by using cloud services on devices like the Raspberry Pi. This means the system can quickly check the data and send warnings in time. SafeTrax can ensure safer roads, and it can help save lives.

INDEX TERMS Traffic, collision, SafeTrax, AWS IoT Greengrass, deep learning, IoT, pedestrian.

I. INTRODUCTION

Metropolitan regions that are rapidly broadening, gridlocked roads, and vehicular crashes have become significant issues for modern-day society. The frequency of catastrophes has spiked as more cars are on the road, which may result in significant consequences like loss of life, and monetary difficulties for families [36]. Traffic accident prevention has been an essential concern due to these worrisome statistics, prompting cutting-edge technologies that may reduce the risk of crashes [1], [2], [35]. Authorities as well as companies are implementing progressively stricter laws regarding traffic, but there is still an urgent requirement for solutions that may predict and avoid collisions from happening before they strike. This emphasizes the significance of establishing cutting-edge technology that improves drivers' awareness of their surroundings in real time by warning them about potential dangers beforehand [3], [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Tariq Umer.

The challenging endeavor is to build a system that can predict accurately and prevent catastrophes. The dynamic and unpredictable nature of road conditions can quickly turn routine driving scenarios into hazardous situations [5]. The economic challenges and modern technologies show the obstacles in the way of developing safer roadways [6], [7]. The dynamic circumstances of traffic and external variables such as weather aggravate these problems, emphasizing the need for innovative, cost-effective solutions [8], [9], [10], [11].

In recent years, so many better models with highly effective factors and frameworks have been developed by a couple of researchers to resolve such challenging scenarios and to prevent crash scenarios on roadways [37]. Traditional methods often come up short of maintaining the level of complexity since they rely upon human attention along with straightforward resources. Several methods like the Chain of Road Traffic Incident (CRTI) framework, Deep Spatio-Temporal Graph Convolutional Networks (DSTGCN), and Hybrid Learning Networks have been introduced for this purpose [12], [13], [14], [15]. These

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

methods use machine learning and artificial intelligence (AI) to improve how well accidents can be predicted. These systems are capable of predicting potential collisions and giving early warnings to drivers by analyzing data from vehicle movements, weather conditions, and traffic flow [16], [17]. Despite being of great progress, some of these approaches are still not fully efficient as they lack the scenarios of testing these in real-world situations or do not incorporate IoT technologies that allow for real-time transmission [18].

Recognizing these challenges, our research offers a proactive and real-time approach to predict whether any possible collision can occur or not. In contrast to the majority of earlier research, which concentrated on collision detection by identifying accidents after they occur. To greatly improve road safety, we have developed a state-of-the-art collision prediction system in our research that makes use of deep learning models and IoT-based communication. The transition from detection to prevention has proven to be pivotal, as it enables prompt notifications and proactive measures, providing drivers with a vital window of time to avert accidents completely.

Affordability while being a practical solution, has been the main focus of our work. To contrast a complete safety system, we have created an effective pipeline that smoothly integrates ambient data, real-time from onboard sensors, and deep learning models. Vehicles can enhance their object detection capabilities and collision prediction using SafeTrax by utilizing real-time sensor data, leading to improved accuracy and responsiveness of the prediction system. Our work not only pushes the boundaries of existing road safety technologies but also demonstrates how deep learning and IoT systems can be effectively utilized.

The significance of this research lies in its potential to make far-reaching changes in road safety. By incorporating such a system that predicts collisions and alerts drivers in advance, we have offered a practical solution that can be implemented in a wide range of scenarios with less economic challenges. Our work has pushed the boundaries of existing collision detection technologies and also has provided a blueprint for how deep-learning models and IoT-based communication systems can work together in harmony to create safer roadways. Addressing a critical need in today's increasingly congested traffic environments, this innovative approach aims to mitigate traffic accidents and ultimately save lives.

The major contributions to our work have been mentioned below:

- We have proposed a forward-thinking solution that integrates IoT and Deep Learning by equipping vehicles with sensors and dashcams, and it creates a network system that continuously feeds real-time data into a central hub where sophisticated algorithms predict potential collisions with high accuracy.
- We have analyzed a predictive algorithmic process to predict potential accidents or collisions by using our highly adaptable and affordable model for all kinds of vehicles.
- We have contributed to the democratization of road safety by developing a system that can operate across both urban and rural contexts, greatly improving accident prediction accuracy and providing timely alerts to drivers, ensuring broad and effective coverage.
- Our core Deep learning architectures have shown strong deep learning performance. ResNet50 has achieved 93.03% accuracy and a 95.57% F1-score, while MobileNetV2 has slightly outperformed with 93.3% accuracy and a 96.08% F1-score. VGG-16, though slightly lower, has still performed well with 90.75% accuracy and a 93.72% F1-score.

The rest of this paper is organized as follows. In Section II, we introduce related work about some previous research closely aligned with our field. Section III proposes information about the dataset and methodologies including MobileNetV2, ResNet50, VGG16, etc., with their model architectures and algorithms. Experiments in Section IV verify the feasibility and performance of the proposed method.

II. RELATED WORKS

We have studied some previous research closely aligned with our field. We have presented our in-depth analysis of them in this section. However, this section is divided into two parts: literature review and gap analysis.

A. LITERATURE REVIEW

Recent advancements in traffic accident prediction have leveraged various computational models and frameworks to enhance accuracy and efficiency in complex traffic scenarios. Xiong et al. [19] have introduced the innovative Chain of Road Traffic Incident (CRTI) framework for improved accident prediction, leveraging vehicle movement data through a two-pronged approach: an initial separation of vehicular lane behavior via a Support Vector Machine (SVM), followed by differentiation between accident and non-accident events using Gaussian-mixture-based Hidden Markov Models (HMM). For accident prevention under complex traffic scenarios, their findings validated by PreScan platform simulations, have underscored the CRTI framework's potential to enhance early warning systems.

Malawade et al. [20] have critiqued past collision prediction methods for autonomous vehicles, highlighting the flaws in early statistical metrics and model-based approaches due to unrealistic assumptions and high false-positive rates, respectively. To more accurately predict collision, they have proposed scene-graph-based methods by integrating Multi-Relational Graph Convolutional Networks (MR-GCN) with LSTM for spatial and temporal analysis, and collisions by understanding dynamic inter-object relationships. Although specific results have not been detailed, this approach has shown improved prediction accuracy and timeliness.

Venkateswaran et al. [21] have explored how deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized vehicle detection, tracking, and distance estimation for forward collision warning systems. They have showcased methods like R-CNN, Fast R-CNN, and YOLO for detection and have employed correlation filters with deep learning for the improved tracking of dynamic objects. Planar homography and Inverse Perspective Mapping (IPM) techniques have been discussed for distance estimation, with IPM's accuracy having been boosted through road boundary detection and automatic calibration. Zhang et al. [22] have reviewed various motion prediction models, distinguishing them as physics-based, maneuver-based, and interaction-aware models. Their methodology has focused on evaluating models like Constant Velocity and Constant Acceleration for real-time applications, and maneuver-based models using techniques such as SVMs and HMMs, while also considering interaction impacts between vehicles. They have highlighted the improvement in prediction accuracy through the use of Interactive Multiple Models (IMM) which blend physics and maneuver-based approaches for dynamic scenarios. The research has confirmed the effectiveness of these models in real-world traffic scenarios, particularly in assessing risks for lane changes and rear-end collisions.

Gus et al. [24] have explored Advanced Driver Assistance Systems (ADAS) technology, focusing on forward-collision warnings, and have employed the Gaussian Mixture Model (GMM) for time-to-collision estimates and used LSTM and Deep Belief Network (DBN) models to predict collision risks. Their methodology has been tested on real-world urban freeway data, demonstrating the predictive accuracy of these models. Dong et al. [25] have developed a model blending unsupervised learning to uncover complex interactions in traffic data, and supervised learning through negative binomial regression, effectively enhancing crash prediction accuracy. When applied to Tennessee data, their dual approach has significantly outperformed traditional models. Using sensors to detect accidents and report their locations via GPS. Sahoo et al. [26] have developed an IoT-based automated system for rapid and accurate traffic crash reporting. Significantly reducing response times and enhancing the effectiveness of emergency services, their approach has integrated low-cost technologies to improve vehicle safety. Yu et al. [27] have introduced the Deep Spatio-Temporal Graph Convolutional Network (DSTGCN) for traffic accident prediction, employing data from various sources such as traffic records and meteorological conditions. In this model, spatial and temporal data analysis have been combined with external data integration to demonstrate notable advancements in traffic safety and management by outperforming existing models.

Kashifi et al. [28] have introduced a Deep Hybrid Learning Framework to forecast short-term traffic crashes through a Deep Spatiotemporal Hybrid Network (DSHN) that integrates CNNs and LSTMs. This approach has processed varied traffic data, increasing prediction accuracy by capturing spatiotemporal patterns. The results have indicated the superior performance of the DSHN model over traditional and machine learning models, evidenced by improved precision, recall, F1-score, and AUC metrics. Thapa et al. [29] have improved duration-based models to predict crash likelihoods and severities by integrating both static and dynamic factors like traffic and weather conditions. Using data from Tennessee, they developed a validated model that has shown effective prediction with a balanced dataset, highlighting the importance of certain variables for accuracy. In the study by Rathore et al., [30], a smart traffic control model has been presented, utilizing vehicular cameras and fog computing for real-time traffic violation monitoring. The approach has employed GPU-based fog devices with Apache Spark for processing, leveraging techniques like SSD for vehicle detection and the Sobel-Feldman edge detection for lane identification to detect violations such as illegal U-turns and driving on central dividers. By implementing it on a GTX750-Ti GPU machine, the system has shown high accuracy and efficiency in urban traffic management. Pramanik et al. [31] have designed a video surveillance system to enhance road safety, incorporating five algorithms (SVD, OTD, OD, IPD, and WDLPD) for detecting traffic pre-events like speed violations and illegal parking. Their methodology has demonstrated effectiveness by analyzing 132 real-time traffic videos from India, outperforming current methods in pre-event detection. Islam et al. [32] have developed a smart device that can detect motorcycle accidents and facilitate immediate assistance for injured individuals. The device aims to minimize casualties by quickly sending alert messages to hospitals, police stations, and registered family members after an accident. This initiative addresses the critical gap in timely medical response to motorcycle crashes in Bangladesh, where road accidents and fatalities are rising.

B. BENCHMARK OF PREVIOUS RESEARCH WORKS

This section presents a benchmark of related research works and compares how the previous studies implemented their proposed methodologies and how well they performed. By doing this, we have decided what has improved in road safety and what still needs more research. This comparison has clarified various methodologies for us and illuminated how our research could contribute to enhancing traffic safety. Table 1 shows results and key points of previous studies on road safety and traffic accident detection.

C. GAP ANALYSIS

In reviewing past studies on road safety, and the detection and forecasting of traffic accidents, it has been found that there is a significant reliance on older techniques and costly setups. This review emphasizes the critical need for improved and more readily available approaches that utilize the latest technology to enhance the accuracy of accident

TABLE 1. Benchmark of literature review.

Authors	Year	Contribution	Methodology	Performance Metrics	Key Techniques	Notable Results
Xiong et al. [19]	2017	Proposed CRTI framework for accident prediction.	Two-stage SVM and HMM.	Not specified	SVM, HMM	Improved early warning for accident prevention.
Malawade et al. [20]	2022	Reviewed collision prediction methods for AVs.	Probabilistic and scene-graph methods.	Prediction accuracy, timeliness	MR-GCN, LSTM	Improved over previous methods (details not included).
Venkateswaran et al. [21]	2021	Advanced vehicle detection and tracking.	CNNs and deep learning techniques.	Detection accuracy, tracking performance	CNN, R-CNN, YOLO	Enhanced object detection and tracking accuracy.
Zhang et al. [22]	2020	Reviewed motion prediction models.	Physics-based categorization.	Prediction accuracy	IMM, SVM, HMM	Improved risk assessment for lane changes and rear-end collisions.
Guo et al. [24]	2022	Reviewed ADAS technologies for collision warnings.	TTC and ML models.	Predictive accuracy	GMM, LSTM, DBN	Effective time-to-collision estimates.
Dong et al. [25]	2018	Introduced hybrid learning model.	Unsupervised interactions with regression.	Crash prediction accuracy	Unsupervised + NB regression	Outperformed traditional models on Tennessee data.
Sahoo et al. [26]	2020	Developed an automated crash reporting system.	Real-time reporting with GPS data.	Response time	GPS, sensors	Reduced emergency response times.
Yu et al. [27]	2021	Introduced DSTGCN for traffic accident prediction.	Spatiotemporal integration.	Model performance comparison	Graph, spatiotemporal analysis	Outperformed existing models.
Kashifi et al. [28]	2023	Presented deep hybrid learning framework.	Integrated CNNs and LSTMs.	Precision, recall, F1-score, AUC	CNN, LSTM	Superior to traditional and ML models.
Thapa et al. [29]	2024	Extended duration-based modeling.	Performance and estimation trade-offs.	Prediction accuracy	Static and dynamic factor integration	Improved crash prediction on high-risk segments.
Rathore et al. [30]	2021	Introduced smart traffic control model.	Real-time analysis with vehicular cameras.	Accuracy, efficiency	SSD, Sobel-Feldman edge detection	High accuracy in urban traffic management.
Pramanik et al. [31]	2021	Proposed video surveillance for traffic pre-events.	Multiple algorithms for anomaly detection.	Pre-event detection accuracy	SVD, OTD, OD, IPD, WDLPD	Outperformed current methods in pre-event detection.

prediction systems and improve road safety. Table 2 offers a comprehensive overview of the results and limitations of past studies, indicating the areas that still have the scope for further research and development.

III. PROPOSED SYSTEM

In this section, we have provided the dataset information, detailed methodology, architecture details, and algorithm of our work. This section is divided into seven subsections: System Overview, Dataset, Model Architecture, IoT Architecture, IoT Implementation Process, Algorithm, and AWS Greengrass in SafeTrax.

A. SYSTEM OVERVIEW

Our system integrates advanced IoT devices, deep learning, and AWS Greengrass for a real-time traffic safety solution. The car has IoT sensors and dash cameras that gather important information about its surroundings, like how close it is to other vehicles. Raspberry Pi collects real-time data from all the connected sensors and cameras. At the central hub, trained deep-learning models (MobileNetV2, VGG-16, and ResNet50) are used. These models check if an accident might happen using data from the cameras. The camera modules continuously monitor the surroundings by providing real-time video input for the deep-learning models. They analyze the camera data to identify objects like cars, people, and other structures. Deep learning models are set up on the Raspberry Pi using AWS Greengrass, which lets the Pi run these models right on the device for quick analysis and sending alerts. AWS Greengrass also helps the Raspberry Pi connect smoothly to the cloud. It allows real-time decision-making on the device and more complex tasks to be done in the cloud. Once the deep learning models detect objects, the system moves to the next stage of analysis. If a potential object is detected, the vehicle's sonar sensors are activated to measure the distance between the vehicle and the object. When the sonar sensors find a possible collision, they check how close nearby objects are. If the object gets too close to the car, the IoT system alerts the driver with an alarm. These sensors also help with spotting obstacles. By using real-time data from the sensors, our system helps vehicles notice nearby collision threats. These sensors work with predefined safety distances; if the distance falls below a safe margin, the system classifies it as a high-risk situation and generates an immediate alert to warn the driver. The combination of Deep Learning, IoT, and AWS Greengrass enables accurate collision predictions, reduces false positives, and enhances real-time decision-making, adapting to changing traffic conditions through a continuous feedback loop. This proactive approach aims to significantly reduce accidents by providing early warnings to drivers and facilitating quick evasive actions. Figure 2 illustrates the simple workflow diagram of SafeTrax.

B. DATASET

The BDD100K dataset, which contains 100,000 images of car collisions, has been taken as the main dataset for our study. We have focused on a subset of 10,000 images for binary classification which have been annotated manually by us. We categorize them into two classes: 'Collision' (label 'y') and 'No Collision' (label 'n'). This dataset has been divided into training (70%), validation (10%), and testing (20%) subsets. It has allowed for robust model training and evaluation. The careful annotation and organization aim to support effective model development in traffic accident detection and analysis. Our customized dataset can be accessed on Kaggle. Table 3 represents the necessary information for the BDD100K dataset.

TABLE 2. Gap analysis table.

Area of Focus	Current State	Gaps Identified	Potential Future Research	Relevance to SafeTrax	Proposed Enhancements in SafeTrax
Data Integration	Multiple studies use various data sources (traffic, weather, IoT sensors)	Lack of standardized approach for integrating diverse data types	Develop unified frameworks for seamless integration of heterogeneous data sources	SafeTrax relies on integrating several data sources (vehicle cameras, IoT sensors, weather data) to predict potential accidents.	SafeTrax integrates multiple data types in real-time using AWS IoT Greengrass, ensuring continuous data processing for accurate predictions.
Real-time Processing	Some studies implement real-time or near-real-time systems	Limited scalability and processing speed for large-scale, real-time applications	Explore edge computing and distributed processing techniques for faster, more efficient real-time analysis	The success of SafeTrax depends on low-latency data processing and alert generation to prevent collisions in real time.	SafeTrax uses edge computing techniques and AWS IoT Greengrass to enhance real-time data processing and reduce latency, ensuring immediate alerts.
Prediction Accuracy	Improved accuracy using advanced ML/DL techniques	High false-positive rates in some models; difficulty in rare event prediction	Investigate hybrid models and ensemble methods to reduce false positives and improve rare event detection	Accurate accident prediction is crucial to SafeTrax's goal to prevent collisions.	By employing ensemble learning models and hyperparameter tuning, SafeTrax reduces false positives and improves accuracy in complex conditions.
Spatiotemporal Modeling	Several studies incorporate spatiotemporal aspects	Limited consideration of long-term temporal dependencies and spatial correlations	Develop advanced spatiotemporal models that capture both short-term and long-term patterns across various spatial scales	SafeTrax must consider both spatial and temporal factors in traffic dynamics to predict collisions effectively.	SafeTrax analyzes data in real time, considering vehicle trajectories over time and space to predict accidents in dynamic traffic conditions.
Interpretability	Focus primarily on predictive performance	Lack of explainable AI approaches in complex models	Incorporate interpretable AI techniques to enhance model transparency and trustworthiness	Explainable AI is key to ensuring trust and adoption of SafeTrax among drivers and regulators.	SafeTrax incorporates interpretable AI techniques to provide clear reasoning behind predictions, improving trust and understanding.
Generalizability	Most studies are region-specific or use limited datasets	Models may not generalize well to different geographic or traffic contexts	Conduct cross-regional studies and develop transfer learning approaches for improved model generalization	SafeTrax must work across various environments (urban and rural) and different road conditions.	SafeTrax uses a diverse dataset that includes urban, rural, and various weather conditions to improve its generalizability across regions.
Multi-modal Analysis	Some studies consider multiple factors (e.g., traffic, weather)	Limited integration of non-traditional data sources (e.g., social media, urban planning data)	Explore the incorporation of diverse data modalities for a more comprehensive understanding of traffic dynamics	SafeTrax must process multiple data sources (cameras, sensors, weather) simultaneously to predict accidents accurately.	SafeTrax integrates vehicle cameras, sonar sensors, and environmental data (like weather) to provide a comprehensive understanding of traffic.
Human Factors	Primarily focus on vehicle and environmental factors	Limited consideration of driver behavior and human factors	Integrate driver behavior models and human-centric approaches in prediction systems	SafeTrax considers both driver behavior and external factors (traffic, weather) in predicting collisions.	SafeTrax integrates behavioral data (speed, lane changes) and environmental data to enhance the accuracy of accident predictions.
Proactive Measures	Emphasis on prediction and early warning	Limited focus on proactive intervention strategies	Developing AI-driven, adaptive traffic management systems for real-time accident prevention	Proactive warnings are central to SafeTrax's goal of preventing accidents before they occur.	SafeTrax continuously monitors traffic and provides real-time alerts to drivers, helping them avoid accidents before they happen.
Validation and Benchmarking	Varied evaluation metrics across studies	Lack of standardized benchmarks for fair comparison of different approaches	Established common datasets and evaluation frameworks for consistent model comparison and validation	Effective validation is critical to ensure SafeTrax's reliability in predicting accidents.	SafeTrax uses standardized evaluation metrics, comparing them with existing models to ensure high accuracy in diverse environments.

1) DATASET PREPROCESSING

Preprocessing our custom dataset is crucial for preparing the data for model training and testing. The applied preprocessing steps are given below:

2) DATA CLEANING

Before starting the preprocessing steps, it is important to clean the dataset. It involves removing corrupted or damaged images, as well as filtering out images that are not relevant to our system. These images could cause confusion during the training of deep learning models. These actions have enhanced the quality of our dataset and ensured more effective training.

a: DATA AUGMENTATION

To make the model more reliable, we have applied data augmentation techniques. We have applied horizontal flipping to some images to introduce variability in object orientation. We have also utilized random rotation with rotation angles ranging from 0 to 75 degrees. It introduces angular variation and helps the model handle different orientations. Additionally, zooming techniques have been applied to create a clearer view of distant objects. It allows the model to better recognize and learn from objects at various distances.

b: RESIZING

All images have been resized to an exact resolution of 256 × 256 pixels. This uniform size ensures that all inputs to the model are the same. It makes training and processing more efficient.

c: NORMALIZATION

After resizing, the pixel values of the images have been normalized to a range between 0 and 1. It has been done by dividing each pixel value by 255. This step has improved the training process and helped the model run more effectively.

3) LABELING PROCESS PIPELINE FOR PRE-COLLISION DETECTION

We have employed an automated and systematic process for labeling images for pre-collision detection. This has been done by measuring the distance of objects from the camera modules and assigning labels to each image. This fully automated method has ensured accurate labeling of the dataset and eliminated the need for manual labeling.

a: CAMERA CALIBRATION AND OBJECT PROPERTIES

The first step involves determining the properties of the camera modules and the real-world dimensions of the objects. The focal length of the camera modules has been estimated based on the following properties:

- **Field of View (FOV):** 120Â°.
- **Image Resolution:** 1080p (1920 × 1080 pixels).
- **Focal Length (f):** The focal length has been calculated using the formula:

$$f = \frac{\text{Image Width}}{2 \times \tan\left(\frac{\text{FOV}}{2}\right)}$$

For estimating object size, we have defined ranges to account for the differences in the dimensions of vehicles and humans:

- **Car Width Range:** 1.6 to 2.2 meters.
- **Human Height Range:** 1.5 to 2.0 meters.

Bounding Box Detection and Object Classification Using a pre-trained object detection model (YOLOv8), bounding boxes are generated around vehicles and humans for each image. Along with the bounding boxes, the object types (e.g., car or human) are classified. The size of the object in pixels is extracted from the bounding box:

- **Vehicles:** The width of the bounding box is used.
- **Humans:** The height of the bounding box is used.

Distance Estimation Using Pinhole Camera Model The distance to each object is estimated using the pinhole camera model:

$$\text{Distance} = \frac{\text{Complete Object Size} \times f}{\text{Object Size in Pixels}}$$

where:

- **Complete Object Size** refers to the object's size based on its classification (vehicle or human), chosen within the defined range (1.6m - 2.2m for vehicles, 1.5m - 2.0m for humans).
- **Object Size in Pixels** is the size of the object in the image (width for vehicles, height for humans) which has been determined from bounding box size.
- **f** is the focal length calculated earlier.

This approach ensures the system can handle objects of varying sizes.

b: THRESHOLDING FOR PRE-COLLISION DETECTION AND AUTOMATIC LABELING

To determine whether an image represents a potential pre-collision scenario, the estimated distance has been compared to predefined thresholds. For vehicles, the threshold has been set within a range of 12 to 18 meters, while for humans, it has been set within a range of 8 to 12 meters. If the estimated distance is below the set threshold, the image has been labeled as a pre-collision scenario ('y'); if it is not, it has been labeled as a non-pre-collision scenario ('n').

This range-based approach allows the system to account for varying object sizes, enhancing the robustness of the labeling process. Figure 1 illustrates the overview of how we have made annotations to our custom dataset.

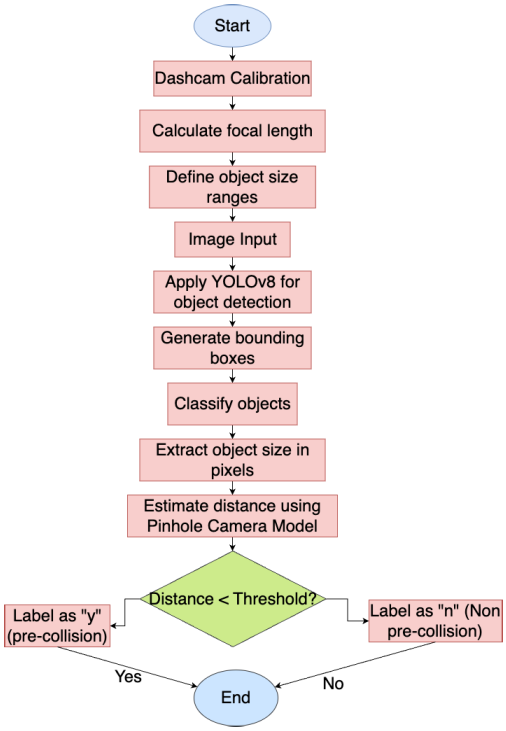


FIGURE 1. Pipeline diagram of the dataset annotation process.

C. MODEL ARCHITECTURE

Our model architecture has been designed as a single pipeline focusing on deep learning-based accident prediction and IoT-based alert system integration. Therefore, this subsection has been divided into four parts: ResNet50 Architecture, VGG16 Architecture, and MobileNetV2 architecture.

1) RESNET50 ARCHITECTURE

We have used the ResNet50 model architecture, which has been trained on the ImageNet dataset and accepts an input size of 224 × 224 × 3. The model begins with a 7 × 7 convolution with 64 filters and a stride of 2, followed by a max-pooling layer. ResNet50 includes residual blocks with increasing filter counts across four stages: 3 blocks with 64 filters, 4 blocks with 128 filters, 6 blocks with 256 filters,

TABLE 3. Overview of BDD100K dataset.

Feature	Description
Dataset Name	BDD100K (Berkeley DeepDrive 100K)
Source	https://www.bdd100k.com/
Total Data Size	100,000 videos and images
Data Types	- Videos: 40-second driving clips - Images: High-resolution frames
Object Classes	Vehicles, pedestrians, cyclists, traffic lights, traffic signs, and other common road elements
Diversity	Collected from various locations in the U.S., with different weather (sunny, rainy, snowy) and times of day (day, night, dawn, dusk)
Applications	- Autonomous Driving - Traffic Scene Understanding - Multi-task Learning - Predicting Collision/Trajectory - Road Safety Research
Weather & Lighting	Supports multiple conditions: rainy, sunny, snowy, day, night

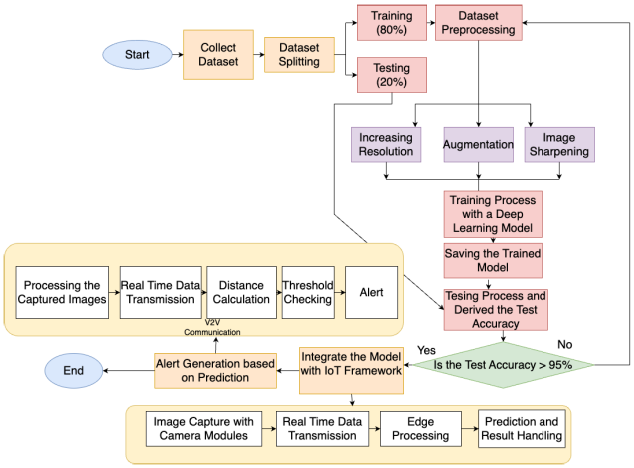


FIGURE 2. Workflow diagram of the research work.

and 3 blocks with 512 filters. A global average pooling layer generates a 2048-dimensional feature vector, and a custom linear layer replaces the fully connected layer to match the number of classes in the dataset. In feature extraction mode, the convolutional layers are frozen, which allows only the weights in the fully connected layer to be updated during training. Figure 4 represents a simple ResNet50 model architecture.

2) VGG-16 ARCHITECTURE

The pre-trained VGG-16 architecture has 13 convolutional layers and 3 fully connected layers that are designed for image classification. It accepts 224 × 224 × 3 RGB images and includes five max pooling layers: 64, 128, 256, and 512 filters in the blocks, all using 3 × 3 kernels with ReLU activations. The output is flattened and processed through three fully connected layers, the first two having 4096 units and Dropout for regularization. The final layer is adapted for binary classification. It produces 2 units with a softmax or sigmoid activation. Figure 5 represents simple VGG-16 model architecture.

3) MOBILENETV2 ARCHITECTURE

The model architecture uses pre-trained MobileNetV2 as the backbone feature extractor. It takes input images sized 256 × 256 pixels with 3 color channels. MobileNetV2 outputs a feature map of 8 × 8x1280. To enable binary classification, a Global Average Pooling layer reduces the feature map to a 1280-dimensional vector. This is followed by a Dense layer with 128 units and ReLU activation to capture non-linear relationships. A Dropout layer with a rate of 0.5 is also included to prevent overfitting. Finally, a Dense layer with a single unit and Sigmoid activation produces a probability score for binary classification. Figure 3 represents simple MobileNetV2 model architecture and table 4 provides breakdowns of the ResNet50, VGG-16, and MobileNetV2 architecture.

TABLE 4. Architecture of ResNet50, VGG-16, and MobileNetV2 for binary classification.

Model	Layer Type	Details
ResNet50	Input Layer	224x224, 3-channel image input
	Conv1 + Max Pool	7x7, 64 filters, stride 2; Output: 112x112
	Conv2	3 Residual Blocks; Output: 56x56
	Conv3	4 Residual Blocks; Output: 28x28
	Conv4	6 Residual Blocks; Output: 14x14
	Conv5	3 Residual Blocks; Output: 7x7
	Global Average Pooling	1x1, 2048 output
VGG-16	Input Layer	224 × 224 × 3 image
	Conv1 and Conv2	64 filters, 3 × 3
	MaxPooling1	2 × 2, stride 2
	Conv3 and Conv4	128 filters, 3 × 3
	MaxPooling2	2 × 2, stride 2
	Conv5, Conv6, and Conv7	256 filters, 3 × 3
	MaxPooling3	2 × 2, stride 2
	Conv8, Conv9, and Conv10	512 filters, 3 × 3
	MaxPooling4	2 × 2, stride 2
	Conv11, Conv12, and Conv13	512 filters, 3 × 3
Mobile-NetV2	Input Layer	Image of size 256×256×3 (RGB)
	MobileNetV2 Backbone	Feature Extraction, Output shape: 8 × 8 × 1280
	Global Average Pooling	Pooling, Reduces to 1 × 1280
	Dense (128 units)	Fully Connected, 128 units, ReLU activation
	Dropout (rate: 0.5)	Regularization, Dropout rate of 0.5 to prevent overfitting
	Dense (1 unit)	Fully Connected, 1 unit, Sigmoid activation for binary classification

D. IoT ARCHITECTURE

To dive into the IoT component, we need to discuss all the sensors, modules, and hardware required to create the prototype. We have used a Raspberry Pi, an ultrasonic sonar sensor, a Raspberry Pi camera module, LEDs, buzzers, and wires. This hardware setup provides a robust platform for implementing the pre-collision alert system, combining visual input, distance sensing, processing power, and alert mechanisms. The Raspberry Pi's capabilities allow for

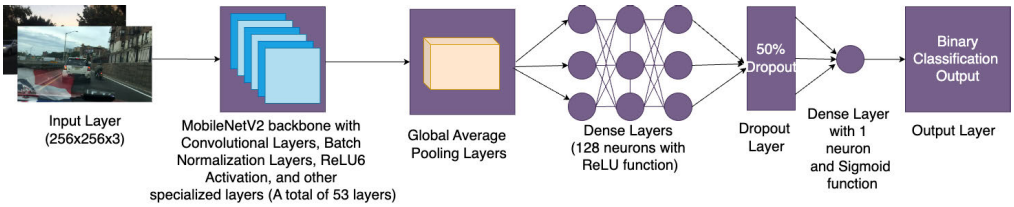


FIGURE 3. Model architecture diagram for MobileNetV2 [33].

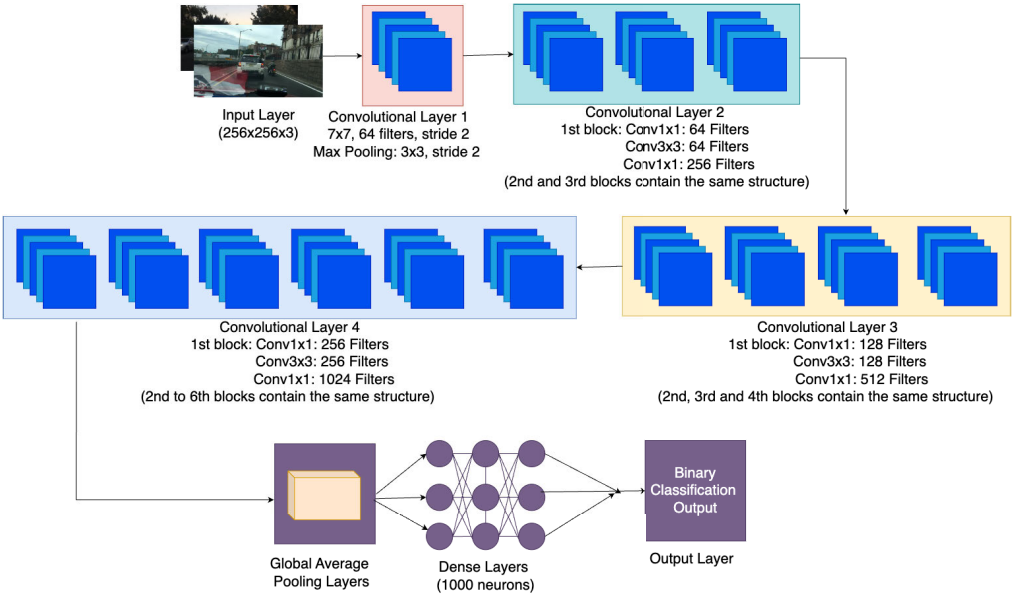


FIGURE 4. Model architecture diagram for ResNet50 [34].

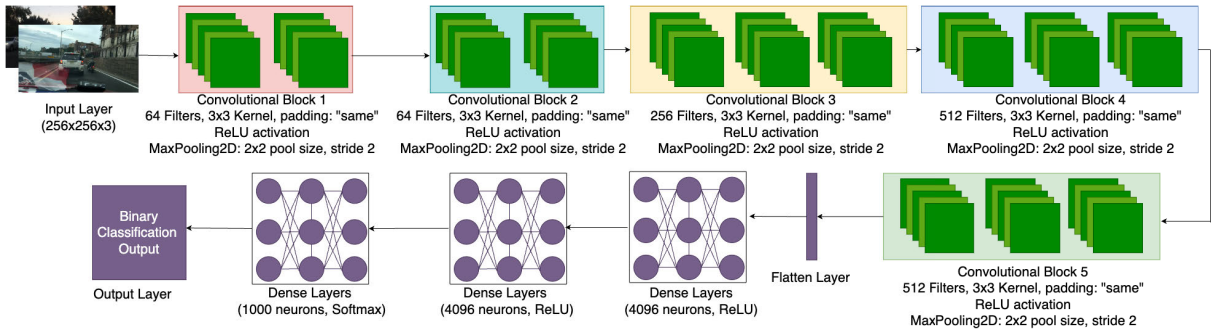


FIGURE 5. Model architecture diagram for VGG-16 [34].

running complex deep learning models, while the additional sensors and output devices create a comprehensive safety system.

E. IoT IMPLEMENTATION PROCESS

1) EDGE DEVICE (RASPBERRY PI 4)

The Raspberry Pi 4 is the main processing unit in our system. It runs AWS IoT Greengrass Core software for local data processing and inference. Our pre-trained deep learning models on the Raspberry Pi enable real-time collision risk assessments using input from connected sensors.

2) CAMERA MODULE

The Raspberry Pi Camera Module connects to the CSI port of the Raspberry Pi. We used the proper CLI commands to enable the camera. We have ensured good lighting for better image capture and regularly tested the camera to confirm it works properly.

3) SONAR SENSOR (HC-SR04)

The sonar sensor connections use 3.3V logic levels. Since we have used a 5V sensor, a voltage divider has been needed for the ECHO pin. We have also ensured that the sensor

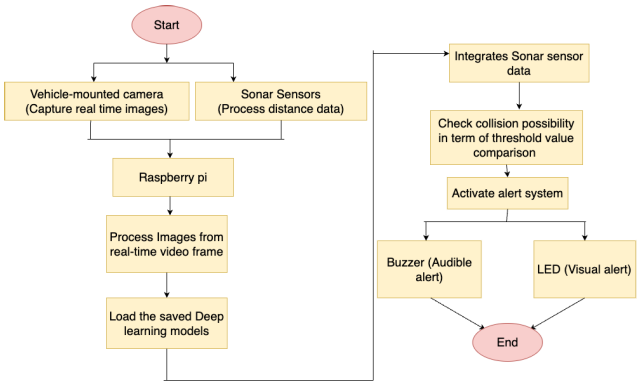


FIGURE 6. Overview diagram of IoT-based alert system.

is securely mounted and correctly positioned for accurate distance measurements. Additionally, we have tested the sensor in different environments to improve its reliability.

4) BUZZER

We have used an active buzzer, so the connections are straightforward. The buzzer is connected to GPIO 22, and we have ensured it is rated for the voltage used and can handle the current from the Raspberry Pi GPIO pin. We have also tested it at different frequencies to achieve the desired sound.

5) LED

We have used a 220 ohm resistor with our LED. The resistor has been placed between the GPIO pin and the LED's anode to prevent excessive current flow. We have ensured that the LED polarity is correct. Table 5 summarizes all the pin number details for the Raspberry Pi and its connected modules.

TABLE 5. Raspberry Pi 4 connections to peripherals.

Raspberry Pi 4	Peripherals
Pin 2 (5V)	Sonar VCC
Pin 6 (GND)	Sonar GND
Pin 11 (GPIO 17)	Sonar TRIG
Pin 13 (GPIO 27)	Sonar ECHO
Pin 15 (GPIO 22)	Buzzer (+)
Pin 34 (GND)	Buzzer (-)
Pin 12 (GPIO 18)	LED Anode (+)
Pin 39 (GND)	LED Cathode (-)
CSI Port	Camera Module

Figure 3, figure 4, and figure 5 illustrates a detailed diagram of our MobileNetV2, ResNet50, and VGG-16 model architecture respectively. These figures also showcase their various components and interactions, while Figure 7 presents the IoT architecture diagram.

F. ALGORITHM

Our research workflow integrates three deep-learning models to predict whether a collision will occur or not based on image features. This approach has leveraged IoT frameworks for real-time data aggregation and utilized smart traffic control systems for automated alert generation and safety enhancement in urban environments. Algorithm 1 is the detailed algorithm outlining our entire research process.

The algorithm is well-structured. It has defined its inputs, outputs, and purpose clearly through the procedure named "Car_Accident_Prediction". The main loop operates continuously while the system is active. In the loop, it encompasses key steps such as data retrieval, model prediction, object detection, and alert generation. In the algorithm, the variable named "distScore" is a metric that evaluates the distance of detected objects (such as vehicles or pedestrians) based on their predicted properties. Additionally, the algorithm uses a generalized distance calculation through "ComputeDistance," which adds flexibility. The use of "current alerts" for managing notifications is effective, while the "TransmitAlerts" function correctly informs both nearby vehicles and the driver, which is crucial for real-time operations. To further improve the algorithm, adding error handling, performance metrics, and clear termination conditions could enhance its reliability and scalability.

G. AWS IoT GREENGRASS IN SAFETRAX

AWS Greengrass enables SafeTrax to work independently and make faster decisions at the edge, even in low-bandwidth conditions. It allows data processing, device management, and machine learning locally. At the same time, devices remain connected to the cloud for updates and long-term data storage. This helps centralize the entire system and enhance its effectiveness. AWS IoT Greengrass enhances SafeTrax in this way:

1. Data Collection: SafeTrax integrated vehicle has a camera module for image data and a sonar sensor for measuring object distance. AWS IoT Greengrass preprocesses this image data locally by filtering out noise and preparing it for analysis.

2. Local Edge Processing: Greengrass can enable Raspberry Pi to perform local edge processing by running deep learning models directly on the device. The system uses sonar data to detect objects that are dangerously close and images to identify obstacles. By executing these models locally, it can provide real-time feedback.

3. Cloud-Based Deep Learning: Preprocessed data can be uploaded to the cloud for advanced analysis using deep learning models on AWS. These models predict potential collision risks by analyzing patterns from multiple vehicles. This transfers heavy computational tasks to the cloud, so edge devices can concentrate on real-time processing.

4. Real-Time Alerts: If there has been a lack of internet connectivity, the system has to rely on the edge processing performed by Greengrass to continue providing essential alerts based on the locally collected data.

1) IMPLEMENTATION PROCESS OF AWS IoT GREENGRASS

We have installed the AWS IoT Greengrass Core software on our Raspberry Pi and set up a Greengrass group in the AWS IoT console. Key components of our system include a Python script as the core of the application and deep learning models for handling machine learning tasks. We have configured

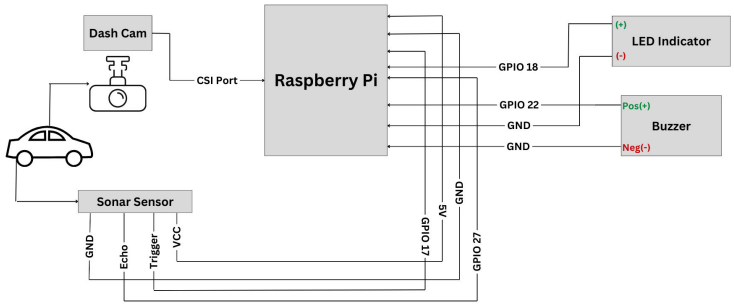


FIGURE 7. Architectural diagram of IoT portion.

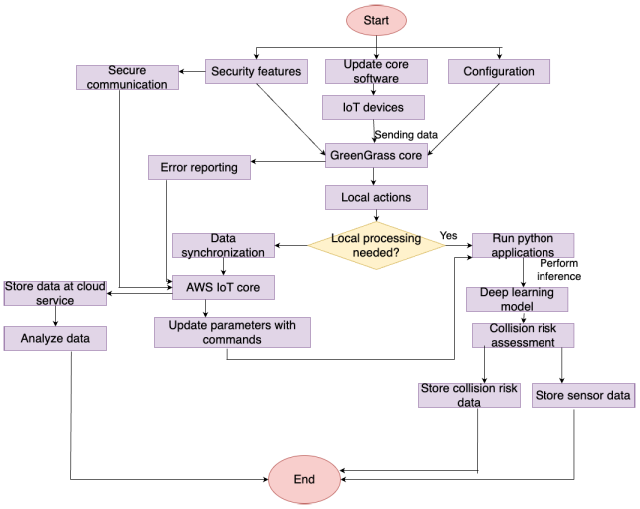


FIGURE 8. Workflow diagram of AWS GreenGrass.

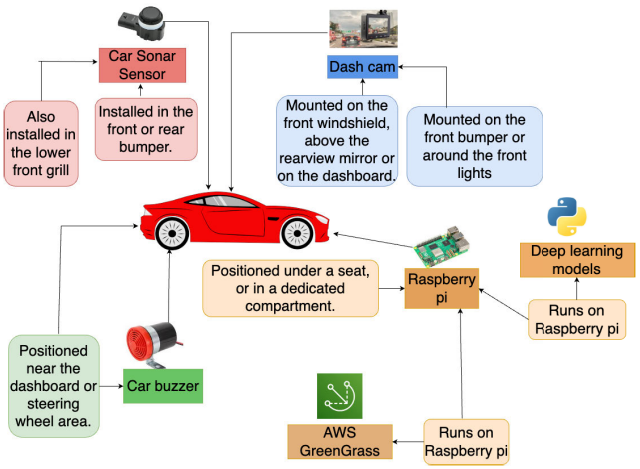


FIGURE 9. Diagram design of a SafeTrax integrated car.

these models as Greengrass components for local inference within the main application. This setup enables us to assess collision risks in real-time. We have also set up local storage to save data on collision risks and sensor readings. To keep everything updated, we implemented a process to sync this data and periodically upload it to AWS IoT Core. Each data message includes the measured distance, collision risk, and a timestamp. This information is sent to AWS IoT Core to ensure consistent availability for further analysis. We have implemented error reporting to AWS IoT Core to facilitate remote monitoring and diagnostics. This system allows us to track issues in real time and enables prompt responses to errors. By sending error logs and relevant information to the cloud, we can efficiently analyze and address problems. Additionally, we have utilized Greengrass's security features to ensure our devices are secure when communicating with AWS IoT Core. We have implemented device authentication to verify the identity of each device and employed encrypted communication to protect data sent over the network. These methods help prevent unauthorized access and protect against data breaches. Figure 8 represents the workflow diagram of AWS GreenGrass. Figure 10 represents the diagram for the working procedure of the key functions.

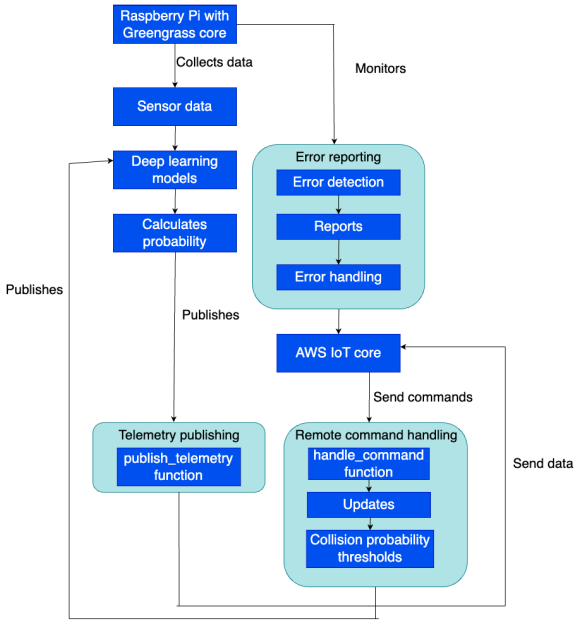


FIGURE 10. Diagram for working procedure of key functions.

IV. PERFORMANCE EVALUATION

This section outlines the evaluation metrics used in our research and provides a detailed assessment of SafeTrax's

Algorithm 1 Car Accident Prediction and Alert System

Input: Real-time sensor data stream S, Pre-trained model M, Distance threshold T
Output: Real-time alerts A

procedure Car_Accident_Prediction
  Initialization:
    Initialize empty alert set $A = \{\}$

  Main Loop:
  while System is active **do**
    $X \leftarrow$ GetNextDataFrame(S)
    $Y_{\text{pred}} \leftarrow M$.Predict(Preprocess($X$))
    objects $\leftarrow$ [o for $o \in$ DetectObjects(Y_{pred})
      if distScore greater than THRESHOLD]
    current_alerts $\leftarrow$ []
    for each pair (i, j) in detected_objects **do**
      $d \leftarrow$ ComputeDistance(i.position, j.position)
      if $d < T$ **then**
        current_alerts.append(GenerateAlert(i, j, d))
      end if
    end for
    TransmitAlerts(current_alerts)
  end while

  Helper Functions:
  Preprocess(X)**:**
    return ApplyDataTransformations(X)

  GenerateAlert(object1, object2, distance):
    return {"object1": object1, "object2": object2, "distance": distance,
      "timestamp": CurrentTime()}

  TransmitAlerts(alerts):
  for each alert in alerts **do**
    BroadcastToNearbyVehicles(alert)
    NotifyDriver(alert)
  end for
end procedure

performance in deep learning and IoT, as well as AWS Greengrass's effectiveness in our system. The section has been divided into six parts: System Environment Setup for Deep Learning Experiments, Evaluation Metrics, Performance Evaluation of Deep Learning Architectures, IOT Performance Analysis, AWS Performance Analysis, and SafeTrax Integrated Vehicle Safety System: Prototype Design.

A. SYSTEM ENVIRONMENT SETUP FOR DEEP LEARNING EXPERIMENTS

Table 6 shows the hardware and software we have used to train and test our deep learning models. The table provides a summary of the equipment and programs used in our study. The most important part of our system is the NVIDIA GeForce RTX 3060 GPU with 12 GB of memory. Additionally, we have used TensorFlow with CUDA 12.6 to speed up deep learning training. Python 3.12 has been the main programming language in our code. We also used libraries like Scikit-learn, Keras, and Numpy to build models and manage data. The system runs on Linux, which provides a stable environment for running deep learning tasks and monitoring GPU performance.

TABLE 6. System environment setup for deep learning experiments.

Component	Details
GPU	NVIDIA GeForce RTX 3060 GDDR6 (12GB GPU VRAM)
CPU	AMD Ryzen 7 5700X 8 Core 16 Threads Processor
RAM	64GB DDR4 3200 BUS
Storage	Samsung 980 Pro 1TB PCIe 4.0 M.2 NVMe SSD
Operating System	Ubuntu 24.04.1 LTS

B. EVALUATION METRICS

To evaluate the performance of our deep learning models, we have used several popular evaluation metrics that are widely used for evaluation purposes in machine learning and deep learning. The metrics we have used include accuracy, precision, recall, and F1 score. A brief overview of these metrics is provided below:

- **Accuracy score:** The accuracy score is one of the most commonly used metrics in the fields of machine learning and deep learning. It represents the proportion of correctly predicted instances (both true positives and true negatives) out of the total instances. These total instances include true positives, true negatives, false positives, and false negatives. This metric provides a comprehensive assessment of the model's overall performance.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

- **Precision:** Precision value evaluates the accuracy of the model's positive predictions. The positive prediction means the ratio of true positive predictions to the total predicted positives (true positives and false positives). This metric provides an overview of how reliable the model is when classifying instances as positive.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall:** Recall can be denoted as the ratio of true positive predictions to the total actual positives. Generally, it only measures the model's ability to identify all relevant instances.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

TABLE 7. Performance of models on the test set.

Model	Accuracy (Test)	Precision (Test)	Recall (Test)	F1-Score (Test)
ResNet50	93.03%	93.76%	97.45%	95.57%
MobileNetV2	93.30%	94.79%	97.40%	96.08%
VGG-16	90.75%	95.78%	91.74%	93.72%

TABLE 8. Performance of models on the validation set.

Model	Accuracy (Validation)	Precision (Validation)	Recall (Validation)
ResNet50	92.60%	90.11%	91.30%
MobileNetV2	90.10%	90.21%	91.44%
VGG-16	92.10%	88.10%	88.40%

- **F1 score:** The F1 score is a metric that combines precision and recall and provides a performance result. It indicates how good a model is at predicting both positive and negative cases. It is especially useful when the classes are imbalanced,

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

C. PERFORMANCE EVALUATION OF DEEP LEARNING ARCHITECTURES

The performance of three deep learning architectures consisting of ResNet50, VGG-16, and MobileNetV2 has been evaluated. The results show that the models have effectively learned the underlying patterns and achieved satisfactory accuracy and loss reduction.

1) TRAINING DYNAMICS

During training, the models have improved accuracy with each epoch, effectively learning and optimizing their weights. Validation accuracy has remained stable with minimal overfitting. It indicates good performance on unseen data. Both training and validation losses have decreased. However, MobileNetV2 has exhibited significant fluctuations in validation accuracy and loss and formed a zigzag pattern. This suggests inconsistent training, possibly due to an incorrect learning rate, overfitting, or insufficient training data.

2) DETAILED MODEL PERFORMANCE METRICS

We have evaluated the performance of the modified architectures using a range of metrics, including accuracy, loss, precision, recall, and F1-score:

- **ResNet50:** Resnet50 has achieved an impressive accuracy of 93.03% on the test set and 92.6% on the validation set. It has demonstrated high precision (93.76% test, 90.11% validation) and recall (97.45% test, 91.3% validation), resulting in an F1-score of 95.57% for the test set. Figure 18 and 19 represent the accuracy and loss curves for the training and validation processes generated during the training of the ResNet50 model.
- **MobileNetV2:** MobileNetV2 have gained an accuracy of 93.3% (test) and 90.1% (validation). It exhibited precision values of 94.79% (test) and 90.21% (validation) alongside recall rates of 97.40% (test) and 91.44% (validation). Also, the F1-score is 96.08% (test). Figure 20 and 21 represent the accuracy and loss curves for the training and validation processes generated during the training of the MobileNetV2 model.
- **VGG-16:** VGG-16 has shown lower overall performance. It has achieved an accuracy of 90.75% (test) and 92.1% (validation). Its precision was 95.78% (test) and 88.1% (validation), with recall values of 91.74% (test) and 88.4% (validation). And its F1-score is 93.72% (test). Figure 22 and 23 depict the accuracy and loss curves for the training and validation processes generated during the training of the VGG-16 model.

3) CLASSIFICATION PERFORMANCE METRICS

In addition to accuracy, key metrics such as precision, recall, and F1-score have been calculated to evaluate classification performance. These metrics demonstrate the model's effectiveness across all classes. Tables 7 and 8 summarize results from the training and evaluation phases for the three Deep Learning models. Figure 24 depicts the confusion matrix for the ResNet50 model's test performance, while Figures 25 and 26 show the confusion matrices for the MobileNetV2 and VGG-16 models, respectively.

D. IoT PERFORMANCE ANALYSIS

In this section, we have presented the result analysis produced by our IoT system and found the outputs in our Raspberry Pi OS CLI. Figures 11, 12, and 13 represent the output logs from three different states: the no-collision state, the collision state, and the specific timestamp with the system closing.

1) NO COLLISION STATE

Each log entry has included a timestamp for tracking events. "INFO" entries have detailed operations, while "DEBUG" entries have provided troubleshooting information for specific camera frames. The collision probability has ranged from 0 to 1 which indicates risk levels. In Figure 11, a Collision Probability of 0.023 in Frame 1 suggests a low collision likelihood, with a Distance of 352.70 cm. It indicates a safe separation. An Alert Status of "OFF" means no collision threat has been detected, allowing for continued monitoring.

```
2024-09-27 15:00:03 [INFO]  - Frame 1 - Collision Probability: 0.023, Distance: 352.70cm,
Alert Status: OFF
2024-09-27 15:00:03 [DEBUG] - Frame 1 - No collision detected, continuing monitoring...
2024-09-27 15:00:04 [INFO]  - Frame 2 - Collision Probability: 0.031, Distance: 350.20cm,
Alert Status: OFF
2024-09-27 15:00:04 [DEBUG] - Frame 2 - No collision detected, continuing monitoring...
2024-09-27 15:00:05 [INFO]  - Frame 3 - Collision Probability: 0.028, Distance: 351.50cm,
Alert Status: OFF
2024-09-27 15:00:05 [DEBUG] - Frame 3 - No collision detected, continuing monitoring...
```

FIGURE 11. CLI output when no collision is detected.

2) COLLISION STATE

The log entry in Figure 12 has shown that, during Frame 845, the collision probability is 0.892, with a distance of 48.30 cm to the nearest object. The alert status is ON, activating the buzzer and LED indicators. This "DEBUG" message has confirmed that the frame data, collision probability, and alert status have been sent to AWS IoT Core for further analysis.

```
2024-09-27 15:00:30 [INFO]  - Frame 845 - Collision Probability: 0.892, Distance: 48.30cm,
Alert Status: ON
2024-09-27 15:00:30 [WARN]  - ALERT: Potential collision detected!
2024-09-27 15:00:30 [INFO]  - Activating Buzzer and LED...
2024-09-27 15:00:30 [DEBUG] - Publishing data to AWS IoT Core...
```

FIGURE 12. CLI output when collision is detected.

3) THE SPECIFIC TIMESTAMP WITH THE SYSTEM CLOSING

The log message in Figure 13 has indicated that the user has initiated the system shutdown process and provides a performance summary for the last 100 processed frames. The average collision probability is 0.098. It indicates a low risk of collision, with an average distance to detected objects of 315.62 cm. Each frame has taken an average of 0.032 seconds to process, using 47.8% of available RAM (Total RAM is 4GB). Since startup, 3600 frames have been processed, resulting in 5 collision alerts. The final entry confirms that the system has shut down safely.

```
2024-09-27 15:30:00 [INFO] - System Shutdown Initiated by User.
2024-09-27 15:30:00 [INFO] - Performance Summary (last 100 frames):
2024-09-27 15:30:00 [INFO] - Avg Collision Probability: 0.098
2024-09-27 15:30:00 [INFO] - Avg Distance: 315.62cm
2024-09-27 15:30:00 [INFO] - Avg Processing Time: 0.032s
2024-09-27 15:30:00 [INFO] - Avg CPU Usage: 34.9%
2024-09-27 15:30:00 [INFO] - Avg Memory Usage: 47.8%
2024-09-27 15:30:00 [INFO] - Total Frames Processed: 3600
2024-09-27 15:30:00 [INFO] - Total Alerts: 5
2024-09-27 15:30:00 [INFO] - Total Auto-braking Events: 2
2024-09-27 15:30:01 [INFO] - System Shutdown Complete.
```

FIGURE 13. CLI Output for the specific timestamp with the system closing.

E. AWS PERFORMANCE ANALYSIS

The performance of AWS Greengrass in our SafeTrax system has been assessed using several metrics. It includes latency, response time, data throughput, and resource utilization. A detailed analysis is given below. The outputs from the CLI have been structured in a tabular format. The commands have been executed on the Raspberry Pi using 'greengrass-cli'.

1) LATENCY AND RESPONSE TIME

Latency and response time are critical for real-time systems. In SafeTrax, IoT devices have sent data to the Greengrass Core for local processing, which forwards results to the AWS IoT Core. Key latency metrics like Round-Trip Time, Sensor Input-to-Action Time, and Cloud Processing Time have been logged. Greengrass Core has retrieved and filtered these logs for the last 10 minutes and displayed them in the CLI. The process for how AWS Greengrass retrieves these data using a CLI command from the user-end has been presented in figure 14. The output after running this command is given in table 9.

According to the table 9, An average RTT of 52 ms ensures efficient communication for real-time applications. The 35 ms sensor-to-action latency enables timely alerts, while the 20 ms cloud processing time indicates fast data handling.

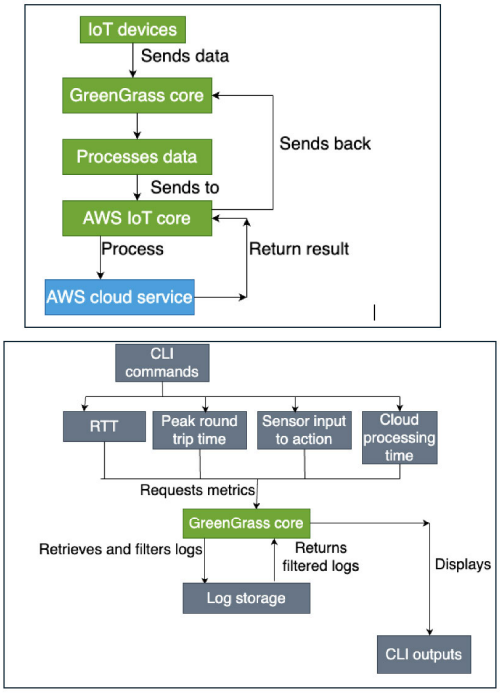


FIGURE 14. Diagram for retrieving latency and response time.

TABLE 9. Latency and response time for AWS Greengrass integration.

Metric	Value	Remarks
Round-Trip Time (RTT)	52 ms	Minimal delay; best for real-time alerts
Peak Round-Trip Time	89 ms	Recorded under high network congestion
Average Sensor Input-to-Action	35 ms	Fast response time from input to alert
Average Cloud Processing Time	20 ms	Efficient cloud processing latency

2) DATA THROUGHPUT

Data throughput is crucial for handling high-definition video and frequent sensor data. The camera has streamed

images to the Greengrass Core, where each frame is processed in real-time by the Pre-Collision Alert System. Performance metrics, like data throughput (MB/s) and frames processed per second (FPS), have been logged. The Greengrass Core has retrieved and displayed logs from the last 15 minutes. The process for how AWS Greengrass retrieves Data throughput using a CLI command from the user-end has been presented in figure 15. The output results are given in table 10.

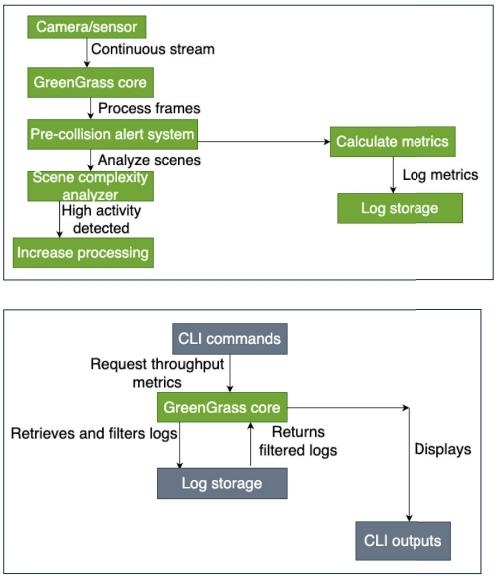


FIGURE 15. Diagram for retrieving data throughput.

TABLE 10. Performance metrics of the pre-collision alert system.

Metric	Value	Remarks
Average Data Throughput	2.5 MB/s	Efficient handling of continuous image streams
Peak Data Throughput	3.8 MB/s	During complex scenes with high activity
Average Frames Processed per Second	25 FPS	Real-time frame processing capability

An average throughput of 2.5 MB/s and peak of 3.8 MB/s show the system handles complex scenes well. Processing 25 FPS ensures real-time responsiveness for timely alerts and improved safety.

3) RESOURCE UTILIZATION

Resource utilization tracks AWS Greengrass overhead on the Raspberry Pi, including CPU, RAM, and bandwidth. The Greengrass Resource Monitor has logged average and peak values, which are processed and displayed in the CLI. The process for how AWS Greengrass retrieves this information using a CLI command from the user-end has been presented in figure 16. The output results are given in table 11. The system on AWS Greengrass efficiently uses Raspberry Pi resources, with an average CPU at 37.5% (peaking at 65%) and memory at 45.3% (peaking at 68%). Network bandwidth peaks at 500 KB/s during high-load tasks that show stable performance and effective management.

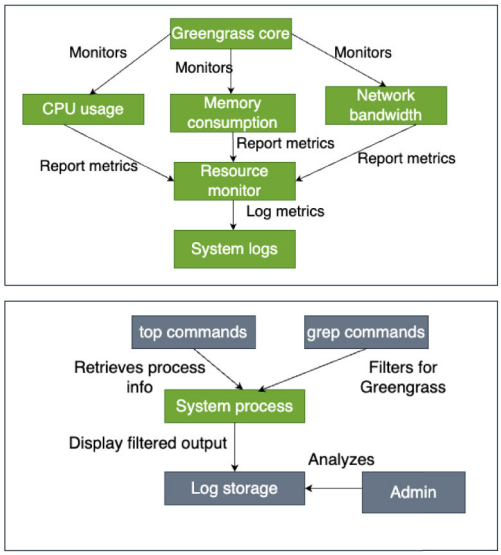


FIGURE 16. Diagram for retrieving resource utilization.

TABLE 11. System performance metrics for AWS Greengrass deployment.

Metric	Value	Remarks
Average CPU Usage	37.5%	Efficient CPU utilization; no throttling observed
Peak CPU Usage	65%	During complex multi-object detection scenarios
Average Memory Consumption (RAM)	45.3%	Memory usage within acceptable limits
Peak Memory Consumption (RAM)	68%	Short spikes during high processing load
Average Network Bandwidth Utilization	250 KB/s	Efficient use of network resources
Peak Network Bandwidth Utilization	500 KB/s	Observed during data-intensive operations

4) RELIABILITY AND FAULT TOLERANCE

The AWS Greengrass Core monitors the network, data integrity, and system stability. In case of a failure, it recovers, stores local data, and logs metrics like recovery success rate, recovery time, false alerts, and failover performance. Reliability metrics from the last 30 minutes have been retrieved and displayed in the CLI. The process for how AWS Greengrass handles reliability and fault Tolerance using a CLI command from the user-end has been presented in figure 17. The outputs are shown in table 12.

TABLE 12. AWS Greengrass system performance under network failure scenarios.

Metric	Outcome	Remarks
Successful Data Recovery	100%	No data loss during reconnection
System Recovery	10s	Recovered within 10 seconds
Number of False Alerts	0	No false alerts
Disconnection Handling	Smooth Failover	Continued local processing during downtime

The table shows that AWS Greengrass recovers from network failures within 10 seconds, with no data loss

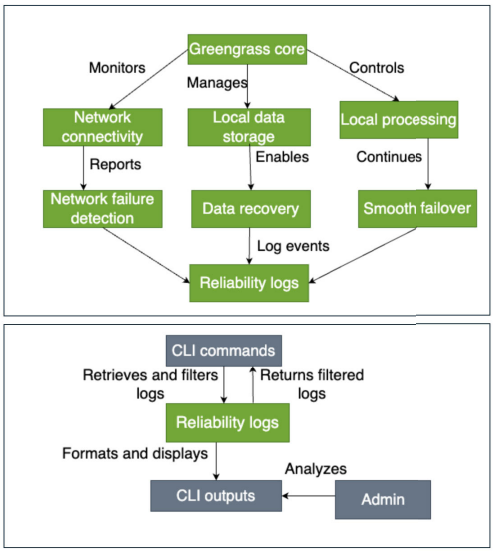


FIGURE 17. Diagram for retrieving reliability and fault tolerance.

or false alerts, indicating robust disconnection handling and smooth failover management.

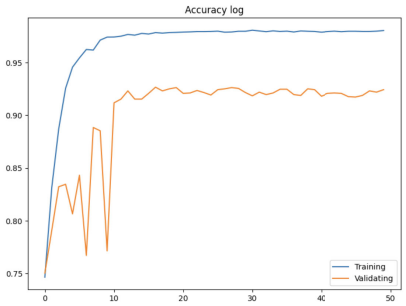


FIGURE 18. Training vs validation accuracy curves for ResNet50.

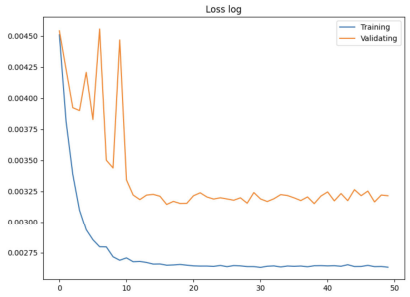


FIGURE 19. Training vs validation loss curves for ResNet50.

F. SAFETRAX INTEGRATED VEHICLE SAFETY SYSTEM: PROTOTYPE DESIGN

We have designed the SafeTrax integrated car prototype, which is shown in Figure 9. It combines advanced sensing and computing technologies to enhance vehicle safety. The system's core components include a car with a sonar sensor installed in the bumpers and front grill for obstacle

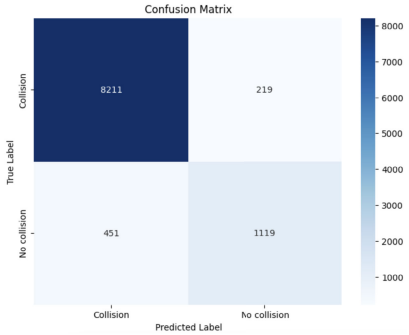


FIGURE 20. Training vs validation accuracy curves for MobileNetV2.

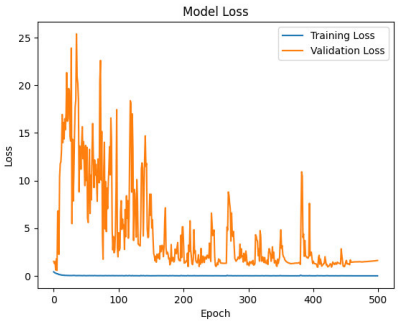


FIGURE 21. Training vs validation loss curves for MobileNetV2.

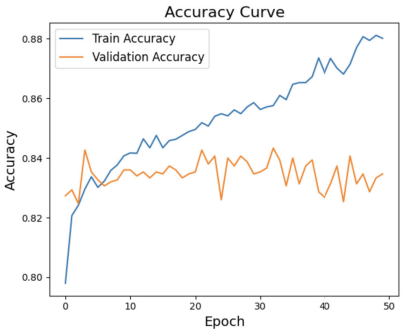


FIGURE 22. Training vs validation accuracy curves for VGG-16.

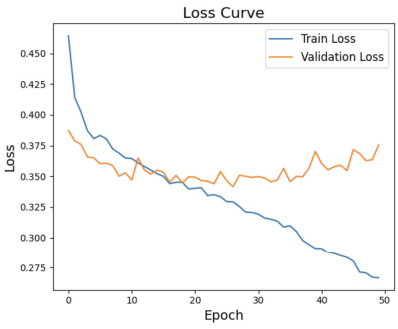


FIGURE 23. Training vs validation loss curves for VGG-16.

detection and a dash cam mounted on the windshield or front of the vehicle for visual monitoring. These sensors feed data to a Raspberry Pi, positioned discreetly within

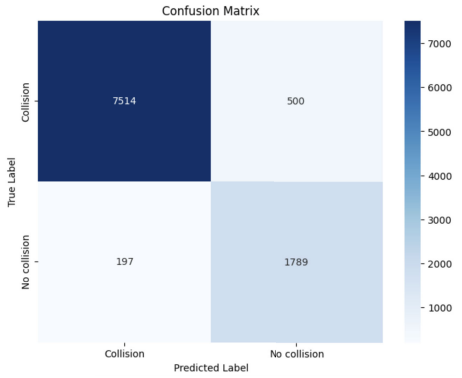


FIGURE 24. Confusion matrix for ResNet50.

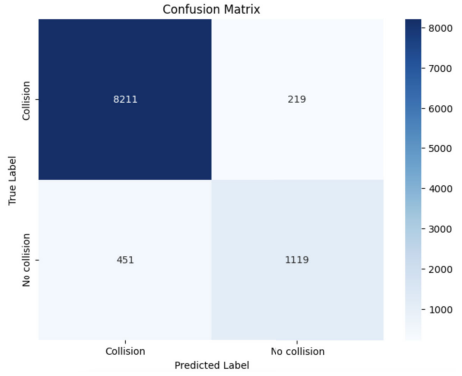


FIGURE 25. Confusion matrix for MobileNetV2.

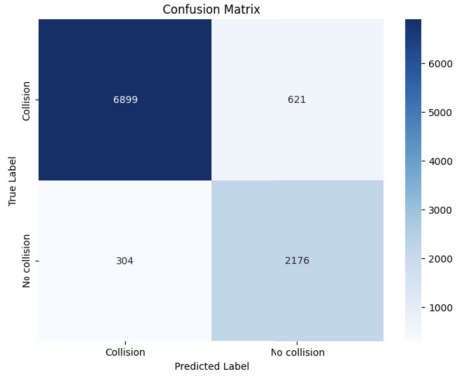


FIGURE 26. Confusion matrix for VGG-16.

the car, which serves as the central processing unit. The Raspberry Pi runs deep learning models and AWS Greengrass enabling both edge computing and cloud connectivity. This setup allows for real-time data processing and decision-making. Python is used as the primary programming language for the system. To alert the driver of potential hazards, a car buzzer is installed near the dashboard or steering wheel.

This is the initial prototype of the SafeTrax car design, which will be further developed in our future research. The prototype will require extensive testing and further development to ensure its effectiveness and compliance with automotive safety standards.

V. CONCLUSION AND FUTURE WORK

This section is divided into three parts consisting of our overall discussion, conclusion, and our future work planning.

A. DISCUSSION

In this research, we have combined IoT and deep learning technologies to create an effective accident prediction and alert system. We have improved cloud services for in-car sensors and cameras by using AWS IoT Greengrass. This allows real-time data collection and processing. As a result, the system's speed, responsiveness, and ability to grow have increased, making it easier to adapt to real driving conditions. To optimize performance, we have used and fine-tuned several pre-trained deep-learning models in our SafeTrax.

This enables the system to handle complex situations, including multi-dimensional data. We have also added an alarm system to enhance safety. It provides drivers with timely warnings about possible accidents. The system has shown flexibility and it can work well in both urban and rural areas, reducing the chances of accidents in different driving conditions.

B. CONCLUSION

We have built a smart system that can predict car crashes and collisions to help make roads safer. Our system uses some available sensors which can be easily found everywhere, deep learning models, and AWS Greengrass to predict the probable collision, access cloud services, and give drivers warnings in real time if an accident might happen. The system is simple and low-cost, and it works in different places. In our tests, the system has worked very well. Two of our models, ResNet50 and MobileNetV2, have been very accurate and predicted crashes correctly more than 93% of the time. Another model, VGG-16, also did a good job, with over 90% accuracy. We have also designed a prototype car using this SafeTrax methodology. It has a car sonar sensor, a dashcam, a car buzzer, and a Raspberry Pi. Images from each video frame can be captured using car dashcams. The Raspberry Pi runs crash-predicting models and connects to AWS Greengrass to process these image data quickly. If there is a danger, a buzzer sounds to warn the driver. In a nutshell, SafeTrax can help make roads safer and reduce car accidents. This smart and affordable system can be integrated into any type of vehicle. In the future, we will keep improving the system to make sure it meets all safety rules.

C. FUTURE WORK

To ensure road safety, we need to focus on enhancing obstacle detection accuracy, enabling flexible real-time data transmission for transportation systems, and ensuring scalability through the use of appropriate IoT sensors in future developments. To do this, we plan to add more convolutional layers and attention mechanisms. Since weather changes are a major concern, we will create a weather-specific detection module to help the system adapt to real-time weather

conditions and provide reliable predictions. Additionally, we will incorporate more advanced sensors.

We will also integrate map-based applications. This will give real-time updates on traffic patterns and promote safer decision-making. Another key goal is to address sensor quality constraints. We will focus on improving the quality of cameras and data processing. This will help ensure more accurate predictions, even in challenging situations, like low-quality image captures. Scalability challenges in remote areas with poor infrastructure can affect SafeTrax's performance. In the future, we will work on improving the software to make it compatible with different networks. Lastly, to reduce processing delays in high-traffic situations, we will optimize AWS IoT Greengrass further. We will also increase the use of edge computing. This will help ensure faster and more responsive performance in various situations.

REFERENCES

[1] R. Talbot, A. Filtness, and A. Morris, "Proposing a framework for evidence-based road safety policy-making: Connecting crash causation, countermeasures and policy," *Accident Anal. Prevention*, vol. 195, Feb. 2024, Art. no. 107409.

[2] P. Vrtal, K. Kocián, J. Nováč ek, Z. Svatý, and T. Kohout, "A unified approach for risk assessment of road crash barriers using Bayesian statistics," *Appl. Sci.*, vol. 13, no. 3, p. 1270, Jan. 2023.

[3] R. Luke, "Current and future trends in driver behaviour and traffic safety scholarship: An African research agenda," *Int. J. Environ. Res. Public Health*, vol. 20, no. 5, p. 4290, Feb. 2023.

[4] R. Ren, H. Li, T. Han, C. Tian, C. Zhang, J. Zhang, R. W. Proctor, Y. Chen, and Y. Feng, "Vehicle crash simulations for safety: Introduction of connected and automated vehicles on the roadways," *Accident Anal. Prevention*, vol. 186, Jun. 2023, Art. no. 107021.

[5] V. Morozov, A. I. Petrov, V. Shepelev, and M. Balfaqih, "Ideology of urban road transport chaos and accident risk management for sustainable transport systems," *Sustainability*, vol. 16, no. 6, p. 2596, Mar. 2024.

[6] N. Sharma and R. D. Garg, "Real-time IoT-based connected vehicle infrastructure for intelligent transportation safety," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 8, pp. 8339–8347, Aug. 2023.

[7] L. Muthugama, H. Xie, E. Tanin, and S. Karunasekera, "Real-time road safety optimization through network-level data management," *GeoInformatica*, vol. 27, no. 3, pp. 491–523, Jul. 2023.

[8] Z. Ye, Y. Wei, S. Yang, P. Li, F. Yang, B. Yang, and L. Wang, "IoT-enhanced smart road infrastructure systems for comprehensive real-time monitoring," *Internet Things Cyber-Phys. Syst.*, vol. 4, pp. 235–249, Aug. 2024.

[9] A. A. Musa, S. I. Malami, F. Alanazi, W. Ounaies, M. Alshammari, and S. I. Haruna, "Sustainable traffic management for smart cities using Internet-of-Things-Oriented intelligent transportation systems (ITS): Challenges and recommendations," *Sustainability*, vol. 15, no. 13, p. 9859, Jun. 2023.

[10] Y. Berhanu, E. Alemayehu, and D. Schröder, "Examining car accident prediction techniques and road traffic congestion: A comparative analysis of road safety and prevention of world challenges in low-income and high-income countries," *J. Adv. Transp.*, vol. 2023, Jul. 2023, Art. no. 6643412.

[11] F. A. Arnob, A. Fuad, A. T. Nizam, S. Barua, A. A. Choudhury, and M. Islam, "An intelligent traffic system for detecting lane based rule violation," in *Proc. Int. Conf. Adv. Emerg. Comput. Technol. (AECT)*, Feb. 2020, pp. 1–6.

[12] A. Kolhe, P. Potabatti, K. Khandagale, A. Jadhav, and J. Kulkarni, "Road companion-vehicle collision predication system," vol. 6, no. 2, pp. 1–20, Apr. 2018.

[13] T. Liu, A. Jiang, J. Zhou, M. Li, and H. K. Kwan, "GraphSAGE-based dynamic spatial–temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 10, pp. 11210–11224, Oct. 2023.

[14] N. Hu, D. Zhang, K. Xie, W. Liang, K. C. Li, and A. Y. Zomaya, "Dynamic multi-scale spatial–temporal graph convolutional network for traffic flow prediction," *Future Generat. Comput. Syst.*, vol. 158, pp. 323–332, Sep. 2024.

[15] A. Razi, X. Chen, H. Li, H. Wang, B. Russo, Y. Chen, and H. Yu, "Deep learning serves traffic safety analysis: A forward-looking review," *IET Intell. Transp. Syst.*, vol. 17, no. 1, pp. 22–71, Jan. 2023.

[16] C. Panda, A. K. Mishra, A. K. Dash, and H. Nawab, "Predicting and explaining severity of road accident using artificial intelligence techniques, SHAP and feature analysis," *Int. J. Crashworthiness*, vol. 28, no. 2, pp. 186–201, Mar. 2023.

[17] N. Shatnawi, A. A. Al-Omari, and S. Alkhateeb, "Prediction of risk factors influencing severity level of traffic accidents using artificial intelligence," *Int. Rev. Civil Eng. (IRECE)*, vol. 14, no. 1, p. 1, Jan. 2023.

[18] Y. Ali, F. Hussain, and M. M. Haque, "Advances, challenges, and future research needs in machine learning-based crash prediction models: A systematic review," *Accident Anal. Prevention*, vol. 194, Jan. 2024, Art. no. 107378.

[19] X. Xiong, L. Chen, and J. Liang, "A new framework of vehicle collision prediction by combining SVM and HMM," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 699–710, Mar. 2018.

[20] A. V. Malawade, S.-Y. Yu, B. Hsu, D. Muthirayan, P. P. Khargonekar, and M. A. A. Faruque, "Spatiotemporal scene-graph embedding for autonomous vehicle collision prediction," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9379–9388, Jun. 2022.

[21] N. Venkateswaran, W. J. Hans, and N. Padmapriya, "Deep learning based robust forward collision warning system with range prediction," *Multimedia Tools Appl.*, vol. 80, no. 14, pp. 20849–20867, Jun. 2021.

[22] L. Zhang, W. Xiao, Z. Zhang, and D. Meng, "Surrounding vehicles motion prediction for risk assessment and motion planning of autonomous vehicle in highway scenarios," *IEEE Access*, vol. 8, pp. 209356–209376, 2020.

[23] J. A. Onesimu, A. Kadam, K. M. Sagayam, and A. A. Elngar, "Internet of Things based intelligent accident avoidance system for adverse weather and road conditions," *J. Reliable Intell. Environ.*, vol. 7, no. 4, pp. 299–313, Dec. 2021.

[24] L. Guo, Y. Jia, X. Hu, and F. Dong, "Forwarding collision assessment with the localization information using the machine learning method," *J. Adv. Transp.*, vol. 2022, May 2022, Art. no. 9530793.

[25] C. Dong, C. Shao, J. Li, and Z. Xiong, "An improved deep learning model for traffic crash prediction," *J. Adv. Transp.*, vol. 2018, Dec. 2018, Art. no. 3869106.

[26] G. K. Sahoo, S. A. Patro, P. K. Pradhan, S. K. Das, and P. Singh, "An IoT-based intimation and path tracing of a vehicle involved in road traffic crashes," in *Proc. IEEE-HYDCON*, Apr. 2020, pp. 1–5.

[27] L. Yu, B. Du, X. Hu, L. Sun, L. Han, and W. Lv, "Deep spatio-temporal graph convolutional network for traffic accident prediction," *Neurocomputing*, vol. 423, pp. 135–147, Jan. 2021.

[28] M. T. Kashifi, M. Al-Turki, and A. W. Sharify, "Deep hybrid learning framework for spatiotemporal crash prediction using big traffic data," *Int. J. Transp. Sci. Technol.*, vol. 12, no. 3, pp. 793–808, 2023.

[29] D. Thapa, S. Mishra, N. R. Velaga, and G. R. Patil, "Advancing proactive crash prediction: A discretized duration approach for predicting crashes and severity," *Accident Anal. Prevention*, vol. 195, Feb. 2024, Art. no. 107407.

[30] M. M. Rathore, A. Paul, S. Rho, M. Khan, S. Vimal, and S. A. Shah, "Smart traffic control: Identifying driving-violations using fog devices with vehicular cameras in smart cities," *Sustain. Cities Soc.*, vol. 71, Aug. 2021, Art. no. 102986.

[31] A. Pramanik, S. Sarkar, and J. Maiti, "A real-time video surveillance system for traffic pre-events detection," *Accident Anal. Prevention*, vol. 154, May 2021, Art. no. 106019.

[32] M. M. Islam, A. E. M. Ridwan, M. M. Mary, M. F. Siam, S. A. Mumu, and S. Rana, "Design and implementation of a smart bike accident detection system," in *Proc. IEEE Region 10 Symp. (TENSYMP)*, Jun. 2020, pp. 386–389.

[33] K. Dong, C. Zhou, Y. Ruan, and Y. Li, "MobileNetV2 model for image classification," in *Proc. 2nd Int. Conf. Inf. Technol. Comput. Appl. (ITCA)*, Dec. 2020, pp. 476–480.

[34] S. Mascarenhas and M. Agarwal, "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for image classification," in *Proc. Int. Conf. Disruptive Technol. Multi-Disciplinary Res. Appl. (CENTCON)*, vol. 1, Nov. 2021, pp. 96–99.

[35] D. D. Hema and T. R. Jaison, "Efficient collision risk prediction model for autonomous vehicle using novel optimized LSTM based deep learning framework," *Int. J. Intell. Transp. Syst. Res.*, vol. 22, pp. 352–362, May 2024.
[36] B. Alafi and S. Moradi, "Multi-class real-time crash risk forecasting using convolutional neural network: Istanbul case study," 2024, *arXiv:2402.06707*.
[37] S. Lee, B. Song, and J. Shin, "Collision prediction in an integrated framework of scenario-based and data-driven approaches," *IEEE Access*, vol. 12, pp. 55234–55247, 2024.



MD. FAHIM BIN AMIN is a highly skilled Machine Learning Engineer and Software Engineer with a robust background in building AI-powered systems and software solutions. With a degree in Computer Science and Engineering, majoring in Data Science from United International University, he has worked on innovative projects like vehicle number plate detection using YOLOv8 and Alzheimer's disease detection using 3D MRI images.

His technical expertise spans a wide range of technologies, including TensorFlow, PyTorch, React, Node.js, and Docker, making him proficient in both software development and machine learning frameworks. His passion for innovation, combined with his expertise in both machine learning and software engineering, positions him as a forward-thinking engineer dedicated to driving technological progress.



ISRAT JAHAN KHAN is a Data Science major at United International University, with a strong passion for Machine Learning (ML) and Deep Learning (DL). Her academic research focuses on computer vision and the development of hybrid ML algorithms, aiming to offer innovative solutions in these areas.

As a Machine Learning researcher, she has gained valuable experience, further fueling her passion for academia. Committed to advancing the field of data science, she strives to contribute to technological progress through continuous learning, innovation, and impactful research in education.



FARIA AKTER is a passionate researcher with a BSc. in Computer Science and Engineering, majoring in Data Science, from United International University. In her academic journey she has gained hands-on research experience as an undergraduate research assistant, where her work focused on advancing Machine Learning, Natural Language Processing (NLP), and Human-Computer Interaction (HCI). Through her projects, she has consistently applied her expertise to address real-life rising challenges.

Currently, her research interests include Image Processing, Large Language Models (LLMs), and Deep Learning. She aspires to broaden her contribution further in the field of Data Science by solving real-life problems, including different scenarios from several aspects.



ADIB AHMED received the degree in computer science and engineering, specializing in software engineering from United International University (UIU), he focuses on creating practical and impactful solutions that merge advanced technologies with real-world applications.

He is dedicated to advancing research and innovation in the fields of machine learning, image processing, and the Internet of Things (IoT). With a profound interest in artificial intelligence and its transformative capabilities, he explores interdisciplinary approaches to address complex problems. His work in AI and the IoT is complemented by a strong emphasis on data science, as he employs data-driven methodologies to devise innovative solutions. Driven by a passion for software development, he aims to contribute to cutting-edge technological advancements and promote significant progress within the technology sector.



MD. MOTAHARUL ISLAM (Member, IEEE) received the Ph.D. degree in computer engineering from Kyung Hee University, South Korea, in 2013.

He has been a Professor with the Department of Computer Science and Engineering, United International University (UIU), Dhaka, Bangladesh, since 2020. Before joining UIU, he worked at many other universities at home and abroad, such as the Islamic University of Madinah, Saudi Arabia, the Islamic University of Technology, Brac University, and the University Grants Commission of Bangladesh. He has published around 100 articles more than the last ten years. His research interests include the smart Internet of Things, IP-based wireless sensor networks (IP-WSN), WSN virtualization, cloud computing, and green computing.