I.J. Information Technology and Computer Science, **2026, 2, 183-201**
Published Online on April 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2026.02.11

Accident Detection and Estimation of Vehicle Speed and Count by Type in Road CCTV Images Using Machine Vision

I. Kadek Rai Pramana
Department of Information Technology, Faculty of Engineering, Udayana University, Badung, Indonesia
E-mail: rai.pramana46@gmail.com
ORCID iD: https://orcid.org/0009-0002-2357-3330
*Corresponding author

I. Putu Agung Bayupati
Department of Information Technology, Faculty of Engineering, Udayana University, Badung, Indonesia
E-mail: bayupati@unud.ac.id
ORCID iD: https://orcid.org/0000-0003-0638-0756

Gusti Made Arya Sasmita
Department of Information Technology, Faculty of Engineering, Udayana University, Badung, Indonesia
E-mail: aryasasmita@unud.ac.id
ORCID iD: https://orcid.org/0000-0001-7242-8039

Ngoc Le
Faculty of Information Technology, Swinburne Vietnam – FPT University, Hanoi, Vietnam
E-mail: ngocla2@fe.edu.vn
ORCID iD: https://orcid.org/0000-0003-3515-5443

Received: 21 November 2025; Revised: 01 January 2026; Accepted: 13 February 2026; Published: 08 April 2026

Abstract: This study presents an integrated traffic monitoring system for accident detection, vehicle counting by type, and vehicle speed estimation using roadside Closed-Circuit Television (CCTV) footage and machine vision based on the YOLOv11 architecture. The proposed methodology comprises data collection from heterogeneous sources, data preprocessing and augmentation, model fine-tuning on a custom Vehicle–Accident dataset, system deployment through a web-based application, and real-world evaluation. The YOLOv11 models were optimized to detect multiple vehicle categories and clearly defined accident classes under real traffic conditions. Experimental results indicate that the YOLOv11 Large (l) model achieves superior detection performance, with 81.8% precision, 75.8% recall, 82.1% mAP50, and 53.3% mAP50–95. Real-world testing further confirms its effectiveness, yielding an object detection accuracy of 99.24% and low speed estimation errors, with a Mean Absolute Percentage Error (MAPE) of 3.56% for video-based evaluation and 5.54% for real-time evaluation. In contrast, the YOLOv11 Nano (n) model offers faster inference and lower computational requirements but exhibits reduced robustness in complex accident scenarios. The trained models are deployed in an interactive web application supporting image, video, and real-time inputs, enabling practical traffic monitoring and decision support. Overall, the YOLOv11l-Vehicle-Accident model is identified as the most suitable configuration for accuracy-critical traffic management systems, while Nano variants are better suited for resource-constrained deployments.

Index Terms: Detection, Accident, Vehicle, Machine Vision, YOLOv11.

1. Introduction

The advancement of digital technologies has significantly contributed to improving traffic safety and efficiency, particularly in urban areas such as Denpasar City, Bali, Indonesia. Among the emerging technologies, machine vision—an artificial intelligence-based approach to processing visual information—has gained attention due to its ability to

This work is open access and licensed under the Creative Commons CC BY 4.0 License.

automatically analyze images and video streams using specific algorithms [1]. Techniques such as object detection, image segmentation, and pattern recognition have enabled the deployment of machine vision in various traffic-related applications, including real-time monitoring and analysis [1, 2]. Prior studies have demonstrated the effectiveness of YOLO (You Only Look Once), one of the leading object detection algorithms, across diverse visual domains, including Balinese manuscript character recognition, satellite-based lightning whistler detection, and vehicle license plate recognition—underscoring its adaptability to complex image conditions [3–5].

In traffic environments, machine vision can be utilized to detect and classify vehicles, estimate their speed, measure traffic volume, and identify traffic accidents with high accuracy [1]. This is achieved through the processing of visual data captured by Closed-Circuit Television (CCTV) cameras using YOLO-based object detection algorithms. The real-time capability of this technology not only enhances traffic surveillance but also supports further analysis for decision-making, such as identifying traffic patterns and understanding the causes of accidents. Previous studies have applied YOLO-based algorithms to traffic tasks such as vehicle detection, accident identification, and speed estimation, though most focus on only one or two functions [6–15].

This study introduces a more comprehensive approach using YOLOv11 to integrate detection, classification, tracking, and speed estimation, offering higher accuracy and efficiency for analyzing complex CCTV footage. Furthermore, this research focuses on the implementation of YOLOv11 for detecting traffic accidents using roadside CCTV footage, as well as its application in estimating both the speed and number of vehicles based on their types. The study also evaluates the model's training performance across various combinations of vehicle and accident datasets and assesses its effectiveness through testing within a web-based dashboard application. By leveraging a comprehensive dataset that includes 22 object classes—comprising four types of vehicles and eighteen accident categories—the proposed system enables more extensive analysis and deeper insights. Deployed in the urban context of Denpasar, the system is designed to support real-time monitoring, facilitate rapid response to traffic incidents, and ultimately enhance road safety and traffic management.

2. Related Works

Several prior studies have addressed traffic-related tasks such as accident detection, vehicle speed estimation, and vehicle counting, laying the foundation for further research in this field. These studies serve as relevant references for the development of this work.

Hutauruk, Matulatan, and Hayaty [6] implemented YOLOv2 for detecting vehicles—motorcycles, cars, trucks, and buses—within an Android-based application. The system, tested on a Xiaomi Redmi 4X using 200 images (4 classes) at a resolution of 768×432 pixels, achieved an accuracy of 83.3%.

Rofii et al. [7] applied YOLOv4 for vehicle detection and unique identification in traffic videos, utilizing 520 images (4 classes). The model achieved 83% accuracy under daylight conditions with optimal camera placement (6-meter height, 50° angle).

Zulfikri et al. [8] employed the Haar Cascade Classifier for vehicle detection and speed estimation by measuring time differentials within a Region of Interest (ROI). Using 1,600 images (1 class), their model attained a detection accuracy of 95.87% and a Mean Absolute Percentage Error (MAPE) of 1.73% for speed estimation.

Mulyana and Rofik [9] utilized YOLOv5 for vehicle type classification from traffic images and videos. With a dataset of 1,332 images (9 classes), the model reached 89.1% accuracy.

Rezky et al. [10] implemented a combination of YOLOv8 and MobileNetv2 for accident detection in traffic videos. The model, trained on 1,008 images (6 classes), achieved a detection accuracy of 89% in real-world conditions.

Kurniawan, Bayupati, and Wibawa [11] developed an automated vehicle counting system using Background Subtraction and MOG2, coupled with classification based on vehicle perimeter within the ROI. Tested on three videos (3 classes), the system achieved 92.19% detection accuracy, though it encountered challenges with small, dark-colored motorcycles blending into the road background.

Kurniawan et al. [12] proposed a model integrating YOLACT with the K-Nearest Neighbors (KNN) algorithm for vehicle detection in traffic videos. Using a dataset of 400 images (10 classes), their model achieved an accuracy of 91.67%.

Lin, Jeng, and Lioa [13] designed a real-time traffic monitoring system combining virtual detection zones, Gaussian Mixture Model (GMM), and YOLOv4 to estimate vehicle count and speed. With 15,951 images (6 classes), the model reported detection accuracy ranging from 98% to 99.5%, and a speed estimation MAPE of 7.6%.

Zhang et al. [14] enhanced YOLOv5 for improved detection of small and occluded vehicles by integrating CIoU loss and Flip-Mosaic data augmentation. Using 2,844 images (8 classes), the model achieved an mAP50 of 90.5% and a real-world detection accuracy of 76.22%.

Chen, Cao, and Wang [15] modified YOLOv5 with an Adaptive Clipping algorithm for detecting small vehicles in high-resolution UAV imagery. Their model, trained on 38,398 images (1 class), achieved 91.9% precision, 82.5% recall, 89.6% mAP50, and 62.4% mAP50–95, with a real-world detection accuracy of 95.1%.

These studies demonstrate a variety of techniques with strong performance in detection, classification, and speed estimation; however, most are limited to specific tasks. In contrast, the present study proposes an integrated system for accident detection, vehicle classification, tracking, and speed estimation using YOLOv11 and a dataset comprising 22 classes (4 vehicle types and 18 accident categories), offering a more robust and comprehensive solution for traffic surveillance applications.

3. Methodology

The proposed methodology extends beyond mere model adoption by incorporating systematic fine-tuning of YOLOv11 on a domain-specific dataset, optimized training configurations, and end-to-end system integration for real-world traffic analysis. The system is designed to support accident detection, vehicle detection, vehicle counting, and speed estimation from roadside CCTV footage. The methodological framework consists of six main stages: data collection, data preprocessing, data augmentation, model fine-tuning and training, application development, and real-world evaluation. An overview of the workflow is illustrated in Fig.1.

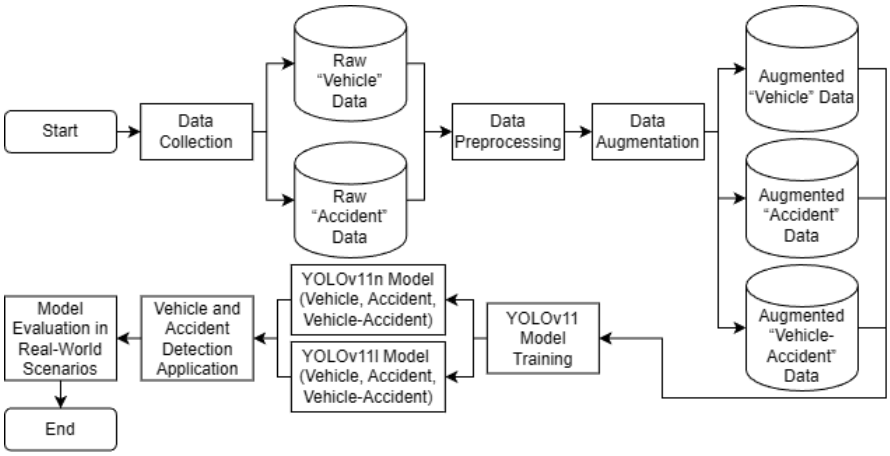


Fig.1. System overview flow

The study begins with the collection of traffic accident and vehicle imagery from both online and offline sources. Public datasets were obtained from Roboflow, Dataset Ninja, YouTube, and Gemini AI, while additional real-world data were collected from the Denpasar City Transportation Department, Bali, Indonesia. This combination enables the construction of a heterogeneous dataset that reflects diverse traffic conditions, camera perspectives, and scene complexities.

Data preprocessing was performed using Roboflow to ensure annotation consistency and data quality. Irrelevant or low-quality frames were filtered, and objects were annotated according to vehicle categories and accident-related classes. This stage ensures compatibility with the YOLOv11 detection framework and supports stable convergence during training.

To improve generalization and mitigate class imbalance, data augmentation techniques were applied to the training subset. These transformations increase variability in object scale, orientation, and illumination, thereby strengthening model robustness under real-world conditions.

The core methodological contribution lies in the fine-tuning of the YOLOv11 model on a custom traffic dataset. Rather than relying on pretrained weights alone, the model was retrained using task-specific data with optimized hyperparameter configurations, including learning rate, batch size, and training epochs, to adapt the detector to accident-prone and dense traffic scenes. Multiple dataset configurations and model scales were evaluated to analyze performance trade-offs between detection accuracy and computational efficiency.

Following training, the fine-tuned models were integrated into a web-based application developed using Streamlit and supporting libraries. This integration enables real-time, video-based, and image-based inference, with outputs presented as annotated frames and interactive visualizations. The deployment stage demonstrates the practical applicability of the proposed methodology beyond offline benchmarking.

Finally, model evaluation was conducted using previously unseen real-world data to assess detection accuracy, robustness, and speed estimation performance under operational conditions. This evaluation validates the effectiveness of the fine-tuned models and the proposed system pipeline for practical traffic monitoring applications.

3.1. Data Collection

This study employed a heterogeneous dataset composed of publicly available datasets, real-world CCTV recordings, and AI-generated images to capture diverse vehicle types, accident scenarios, and environmental conditions. Public data were sourced from Roboflow's *Real Time Accident Detection Computer Vision Project* [16], Dataset Ninja's VisDrone2019-DET [17], and Vehicle Dataset for YOLO [18], as well as traffic videos from the Demak Regency Transportation Department. High-resolution CCTV footage from Denpasar intersections was obtained directly from local authorities and reserved exclusively for real-world evaluation.

AI-generated accident images produced using Gemini AI were incorporated to address the scarcity and class imbalance of real accident data. Consistent with prior studies on synthetic datasets in autonomous driving, these images were used solely for training augmentation rather than evaluation, serving as a complementary source to enhance scenario diversity and model robustness [19]. Similar findings have demonstrated that AI-generated data can improve detection

performance and generalization when combined with real data, particularly for underrepresented classes [20].

To mitigate potential bias from mixed data sources, all datasets were standardized to a unified annotation schema, visually validated for scene consistency, and segregated to prevent data leakage. Synthetic images were excluded from the test set, ensuring that all reported results reflect performance under real-world conditions. This strategy balances data diversity and validity while supporting reliable evaluation of the proposed models.

3.2. Data Preprocessing

Data preprocessing was conducted using the Roboflow platform to ensure label consistency, data quality, and clear semantic separation between object and accident classes. The process included standardizing class names, removing irrelevant classes, null images, and duplicates, as well as extracting CCTV video frames at one frame per ten seconds followed by duplicate filtering prior to annotation.

In this study, an accident is operationally defined as a traffic event involving direct physical contact or collision between a vehicle and another vehicle, a person, or a static object. Accident labeling was guided by observable visual cues, including vehicle deformation, abnormal orientation, visible collision points, debris, or abrupt changes in relative position across consecutive frames [10]. Situations in which vehicles were merely adjacent or passing without contact or abnormal motion were explicitly excluded from accident annotations.

To ensure annotation consistency and reliability, all labels were generated through iterative discussions and consensus among the authors, based on a shared annotation guideline derived from the above definition. Ambiguous cases were jointly reviewed and resolved prior to final labeling, ensuring uniform interpretation across datasets. Accident classes were designed to encode both the vehicle type and the collision counterpart (vehicle, object, or person), minimizing semantic ambiguity. The complete list of object and accident classes is presented in Table 1.

Table 1. Class list. This table lists object and accident classes used in the dataset, labeled by vehicle type or accident combination

Class Name	Description
“bus”	Bus vehicle
“bus_bus_accident”	Accident involving a bus and another bus
“bus_object_accident”	Accident involving a bus and an object (e.g., road barrier, pole, tree)
“bus_person_accident”	Accident involving a bus and a person
“bus_truck_accident”	Accident involving a bus and a truck
“car”	Car vehicle
“car_bus_accident”	Accident involving a car and a bus
“car_car_accident”	Accident involving a car and another car
“car_motorcycle_accident”	Accident involving a car and a motorcycle
“car_object_accident”	Accident involving a car and an object
“car_person_accident”	Accident involving a car and a person
“car_truck_accident”	Accident involving a car and a truck
“motorcycle”	Motorcycle vehicle
“motorcycle_bus_accident”	Accident involving a motorcycle and a bus
“motorcycle_motorcycle_accident”	Accident involving two motorcycles
“motorcycle_object_accident”	Accident involving a motorcycle and an object
“motorcycle_person_accident”	Accident involving a motorcycle and a person
“motorcycle_truck_accident”	Accident involving a motorcycle and a truck
“truck”	Truck vehicle
“truck_object_accident”	Accident involving a truck and an object
“truck_person_accident”	Accident involving a truck and a person
“truck_truck_accident”	Accident involving two trucks

After preprocessing, the datasets were divided into training, validation, and testing subsets. Additional filtering was applied in certain cases, such as excluding frames where objects were excessively distant or visually ambiguous, to maintain annotation reliability and support robust model learning.

3.3. Data Augmentation

Data augmentation was applied to expand and diversify the training dataset while preserving label integrity, thereby improving model robustness and generalization to unseen conditions. The augmentation process was performed using Roboflow, incorporating preprocessing steps such as automatic image orientation based on metadata, resizing to ensure uniform dimensions, and grayscale conversion to reduce computational complexity while retaining essential visual features, including edges and object shapes.

Augmentation techniques included horizontal flipping, cropping, rotation, shearing, and adjustments to brightness

and exposure. These transformations were applied randomly, resulting in approximately a twofold increase in the vehicle dataset and a fourfold increase in the accident dataset. For close-range images—such as those from the "Vehicle Dataset for YOLO" and selected accident datasets—additional blur and noise augmentations were introduced to enhance generalization and mitigate overfitting.

Following augmentation, the datasets were organized into three categories: Vehicle, Accident, and a combined Vehicle-Accident dataset. These augmented datasets constitute the final training data used in subsequent model training and evaluation stages.

3.4. Model Training

Model training in this study employed two architectures from the YOLOv11 family, namely YOLOv11 Large (l) and YOLOv11 Nano (n), which differ primarily in network depth, parameter count, and representational capacity. The Large variant is designed for higher accuracy in complex scenes, whereas the Nano variant prioritizes computational efficiency for deployment in resource-constrained environments.

All primary comparative experiments between YOLOv11l and YOLOv11n were conducted using the same "Vehicle-Accident" dataset, identical preprocessing pipelines, and consistent hyperparameter configurations to ensure fairness and reproducibility. Training was performed locally using the Ultralytics framework with Torch-based GPU acceleration, as cloud-based platforms were found to be impractical due to runtime and connectivity constraints.

An additional exploratory experiment involving background-removed images was conducted exclusively for the Nano architecture to investigate whether reducing contextual complexity could compensate for its limited model capacity. However, this configuration resulted in substantial performance degradation during both training and real-world evaluation. The findings indicate that background context plays a critical role in accident and vehicle detection, particularly for tasks involving spatial interactions, occlusions, and scene-level cues. Removing background information disrupted spatial coherence and weakened feature associations necessary for reliable detection.

Given its consistently inferior performance and the lack of practical benefits, the background-removed configuration was excluded from the core comparative analysis between YOLOv11l and YOLOv11n. Consequently, all reported architectural comparisons are based solely on models trained with full-scene images, ensuring a fair and consistent evaluation. The background-removal experiment is therefore treated as an exploratory study rather than a primary methodological component.

Each finalized model was evaluated using standard detection metrics, including precision, recall, mAP50, and mAP50–95. This training strategy enables a clear and unbiased assessment of architectural trade-offs between accuracy and efficiency, while maintaining relevance to real-world traffic surveillance applications.

3.4.1. YOLOv11 Model

YOLOv11 is the latest advancement in the YOLO object detection series, offering fast and accurate identification and classification of objects. Building upon the strengths of its predecessors, it integrates several architectural innovations to enhance performance and flexibility. Designed to address a wide range of computer vision tasks—such as object detection, segmentation, pose estimation, tracking, and classification—YOLOv11 demonstrates high reliability across diverse applications, including traffic surveillance and medical image analysis. Its core advantages lie in computational efficiency and high detection accuracy, enabling real-time analysis of high-resolution images and videos [21].

The architecture of YOLOv11 is modular, comprising three main components: backbone, neck, and head, as shown in Fig.2 [22]. This one-stage detection framework unifies bounding box regression and object classification into a single, end-to-end differentiable neural network, offering significant efficiency gains over traditional two-stage methods [23]. Enhancements to these components—such as architectural refinements and parameter optimization—result in superior speed and accuracy [21, 24].

a. Backbone

The backbone serves to extract multi-scale feature representations through progressive down-sampling, using a modified Cross Stage Partial (CSP) architecture integrated with C3K2 blocks and Spatial Pyramid Pooling-Fast (SPPF). These modules improve computational efficiency and preserve critical visual features. Additionally, the C2PSA (Cross Stage Partial with Spatial Attention) module refines feature focus via spatial attention mechanisms. The output features span various resolutions (e.g., 320×320 to 20×20), capturing low-level to high-level information essential for detecting objects of varying sizes and contexts [22].

b. Neck

The neck connects the backbone and head, enhancing spatial and semantic richness through multi-scale feature fusion. It employs a PANet structure comprising convolutional layers, C3K2 blocks, up-sampling, and concatenation layers. The integration of C2PSA and SPPF modules strengthens object localization, particularly for small or partially occluded objects. This component maintains feature integrity across resolutions, supporting robust detection in complex scenarios such as dense traffic environments [22, 23].

c. Head

The head component produces final predictions, including bounding box coordinates, objectness scores, and class probabilities [25]. YOLOv11 adopts an anchor-free detection strategy, reducing computational complexity by directly predicting bounding boxes from feature maps. A decoupled head design separately optimizes regression and classification paths, enhancing both localization precision and class discrimination. The objectness score, calculated via a sigmoid function, represents the likelihood of object presence, while class probabilities are determined using softmax for exclusive multi-class classification [26]. Multi-scale outputs from feature maps (e.g., 80×80, 40×40, 20×20) enable the simultaneous detection of various object sizes with real-time inference capabilities.

d. Inference and Post-Processing

The evaluation of the YOLO model constitutes a critical step in assessing the performance of object detection algorithms across diverse scenarios. This process incorporates several key metrics, including precision, recall, and mean Average Precision (mAP). These metrics collectively provide comprehensive insights into the model's ability to accurately detect and classify objects [23].

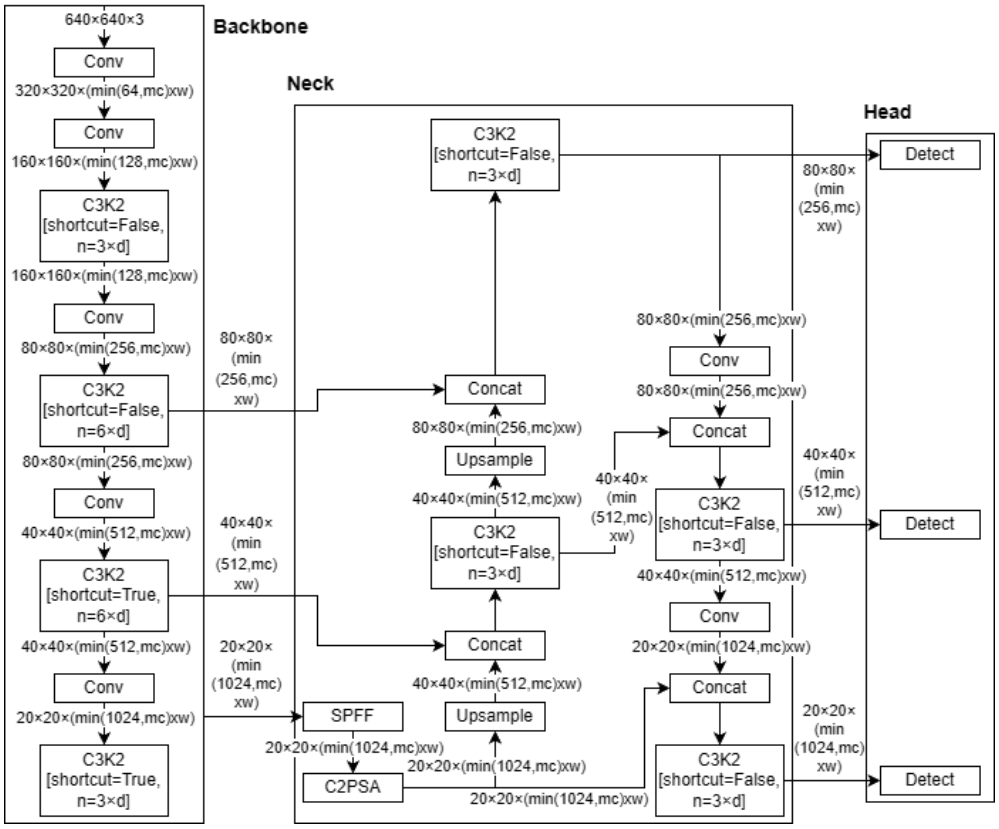


Fig.2. YOLOv11 architecture

3.4.2. Detection Workflow of YOLOv11

YOLOv11 detects and classifies objects—such as vehicles and accidents—in a single pass. Given an input image (e.g., 640×640 pixels), typically sourced from traffic surveillance, the backbone extracts complex visual features using C3K2, SPPF, and C2PSA modules. These are passed to the PANet-based neck for multi-scale feature integration and subsequently to the head for final prediction [23, 26].

Unlike earlier YOLO versions, YOLOv11 employs an anchor-free and NMS-free detection strategy. The decoupled head independently generates bounding boxes and class predictions, enhancing localization precision. A consistent dual assignment mechanism replaces traditional Non-Maximum Suppression (NMS), thereby reducing post-processing latency. Final predictions are determined based on a confidence score, calculated as follows (1):

$$Confidence\ Score = P(Object) \times P(Class_i|Object) \tag{1}$$

In this formulation, $P(Object)$ represents the objectness score, indicating the probability that a particular region contains an object rather than background. It is derived using a sigmoid activation function that normalizes the prediction to a value between 0 and 1. The term $P(Class_i|Object)$ refers to the conditional class probability, which quantifies the

likelihood that the detected object belongs to a specific class i, given that an object is indeed present. This is computed using a softmax function across all predefined classes to ensure exclusivity and probabilistic consistency. The resulting confidence score thus reflects both the presence and classification certainty of an object within a region [23, 25]. An additional filtering step using the Intersection over Union (IoU) metric is applied to refine results, wherein overlapping predictions are suppressed based on a predefined threshold (e.g., 0.5), further enhancing precision by eliminating redundant detections [27].

3.4.3. Model Evaluation Metrics

The evaluation of the YOLO model plays a critical role in assessing the performance of object detection algorithms across various contexts. This evaluation process employs several key metrics, including precision, recall, mAP, accuracy, and MAPE, each of which serves to measure the model's effectiveness in accurately detecting and classifying objects.

a. Precision

Precision measures the proportion of correctly identified positive detections among all positive predictions made by the model. It is particularly important when the cost of false positives is high, such as in email spam detection systems [28]. The mathematical formulation for precision is presented in (2):

$$Precision = \frac{TP}{TP+FP} \tag{2}$$

Where TP (True Positive) indicates the count of positive instances that the model has correctly classified. Conversely, FP (False Positive) corresponds to the number of negative instances that the model has erroneously labeled as positive.

b. Recall

Recall quantifies the model's ability to identify all relevant objects in the dataset. This metric is crucial in applications such as medical diagnostics, where missing positive cases (false negatives) can lead to serious consequences [28]. The mathematical formulation for recall is presented in (3):

$$Recall = \frac{TP}{TP+FN} \tag{3}$$

Where TP refers to the number of positive instances that the model successfully identified. Meanwhile, FN (False Negative) denotes the number of actual positive instances that were not recognized by the model and were instead incorrectly classified as negative.

c. Mean Average Precision (mAP)

Mean Average Precision (mAP) is a comprehensive evaluation metric commonly used to measure the performance of object detection models across multiple classes and thresholds. It represents the mean of the Average Precision (AP) values computed for each object class. The AP is derived from the area under the precision-recall curve at various IoU thresholds [29]. The formulas for mAP are given in (4) and (5):

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \tag{4}$$

$$AP = \sum_{r_n}^{r_{n+1}} (r_{n+1} - r_n) \cdot P_{interp}(r_{n+1}) \tag{5}$$

Where n represents the total number of object classes considered in the evaluation. AP_k refers to the AP for the k^{th} class, which corresponds to the area under the interpolated precision-recall curve specific to that class. The terms r_n and r_{n+1} indicate consecutive recall thresholds used in the calculation. $P_{interp}(r)$ denotes the interpolated precision at a specific recall level. In computing AP, the model's performance is assessed across multiple IoU thresholds. IoU itself quantifies the degree of overlap between the predicted bounding box and the ground truth [29]. The mathematical formulation for IoU is presented in (6):

$$IoU = \frac{Overlap\ Area}{Combined\ Area} \tag{6}$$

Where the $Overlap\ Area$ indicates the intersecting region shared by the predicted bounding box and the ground truth box, while the $Combined\ Area$ represents the union of both boxes, encompassing the total area covered by either. Variants of this metric include mAP50, which calculates mAP using a fixed IoU threshold of 0.5, and mAP50–95, which computes the average of AP values across multiple IoU thresholds ranging from 0.5 to 0.95 in 0.05 increments [29]. This latter variant offers a more comprehensive evaluation by reflecting the model's performance across varying degrees of localization precision.

d. Accuracy

The accuracy evaluation in this study employs a proportional deviation-based metric, as adopted in prior traffic counting and surveillance studies [13], to quantify prediction error relative to ground-truth values while accounting for both under-detection and over-detection. This metric is particularly suitable for vehicle counting and speed estimation tasks, where performance is better reflected by the magnitude of deviation between predicted and actual quantities rather than by discrete classification correctness. Unlike conventional accuracy measures that rely on true–false classifications, the selected approach evaluates the relative prediction error and transforms it into an interpretable accuracy score, thereby providing a more representative assessment of counting-based outputs [13, 30]. The mathematical formulation of the accuracy metric is defined in (7):

$$Accuracy = 1 - \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \times 100 \tag{7}$$

Where A_t denotes the actual (ground-truth) value at time or instance t, F_t represents the corresponding predicted value, and n indicates the total number of observations. By normalizing prediction errors with respect to the actual values, this metric ensures scale invariance and enables fair comparison across different traffic densities. Consequently, it offers a more robust and application-relevant evaluation for real-world traffic analysis scenarios, as demonstrated in [13].

e. Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is used to measure the average proportional deviation between predicted and actual values, complementing the proportional deviation-based accuracy metric. Unlike accuracy, which converts relative error into a correctness score, MAPE directly represents the magnitude of prediction error in percentage form, making it suitable for continuous estimation tasks such as vehicle speed measurement [13, 30]. The mathematical formulation for MAPE is defined in (8):

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \times 100 \tag{8}$$

Where A_t is the actual value, and F_t is the predicted value, and n is the number of observations. By normalizing errors relative to ground-truth values, MAPE provides an interpretable and scale-invariant measure of estimation precision that aligns conceptually with the accuracy metric [30].

3.5. Vehicle and Accident Detection Application

The Vehicle and Accident Detection Application is a web-based interface developed using the Streamlit library, designed to implement YOLOv11-based object detection for identifying vehicles and traffic accidents. The application provides an intuitive platform for users to conduct detection tasks by uploading images, videos, or real-time input (screen record), selecting the desired detection model, and interactively viewing the detection results. The system workflow is illustrated in Fig.3.

The detection process begins with model selection, followed by the configuration of several core parameters that govern the detection and analysis process. These include the confidence threshold (expressed as a percentage to filter low-confidence predictions), IoU threshold (to refine overlapping bounding boxes), target width and height (in meters, used to convert pixel dimensions to real-world scale for speed estimation), and detection zone coordinates (defined in pixel format to limit detection to a specific area within the frame). Additionally, each frame is timestamped to enable temporal tracking and correlation with external events.

Input data can be provided through video files, live webcam feeds, or screen recordings. Each input frame is extracted and converted to grayscale using NumPy, OpenCV, and Pillow libraries to enhance processing efficiency. Object detection is then performed using the selected YOLOv11 model and the configured thresholds, utilizing Ultralytics, Supervision, and Torch libraries. Vehicle counts and speed estimates are calculated using Pandas, and accident detection events trigger notifications on the dashboard.

Annotated color frames, along with detection visualizations created using Plotly, are displayed in real time. Once processing is complete, all annotated frames are compiled into a final video output. Simultaneously, detection results—such as vehicle counts, speed data, and timestamps—are aggregated and exported. Users may download these results in spreadsheet format (.xlsx) and as a compiled video for further analysis or reporting.

To install and operate the application optimally, users must ensure that their system meets the minimum hardware and software requirements and that all necessary libraries are properly installed. In this study, the application is deployed locally on a development device equipped with an AMD Ryzen 5 5600 CPU (6 cores, 12 threads, up to 4.4 GHz), an NVIDIA GeForce RTX 3060 GPU (8GB GDDR6, 3584 CUDA cores, NVIDIA Studio Driver 572.16), 16GB dual-channel RAM (3200MHz), and a 500GB NVMe SSD (PCIe Gen 3x4). The operating system is Windows 11 (version 24H2), and development was carried out using Visual Studio Code (version 1.96.4) and Python (version 3.12.7).

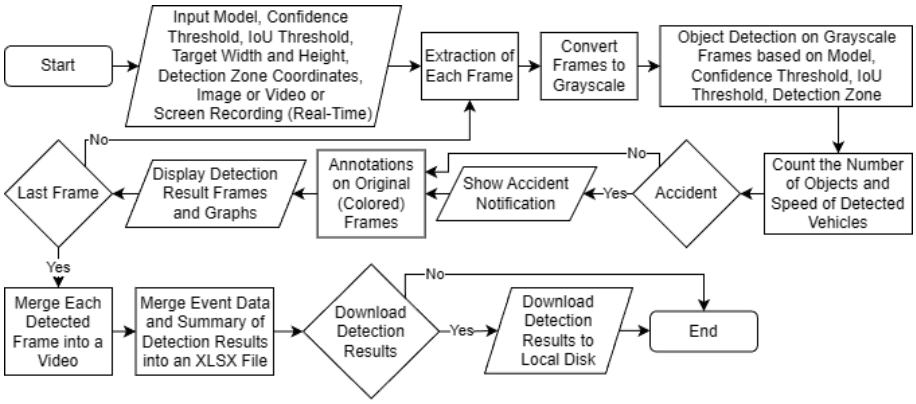


Fig.3. Vehicle and accident detection application flow

Required Python libraries include NumPy (2.2.2) for numerical computations, OpenCV (4.10.0.84) for image processing, Pandas (2.2.3) for data manipulation, Pillow (11.1.0) for image operations, Plotly (5.24.1) for data visualization, Streamlit (1.40.2) for the web interface, Torch (2.5.1) for deep learning acceleration, Ultralytics (8.3.40) for YOLO-based object detection, and Supervision (0.25.1) for detection result processing. The application is available in the official GitHub repository [31], which includes detailed installation instructions, covering Python environment setup, dependency installation via pip, and CUDA configuration for accelerated model inference.

This application is deployed locally rather than on cloud platforms due to the high computational demands of deep learning inference using YOLO, particularly for real-time video analysis. Deploying to a cloud service—such as AWS EC2 with NVIDIA GPU support, Google Cloud AI Platform, or Microsoft Azure—typically incurs significant operational costs based on usage time, compute resources, and storage, especially for continuous video processing. Local deployment allows the developer to utilize a high-performance personal device (with a capable CPU and GPU) without incurring additional operational expenses. Furthermore, local deployment provides greater flexibility during development and testing phases, enabling performance optimization without concerns related to latency or bandwidth limitations often encountered in cloud environments.

3.5.1. Estimation of Vehicle Speed and Count

The estimation of vehicle speed and count in traffic surveillance involves a multi-step process combining advanced object detection, tracking, perspective transformation, and speed calculation techniques. This integrated approach addresses challenges such as perspective distortion and temporal consistency to yield accurate estimations.

Object Detection is the initial step, employing a deep learning-based model (YOLO) that efficiently detects vehicles within each video frame in real time. The model generates bounding boxes identifying vehicle locations, confidence scores indicating detection certainty, and class labels specifying vehicle types [32, 33]. These detections form the basis for subsequent analysis.

Object Tracking follows, utilizing the ByteTrack algorithm to assign consistent unique IDs to each detected vehicle across frames. This ensures reliable correspondence of vehicle positions over time despite occlusions or missed detections, which is critical for accurately measuring movement trajectories and speed [33].

Perspective Transformation compensates for distortions caused by the camera's angle relative to the road. Given source points in the image and their corresponding real-world coordinates, a transformation matrix is computed using OpenCV's perspective transformation functions. This matrix converts pixel-based coordinates of detected vehicles (typically the bottom-center of bounding boxes) into real-world spatial coordinates, enabling precise distance measurements [33].

Speed Calculation is performed by tracking the change in a vehicle's real-world position over time. For each tracked vehicle, a history of vertical world coordinates is maintained over several frames. Once sufficient data is collected, the displacement is calculated as the absolute difference between the initial and final coordinate values. The elapsed time is derived from the number of frames in the history divided by the video's frame rate (FPS) [14, 33]. The vehicle's speed in kilometers per hour (km/h) is then computed using the formula (9):

$$Speed(km/h) = \frac{Distance\ (meters)}{Time\ (seconds)} \times 3.6 \tag{9}$$

Where $Distance$ refers to the absolute vertical displacement of the vehicle measured in meters, while $Time$ represents the interval during which this displacement occurs, calculated by dividing the number of frames by the video's FPS. The multiplication factor of 3.6 serves to convert the resulting speed from meters per second (m/s) to km/h. This methodology effectively reduces the impact of minor fluctuations caused by bounding box variations across consecutive frames by averaging the vehicle's movement over a series of frames, thereby producing speed estimates that are both more stable and realistic [33]. Collectively, the combination of YOLO-based detection, ByteTrack tracking, perspective transformation, and speed calculation constitutes a comprehensive and reliable system for real-time traffic analysis.

3.5.2. Accident Detection

Accident detection is the automated identification of traffic collisions using CCTV footage and deep learning models such as YOLO for real-time video analysis. Rather than merely detecting the proximity of two vehicles, the model assesses interaction patterns, including vehicle tilt angles, deformation from impacts, and abrupt changes in relative positions across sequential frames. Detection criteria include abnormal vehicle overlap, overturned vehicles, or the presence of displaced objects such as thrown pedestrians, which increase the likelihood of classifying an event as an accident [10]. Conversely, vehicles traveling close together but maintaining normal trajectories without sudden positional or shape changes are classified as merely passing each other. The model, trained on a dedicated accident dataset, also incorporates features such as vehicle debris and visible damage patterns to enhance accuracy.

3.6. Model Evaluation in Real-World Scenarios

The evaluation of the trained YOLOv11 models in real-world scenarios involves assessing their performance on previously unseen data using the developed application. This process is conducted through two testing scenarios: (1) detection of vehicle counts and accident occurrences, and (2) estimation of vehicle speeds. The outputs from each model are assessed using appropriate evaluation metrics to quantify their performance under real-world conditions—accuracy for the first scenario and MAPE for the second scenario.

In the first scenario, the detection performance of each trained YOLOv11 model—regarding the number of vehicles and accidents—is tested using the application's video and real-time detection features. A test video of 23 seconds in duration, recorded at a resolution of 1280×720 pixels and a frame rate of 25 FPS, captures traffic and an accident incident at the CCTV location “SIMP. PIDADA”. Detection parameters include a confidence threshold of 50% and an IoU threshold of 70%. The resulting detection outputs are saved in a spreadsheet file under the “Summary” sheet, containing the total number of detected vehicles and accidents per class. These results are then compared with manually verified ground-truth counts. Detection performance is evaluated in terms of accuracy, calculated as the percentage of correctly detected objects relative to the actual counts.

In the second scenario, the evaluation focuses on vehicle speed estimation, excluding the accident-specific model. This test is also conducted using the application's video and real-time detection features. A 29-second test video with a resolution of 1280×720 pixels and a frame rate of 25 FPS captures a motorcycle traveling at predefined speeds of 9, 13.5, 18, 22.5, and 27 km/h along Batu Intan 7 Street. The model is configured with a confidence threshold of 35% and an IoU threshold of 70%. The estimated speeds are logged in a spreadsheet under the “Events” sheet, which records detected objects (excluding accident classes) along with their corresponding speeds and object IDs per second. For each object ID, the estimated speed is averaged over time and evaluated using the MAPE to determine the percentage error relative to the actual speeds.

4. Results and Discussion

The Results and Discussion section presents the outcomes of the conducted experiments, including data collection, preprocessing, augmentation, YOLOv11 model training, the Vehicle and Accident Detection Application, and the evaluation of the model in real-world scenarios.

4.1. Data Collection

This study employed a diverse dataset encompassing both traffic accident scenarios and general vehicle scenes, collected from various online platforms such as Roboflow, Dataset Ninja, YouTube, and Gemini AI, as well as offline contributions from the Denpasar City Transportation Department. The data include both annotated images and raw video recordings, covering a wide range of conditions in terms of resolution, object scale, and distance from the camera. Several publicly available datasets were utilized to enhance model generalization across varying viewpoints, such as aerial drone footage and fixed roadside surveillance. A detailed overview of the datasets, including their sources, formats, and quantities, is provided in Table 2.

Table 2. Summary of collected datasets. This table summarizes the datasets used, detailing their sources, formats, and data volumes in images and videos

Dataset Name	Source	Format	Total/Duration
Real-Time Accident Detection Computer Vision Project	Roboflow (online), by Puyush	Image	2,629 images (2,375 training, 254 validation)
VisDrone2019-DET Dataset	Dataset Ninja (online), by Zhu et al.	Image	10,209 images (6,471 training, 548 validation, 1,610 test-dev, 1,580 test-challenge)
Vehicle Dataset for YOLO	Dataset Ninja (online), by Pethiyagoda	Image	3,000 images (2,100 training, 900 validation)
Roadside CCTV Footage Dataset – Demak Regency	YouTube channel (online) of the Demak Regency Transportation Department	Video	10 videos (total duration: 8 hours 42 minutes 50 seconds)
Roadside CCTV Footage Dataset – Denpasar City	Denpasar City Transportation Department (offline)	Video	42 videos (total duration: 51 hours 30 minutes 17 seconds)
AI-Generated Accident Dataset	Gemini AI (online)	Image	3,085 images

In addition to the existing datasets, real-world CCTV recordings were incorporated to better represent typical urban and suburban traffic environments. Due to the limited availability of annotated accident imagery, synthetic images were also generated using AI-based tools to simulate diverse accident scenarios.

4.2. Data Preprocessing

Data preprocessing was conducted using the Roboflow platform, focusing on standardizing class labels and filtering irrelevant or redundant data. Class names were adjusted to align with a predefined object list, including merging synonymous labels (e.g., "bike" and "motor" renamed to "motorcycle"). The "van" class required manual relabeling due to inconsistencies with vehicle size and function. Non-essential classes such as pedestrians, bicycles, and undefined regions were excluded, and any resulting images without valid annotations were removed.

Additional cleaning steps were applied to specific datasets: duplicates were manually removed from the Roboflow accident dataset, while the VisDrone2019 set was filtered to retain only relevant vehicle types and discard overly distant aerial views. CCTV video footage from Demak and Denpasar was manually annotated frame-by-frame at 10-second intervals. One representative Denpasar video was selected for annotation due to time constraints. AI-generated accident images from Gemini were also annotated in accordance with the established class list. Roboflow's dataset splitting feature ensured balanced distribution of object classes across training, validation, and test sets, supporting robust model generalization. A summary of the resulting datasets after preprocessing is provided in Table 3.

Table 3. Summary of preprocessed datasets. This table presents the number of images in each dataset after preprocessing, divided into training, validation, and testing subsets

Dataset Name	Total Images	Train	Validation	Test
Accident Dataset	4,507	3,155	902	450
Filtered "Vehicle Dataset for YOLO"	2,470	1,729	494	247
Filtered "VisDrone2019-DET Dataset"	1,042	730	209	103
CCTV Traffic Image Dataset	4,476	3,135	896	445

These refined datasets, covering both general traffic and accident scenarios, served as the basis for subsequent data augmentation and model development processes.

4.3. Data Augmentation

To improve model generalization and address class imbalance, data augmentation was applied to the training subsets using Roboflow. Augmentation techniques included resizing, horizontal flipping, rotation, cropping, brightness adjustment, and shearing, with parameters adapted to object scale and image characteristics. Additional noise and blur effects were selectively applied to enhance robustness under varying visual conditions. All augmentations were performed exclusively on the training data, thereby increasing dataset diversity without affecting validation and testing integrity.

After augmentation, the datasets were finalized and organized into three categories: a vehicle detection dataset, an accident detection dataset, and a combined vehicle–accident dataset. The distribution of images across training, validation, and testing subsets for each dataset is summarized in Table 4.

Table 4. Summary of augmented datasets. This table presents the distribution of images in the augmented datasets across training, validation, and testing

Dataset Name	Total Images	Train	Validation	Test
Vehicle	13,581	11,187	1,599	795
Accident	12,818	11,569	834	415
Vehicle-Accident	27,531	23,785	2,501	1,245

Fig.4 illustrates representative samples from the augmented "Vehicle-Accident" dataset. All datasets presented in Table 4 constitute the final versions used for subsequent model training and evaluation.





Fig.4. Sample augmented image

4.4. Model Training

The training of YOLOv11 models was conducted using consistent hyperparameter settings across all experiments to ensure a fair and reliable comparison. Two model architectures were evaluated—YOLOv11 Large (l) and YOLOv11 Nano (n)—using three finalized datasets: Vehicle–Accident, Vehicle, and Accident. Based on the combination of architectures and datasets, six detection models were produced. The training performance of each model is summarized in Table 5.

Table 5. Summary of YOLOv11 model training results. This table presents key training metrics and model specifications

Model	Precision	Recall	mAP50	mAP50–95	Parameters	Inference (ms)	Size (MB)
l-Vehicle-Accident	81.8%	75.8%	82.1%	53.3%	25.3M	12.2	51.2
l-Vehicle	88.3%	84.6%	90.8%	70.3%	25.3M	13.4	51.2
l-Accident	75.7%	74.3%	79.2%	47.2%	25.3M	11.9	51.2
n-Vehicle-Accident	77.0%	72.6%	79.0%	49.3%	2.6M	2.4	5.5
n-Vehicle	86.7%	78.1%	86.3%	63.4%	2.6M	2.4	5.5
n-Accident	74.3%	72.2%	77.2%	45.7%	2.6M	2.1	5.5

The results show that YOLOv11 Large models consistently outperform Nano models in terms of detection accuracy, particularly for vehicle-related tasks, albeit at a higher computational cost. In contrast, YOLOv11 Nano models offer substantially faster inference and smaller model sizes, making them more suitable for deployment in resource-constrained environments.

Detection of accident-specific classes remains more challenging for both architectures, as indicated by relatively lower precision and recall values compared to vehicle detection tasks. Overall, these results demonstrate a clear trade-off between accuracy and efficiency, where the large models are better suited for complex detection scenarios, while the Nano models provide a lightweight alternative with acceptable performance.

4.5. Vehicle and Accident Detection Application

The trained YOLOv11 models were integrated into a web-based application developed using Streamlit, enabling real-time detection of vehicles and accidents through an interactive dashboard interface. The application (Fig.5) supports image, video, and live stream inputs, displaying detection outputs directly on the dashboard alongside bar charts that summarize object counts per class.

Users can select models, adjust detection parameters such as confidence and IoU thresholds, and configure detection settings including frame dimensions, detection zones, and timestamping. The interface also provides controls to start, pause, or continue detection, and automatically generates a downloadable output upon completion. In the event of an accident, a real-time notification is displayed.

Detection results using the YOLOv11l-Vehicle-Accident model are shown in Fig.6. The model was applied to a 23-second CCTV recording from the “SIMP. PIDADA” location, using a confidence threshold of 50% and an IoU threshold of 70%. The output includes annotated video displaying object class labels, estimated speeds, and total object counts. Real-time detection operates at a resolution of 640×360 pixels and achieves approximately 11.364 FPS on an NVIDIA RTX 3060 GPU, while the video detection feature processes inputs at their original resolution (1280×720 pixels, 25 FPS).

Both detection modes differ significantly in processing strategy. The video detection feature analyzes every frame sequentially, preserving full temporal and spatial fidelity, resulting in higher accuracy for speed estimation and object counting. However, this method is computationally intensive and slower. Conversely, the real-time feature prioritizes responsiveness by selectively skipping frames depending on hardware capacity, resulting in faster processing at the expense of reduced estimation accuracy—especially on less capable devices. Therefore, video detection is better suited for high-precision analysis, while real-time detection is ideal for live monitoring where processing speed is critical.

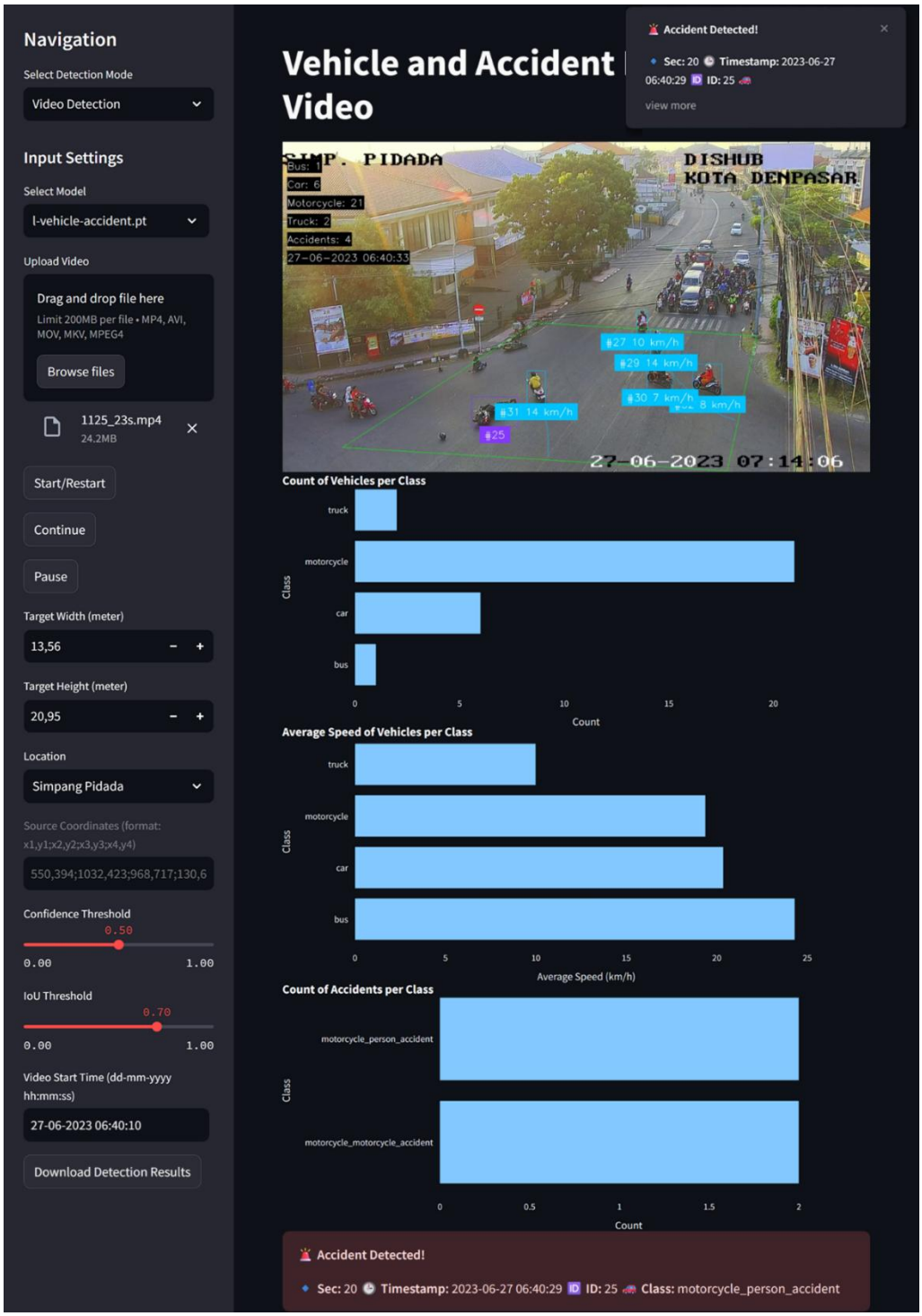


Fig.5. Vehicle and accident detection application user interface

In addition to annotated video outputs, the system generates structured tabular data in a spreadsheet format (Fig.7). The file includes two sheets: “Events” logs each detected object with timestamp, ID, class, and speed; “Summary” compiles overall statistics such as object counts per class and average speed. This output supports data-driven analysis and facilitates precise incident tracking.

Finally, detection results are visualized through downloadable bar charts (Fig.8), which illustrate object distributions by class and variations in vehicle speed over time. These visualizations enhance interpretability, support documentation, and serve as useful tools for communicating findings to relevant stakeholders.



Fig.6. Video output

Sec	Timestamp	ID	Class	Speed (km/h)
20,92	2023-06-27 06:40:31	24	motorcycle	18
20,92	2023-06-27 06:40:31	26	motorcycle_person_accident	
20,96	2023-06-27 06:40:31	25	motorcycle_person_accident	
20,96	2023-06-27 06:40:31	28	motorcycle_motorcycle_accident	
20,96	2023-06-27 06:40:31	24	motorcycle	18
20,96	2023-06-27 06:40:31	27	motorcycle	14,4
20,96	2023-06-27 06:40:31	26	motorcycle_person_accident	
21	2023-06-27 06:40:31	25	motorcycle_person_accident	
21	2023-06-27 06:40:31	27	motorcycle	14,4
21	2023-06-27 06:40:31	24	motorcycle	18
21	2023-06-27 06:40:31	28	motorcycle_motorcycle_accident	
21,04	2023-06-27 06:40:31	25	motorcycle_person_accident	
21,04	2023-06-27 06:40:31	27	motorcycle	14,4
21,04	2023-06-27 06:40:31	24	motorcycle	18
21,04	2023-06-27 06:40:31	28	motorcycle_motorcycle_accident	

Events | Summary

Class	Count	Average Speed (km/h)
bus	1	24,30316078
bus_bus_accident	0	
bus_object_accident	0	
bus_person_accident	0	
bus_truck_accident	0	
car	6	20,35017735
car_bus_accident	0	
car_car_accident	0	
car_motorcycle_accident	0	
car_object_accident	0	
car_person_accident	0	
car_truck_accident	0	
motorcycle	21	19,36773464
motorcycle_bus_accident	0	
motorcycle_motorcycle_accident	2	

Events | Summary

Fig.7. Spreadsheet file output

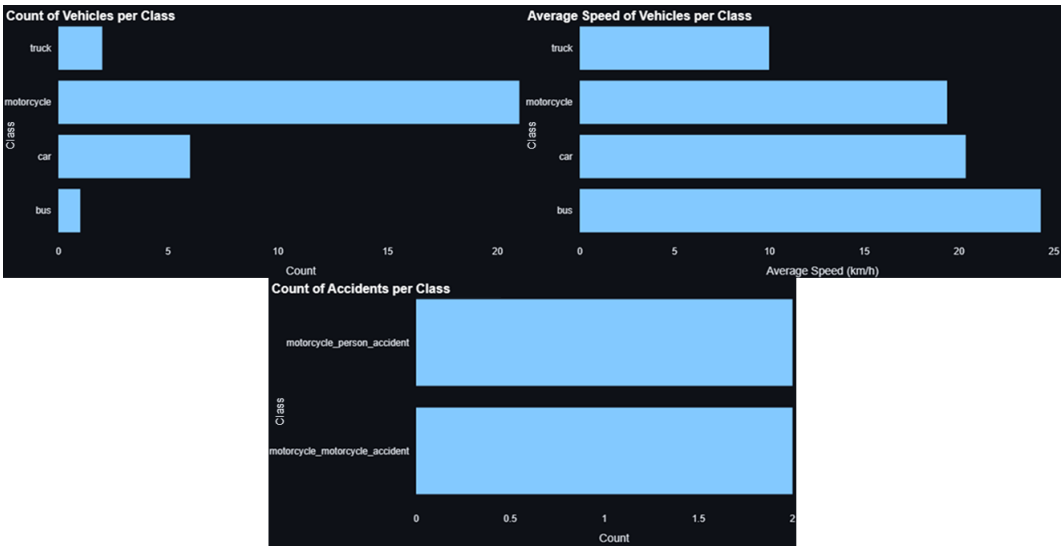


Fig.8. Chart output

4.6. Model Evaluation in Real-World Scenarios

The trained YOLOv11 models were evaluated using real-world data through the developed application. The evaluation employed CCTV footage from the “SIMP. PIDADA” intersection for vehicle and accident detection, and footage from Batu Intan 7 Street for vehicle speed estimation. Detection and speed estimation outputs were recorded in spreadsheet format, with vehicle and accident counts stored in the “Summary” sheet and speed estimation results in the “Events” sheet.

Vehicle and accident detection performance was assessed using a 23-second video (1280×720 pixels, 25 FPS) from the “SIMP. PIDADA” CCTV, with a confidence threshold of 50% and an IoU threshold of 70%. Both video-based and real-time detection modes were evaluated, and the detection results are summarized in Table 6.

Table 6. Summary of YOLOv11 vehicle and accident detection results in real-world conditions. This table summarizes YOLOv11's vehicle and accident detection accuracy in real-world scenarios

Actual/Model (YOLOv11)		Count of Objects per Class						Accuracy (%)
		bus	car	motorcycle	truck	motorcycle_ motorcycle_accident	motorcycle_ person_accident	
Actual Count		1	6	22	2	2	2	-
Video Detection Feature	l-Vehicle-Accident	1	6	21	2	2	2	99.24
	l-Vehicle	1	6	25	2	-	-	96.59
	l-Accident	-	-	-	-	2	2	100
	n-Vehicle-Accident	1	6	22	2	1	1	83.33
	n-Vehicle	1	6	26	2	-	-	95.45
	n-Accident	-	-	-	-	2	3	75
Real-Time Detection Feature	l-Vehicle-Accident	1	6	21	2	2	2	99.24
	l-Vehicle	1	6	24	2	-	-	97.73
	l-Accident	-	-	-	-	1	2	75
	n-Vehicle-Accident	1	6	22	2	1	1	83.33
	n-Vehicle	1	6	26	2	-	-	95.45
	n-Accident	-	-	-	-	2	4	50

As shown in Table 6, detection accuracy varied across models, with YOLOv11l-Vehicle-Accident demonstrating the most consistent and reliable performance in both modes, achieving an accuracy of 99.24%. Minor under-detections were observed in motorcycle-related classes, while Nano models exhibited more pronounced over- and under-detection, particularly in real-time operation due to frame skipping.

Vehicle speed estimation was conducted using a 29-second video (1280×720 pixels, 25 FPS) capturing motorcycles traveling at predefined speeds ranging from 9 to 27 km/h. Speed estimation relied on manual calibration of real-world dimensions, performed prior to evaluation. Calibration was based on direct measurement of known physical references in the scene, including the approximate dimensions of a representative vehicle (Toyota Avanza) at the "SIMP. PIDADA" intersection and manual field measurements of lane width and detection zone dimensions at Batu Intan 7 Street. These measurements were used to map pixel displacement into real-world distance (meters), forming the basis for speed computation.

To reduce calibration bias and measurement noise, the evaluation results reported in Table 7 were obtained from multiple repeated trials under identical settings, with final values computed as aggregated outcomes across runs. Performance was evaluated using MAPE, which directly reflects proportional deviation between estimated and actual speeds.

Table 7. Summary of YOLOv11 vehicle speed estimation results in real-world conditions. This table summarizes YOLOv11's vehicle speed estimation accuracy using MAPE in real-world scenarios

Actual/Model (YOLOv11)		Vehicle Speed per ID (km/h)					MAPE (%)
		#1	#2	#3	#4	#5	
Actual Speed		9.00	13.50	18.00	22.50	27.00	-
Video Detection Feature	l-Vehicle-Accident	10.07	13.80	17.73	22.60	26.50	3.56
	l-Vehicle	9.34	16.87	20.13	24.60	31.80	13.49
	n-Vehicle-Accident	10.83	14.08	18.23	22.50	27.00	5.21
	n-Vehicle	11.24	14.17	20.83	22.60	26.40	9.60
Real-Time Detection Feature	l-Vehicle-Accident	8.59	12.95	16.07	20.70	26.80	5.54
	l-Vehicle	8.50	12.60	14.18	23.50	28.40	8.58
	n-Vehicle-Accident	9.42	15.49	16.84	22.50	29.90	7.34
	n-Vehicle	9.17	11.83	15.98	20.10	27.40	7.53

As shown in Table 7, YOLOv11l-Vehicle-Accident achieved the lowest estimation error, with MAPE values of 3.56% in video mode and 5.54% in real-time mode, indicating stable and accurate speed estimation. Nano models exhibited higher variance, particularly at lower speeds, reflecting greater sensitivity to detection jitter and frame skipping.

Overall, video-based evaluation yielded more reliable results due to complete frame processing, whereas real-time evaluation involved a trade-off between accuracy and processing speed. Although formal statistical significance testing and confidence intervals were not explicitly computed, the consistency of performance trends across repeated trials and across multiple evaluation modes supports the robustness of the observed results. Under real-world conditions, YOLOv11l-Vehicle-Accident consistently outperformed other configurations, demonstrating its suitability for integrated traffic monitoring applications, while Nano models provided faster inference at the cost of reduced accuracy.

5. Conclusions

This study demonstrates the effectiveness of YOLOv11 for integrated traffic scene analysis, enabling accident detection, vehicle counting, and speed estimation from roadside CCTV footage. The trained models were deployed in a Streamlit-based application supporting image, video, and real-time inputs, providing practical functionality for traffic monitoring and decision support.

Experimental results show that model architecture strongly influences the trade-off between accuracy and efficiency. YOLOv11l-Vehicle-Accident achieved the best overall performance, with 81.8% precision, 75.8% recall, 82.1% mAP50, and 53.3% mAP50–95, and delivered stable real-world results with 99.24% detection accuracy and low speed estimation error (MAPE 3.56%–5.54%). In contrast, Nano models offered faster inference and lower resource requirements but exhibited reduced robustness, particularly for accident detection and real-time speed estimation.

From a deployment perspective, large models are more suitable for centralized systems prioritizing accuracy, while Nano variants better support resource-constrained and scalable deployments where efficiency is critical. Future work will focus on improving robustness through richer data augmentation, improved class balancing, and hardware-aware optimization, as well as extending the system with congestion analysis based on vehicle density and speed patterns to enhance its value for traffic management applications.

Author Contributions Statement

I Kadek Rai Pramana: Conceptualization, Methodology, Data Curation, Software Development, Model Training, and Overall Project Coordination: Developed the core research idea, designed the methodological framework, led data collection, preprocessing, and augmentation, implemented the vehicle and accident detection system, conducted model training, and coordinated all stages of the project.

I Putu Agung Bayupati: Supervision, Technical Guidance, and Methodological Refinement: Provided supervisory oversight, contributed to research direction and methodological refinement, and assisted in validating the design and implementation of the proposed system.

Gusti Made Arya Sasmita: Model Training Support, Evaluation Input, and Technical Review: Assisted in model training and experimental design, contributed to the evaluation of system performance in real-world scenarios, and provided technical insights to strengthen the study.

Ngoc Le: Writing (Review & Editing) and Advisory Support: Reviewed and edited the entire manuscript for academic accuracy and clarity, provided advisory input for scientific presentation, and ensured the overall coherence and quality of the final submission.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare that there are no conflicts of interest regarding the publication of this study.

Funding Declaration

This research received no external funding and was not supported by any institutional, organizational, or commercial grants.

Data Availability Statement

All data supporting the findings of this study are publicly accessible. The processed datasets used for model training and evaluation are available from the following sources: the Vehicle Dataset (https://universe.roboflow.com/nyankosenseiroku/vehicle-kxrjg), the Accident Dataset (https://universe.roboflow.com/nyankosenseiroku/accident-wwgfl), and the Vehicle-Accident Dataset (https://universe.roboflow.com/nyankosenseiroku/vehicle-accident-x2puh). The public source datasets referenced in this research can be obtained from openly available repositories, including the Real Time Accident Detection Computer Vision Project Dataset (https://universe.roboflow.com/puyush-fipgg/real-time-accident-detection), the Vehicle Dataset for YOLO (https://datasetninja.com/vehicle-dataset-for-yolo), the VisDrone2019-DET Dataset (https://datasetninja.com/vis-drone-2019-det), and the Demak District Transportation Agency YouTube Channel (https://www.youtube.com/@cctvdishubdemak). All public source datasets were accessed on 8 November 2024.

Ethical Declarations

This study did not involve human participants, animal subjects, or any procedures requiring institutional ethical approval. All datasets utilized in this research were publicly available and were used strictly in accordance with their respective terms of use.

Acknowledgments

The authors express their sincere appreciation to the reviewers and subject-matter experts for their professional evaluation

and constructive recommendations, which have significantly contributed to improving the quality, clarity, and rigor of this research. The authors also acknowledge the public institutions and dataset providers whose openly accessible resources enabled the completion of this study.

Declaration of Generative AI in Scholarly Writing

No generative artificial intelligence (AI) or AI-assisted technologies were used in the writing or editing of this manuscript. All textual content was prepared and fully validated by human contributors. AI tools (Gemini AI) were used solely for data augmentation within the research process and not for scholarly writing. The authors bear full responsibility for the accuracy and integrity of the manuscript.

Abbreviations

The following abbreviations are used in this manuscript:

AI - Artificial Intelligence
CCTV - Closed-Circuit Television
MAPE - Mean Absolute Percentage Error
YOLO - You Only Look Once
ROI - Region of Interest
YOLACT - You Only Look At CoefficienTs
KNN - K-Nearest Neighbors
GMM - Gaussian Mixture Model
CIoU - Complete Intersection Over Union
UAV - Unmanned Aerial Vehicle
CSP - Cross Stage Partial
C3K2 - Cross Convolution Block with 3×3 Kernels
SPPF - Spatial Pyramid Pooling-Fast
C2PSA - Cross Stage Partial with Spatial Attention
PANet - Path Aggregation Network
mAP - Mean Average Precision
NMS - Non-Maximum Suppression
IoU - Intersection over Union
TP - True Positive
FP - False Positive
FPS - Frames per Second

References

[1] M. Tzampazaki, C. Zografos, E. Vrochidou, and G. A. Papakostas, "Machine Vision—Moving from Industry 4.0 to Industry 5.0," *Applied Sciences (Switzerland)*, vol. 14, no. 4, pp. 1471–1503, 2024, doi: 10.3390/app14041471.

[2] I. G. P. S. Wijaya, M. N. Azmi, and A. Y. Husodo, "Comparative Analysis of YOLOv8 and HSV Methods for Traffic Density Measurement," *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, vol. 15, no. 2, pp. 134–148, 2024, doi: 10.24843/LKJITI.2024.v15.i02.p06.

[3] N. Suciati, N. P. Sutramiani, and D. Siahaan, "LONTAR_DETC: Dense and High Variance Balinese Character Detection Method in Lontar Manuscripts," *IEEE Access*, vol. 10, pp. 14600–14609, 2022, doi: 10.1109/ACCESS.2022.3147069.

[4] I. M. A. D. Suarjaya *et al.*, "Deep Learning Model Size Performance Evaluation for Lightning Whistler Detection on Arase Satellite Dataset," *Remote Sens (Basel)*, vol. 16, no. 22, 2024, doi: 10.3390/rs16224264.

[5] I. K. Gunawan, I. P. A. Bayupati, K. S. Wibawa, I. M. Sukarsa, and L. A. Kurniawan, "Indonesian Plate Number Identification Using YOLACT and Mobilenetv2 in the Parking Management System," *JUITA: Jurnal Informatika*, vol. 9, no. 1, pp. 69–76, 2021.

[6] J. S. W. Hutauruk, T. Matulatan, and N. Hayaty, "Deteksi Kendaraan Secara Real Time Menggunakan Metode YOLO Berbasis Android," *Jurnal Sustainable: Jurnal Hasil Penelitian dan Industri Terapan*, vol. 9, no. 1, pp. 8–14, 2020.

[7] F. Rofii, G. Priyandoko, M. I. Fanani, and A. Suraji, "Peningkatan Akurasi Penghitungan Jumlah Kendaraan dengan Membangkitkan Urutan Identitas Deteksi Berbasis Yolov4 Deep Neural Networks," *TEKNIK*, vol. 42, no. 2, pp. 169–177, 2021, doi: 10.14710/teknik.v42i2.37019.

[8] M. Zulfikri, Hairani, Ahmad, K. Abd. Latif, R. Hammad, and Moch. Syahrir, "Deteksi dan Estimasi Kecepatan Kendaraan dalam Sistem Pengawasan Lalu Lintas Menggunakan Pengolahan Citra," *Techno.Com*, vol. 20, no. 3, pp. 455–467, 2021.

[9] D. I. Mulyana and M. A. Rofik, "Implementasi Deteksi Real Time Klasifikasi Jenis Kendaraan Di Indonesia Menggunakan Metode YOLOV5," *Jurnal Pendidikan Tambusai*, vol. 6, no. 3, pp. 13971–13982, 2022.

[10] A. Rezky, A. Bagir, D. Pamerean, and F. Makhrus, "Deteksi Kecelakaan Lalu Lintas Otomatis Pada Rekaman CCTV Indonesia Menggunakan Deep Learning," *Buletin Pagelaran Mahasiswa Nasional Bidang Teknologi Informasi dan Komunikasi*, vol. 1, no. 1, pp. 1–5, 2023.

[11] L. A. Kurniawan, I. P. A. Bayupati, and K. S. Wibawa, "Sistem Hitung Kendaraan Berdasarkan Jenis Menggunakan Metode Background Subtraction," *Jurnal Ilmiah Teknologi dan Komputer*, vol. 1, no. 2, pp. 265–273, 2021.

[12] L. A. Kurniawan, I. P. A. Bayupati, K. S. Wibawa, I. M. Sukarsa, and I. K. Gunawan, "Sistem Klasifikasi Jenis dan Warna Kendaraan Secara Real-time Menggunakan Metode k-Nearest Neighbor dan Framework YOLACT," *Jurnal Edukasi dan Penelitian Informatika (JEPIN)*, vol. 7, no. 1, pp. 12–17, 2021.
[13] C. J. Lin, S. Y. Jeng, and H. W. Lioa, "A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO," *Math Probl Eng*, vol. 2021, no. 1, pp. 1–10, 2021, doi: 10.1155/2021/1577614.
[14] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, and X. Guo, "Real-Time Vehicle Detection Based on Improved YOLOv5," *Sustainability (Switzerland)*, vol. 14, no. 19, pp. 12274–12292, 2022, doi: 10.3390/su141912274.
[15] Z. Chen, L. Cao, and Q. Wang, "YOLOv5-Based Vehicle Detection Method for High-Resolution UAV Images," *Mobile Information Systems*, vol. 2022, no. 1, pp. 1–11, 2022, doi: 10.1155/2022/1828848.
[16] Puyush, "Real Time Accident Detection Dataset," Roboflow Universe. Accessed: Nov. 08, 2024. [Online]. Available: https://universe.roboflow.com/puyush-fipgg/real-time-accident-detection
[17] P. Zhu *et al.*, "Detection and Tracking Meet Drones Challenge," *IEEE Trans Pattern Anal Mach Intell*, vol. 44, no. 11, pp. 7380–7399, 2021, [Online]. Available: http://arxiv.org/abs/2001.06303
[18] N. Pethiyagoda, "Vehicle Dataset for YOLO," Dataset Ninja. Accessed: Nov. 08, 2024. [Online]. Available: https://datasetninja.com/vehicle-dataset-for-yolo
[19] Z. Song *et al.*, "Synthetic Datasets for Autonomous Driving: A Survey," *ArXiv*, vol. 2304.12205, pp. 1–19, 2024, [Online]. Available: http://arxiv.org/abs/2304.12205
[20] A. Kazemi, Q. ul ain Fatima, V. Kindratenko, and C. Tessum, "AIDOVECL: AI-generated Dataset of Outpainted Vehicles for Eye-level Classification and Localization," *ArXiv*, vol. 2410.24116, pp. 1–34, 2025, [Online]. Available: http://arxiv.org/abs/2410.24116
[21] N. Jegham, C. Y. Koh, M. Abdelatti, and A. Hendawi, "YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLO11, and Their Previous Versions," *ArXiv*, vol. 2411.00201v2, pp. 1–22, 2024, [Online]. Available: http://arxiv.org/abs/2411.00201
[22] A. F. Rasheed and M. Zarkoosh, "YOLOv11 Optimization for Efficient Resource Utilization," *ArXiv*, vol. 2412.14790, pp. 1–12, 2024, [Online]. Available: http://arxiv.org/abs/2412.14790
[23] R. Khanam and M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," *ArXiv*, vol. 2410.17725, pp. 1–9, 2024, [Online]. Available: http://arxiv.org/abs/2410.17725
[24] T. Jiang and Y. Zhong, "ODverse33: Is the New YOLO Version Always Better? A Multi Domain benchmark from YOLO v5 to v11," *ArXiv*, vol. 2502.14314v2, pp. 1–20, 2025, [Online]. Available: http://arxiv.org/abs/2502.14314
[25] M. A. R. Alif, "YOLOv11 for Vehicle Detection: Advancements, Performance, and Applications in Intelligent Transportation Systems," *ArXiv*, vol. 2410.22898, pp. 1–16, 2024, [Online]. Available: http://arxiv.org/abs/2410.22898
[26] R. Sapkota *et al.*, "YOLO Advances to Its Genesis: A Decadal and Comprehensive Review of the You Only Look Once (YOLO) Series," *ArXiv*, vol. 2406.19407v5, pp. 1–26, 2025, [Online]. Available: http://arxiv.org/abs/2406.19407v5
[27] N. Dahiya *et al.*, "Hyper-parameter Tuned Deep Learning Approach for Effective Human Monkeypox Disease Detection," *Sci Rep*, vol. 13, no. 1, pp. 15930–15947, 2023, doi: 10.1038/s41598-023-43236-1.
[28] Z. Ning, X. Wu, J. Yang, and Y. Yang, "MT-YOLOv5: Mobile Terminal Table Detection Model Based on YOLOv5," *J Phys Conf Ser*, vol. 1978, no. 1, pp. 1–9, 2021, doi: 10.1088/1742-6596/1978/1/012010.
[29] A. E. Maxwell, T. A. Warner, and L. A. Guillén, "Accuracy Assessment in Convolutional Neural Network-Based Deep Learning Remote Sensing Studies—Part 1: Literature Review," *Remote Sens (Basel)*, vol. 13, no. 13, pp. 209–220, 2021, doi: 10.3390/rs13132450.
[30] F. Liantoni and A. Agusti, "Forecasting Bitcoin using Double Exponential Smoothing Method Based on Mean Absolute Percentage Error," *JOIV: International Journal on Informatics Visualization*, vol. 4, no. 2, pp. 91–95, 2020.
[31] I. K. R. Pramana, "Vehicle-AccidentYoloApp," GitHub. Accessed: Jan. 08, 2025. [Online]. Available: https://github.com/rai-pramana/Vehicle-AccidentYoloApp
[32] Ultralytics Inc, "Ultralytics YOLO Docs," Ultralytics. Accessed: Jan. 08, 2025. [Online]. Available: https://docs.ultralytics.com/#how-can-ultralytics-yolo-be-used-for-real-time-object-tracking
[33] P. Skalski, "How to Estimate Speed with Computer Vision," Roboflow Blog. Accessed: Jan. 08, 2025. [Online]. Available: https://blog.roboflow.com/estimate-speed-computer-vision/

Authors' Profiles



I. Kadek Rai Pramana received his Bachelor of Computer Science degree in Information Technology from the Faculty of Engineering, Udayana University, Indonesia. His research interests include Data Analytics, Machine Learning, and Computer Vision. He has experience developing data-driven and computer vision applications using Python and related frameworks, including NumPy, OpenCV, Torch, Ultralytics, and Streamlit, as well as working with databases and web-based systems.



I. Putu Agung Bayupati received his Bachelor of Engineering degree in Electrical Engineering from Udayana University in 2001, Master of Engineering degree in Information Technology from Bandung Institute of Technology in 2006, and Ph.D. degree in Electrical Engineering and Computer Science from Kanazawa University in 2012. He joined the Faculty of Engineering at Udayana University as a lecturer in 2003. His research interests include intelligent signal processing, computer vision, and business analytics.



Gusti Made Arya Sasmita received his Bachelor of Engineering degree in Electrical Engineering from Udayana University in 1997 and his Master of Engineering degree in Electrical Engineering (Computer and Informatics Systems) from Gadjah Mada University, Yogyakarta, in 2003. He currently serves as a Lektor (Assistant Professor) at the Information Technology Study Program, Faculty of Engineering, Udayana University. His teaching areas include information systems, computer networks, information security, and network programming. His research interests focus on computer networks, cybersecurity, and information systems.



Ngoc Le received his Ph.D. degree in Communication and Information Technology from Kyungpook National University, South Korea. He is currently a Professor of Information Technology and Director of the Swinburne Innovation Space at Swinburne Vietnam – FPT University, where he leads the Intelligent Systems and Networks Research Group (ICISN). Since 2025, he has also served as Dean of the Faculty of Semiconductor and Artificial Intelligence at Asia University, Vietnam, and as Deputy Director of the Quantum AI Cybersecurity Institute (QACI) under FPT Corporation. His research interests include intelligent and embedded systems, communication networks, artificial intelligence, image and video processing, IoT, and big data analytics.

How to cite this paper: I. Kadek Rai Pramana, I. Putu Agung Bayupati, Gusti Made Arya Sasmita, Ngoc Le, "Accident Detection and Estimation of Vehicle Speed and Count by Type in Road CCTV Images Using Machine Vision", International Journal of Information Technology and Computer Science(IJITCS), Vol.18, No.2, pp.183-201, 2026. DOI:10.5815/ijitcs.2026.02.11