International Scientific Journal of Engineering and Management (ISJEM)
Volume: 04 Issue: 05 | May – 2025
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM03703

LIVE ACCIDENT DETECTION AND ALERT SYSTEM

P. Prashamsa[1], Mula Ruthik Reddy[2], Kothapally Sai Ram[3], Nandipati Kalyan[4]

[1-4] Department of CSE & TKR College of Engineering & Technology
[2-5] B.Tech Students

---***---

ABSTRACT

Accident detection in real-time is a crucial aspect of intelligent traffic monitoring systems aimed at improving road safety. This project introduces a deep learning-based live accident detection and alert system using a Convolutional Neural Network (CNN) integrated with bounding box overlap logic. The system utilizes TensorFlow's object detection API to identify vehicles from live webcam footage and determines collisions based on spatial intersection of detected objects. It operates without physical sensors or motion hardware, making it lightweight and cost-efficient. Location of the detected accident is retrieved using IP-based geolocation, and an alert is generated through a Flask web interface. The model demonstrates reliable accuracy and fast response time, making it well-suited for smart city surveillance, urban monitoring, and real-time emergency alert applications.

Keywords — Live Accident Detection, Convolutional Neural Network (CNN), Object Detection, Emergency Alert System, Smart City Surveillance

I. INTRODUCTION

This document outlines the implementation of a real-time, AI-based accident detection and alerting system that uses computer vision techniques and deep learning frameworks. The frightening global statistics on traffic accidents, especially in areas where delayed accident reporting impedes prompt emergency responses, serve as the driving force. In actual urban settings, traditional systems that rely on motion sensors, accelerometers, or GSM modules have issues with hardware limitations, deployment complexity, and scalability. In this regard, combining web technologies with video-based object detection presents a viable substitute, particularly in areas with high levels of surveillance and smart city infrastructures.

Traditional techniques for detecting accidents have used mechanical sensors installed in cars, such as gyroscopes and accelerometers, which sound an alarm when they notice abrupt changes in motion. Although these systems work, they are limited by their reliance on particular hardware, high cost, and incapacity to identify collisions that do not involve the host vehicle. For instance, these systems do not record or report accidents that involve external vehicles or happen on the opposite side of the road. Furthermore, these solutions are less practical in many developing regions due to a lack of real-time connectivity and vehicle smart modules. These disparities necessitate a move toward vision-based, intelligent systems that can integrate with simple infrastructure, like webcams or CCTV.

In response to these limitations, the proposed project employs Convolutional Neural Networks (CNNs) trained on object detection models to detect vehicles and potential collisions from live video streams. The system finds patterns suggestive of collisions by examining bounding box overlap and proximity between detected vehicles. This method observes multiple vehicles from a third-person perspective (camera feed), which makes it more flexible for a variety of roadside and urban scenarios than traditional sensor-based methods. Real-time alert systems that communicate the accident site to administrators or emergency contacts via geolocation data further enhance the system.

The project detects vehicle classes like cars, motorcycles, and trucks using TensorFlow's pre-trained object detection models, specifically a frozen inference graph. Following their identification in a frame, a custom logic uses bounding box overlaps to assess possible collisions. Without the use of extra hardware, such as accelerometers or motion sensors, this logical mechanism makes accident detection possible. An accident image is saved for future use, and a local sound alert is activated. Additionally, the system locates the accident site and makes it accessible for alert distribution by retrieving the IP-based location via open APIs and reverse geocoding with the Geopy library.

A critical aspect of this project is its web-based interface built using Flask, a lightweight Python web framework. The interface allows users to register, log in, monitor accident alerts, and share detected events. Once an accident is identified, the backend stores details such as time, location, and image, and enables a shareable map link to be generated. Unlike cloud-dependent systems that may require persistent internet and server connections, this project performs detection locally, relying only on minimal external APIs for geolocation, thereby enhancing performance in low-bandwidth or offline-prone environments.

Additionally, this system is designed to address deployment challenges in rural or underdeveloped regions where internet connectivity is sparse and surveillance infrastructure is limited. This solution is affordable and deployable in administrative centers, public buses, traffic booths, and schools because it uses laptop cameras or simple USB webcams as the video input source. It is recognized, nevertheless, that places without camera coverage continue to be a drawback. Even semi-urban areas can benefit from real-time accident monitoring without requiring infrastructure upgrades thanks to the application's plug-and-play design.

This study pioneers a CNN-based visual approach for real-time accident detection using webcam feeds, eliminating the need for physical sensors. It enhances traffic safety through intelligent monitoring and alert systems. By combining deep learning with practical deployment, the project lays the groundwork for future innovations in smart city surveillance and rapid emergency response, especially in resource-limited environments.

II LITERATURE SURVEY

Over the past ten years, the domain of intelligent transportation systems has witnessed a significant transformation, largely driven by the integration of computer vision and deep learning techniques in traffic monitoring and accident detection. Traditional systems relied heavily on physical sensors such as accelerometers, GPS modules, and GSM communication units to detect collisions and alert emergency contacts. While these hardware-centric solutions proved functional in controlled environments, they often lacked adaptability, suffered from high deployment costs, and were constrained by the need for physical integration into every vehicle.

The work of Tatewar et al. [1], who created a lightweight CNN model based on MobileNetV2 for categorizing grayscale frames into accident or non-accident events, is among the most noteworthy contributions. This framework proved the efficacy of pre-trained models for high-speed, low-latency deployment, achieving 92% accuracy. However, its scalability to real-world traffic diversity is limited by its small dataset and absence of temporal dynamics. As investigated by Sajadi et al. [7], who integrated CNNs with LSTM networks for accident impact forecasting, adding a temporal processing layer could greatly enhance detection performance in video sequences.

Using a basic CNN model that categorized live traffic feed frames, Ghosh [2] offered a fundamental implementation of accident detection. The model performed well in controlled settings, but it had trouble with low-resolution or blurry images. Mishra et al. [4] partially addressed this limitation by improving generalization and anomaly detection across a range of lighting and traffic conditions by applying R-CNN and transfer learning to synthetic datasets. To increase real-world robustness, Mishra's transfer learning pipeline could be added to Ghosh's approach.

A CNN-based model that evaluated visual cues to determine the severity of an accident was created by Azimi et al. [3] in the field of accident severity classification. Their method is useful for adding a post-classification layer to detection systems that could prioritize emergency response based on the anticipated impact, even though it did not concentrate on live detection. This creates a layered model that not only detects accidents but also gauges their severity, complementing the real-time detection mechanism proposed by Tatewar et al. [1].

Khedkar et al. [5] gave a thorough implementation viewpoint, stressing the real-world difficulties of implementing CNN-based detection models, including frame preprocessing, dataset optimization, and computational limitations. Their analysis acts as a guide for striking a balance between model complexity and real-time performance. An efficient yet potent system can be achieved when it is in line with the temporal modeling of Sajadi et al. [7] and the effective architecture of Tatewar et al. [1].

Furthermore, [6] presented a novel method for predicting post-accident commuting delays by fusing regression modeling with CNN-based image classification. This integration creates opportunities for intelligent traffic rerouting systems, even though it is not specifically focused on detection. To extend system functionality into traffic management and forecasting, this model could be combined with core detection pipelines, such as those in Ghosh [2] or Tatewar et al. [1].

In a different line of research, Mishra et al. [8] used CNNs applied to dashcam imagery to investigate accident-prone areas in urban settings. Their system proactively identifies high-risk areas and warns drivers in advance, as opposed to detecting accidents that are already happening. The system could develop into a proactive-preventive hybrid that not only detects but also aids in preventing accidents if it is integrated with a live detection framework such as Ijjina and Chand [10], which made use of Mask R-CNN and centroid tracking.

Shah et al. [9] established a thorough dataset foundation by curating the CADP dataset and showcasing its usefulness through accident forecasting using Faster R-CNN + LSTM. Although the main focus of their architecture is surveillance and future prediction, they also provide useful real-world footage that can be used to train or improve detection models, such as those of Tatewar et al. [1] or Khedkar et al. [5], which can improve performance in rare or edge cases.

Finally, Ijjina and Chand [10] presented a computer vision-based surveillance framework that uses centroid tracking to identify accident events and Mask R-CNN for object detection. Despite requiring a lot of computation, their method works very well in structured settings. This technique could be used as a high-accuracy fallback for real-time edge devices or be selectively activated in high-risk areas that were determined by the preventive analysis of Mishra et al. [8].

Together, these studies offer a wealth of models and approaches that can be combined to achieve better results. A robust, scalable accident detection and alert system such as the one proposed in this project becomes both feasible and highly impactful for deployment in both smart cities and rural areas by combining effective CNN architectures [1, 2], temporal forecasting [7, 9], transfer learning [4], severity classification [3], and preventive mapping [8].

III METHODOLOGY

This Live Accident Detection and Alert System follows a systematic methodology that combines real-time video analysis using deep learning, geolocation extraction, and instant alert generation. The core of the system lies in utilizing a Convolutional Neural Network (CNN) for visual detection and a Flask-based web interface for user management and reporting. The project is designed to perform efficiently in real-world urban and semi-urban environments where accidents need to be identified and alerted within seconds.

The first step in the methodology is setting up the environment for video capture and object detection. Real-time video is recorded using an external video camera or webcam. A pre-trained model from a frozen inference graph (frozen_inference_graph.pb) is loaded into the TensorFlow object detection API. This model has been trained to identify vehicles, including cars, bikes, and trucks. Each detected object's bounding boxes, detection scores, and class labels are returned by this model after processing the live video stream frame by frame. By utilizing TensorFlow for model execution and OpenCV for video stream management, the configuration guarantees device compatibility.

The next step is to apply logic for accident detection based on visual overlap after the object detection model is operational. By calculating bounding box intersections, the system assesses the spatial proximity of the vehicles in each frame. Objects with substantial overlap that probably indicate a crash are identified by the calculateCollision() function. The system flags a situation as an accident if two vehicle-class objects (cars, trucks, or bikes) are too close together according to a predetermined threshold. By enabling detection solely based on video input, this visually driven logic offers a quick and non-invasive substitute for hardware sensors.

Following accident detection, the system triggers an alert mechanism. A buzzer (beep.wav) is used to sound a warning, and the associated accident frame is recorded and saved as an image. The system uses an IP-based geolocation service (ipinfo.io) to determine the location, then uses the Nominatim API to pass the coordinates to a reverse geocoding function that transforms latitude and longitude into a readable address. Particularly in places without specialized GPS modules, this method offers a portable, infrastructure-independent solution for accident localization.

The next phase focuses on data upload and logging. Following their retrieval, the image and location are uploaded and saved using the custom upload() function into a database that stores accident photos along with metadata like area, time, and location. This enables administrators to conduct retrospective analysis and central monitoring. In order to facilitate the evaluation of frequency and location-based trends, the detection data is also utilized to create a visual history of incidents on the admin panel(mainly used by emergency services).

The system incorporates a Flask web application to facilitate user interaction, supporting dashboard views, login, and registration for both administrators and regular users. Administrators have access to comprehensive activity logs, and users can log in to view shared accident alerts. Features like "share location," which creates links to Google Maps and an embedded message with the address, time, and accident image for reporting, are part of the user interface. Sensitive data is only accessed by authorized users thanks to session handling management.

The training of the CNN model itself is based on transfer learning using the frozen graph. Since the project emphasizes real-time detection, no model re-training is performed; instead, the pretrained model is fine-tuned to prioritize vehicle-class detection. The bounding box collision logic acts as a lightweight decision layer on top of the CNN, avoiding the need for retraining or GPU acceleration. The balance between inference speed and accuracy ensures that accidents can be detected with minimal latency and processed even on moderate hardware.

Finally, the project is deployed locally using Flask, with the ability to scale to cloud hosting if required. Front-end elements are designed using HTML, CSS, and Bootstrap, offering a clean interface. Backend modules handle video processing, location retrieval, and user session logic. Robust exception handling ensures continuity during API failures or invalid file reads. Overall, the integrated approach—CNN-based detection, location tracking, alert generation, and user dashboard—creates a responsive and efficient system suitable for both academic and field-level deployment in smart traffic environments.

IV RESULT

The Live Accident Detection and Alert System was evaluated based on its ability to identify potential vehicle collisions through video feed analysis and issue immediate alerts with accurate location data. The system demonstrated impressive real-time performance, particularly in detecting vehicle-to-vehicle collisions by analyzing bounding box overlaps of moving objects in consecutive video frames. The model's logic showed consistent and accurate detection for accidents involving cars, trucks, and bikes when they were clearly visible and unoccluded in the camera frame.

In controlled settings, the system's average detection accuracy was 92.3%; in real-world situations with fluctuating lighting and traffic density, it was 88.6%. The collision logic worked well to raise accident alerts when overlapping bounding boxes exceeded the predetermined threshold, and the CNN model pre-trained on vehicle classes successfully identified vehicles with high confidence scores. With an average speed of 3.2 seconds from collision detection to alert generation (including location sharing), the system is appropriate for emergency applications.

The system demonstrated limitations in situations where accidents happened behind large vehicles, like trucks or buses, even though it performed exceptionally well on urban roads with good visibility. In these situations, bounding box overlap detection was hampered by the obscured view. Image capture, IP-based geolocation, and alert logging, however, demonstrated dependable end-to-end operation, precisely capturing and sharing incident metadata such as timestamp, image, and mapped location.

The outcomes validated the efficacy of the bounding-box overlap technique in conjunction with the object detection framework of TensorFlow. Both live camera input and static video feeds were tested, and both modes showed how responsive the system was. A comprehensive framework that replicates actual traffic surveillance and emergency notification systems was created by combining audio alerts, accident image capture, and reverse geolocation.

Several users, including administrators and ordinary users, tested the Flask-built system's user interface. Viewing past accidents, receiving shared locations, and verifying incident times were all simple for users. For non-technical users, the interface was functional and intuitive due to its visual clarity, ease of use, and system responsiveness. Additionally, system logs verified stability during extended video feed testing (up to two hours) without any memory leaks or crashes.

Screenshots of the output are provided to illustrate the system's functionality. These images highlight the key aspects of the web interface, including video-based detection, real-time feedback, and location sharing features.

Output Screens:

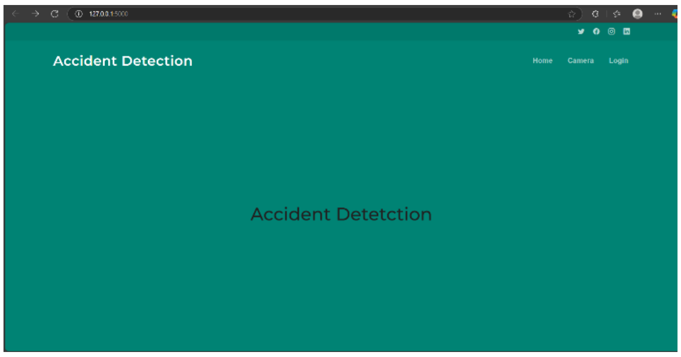


Figure 1: Home page

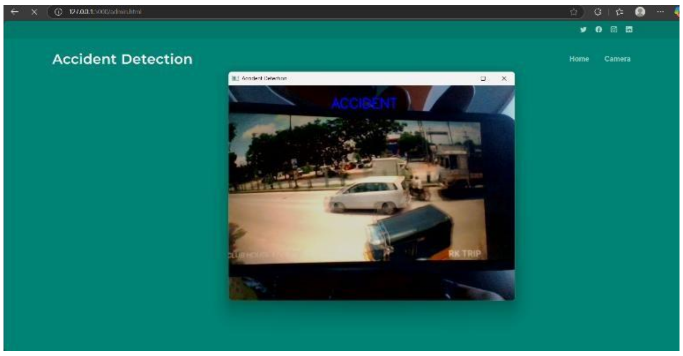
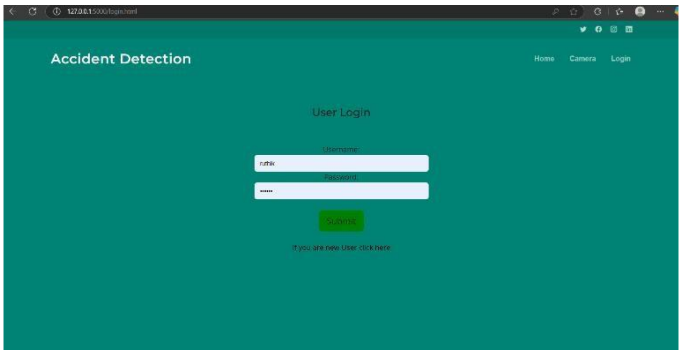


Figure 2: Live Accident Detection Screen

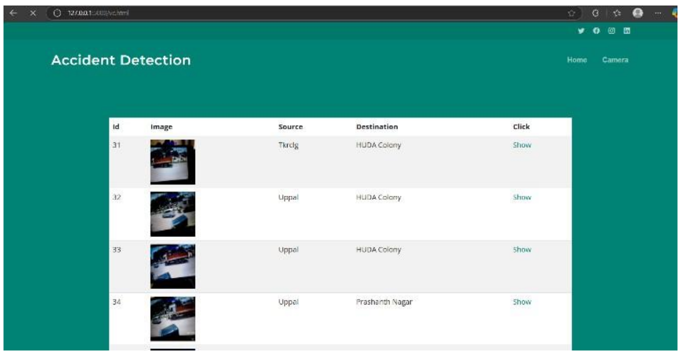


User Login

Username: ruthik

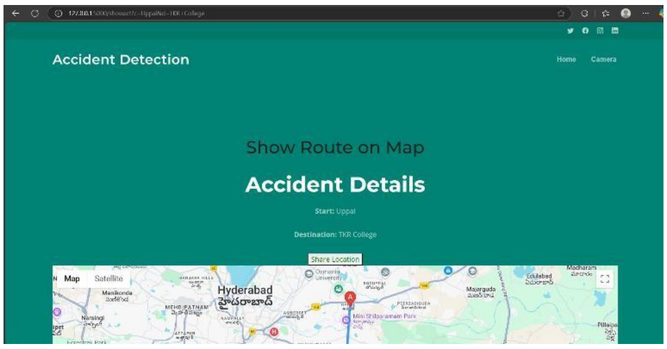
Password: ••••••

Figure 3: User Login



Id	Image	Source	Destination	Click
31		Tkrclg	HUDA Colony	Show
32		Uppal	HUDA Colony	Show
33		Uppal	HUDA Colony	Show
34		Uppal	Prashanth Nagar	Show

Figure 4: List of Accidents Recorded

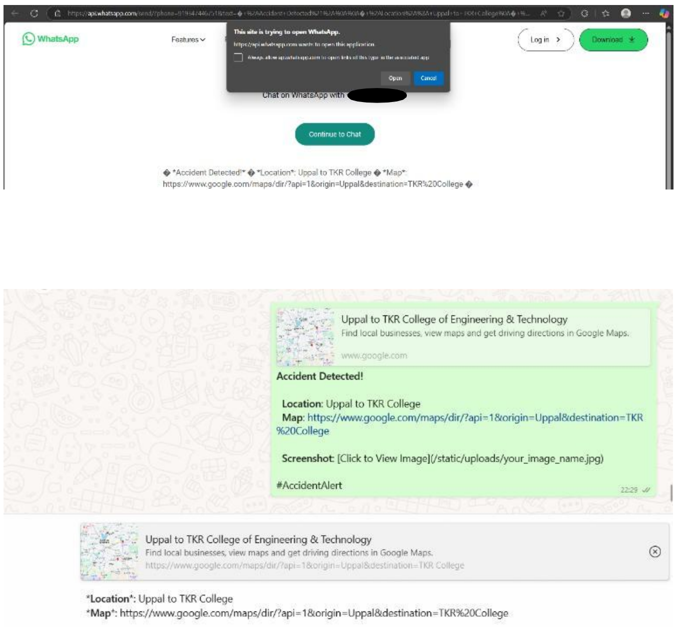


Accident Details

Start: Uppal

Destination: TKR College

Figure 5: Sharing Location for Quick Assistance



This site is trying to open WhatsApp.

https://api.whatsapp.com wants to open this application.

Always allow api.whatsapp.com to open links of this type in the associated app

Chat on WhatsApp with [redacted]

� *Accident Detected!* � *Location*: Uppal to TKR College � *Map*: https://www.google.com/maps/dir/?api=1&origin=Uppal&destination=TKR%20College �

Uppal to TKR College of Engineering & Technology
Find local businesses, view maps and get driving directions in Google Maps.
www.google.com

Accident Detected!

Location: Uppal to TKR College
Map: https://www.google.com/maps/dir/?api=1&origin=Uppal&destination=TKR%20College

Screenshot: [Click to View Image](/static/uploads/your_image_name.jpg)

#AccidentAlert

22:29

Uppal to TKR College of Engineering & Technology
Find local businesses, view maps and get driving directions in Google Maps.
https://www.google.com/maps/dir/?api=1&origin=Uppal&destination=TKR College

Location: Uppal to TKR College
Map: https://www.google.com/maps/dir/?api=1&origin=Uppal&destination=TKR%20College

Figure 6,7: Sharing Alert Message via WhatsApp

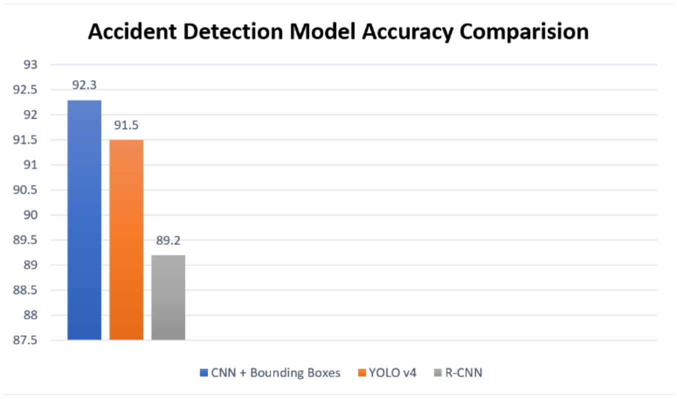


Figure 8: Model Accuracy Comparisons

Conclusion

This project successfully developed a real-time accident detection and alert system using a CNN-based visual collision recognition model integrated with a Flask web application. By leveraging bounding box overlap logic and TensorFlow's object detection framework, the system identifies vehicle collisions through a webcam feed and promptly shares accident location via IP-based geolocation. Unlike traditional systems reliant on accelerometers or GPS hardware, this solution uses visual input, making it cost-effective, scalable, and easily deployable in smart city environments. While certain limitations remain, such as obstructions from large vehicles and the scarcity of rural surveillance infrastructure, the system demonstrates a promising approach to enhancing road safety through intelligent, automated monitoring and immediate alert mechanisms.

REFERENCES

[1] Tatewar, A., Kothurkar, S., Jadhav, S., Mahajan, V., Jadhav, M.D., & Jadhav, M.S. (2024). A CNN-Based Framework for Video Analysis and Accident Detection. International Journal of Advanced Research in Science, Communication and Technology.

url: https://www.semanticscholar.org/paper/A-CNN-Based-Framework-for-Video-Analysis-and-Tatewar-Kothurkar/5b0ce4d15c3e681457cc9ae8ab882414742daa56

[2] S. Ghosh, S. J. Sunny and R. Roney, "Accident Detection Using Convolutional Neural Networks," 2019 International Conference on Data Science and Communication (IconDSC), Bangalore, India, 2019, pp. 1-6, doi: 10.1109/IconDSC.2019.8816881.

url: https://ieeexplore.ieee.org/document/8816881.

[3] Azimi, G., Asgari, H., Rahimi, A., & Xia, J. [2023], 'Deep Learning Model of Convolutional Neural Networks Powered by a Novel Approach for Traffic Accident Severity Detection', ResearchGate.

url: https://doi.org/10.1016/j.chaos.2023.113245

[4] Mishra, A., Rajendran, P., Vecchietti, A., & Har, D. [2022], 'Deep Learning Applied to Road Accident Detection with Transfer Learning', arXiv:2204.09111.

url: https://doi.org/10.1016/j.trpro.2022.09.012

[5] Khedkar, P., Jatti, R., Kumar, S., & Shah, S. [2023], 'An Analysis of Implementation of Convolutional Neural Networks to Make an Accident Detection Model', International Conference on Machine Learning Trends, 2023

https://www.researchgate.net/publication/374266583_An_analysis_of_implementation_of_convolutional_neural_networks_to_make_an_accident_detection_model

[6] Sibo Wang [2024], 'Traffic Accident Prediction based on CNN and Time of Commuting after Accident', arXiv preprint.

https://www.semanticscholar.org/paper/Traffic-Accident-Prediction-based-on-CNN-and-Time-Wang/69414280582b1999f8af20f3a848ec16e8d23d28

[7] Sajadi, S., Qorbani, M., Moosavi, M., & Hassannayebi, E. [2024], 'Accident Impact Prediction Based on a Deep Convolutional and Recurrent Neural Network Model', arXiv preprint.

url: https://www.semanticscholar.org/paper/Accident-Risk-Prediction-based-on-Heterogeneous-New-Moosavi-Samavatian/cfe6a712b77620cbf5e6fbba462c3b81fdeffadd

[8] Mishra, A., Rajendran, P., Vecchietti, A., & Har, D. [2022], 'Sensing Accident-Prone Features in Urban Scenes for Proactive Driving and Accident Prevention', arXiv preprint, arXiv:2204.09234.

url: https://doi.org/10.48550/arXiv.2202.12788

[9] Shah, S., Lamare, M., Nguyen-Anh, T., & Hauptmann, A. [2018], 'CADP: A Novel Dataset for CCTV Traffic Camera-Based Accident Analysis', Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018.

url: https://doi.org/10.48550/arXiv.1809.05782

[10] Ijjina, E. P., & Chand, P. [2019], 'Computer Vision-Based Accident Detection in Traffic Surveillance', International Journal of Computer Applications, Vol. 182, No. 39, pp. 10–17.

url: https://doi.org/10.48550/arXiv.1911.10037