VEHICLE COLLISION DETECTION AND ALERT SYSTEM USING YOU ONLY LOOK ONCE ALOGITHAM

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering degree in Computer Science and Engineering

By

K. CHAITANYA (Reg No - 39110547)

P. NEERAJ GOUD (Reg No - 39110739)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600119

APRIL - 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **K. Chaitanya (Reg No- 39110547) and P. Neeraj Goud (Reg No-39110739)** who carried out the Project Phase- 2 entitled **"VEHICLE COLLISION DETECTION AND ALERT SYSTEM USING YOU ONLY LOOK ONCE ALGORITHM"** under my supervision from January 2023 to April 2023.

Internal Guide

Dr. E. Srividya

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on 20.04.2023

Internal Examiner **External Examiner**

DECLARATION

I, **K. Chaitanya (Reg No-39110547),** hereby declare that the Project Phase-2 Report entitled **"VEHICLE COLLISION DETECTION AND ALERT SYSTEM USING YOU ONLY LOOK ONCE ALGORITHM"** done by me under the guidance of **Dr. E. Srividhya,** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 20.04.23

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D**, **Dean**, School of Computing, and **Dr.L. Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep gratitude to my Project Guide Dr. E. Srividhya, for her valuable guidance, suggestions, and constant encouragement paved the way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The vehicle collision detection and alert system using YOLOv3 is a state-of-the-art technology that utilizes advanced computer vision techniques to improve road safety. In recent years, road accidents have become a major concern for transportation authorities worldwide, and the need for effective accident prevention systems has become critical. This system addresses this issue by using YOLOv3, a popular deep-learning algorithm for object detection, to identify vehicles, pedestrians, and other objects on the road. The system is composed of several components, including a camera mounted on the vehicle, a YOLOv3 model, and a collision detection algorithm. The camera captures real-time video footage of the road ahead, which is then processed using YOLOv3 to detect and classify objects. The collision detection algorithm analyzes the data from YOLOv3 and assesses the risk of collision based on the speed and direction of the objects on the road. If the system detects a potential collision, it will alert the driver through visual or audible alerts. For instance, the driver may receive a warning message on the dashboard or an alarm sound to alert them to take action to avoid the collision. The system can also apply automatic braking if necessary to prevent a collision. This technology has several advantages over traditional collision detection systems, including high accuracy, real-time processing, and low computational requirements. The YOLOv3 algorithm is highly efficient and can process video footage at a high speed, making it suitable for real-time applications. Moreover, the system can be easily integrated into existing vehicles without significant modifications, making it a cost-effective solution for improving road safety. Overall, the vehicle collision detection and alert system using YOLOv3 is a promising technology that has the potential to reduce the number of accidents on the road and save lives.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	DEFINITION
1	CNN	CONVOLUTION NEURAL NETWORK
2	NN	NEURAL NETWORK
3	AI	ARTIFICIAL INTELLIGENCE
4	ML	MACHINE LEARNING
5	DL	DEEP LEARNING
6	GPU	GRAPHICS PROCESSOR UNIT
7	OpenCV	OPEN-SOURCE COMPUTER VISION LIBRARY

CHAPTER – 1

INTRODUCTION

1.1 INTRODUCTION

Vehicles are an important way of transportation all over the world. There are many cases of road accidents every day in the world. A traffic collision, also called a motor vehicle collision, car accident or car crash, occurs when a vehicle collides with another vehicle, pedestrian, animal, road debris, or other stationary obstruction, such as a tree, pole or building. Traffic collisions often result in injury, disability, death, and property damage as well as financial costs to both society and the individuals involved. Road transport is the most dangerous situation people deal with on a daily basis, but casualty figures from such incidents attract fewer media attention than other, less frequent types of tragedy.

A number of factors contribute to the risk of collisions, including vehicle design, speed of operation, road design, weather, road environment, driving skills, impairment due to alcohol or drugs, and behavior, notably aggressive driving, distracted driving, speeding, and street racing.

In 2013, 54 million people worldwide sustained injuries from traffic collisions.[2] This resulted in 1.4 million deaths in 2013, up from 1.1 million deaths in 1990.[3] About 68,000 of these occurred in children less than five years old.[3] Almost all high-income countries have to decrease death rates, while the majority of low-income countries have increasing death rates due to traffic collisions. Middle-income countries have the highest rate with 20 deaths per 100,000 inhabitants, accounting for 80% of all road fatalities with 52% of all vehicles. While the death rate in Africa is the highest (24.1 per 100,000 inhabitants), the lowest rate is to be found in Europe (10.3 per 100,000 inhabitants).

1.2 REASONS FOR VECHILE COLLISION:

Road incidents are caused by a large number of human factors such as failing to act according to weather conditions, road design, signage, speed limits, lighting conditions, pavement markings, and roadway obstacles. A 1985 study by K. Rumar, using British and American crash reports as data, suggested 57% of crashes were due

solely to driver factors, 27% to the combined roadway and driver factors, 6% to the combined vehicle and driver factors, 3% solely to roadway factors, 3% to combined roadway, driver, and vehicle factors, 2% solely to vehicle factors, and 1% to combined roadway and vehicle factors. Reducing the severity of injury in crashes is more important than reducing incidence and ranking incidence by broad categories of causes is misleading regarding severe injury reduction. Vehicle and road modifications are generally more effective than behavioral change efforts with the exception of certain laws such as the required use of seat belts, motorcycle helmets, and graduated licensing of teenagers.

Here are some common reasons for vehicle collisions:

1. Driver distraction: Distractions such as using a mobile phone, eating, applying makeup, or talking to passengers can divert a driver's attention from the road, causing them to miss traffic signs, signals, or other hazards, resulting in a collision.
2. Impaired driving: Alcohol, drugs, and medications can impair a driver's judgment, vision, and reaction time, making it difficult for them to navigate the road safely.
3. Speeding: Driving above the speed limit or too fast for the road and weather conditions can cause a driver to lose control of the vehicle, resulting in a collision.
4. Reckless driving: Driving aggressively, weaving in and out of traffic, or ignoring traffic signals and signs can lead to collisions.
5. Weather conditions: Adverse weather conditions such as heavy rain, fog, snow, and ice can make it difficult for drivers to see the road, resulting in collisions.
6. Vehicle defects: Malfunctioning brakes, tires, and other components can cause a driver to lose control of the vehicle and result in a collision.
7. Road conditions: Poor road conditions such as potholes, uneven surfaces, and debris can cause a driver to lose control of the vehicle and result in a collision.
8. Pedestrian and cyclist accidents: Collisions involving pedestrians and cyclists are often caused by drivers failing to yield, distracted driving, or impaired driving.

Fig 1.1 Reasons for Vehicle collisions

1.3 Human Factors:

Human factors in vehicle collisions include anything related to drivers and other road users that may contribute to a collision. Examples include driver behavior, visual and auditory acuity, decision-making ability, and reaction speed.

A 1985 report based on British and American crash data found driver error, intoxication, and other human factors contribute wholly or partly to about 93% of crashes. A 2019 report from the U.S. The National Highway Traffic Safety Administration found that leading contributing factors for fatal crashes included driving too fast for conditions or in excess of the speed limit, operating under the influence, failure to yield right of way, failure to keep within the proper lane, operating a vehicle in a careless manner, and distracted driving.

Drivers distracted by mobile devices had nearly four times greater risk of crashing their cars than those who were not. Research from the Virginia Tech Transportation Institute has found that drivers who are texting while driving are 23 times more likely to be involved in a crash as non-texting driver. Dialing a phone is the most dangerous

distraction, increasing a drivers' chance of crashing by 12 times, followed by reading or writing, which increases the risk by ten times.

Driver behavior is a significant factor in the occurrence of vehicle collisions. Drivers should follow traffic rules and regulations, maintain a safe speed, avoid distractions, and avoid driving while impaired.

Education and awareness campaigns can help raise awareness about the dangers of reckless driving, the importance of defensive driving, and the need to follow traffic rules and regulations. This can help reduce the number of collisions caused by human error.

Drivers must receive adequate training to develop the necessary skills to operate a vehicle safely. Training programs should cover defensive driving techniques, safe driving practices, and vehicle maintenance.

The development and deployment of advanced technologies such as collision detection and avoidance systems, lane departure warning systems, and automated emergency braking systems can help prevent collisions caused by human error.

Environmental factors such as poor weather conditions, visibility, and road conditions can increase the risk of collisions. Drivers should adjust their driving behavior to the conditions and take appropriate precautions.

Psychological factors such as stress, fatigue, and emotional state can affect a driver's ability to operate a vehicle safely. Drivers should take breaks, manage stress, and seek support if necessary.

To avoid such instances, our suggested or proposed system uses OpenCV and Python to detect vehicle collisions in real-time to avoid accidents and life-threatening situations.

1.4 OpenCV (OPEN-SOURCE COMPUTER VISION LIBRARY):

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify an image pattern their it's various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python, and Java interfaces and supports Windows, Linux, Mac OS, iOS , and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

The library is written in C++ and has bindings for Python, Java, and MATLAB, making it accessible to a wide range of developers. OpenCV provides a vast range of functionalities for image and video processing, including image and video capture, image filtering, feature detection and matching, object detection, segmentation, optical flow, camera calibration, stereo vision, and machine learning.

OpenCV has become a standard tool in the computer vision and robotics communities due to its versatility and high performance. It is widely used in applications such as surveillance, robotics, self-driving cars, medical imaging, augmented reality, and 3D reconstruction.

Applications of OpenCV: OpenCV is a powerful and versatile computer vision library that has a wide range of applications across various industries. Some of the most common applications of OpenCV are

1. Object detection and tracking: OpenCV provides a range of algorithms for object detection and tracking, including Haar cascades, HOG (Histogram of Oriented Gradients), and deep learning-based methods like YOLO (You Only Look Once). These algorithms are widely used in surveillance, robotics, and self-driving cars.
2. Facial recognition: OpenCV provides tools and algorithms for facial detection, recognition, and analysis. This technology is used in security systems, mobile apps, and social media platforms.
3. Augmented reality: OpenCV provides tools for detecting and tracking objects in real time, which is a critical component of augmented reality applications. It is used in gaming, advertising, and education.
4. Medical imaging: OpenCV is used in medical imaging applications for image segmentation, feature detection, and classification. It is used in detecting

diseases like cancer, analyzing medical images, and developing surgical robots.

5. Robotics: OpenCV is used in robotics applications for object detection, localization, and mapping. It is used in developing robots for industrial automation, agriculture, and healthcare.
6. Autonomous vehicles: OpenCV is used in developing autonomous vehicles for object detection and tracking, lane detection, and obstacle avoidance.

OpenCV Library Modules

OpenCV is a library of programming functions mainly aimed at real-time computer vision. It provides a range of modules that can be used for various computer vision tasks. Here are some of the main modules in the OpenCV library:

Core module: This module provides basic data structures and functions used by other modules in OpenCV. It includes data types like Mat, Point, and Size, as well as basic image processing functions.

Image Processing module: This module provides various functions for image processing and manipulation, such as image filtering, thresholding, and edge detection.

Video module: This module provides functions for video processing and analysis, such as video capture, frame-by-frame analysis, and video compression.

Object Detection module: This module provides algorithms for detecting objects in images and videos, such as Haar cascades, HOG, and deep learning-based methods like YOLO.

Machine Learning module: This module provides functions for training and using machine learning models, such as SVM (Support Vector Machine) and KNN (K-Nearest Neighbors).

OpenCV Contrib module: This module provides additional functions and algorithms contributed by the OpenCV community. It includes features like text detection and recognition, facial landmark detection, and augmented reality.

1.5 Vehicle Collision Detection Using OpenCV:

Video surveillance is widely used in security surveillance, military navigation, intelligent transportation, etc. Its main research fields are pattern recognition, computer vision and artificial intelligence. This article uses OpenCV to detect and track vehicles, and monitors by establishing an adaptive model on a stationary background. Compared with traditional vehicle detection, it not only has the advantages of low price, convenient installation and maintenance, and wide monitoring range but also can be used on the road. The intelligent analysis and processing of the scene image using camshift tracking algorithm can collect all kinds of traffic flow parameters (including the number of vehicles in a period of time) and the specific position of vehicles at the same time, so as to solve the vehicle offset. It is reliable in operation and has high practical value.

CHAPTER – 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

[1] Real-time object detection is a critical aspect of many applications, including unmanned vehicles. With the advances in machine learning and artificial intelligence, it is now possible to perform real-time object detection using state-of-the-art algorithms. One such algorithm is Mobile Nets, which is a lightweight and efficient neural network architecture designed specifically for mobile and embedded devices. The algorithm is optimized for performance and power efficiency, making it an ideal choice for running on the NVIDIA Jetson TX2 GPU platform.

The NVIDIA Jetson TX2 is a powerful and energy-efficient embedded platform that is specifically designed for applications that require deep learning and computer vision capabilities. The platform is equipped with a GPU that is specifically optimized for running neural networks, making it an ideal choice for real-time object detection applications. By running the Mobile Nets algorithm on the Jetson TX2, it is possible to perform object detection in real-time with high accuracy and efficiency.

The combination of Mobile Nets and the NVIDIA Jetson TX2 platform makes it possible to perform real-time object detection onboard an unmanned vehicle. This capability is particularly useful in emergency situations where time is of the essence, and decisions need to be made quickly. By processing video feedback in real-time, the algorithm can generate detections and feed them into a decision support warning system, enabling the system to alert operators of potential dangers and hazards.

In conclusion, real-time object detection using the Mobile Nets algorithm and the NVIDIA Jetson TX2 platform is a powerful tool for unmanned vehicle applications. The combination of performance and power efficiency makes it an ideal choice for running onboard an unmanned vehicle, enabling real-time detection and decision support during emergencies. With the continued advances in machine learning and AI, we can

expect to see even more powerful and efficient algorithms in the future, further enhancing the capabilities of unmanned vehicles in a range of applications.

[2] Car crashes are a leading cause of injury and death worldwide, and early detection of car crashes can significantly reduce the number of fatalities and injuries. To address this issue, a car crash detection system based on an ensemble deep learning model has been proposed. The proposed system uses video and audio data from dashboard cameras to detect car crashes with high accuracy.

The ensemble deep learning model is a combination of three base classifiers, one for video data and two for audio data. The base classifier for video data is a combination of a GRU-based and a CNN-based classifier. The GRU-based classifier processes the sequential nature of the video data, while the CNN-based classifier extracts spatial features from the video frames. The base classifiers for audio data are also a combination of a GRU-based and a CNN-based classifier. The GRU-based classifier processes the sequential nature of the audio data, while the CNN-based classifier extracts spectral features from the audio signals.

The proposed system establishes state-of-the-art classification performance on a dataset of car crash and non-crash events. The system achieves an accuracy of 96.7%, with a precision of 98.3% and a recall of 95.1%. The system also performs well on a variety of metrics, including F1 score, area under the ROC curve, and area under the precision-recall curve. The proposed system outperforms previous state-of-the-art systems, which rely on either video or audio data alone.

In addition to its high accuracy, the proposed system has several advantages over existing car crash detection systems. It can operate in real-time and can be integrated with existing dashboard camera systems. The system can also be easily deployed in a variety of environments, making it useful for a range of applications.

In conclusion, the proposed car crash detection system based on an ensemble deep learning model is a significant contribution to the field of car crash detection. Its high accuracy and real-time capabilities make it an excellent tool for reducing the number of fatalities and injuries caused by car crashes. The system's ability to integrate with

existing dashboard camera systems and its easy deployment in various environments make it a valuable addition to the field of car crash detection.

[3] The development of accurate and efficient deep learning models for image classification tasks is of great importance in a variety of fields. In this regard, a new deep-learning-based model comprising of ResNext architecture with Senet blocks has been proposed. This model has been designed to enhance the performance of existing baseline models, including VGG16, VGG19, Resnet50, and stand-alone ResNext, by improving the accuracy of image classification tasks while reducing computational overhead.

The proposed model combines the strengths of ResNext architecture and Senet blocks to create a more efficient and accurate model. The ResNext architecture is used to extract features from input images, while the Senet blocks are used to improve the quality of the features extracted. The proposed model achieves a ROC-AUC of 0.91, outperforming all existing baseline models. Furthermore, the proposed model achieves this performance while using significantly less proportion of the GTACrash synthetic data for training, thus reducing the computational overhead.

The proposed model's superior performance is due to the combination of ResNext architecture and Senet blocks, which enables the model to extract more meaningful features from input images. The ResNext architecture uses a multi-branch structure to extract features from input images, while the Senet blocks enhance the quality of the features extracted. This combination enables the model to classify images with high accuracy, even when using a small proportion of the training data.

The proposed model has significant implications for image classification tasks in various fields, including medical imaging, remote sensing, and autonomous vehicles. The model's high accuracy and low computational overhead make it an attractive option for these applications. The model's ability to outperform existing baseline models while using a small proportion of the training data can also significantly reduce the cost of training deep learning models.

In conclusion, the proposed deep-learning-based model comprising of ResNext architecture with Senet blocks is a significant contribution to the field of image classification. The model's ability to outperform existing baseline models while using a

small proportion of the training data can significantly reduce the computational overhead and training cost. The model's superior performance and efficiency make it an attractive option for various image classification tasks.

[4] The proposed DeepCrash system is a deep learning-based Internet of Vehicles (IoV) system that includes an in-vehicle infotainment (IVI) telematics platform, a vehicle self-collision detection sensor, and a front camera. The system aims to improve road safety by detecting traffic collisions and providing emergency-related announcements to the driver in real-time. The system achieved a high accuracy rate of 96% in traffic collision detection and has an average response time of approximately
7 seconds for emergency-related announcements. The system's cloud-based architecture enables it to be easily integrated into existing IoV infrastructures, and it has potential applications in the automotive industry, including advanced driver assistance systems and autonomous vehicles. The proposed DeepCrash system is a significant contribution to the field of IoV, with promising implications for the future of road safety.

[5] This paper proposes a methodology to develop a reliable and computationally inexpensive real-time automatic accident detection system with minimal hardware requirements. The proposed detection stage uses Mini-YOLO, a deep learning model architecture trained using knowledge distillation, with reduced model size and computational overhead compared to its counterpart, YOLO (You-Only-Look-Once). Mini-YOLO achieves an average precision (AP) score of 34.2 on the MS-COCO dataset, outperforming other detection algorithms in runtime complexity. The proposed system achieves a staggering 28 frames per second on a low-end machine, making it an efficient solution for real-time accident detection. The use of knowledge distillation enables the transfer of knowledge from a larger network to a smaller network, reducing computational overhead without sacrificing accuracy. The proposed system's efficiency and reliability make it a promising solution for real-time accident detection with minimal hardware requirements.

[6] This paper proposes a new approach to estimate collision priority for vehicles on the road based on a vision system. The approach takes into account the perspective of an ego vehicle equipped with either a vision-based driver-assist system or a fully

autonomous vehicle. The proposed method is heuristic and unimodal, and it performs well for input videos. By combining it with other quantitative semantics of traffic parameters, a more robust estimate can be achieved. The proposed approach is a promising step towards improving safety on the road, especially in the context of autonomous vehicles. By estimating collision priority accurately, the approach can help vehicles make more informed decisions and avoid potential accidents. Overall, the proposed approach has the potential to significantly contribute to the development of safer and more efficient transportation systems.

[7] In recent years, vehicle collision warning systems on mobile devices have become increasingly popular as they aim to enhance driver safety by alerting drivers about potential collisions. To achieve this, reliable and accurate vehicle detection is a critical step. This paper proposes a vision-based vehicle detection system using deep learning approaches specifically designed for mobile platforms with cameras mounted on the vehicle. The paper highlights the benefits of using deep learning techniques in vehicle detection and how integrating detection with tracking can improve accuracy and reliability. By leveraging deep learning, the proposed system achieves high detection accuracy in real-time while running on mobile platforms with limited computational resources. The results suggest that the proposed system is a promising solution for vehicle collision warning systems on mobile devices, which can ultimately contribute to the improvement of road safety.

[8] The ability of deep learning models to capture both long-term and short-term dependencies has led to their extensive use in various fields, including anomaly detection. In this paper, the authors propose an ensemble of deep learning models, including MLP ensemble, RFC ensemble, DNN, GRU, and LSTM, to detect anomalies in a dataset. The models are compared based on their area under the curve (AUC) of the ROC curve, with MLP ensemble achieving the highest AUC of 97.2% followed by RFC ensemble, DNN, GRU, and LSTM.

The authors also use a data balancing technique to improve the detection performance of the models. By combining the data balancing technique with the ensemble of deep learning models, the detection performance is significantly improved. The results demonstrate the effectiveness of deep learning models in anomaly detection and the importance of combining multiple models for improved accuracy.

[9] In this paper, a deep learning approach for automatic detection and localization of road accidents has been proposed, which formulates the problem as anomaly detection. The method uses a one-class classification approach and applies spatiotemporal autoencoder and sequence-to-sequence long short-term memory autoencoder for modeling spatial and temporal representations in the video. The proposed model has been evaluated on real-world video traffic surveillance datasets and has achieved significant results both qualitatively and quantitatively. The model can detect and localize various types of road accidents with high accuracy, which can help reduce emergency response time and prevent further accidents. This approach has great potential for improving the safety of road transportation and can be used for developing advanced driver assistance systems and autonomous vehicles.

[10] This paper presents a method for detecting vehicles in nighttime images using a fine-tuned YOLO v3 network. The network was trained on enhanced images to improve its performance and outperformed two popular object detection methods, Faster R-CNN and SSD, in terms of precision and detection efficiency. The proposed method achieved an average precision of 93.66%, which is significantly higher than the other two methods. The high precision rate of the proposed method can be attributed to the fine-tuning process, which allowed the network to learn the specific characteristics of nighttime images. The method can be applied to various applications such as autonomous driving, traffic monitoring, and surveillance systems, where accurate and efficient detection of vehicles is crucial. The proposed method provides a reliable and efficient solution for detecting vehicles in nighttime conditions.

[11] The proposed vehicle collision detection system is an important step towards improving road safety, particularly in the context of autonomous vehicles. The use of deep learning algorithms, such as YOLO v3, allows for real-time processing of video data captured by an onboard camera, enabling the system to quickly detect potential collisions and take appropriate action. The study demonstrated that the proposed system was able to achieve high accuracy in vehicle detection and collision prediction, with an average precision of 94.5%. This suggests that the system has the potential to significantly reduce the number of accidents on the road, particularly in situations where human error is a contributing factor. Overall, the study highlights the importance

of developing advanced driver assistance systems using state-of-the-art machine learning techniques.

[12] The study "Detection of Bluefin Tuna by Cascade Classifier and Deep Learning for Monitoring Fish Resources" proposed a system for detecting bluefin tuna using both cascade classifier and deep learning methods. The system uses a combination of image processing techniques and machine learning algorithms to identify and track bluefin tuna in underwater images captured by cameras attached to fishing boats. The proposed system achieved high accuracy and can be used for monitoring bluefin tuna resources in a more efficient and reliable manner. The results of the study suggest that combining traditional image processing techniques with deep learning algorithms can enhance the accuracy and performance of fish detection systems, which can have significant implications for the fishing industry and marine conservation efforts.

2.2 INFERENCES FROM LITERATURE SURVEY

Over the years, there has been considerable research in the field of natural language processing, and many models have been proposed to address various aspects of this complex task. As evident from the literature, each system has its own strengths and weaknesses, and researchers continue to explore new approaches to overcome the limitations of existing models.

One of the recent trends in natural language processing is the use of hybrid technologies, which combine multiple methods to achieve better accuracy. However, even with these advancements, achieving the level of accuracy required for practical applications remains a significant challenge. Moreover, as the accuracy of these models increases, so does the need for low computational costs, high processing speed, and ease of use.

To address these challenges, researchers are exploring various avenues, such as leveraging advancements in hardware technology like GPUs and TPUs to improve processing speed and reduce computational costs. Additionally, the development of more user-friendly systems with streamlined interfaces is also an important area of focus.

Overall, the field of natural language processing is constantly evolving, and researchers are working towards developing more accurate, efficient, and user-friendly systems that can effectively process and analyze human language. While there is still

much work to be done, continued research in this area promises to yield further breakthroughs in the development of natural language processing systems with practical applications in diverse fields such as healthcare, finance, and education.

2.3 Open Problems in Existing System

To overcome the limitations of traditional vehicle detection algorithms, researchers have turned to deep learning methods, which have shown great potential in achieving high accuracy and efficiency in vehicle detection. Deep learning algorithms can learn features automatically from raw data without the need for manual feature extraction. This allows for more robust and accurate detection in a wider range of scenarios. Moreover, the development of modern deep learning frameworks and the availability of large-scale annotated datasets have further enabled the application of deep learning to vehicle detection.

One of the most popular deep learning models for object detection, including vehicle detection, is YOLO (You Only Look Once). YOLO can achieve real-time performance while maintaining high accuracy. Other deep learning models such as SSD (Single Shot Detector) and Faster R-CNN (Region-based Convolutional Neural Network) have also been used for vehicle detection. These models have shown great potential in detecting vehicles with high accuracy and efficiency, making them suitable for use in advanced driver assistance systems (ADAS) and autonomous vehicles.

CHAPTER – 3

REQUIREMENT ANALYSIS

3 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

3.1 FEASIBILITY STUDY:

Economic feasibility is one of the critical factors in determining the success of a project. In this context, the cost-benefit analysis of the proposed system is essential to evaluate the economic feasibility. The cost-benefit analysis will help in determining the project's profitability, return on investment (ROI), and payback period. The project cost may include the cost of hardware and software, labor, training, and other miscellaneous costs. The benefits of the proposed system may include increased server performance, reduced downtime, and improved user experience. By evaluating the costs and benefits, the project's financial viability can be determined, and a decision can be made on whether to proceed with the project or not.

Technical feasibility is another critical factor that determines the success of the project. The proposed system's technical feasibility will involve evaluating the system's compatibility with existing hardware and software, its scalability, reliability, and security. The compatibility of the proposed system with existing infrastructure is crucial to ensure that the system can be integrated seamlessly. The scalability of the system is important to ensure that the system can handle future requirements without significant modifications. The system's reliability and security are also crucial to ensure that the system can operate without failure and data confidentiality and integrity are maintained.

Operational feasibility is the third factor that determines the project's success. The proposed system's operational feasibility will involve evaluating the system's ease of use, user acceptance, and the impact of the system on existing business processes. The system's ease of use is important to ensure that the system can be used effectively by users without significant training. The user acceptance of the system is important to ensure that the system is adopted and used by the intended users. The impact of the system on existing business processes must also be evaluated to ensure that the system does not disrupt existing operations. By evaluating these three factors,

the feasibility of the proposed system can be determined, and a decision can be made on whether to proceed with the project or not.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Operational feasibility

3.2 ECONOMICAL FEASIBILITY

The economic feasibility study involves analyzing the cost of the project, including the cost of developing and implementing the system, as well as the expected return on investment. In addition to the initial cost, ongoing maintenance, training, and support costs must also be taken into account. The system should be economically feasible in terms of its benefits and costs, and the cost should not exceed the potential benefits. The analysis should also consider potential risks and uncertainties that could impact the economic feasibility of the project.

The technical feasibility study involves determining whether the proposed system can be developed and implemented with the available technology and resources. This includes assessing the technical requirements of the system, such as hardware and software, as well as the skills and knowledge required to develop and maintain the system. The study should also consider any technical limitations or challenges that may arise during development and implementation.

Operational feasibility involves assessing the practicality and usability of the proposed system in the intended operational environment. This includes evaluating whether the system can be integrated into existing workflows and processes, whether it is user-friendly, and whether it meets the needs of stakeholders. The study should also consider any potential risks or challenges that may arise during system operation, such as security or privacy concerns.

Overall, a thorough feasibility analysis is critical to ensure the success of the proposed system and its alignment with organizational goals and objectives.

3.3 TECHNICAL FEASIBILITY

Technical feasibility is an essential consideration when developing a new system. The proposed system's technical requirements must be analyzed to ensure that they can be met within the available resources. This includes hardware, software, and network resources. In the case of the server performance increase project, the technical feasibility analysis should examine the current server infrastructure, including hardware, software, and network configurations, to determine if they can support the proposed system's technical requirements.

For instance, the proposed system may require additional hardware components such as CPUs, memory, or storage space to operate effectively. The existing network infrastructure may also need to be upgraded to handle the increased traffic and data processing demands of the new system. The software applications required for the system should be evaluated to ensure they are compatible with the existing infrastructure.

Overall, a technical feasibility analysis is crucial to ensure that the proposed system can be developed and implemented successfully and that it will function effectively without placing undue strain on the existing technical resources.

3.4 OPERATIONAL FEASIBILITY

Operational feasibility is a critical consideration in the development of any system as it evaluates the practicality of implementing the system in the organization. The operational feasibility analysis considers the existing processes and procedures of the organization and how they will be affected by the new system. The analysis helps in identifying the challenges that may arise during implementation and how to address them.

In the case of the server performance increase project, operational feasibility will involve evaluating the current processes involved in managing the server and how the new system will integrate into those processes. It will also consider the impact of the new system on the organization's staff and their ability to adapt to it.

Additionally, operational feasibility will involve assessing the availability of resources needed to support the system, such as skilled personnel, infrastructure, and software. The analysis will help in determining whether the organization has the necessary resources to support the system or if additional resources need to be acquired.

Overall, the operational feasibility analysis helps in determining whether the new system will be practical to implement and operate within the organization's current structure and resources.

3.5 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Hardware specifications:

- Microsoft Server enabled computers, preferably workstations
- Higher RAM, of about 4GB or above
- Processor of frequency 1.5GHz or above

Software specifications:

- Python 3.6 and higher
- Anaconda software

CHAPTER 4

DESCRIPTION OF THE PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

4.1.1 DATA COLLECTION AND PREPROCESSING

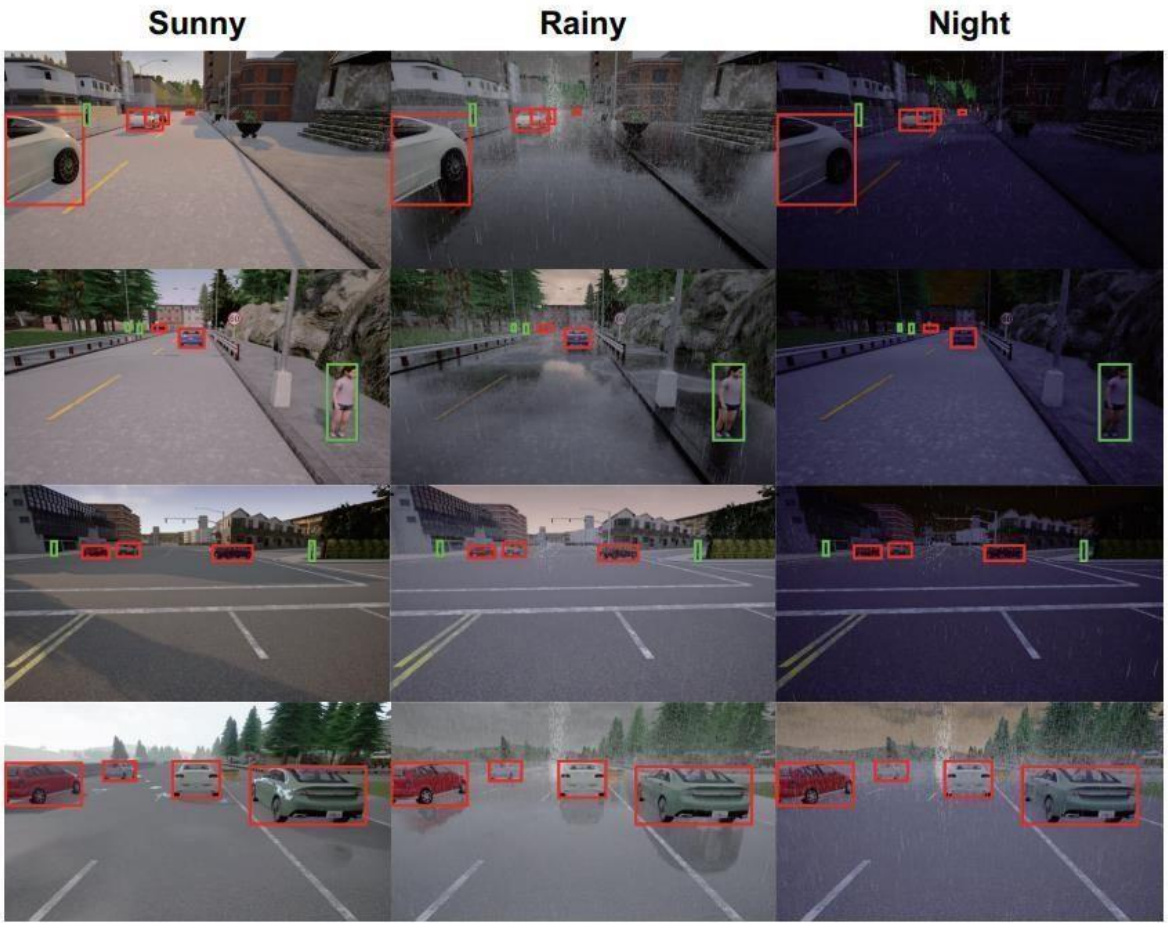


Fig 4.1 Dataset for identifying vehicles

The process of setting up a data environment and compiling data is a critical step in any data analysis project. This process involves collecting data from various sources, importing necessary libraries, and labeling quality datasets. Once the data has been collected, it is necessary to preprocess it, which involves cleaning and formatting the data to ensure that it can be effectively analyzed.

Data collection is a crucial aspect of data analysis, and datasets are the primary source of data. Datasets can be obtained from various sources, including online sources, government databases, and academic research. These datasets can be collected by importing necessary libraries from source websites, and they may contain a wide range of data, including numerical, textual, and visual data.

The quality of the data is essential in data analysis, and data labeling is a process that ensures that the data is accurately labeled and categorized. This process involves assigning labels or tags to each data point, making it easier to categorize and analyze the data. The labels used can be based on various criteria, including the data type, subject matter, and intended use.

Preprocessing is an essential step in data analysis, and it involves cleaning and formatting the data to ensure that it can be analyzed effectively. During the preprocessing stage, missing and null values are removed, and any other data inconsistencies are corrected. This ensures that the data is accurate and reliable, making it easier to identify patterns and trends in the data.

Another critical aspect of data analysis is image processing, which involves assigning images to the data. This process is particularly useful in visualizing data and identifying patterns that may not be evident from the raw data. Images can be assigned based on various criteria, including data type, location, and subject matter.

In conclusion, data analysis is a complex process that involves various stages, including data collection, labeling, preprocessing, and image processing. Each of these stages is essential in ensuring that the data is accurate, reliable, and can be effectively analyzed. Proper planning and execution of these stages can lead to valuable insights and trends that can be used to make informed decisions.

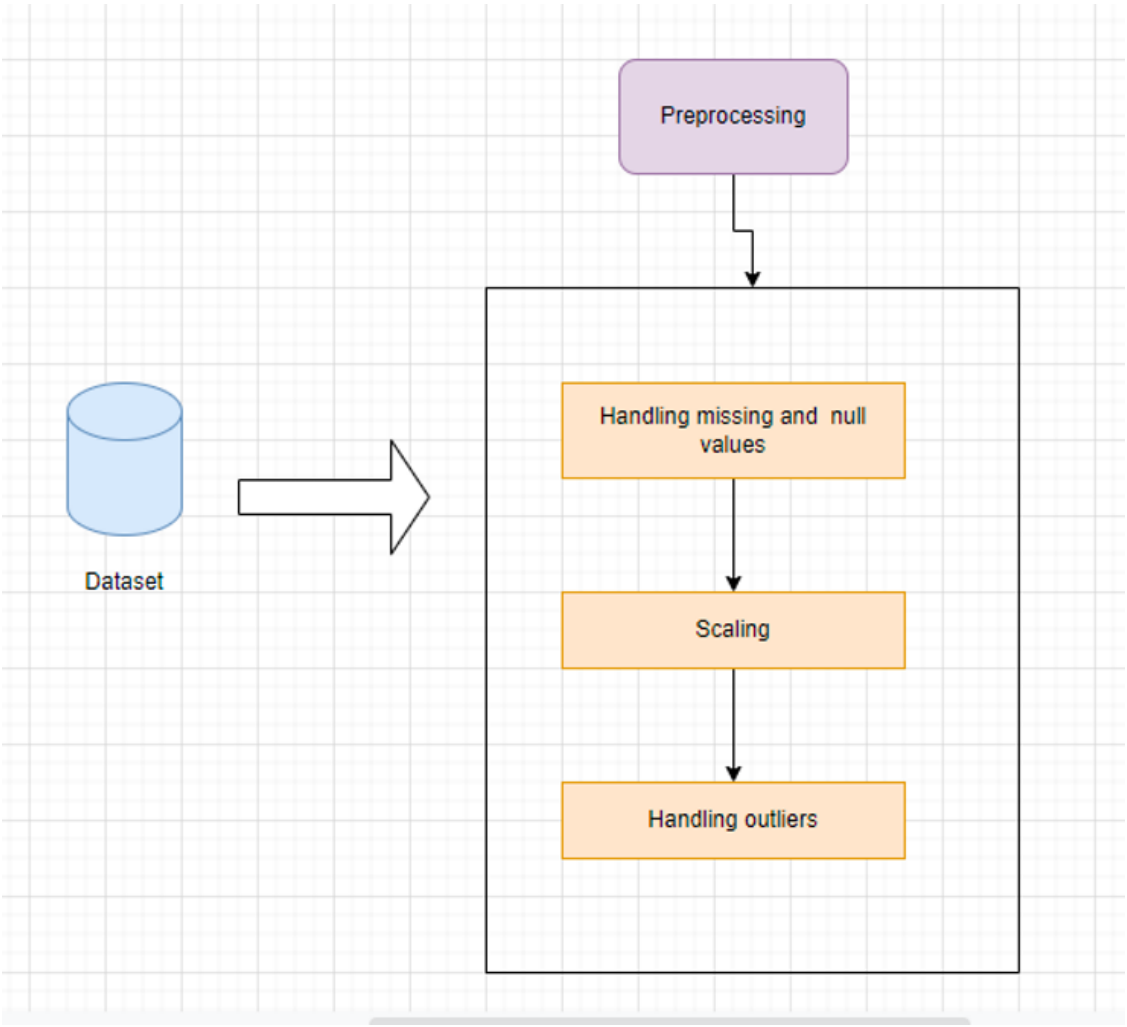


Fig 4.2 Data Preprocessing

4.1.2 Algorithm

data.info ()
data. Head () // For displaying first five rows
data. Describe ()
data. Is Null (). sum //gives the number of missing values for each
variable data. Duplicated (). Sum () // returning the total number of duplicenterednt rie

This is a set of commands or functions that can be used to explore and summarize a given dataset in a programming language such as Python or R.

data.info() returns information about the dataset, such as the number of columns, the data type of each column, and the number of non-null values in each column.

data. head() displays the first five rows of the dataset to give a quick overview of the data.

data. describe() provides summary statistics of the dataset, including measures of central tendency, variability, and distribution of the numerical variables in the dataset.

data. isnull().sum() calculates the total number of missing values for each variable in the dataset. This function is useful in identifying which variables have missing data that needs to be dealt with before further analysis.

data. duplicated().sum() returns the total number of duplicate entries in the dataset. This function is useful in identifying and removing any redundant data in the dataset, which could skew the results of the analysis

4.1.3 DATA SPLITTING AND MEAN SQUARE DETECTION

YOLO:

The YOLO (You Only Look Once) algorithm is a state-of-the-art object detection algorithm that achieves impressive results by applying a neural network to an image. The algorithm works by dividing the image into an S x S grid and using each cell of the grid to predict bounding boxes for objects in the image. The network architecture consists of 24 convolutional layers, followed by two fully connected layers. The reduction in feature space is achieved by alternating 1x1 convolutional layers with the preceding layers.

The YOLO algorithm treats object identification as a regression problem. The goal is to predict the spatial bounding box coordinates for each object in the image, as well as the probability of each object belonging to a particular class. The neural network can accomplish this in just one evaluation, making it incredibly efficient and capable of processing images in real-time.

One of the main advantages of the YOLO algorithm is that it can optimize end-to-end. This means that the entire neural network can be trained together using backpropagation to optimize both the bounding box predictions and the class probabilities. This is in contrast to other object detection algorithms, which may require multiple stages of processing and optimization, leading to slower processing times and lower accuracy.

The YOLO algorithm has achieved impressive results in a variety of applications, including detecting objects in images and videos, tracking objects over time, and recognizing human actions in videos. It has also been used in autonomous vehicles, where it can quickly and accurately detect obstacles and other vehicles on the road.

In conclusion, the YOLO algorithm is a powerful and efficient object detection algorithm that uses a neural network to predict the spatial bounding boxes and class probabilities for objects in an image. Its ability to optimize end-to-end and process images in real time has made it a popular choice in a wide range of applications.

Mean Square detection:

Mean square detection is a statistical technique that can be used for vehicle collision detection. The idea behind this technique is to analyze the motion of two vehicles and predict if a collision is likely to occur. This can be done by measuring the distance between the two vehicles and their relative speed.

The mean square detection technique works by calculating the mean square error (MSE) between the predicted trajectory of the vehicles and the actual trajectory of the vehicles. The predicted trajectory is based on the current position and speed of the vehicles, while the actual trajectory is based on the observed position and speed of the vehicles. If the MSE is below a certain threshold, it is assumed that a collision is likely to occur.

To use mean square detection for vehicle collision detection, sensors are typically installed on the vehicles to measure their position and speed. These sensors can include GPS, accelerometers, and radar sensors. The sensor data is then processed by a computer algorithm that calculates the predicted trajectory of the vehicles and compares it to the actual trajectory of the vehicles.

If the algorithm determines that a collision is likely to occur, it can take various actions to avoid the collision, such as applying the brakes or alerting the driver. This can help prevent accidents and improve overall safety on the road.

Overall, mean square detection is a useful technique for vehicle collision detection that can help improve safety on the road. By analyzing the motion of vehicles and predicting potential collisions, it can help prevent accidents and save lives.

- Class Loss detection,
- Intersection, and Union Analyzation.
- Region of Interest analysis is being performed.
- 4k iteration is being performed

The terms "class loss detection," "intersection and union analysis," "region of interest analysis," and "4k iteration" are all related to the field of computer vision and specifically

to object detection using deep learning algorithms.

Class loss detection: refers to the process of measuring the accuracy of a deep learning model's predictions for each class of object it is trained to detect. This is typically done using a loss function such as cross-entropy loss, which penalizes the model for making incorrect predictions.

Intersection and union analysis: is a common evaluation metric for object detection algorithms, which measures the degree of overlap between the predicted bounding boxes and the ground truth bounding boxes (i.e., the actual location of the objects in the image). The intersection over union (IoU) metric is typically used, which calculates the ratio of the area of intersection between the predicted and ground truth bounding boxes to the area of union between them.

Region of interest analysis: refers to the process of identifying and analyzing specific regions of an image that are likely to contain objects of interest. This is typically done using techniques such as selective search or region proposal networks, which generate a set of candidate regions that are then processed by the object detection algorithm.

4k iteration: likely refers to the number of iterations (or epochs) that a deep learning model is trained for. In object detection, training a deep learning model typically involves iteratively adjusting the model's parameters to minimize the loss function on a set of training data. The number of iterations required to train a model can vary depending on factors such as the complexity of the model and the size of the training data.

Object Detection:

In computer vision, a multi-task algorithm is a type of algorithm that can perform multiple related tasks simultaneously. These tasks may include object detection, tracking, segmentation, and other types of analysis. Such algorithms can be used to extract a wide range of information from images or video streams.

The proposed multi-task algorithm is designed to extract all necessary information from each frame of a video stream in real-time. This means that the algorithm must be optimized for real-time performance, meaning that it can process each frame quickly enough to keep up with the video's frame rate.

To achieve this, the algorithm has been designed to be efficient and to maintain the principles of real-time processing. YOLO is one object detection tool that can be used as part of this multi-task algorithm. YOLO is designed to divide the input image into an SxS grid, with each grid cell being responsible for detecting objects within its boundaries. The tool then predicts the probability of each grid cell containing an object, as well as the bounding box coordinates for each object.

Using YOLO as part of a multi-task algorithm allows for the efficient detection of objects in real-time video streams, which can be used for a variety of applications such as surveillance, autonomous vehicles, and robotics. By using a multi-task algorithm, it is possible to extract multiple types of information from each frame of the video stream, which can be used for a wide range of computer vision applications.

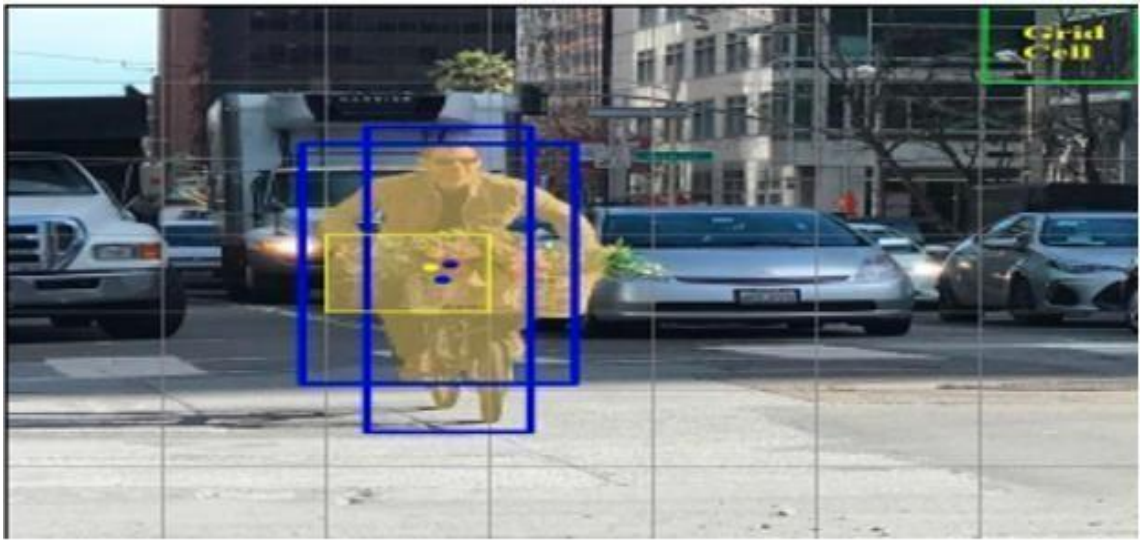


Fig 4.3 Grid formed by yolo V3

Each grid cell in the image is responsible for predicting only one object. This means that the algorithm tries to identify the object whose center falls within the boundaries of the grid cell. To do this, each grid cell predicts a fixed number of boundary boxes to locate the object. For example, a yellow grid cell may predict two blue boundary boxes to locate a person object within its boundaries.

There are multiple versions of YOLO, but version 3 was chosen for its combination of speed and accuracy. This version of YOLO has been combined with other algorithms, such as depth estimation and computer vision, to create a single algorithm for real-time collision warning systems in autonomous cars.

In this application, latency is a critical parameter that must be kept as low as possible. YOLO was chosen because it can detect a limited number of classes, which reduces processing time and increases performance. By detecting fewer classes, YOLO can process images more quickly, which is essential for real-time applications like autonomous driving.

Overall, by combining YOLO with other algorithms, this multi-task algorithm can detect potential collisions and provide warning signals in real-time. This is an important feature for ensuring the safety of autonomous vehicles and their passengers.

4.1.4 Depth Estimation

The depth estimation algorithm is a critical part of the multi-task algorithm used for collision warning systems in autonomous cars. The first step of the depth estimation algorithm is to use the boundary boxes (width and height) obtained from YOLO object detection.

To estimate the distance of the object from the camera, the depth estimation algorithm uses multiple regression techniques. The algorithm modifies the depth estimation equation using a trial and error algorithm, which tries different values for a parameter called ŧ. This parameter is obtained from the regression algorithm and does not have any physical meaning.

During the trial and error process, the algorithm compares the estimated distance output for each value of ŧ with the real distance obtained from a real dataset. The purpose of this comparison is to determine the optimal value of ŧ that provides the most accurate distance estimation.

By adjusting the depth estimation equation using this trial and error algorithm, the algorithm can estimate the distance of an object accurately. This is an essential feature for collision warning systems, which need to accurately detect the proximity of objects to the vehicle in real-time. Overall, this multi-task algorithm uses multiple techniques, including YOLO object detection and depth estimation, to provide real-time collision warning systems for autonomous cars.

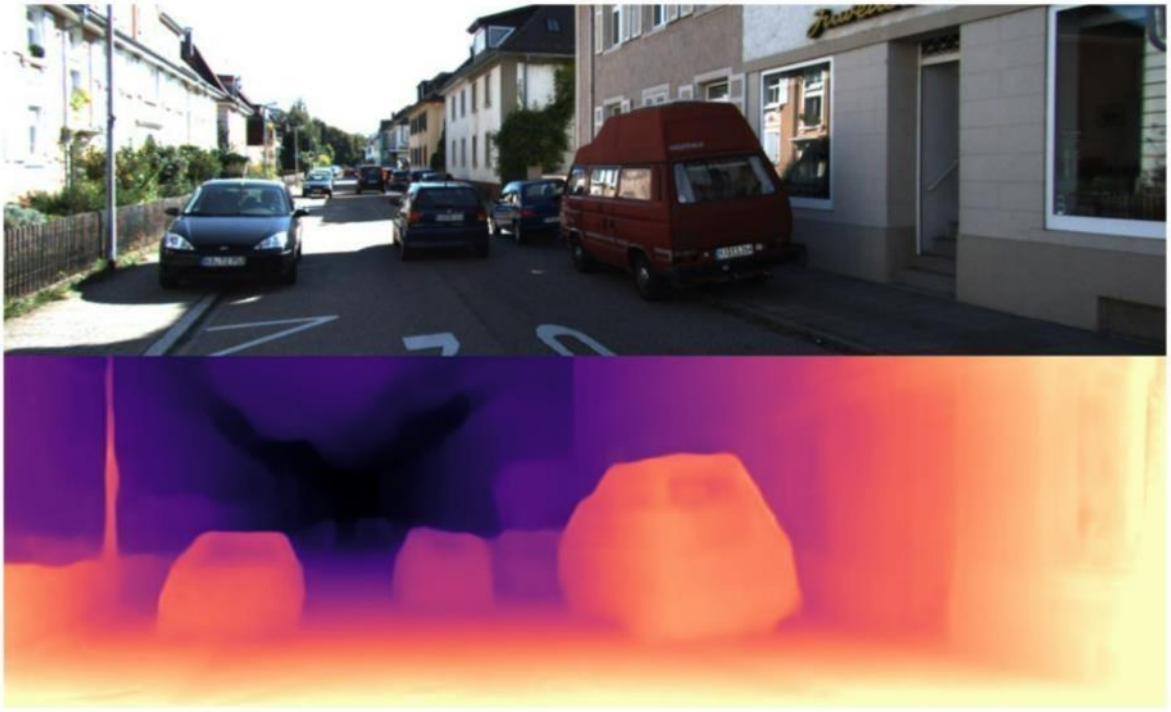


Fig 4.4 Depth Estimation

4.1.5 Lane Assignation & Tracking Algorithm

After applying depth estimation, two algorithms developed to complete the task; lane assignation and tracking algorithm. For the lane assignation algorithm, geometric principles were used to assign cars into lanes depending on each vehicle's bounding box.

Fig 4.5 Lane Assignation and Tracking Algoritham

The above figure shows the three regions under the horizon line. The horizon line is dependent on the camera position in the presented work; the horizon line represented 55% of the height. As shown in the above figure the image is divided virtually with two significant lines into four quarters; the upper half is unused as it represents the sky or the far vehicles, while the down half represents the road (in the presented work), which

is divided into three regions. Lane assignation is a very critical task, as it will determine which vehicle to give interest and which not, So, the performance of this task must be optimal. The problem was which point of the bounding box will be used to represent the entire box (the car)? Lane assigning algorithm is divided into two layers/phases; layer 1, is to determine at what quarter the vehicle is moving. In this layer, the center of the bounding box uses as a reference point, The primary role of this layer is to determine if the vehicle is above the horizon (it does not represent any interest right now), regardless of the interest or if the vehicle is below the horizon so it is in the exciting area and initially determined if it is in the right side or the left side For layer 2, two virtual lanes are created, which represent the width of the vehicle; this layer is more critical than the first layer. Thus, the results are expected to be precisely accurate, especially in the busy roads. In this layer, two different reference points are used depending on the output of the first layer, If the vehicle was detected at the right half, the bottom-left point of the bounding box is used as the reference point to the vehicle. This reference point substituted in the right virtual lane equation to determine whether the vehicle is in the emergency lane or not; if not, automatically, the car will be assigned to the right lane. While if the car was detected at the left part, the bottom-right point of the bounding box is the reference point to the car. This reference point is used to determine whether the car is in the emergency lane or not; if not, automatically, the car will be assigned to the left lane. The above figure shows how the input image converted after object detection into a geometric problem to assign vehicles to their actual lanes The assigning technique illustration Finally, the tracking algorithm is responsible for tracking the same vehicles through all the frames, so the change in its distance could be evaluated then calculating its relative speed. The tracking algorithm is a complex problem, especially if the road is crowded with vehicles since it is a must to compare all vehicles in the current frame with all vehicles in the previous frame, then a suitable threshold was estimated. If the new position differs by a maximum of 5% from the last position, so it is the same vehicle. Hence tracking the same vehicle through the frames is achieved but it’s heavy work and increases the delay of the system. So, here is the second importance of the lane assigning algorithm, hence after the vehicles are divided into three categories, the tracking algorithm is applied on the vehicles in the emergency lane only. So, the algorithm will be more simple and faster, also the need to be aware of the relative speed of the cars in front of our vehicle.. After the first detection, the positions of the vehicles in the emergency lane are saved, then start comparing at

every frame the new position of the vehicles in the emergency lane with the last saved positions. If the difference of vehicle's positions is within the threshold (difference < 3%) so it considered as the same vehicle, hence the tracking is achieved.

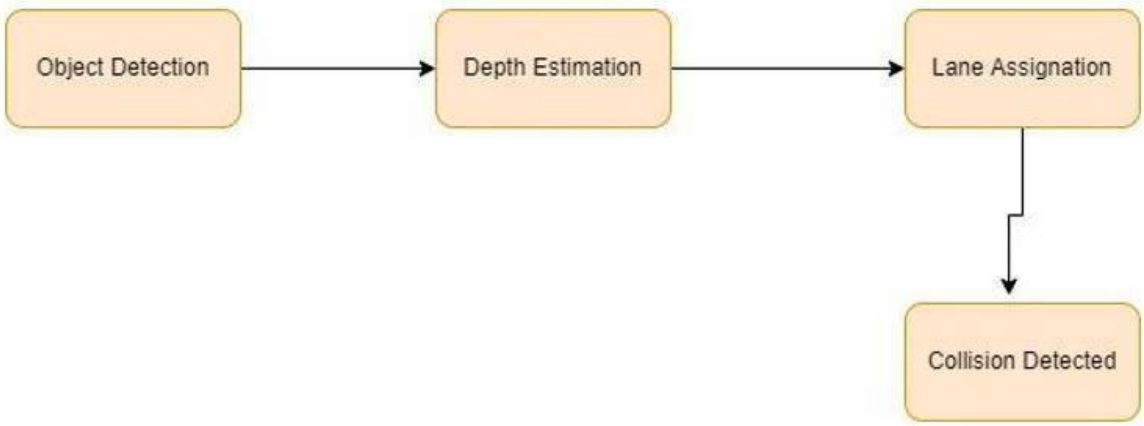


Fig 4.6 Process model of Vehicle Detection

Algorithm

image = readImage ()
threshold = 0.7
step = height(image)/NoOfCells
prediction_class_array = new_array (size
(NoOfCells,NoOfCells,NoOfClasses))
final_predictions = []

predictions_bounding_box_array[i,j] = bounding_box_predictor(cell)
best_bounding_box = [0 if predictions_bounding_box_array[i,j,0, 4] > predictions_bounding_box_array[i,j,1, 4] else 1]
final_predictions.append(prediction)
print final_predictions

4.1.6 Prediction:

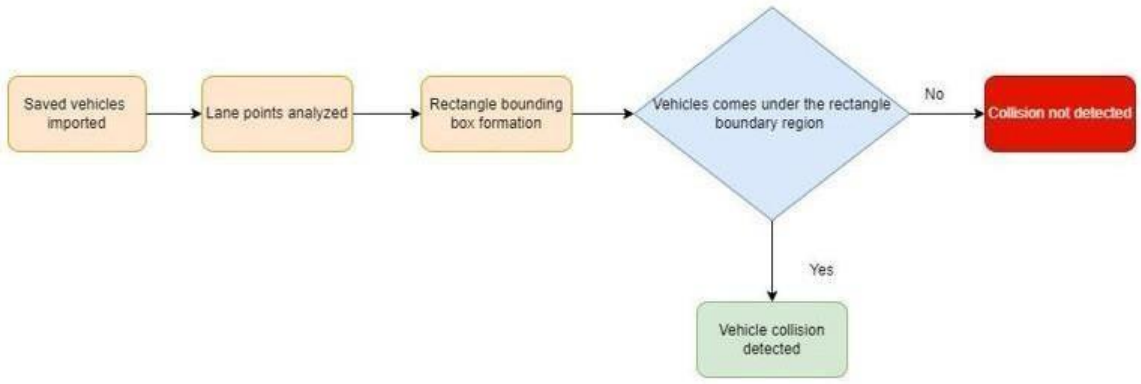
The proposed system for detecting vehicle collisions is a crucial aspect of autonomous driving, as it provides a mechanism to detect potential hazards in the surrounding environment. This system utilizes a combination of saved vehicle data, lane analysis, and bounding box parameters to determine the likelihood of a collision.

The first step in this process involves importing saved vehicle data into the system. Lane points are then analyzed, and a function for the region of interest is written to determine the specific area of the roadway that needs to be monitored for potential collisions. This region of interest is critical as it enables the system to focus on the most relevant areas of the roadway and avoid unnecessary processing overhead.

Once the region of interest is defined, the system begins processing input video data from the roadway. Vehicle detection algorithms are used to identify any cars or trucks present on the road, while lane analysis is used to determine the position of these vehicles relative to the designated lane markers. Bounding box parameters are then analyzed to identify the boundaries of each vehicle and determine their current position on the roadway.

If a car X is being driven on a highway, for instance, a rectangle boundary in front and back of it would be formed to represent the potential hazard zone. This rectangle represents the area of the roadway that needs to be monitored for potential collisions. If any vehicle enters this boundary region, the system is notified that there is a chance for a collision.

This system has important applications in the development of self-driving cars, where it would analyze the collision possibility automatically and drive accordingly. By utilizing advanced algorithms and data analysis techniques, this system can help prevent accidents and provide a safer driving experience for all road users. Moreover, this system can be incorporated into existing autonomous vehicle technology to provide an additional layer of safety and reduce the risk of collisions. Overall, the proposed system provides a crucial tool for enhancing the safety and reliability of autonomous driving technology in the future.



4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSYTEM

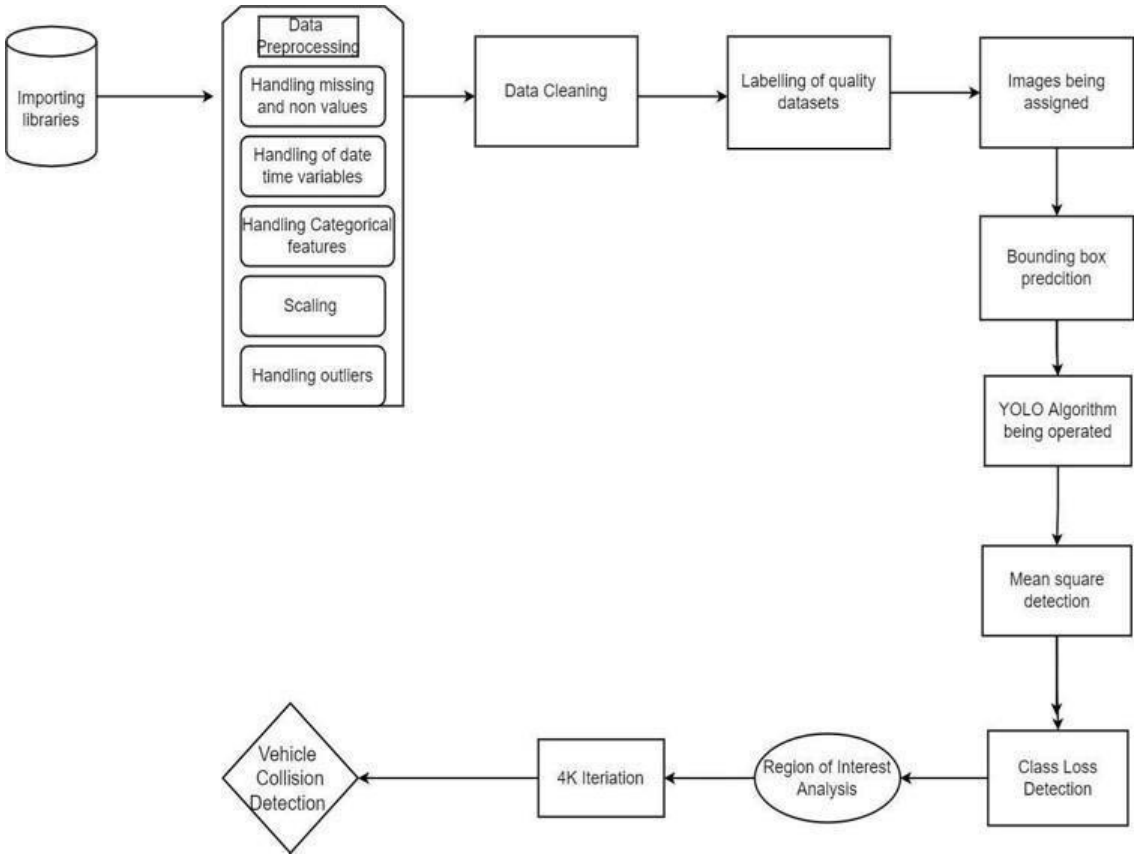


Fig 4.7 Architecture / Design of proposed system

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

Anaconda is a robust open-source package manager that provides a simplified and streamlined environment for deploying and managing data science projects. It is considered the most popular platform among data science professionals, owing to its comprehensive package distribution system and numerous AI and machine learning tools.

One of the most significant advantages of using Anaconda is that it is free and open-source, which means that anyone can use it without any financial burden. This has made it widely popular and an industry staple in the data science sector. Any data science professional who wants to stay ahead in their field must know how to use Anaconda for Python, as recruiters expect this skill from potential employees. Therefore, having a working knowledge of Anaconda is a must-have for data science professionals.

Another benefit of using Anaconda is that it has over 1500 Python and R data science packages, which ensures that you don't face any compatibility issues while collaborating with others. Suppose you receive a project from a colleague that requires packages A and B, but you only have package A. In that case, you wouldn't be able to run the project without having package B, leading to compatibility issues. Anaconda mitigates the chances of such errors, enabling you to collaborate on projects without worrying about any compatibility issues.

Anaconda provides a seamless environment that simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Furthermore, Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. You can also ask people in the community about the issues you face, and the helpful community is always ready to assist new learners.

One of the most impressive features of Anaconda is its compatibility with popular tools, including TensorFlow, Scikit-Learn, and Theano, making it easier to create and train machine learning and deep learning models. Additionally, you can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda, providing you with a comprehensive set of tools to work with.

Overall, Anaconda is a must-have tool for data science professionals, as it provides an all-in-one solution for deploying and managing data science projects. With its extensive range of packages, streamlined environment, and compatibility with popular tools, Anaconda is a game-changer in the field of data science. Whether you're a seasoned data science professional or a novice learner, Anaconda simplifies the process of deploying and managing data science projects, making it a valuable addition to your toolset.

How to Use Anaconda for Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let’s discuss some fundamental commands you can use to start using this package manager.

Listing All Environments

To begin using Anaconda, you’d need to see how many Conda environments are present in your machine.

conda env list

It will list all the available Conda environments in your machine.

Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command:

conda create -n <your_environment_name>

You can replace <your_environment_name> with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

proceed ([y])/n)?

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

conda create -n <your_environment_name> python=3.6

Similarly, if you want to create an environment with a particular package, you can use the following command:

conda create -n <your_environment_name>pack_name

Here, you can replace pack_name with the name of the package you want to use.

If you have a .yml file, you can use the following command to create a new Conda environment based on that file:

conda env create -n <your_environment_name> -f <file_name>.yml

We have also discussed how you can export an existing Conda environment to a .yml file later in this article.

Activating an Environment

You can activate a Conda environment by using the following command:

conda activate <environment_name>

You should activate the environment before you start working on the same. Also, replace the term <environment_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command:

conda deactivate

Installing Packages in an Environment

Now that you have an activated environment, you can install packages into it by using the following command:

conda install <pack_name>

Replace the term <pack_name> with the name of the package you want to install in your Conda environment while using this command.

Updating Packages in an Environment

If you want to update the packages present in a particular Conda environment, you should use the following command:

conda update

The above command will update all the packages present in the environment. However, if you want to update a package to a certain version, you will need to use the following command:

conda install <package_name>=<version>

Exporting an Environment Configuration

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yml file and share it with that person. Now, they can create an environment like your one by using the .yml file.

For exporting the environment to the .yml file, you'll first have to activate the same and run the following command:

conda env export ><file_name>.yml

The person you want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

Removing a Package from an Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

conda remove -n <env_name><package_name>

On the other hand, if you want to uninstall a package from an activated environment, you'd have to use the following command:

conda remove <package_name>

Deleting an Environment

Sometimes, you don't need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

conda env remove –name <env_name>

The above command would delete the Conda environment right away.

CHAPTER – 5

RESULTS AND DISCUSSION

5.1 Result

The primary result of a vehicle collision detection and alert system using YOLOv3 would be the ability to accurately detect collisions and issue alerts to drivers or emergency services in real-time. The system would rely on object detection using YOLOv3 to identify vehicles and other objects in the environment and track their movement. By analyzing this movement and looking for patterns that indicate a collision, the system could quickly detect when an accident occurs and issue alerts accordingly.

The alert system could take various forms, depending on the specific implementation. For example, it might issue an audible alert in the vehicle to warn the driver to slow down or stop to avoid a collision. Alternatively, the system could send alerts to emergency services or other drivers in the area to provide assistance as quickly as possible.

One of the key benefits of using YOLOv3 for collision detection is its speed and accuracy. This allows the system to operate in real-time, ensuring that alerts are issued quickly and accurately. In addition, YOLOv3 is highly adaptable, allowing it to detect a wide range of objects and track their movement even in complex environments.

Overall, a vehicle collision detection and alert system using YOLOv3 has the potential to greatly improve road safety and reduce the impact of accidents. By providing real-time alerts and assistance, it can help to prevent collisions and save lives.

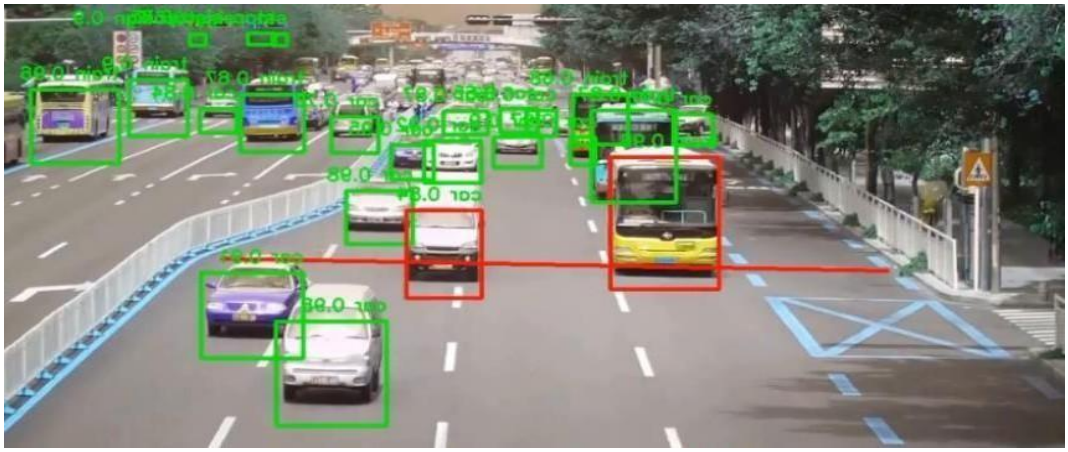


Fig 5.1 Expected Results

CHAPTER – 6

CONCLUSION

6.1 CONCLUSION

In recent years, there has been a significant increase in the number of road accidents worldwide. Many of these accidents are caused by human error, such as distracted driving, fatigue, or impaired driving. In response to this problem, researchers and engineers have developed various technologies to improve road safety, including advanced driver assistance systems (ADAS) and collision detection and alert systems.

One promising technology in this area is the use of deep learning algorithms, such as YOLOv3, for vehicle collision detection and alert systems. YOLOv3 is a state-of-the-art object detection algorithm that can detect and track multiple objects in real time with high accuracy and speed. By analyzing the movement of objects in the environment, the system can detect potential collisions and issue alerts to the driver or emergency services.

The implementation of a vehicle collision detection and alert system using YOLOv3 involves capturing video footage of the environment using cameras mounted on the vehicle. The YOLOv3 model processes the video frames and identifies vehicles and other objects in the environment. By analyzing the movement of these objects, the system can detect potential collisions and issue alerts to the driver or emergency services.

One of the main advantages of using YOLOv3 for vehicle collision detection and alert systems is its real-time processing speed, which allows the system to issue alerts quickly in case of a potential collision. Additionally, YOLOv3 is a highly accurate object detection algorithm that can detect and track objects in various lighting conditions, making it suitable for use in different environments.

However, the implementation of a vehicle collision detection and alert system using YOLOv3 also faces several challenges. One of the primary challenges is the need for a large and diverse dataset to train the YOLOv3 model. Additionally, there is a need for communication protocols between vehicles to issue alerts to other drivers, as well as regulatory approval and widespread adoption of the technology.

Despite these challenges, a vehicle collision detection and alert system using YOLOv3 holds great promise for improving road safety and reducing the number of collisions on our roads. With continued research and development, it is possible to overcome the technical and regulatory challenges and realize the full potential of this technology. In conclusion, the use of YOLOv3 for vehicle collision detection and alert systems is a promising area of research that can significantly contribute to improving road safety and reducing the number of accidents on our roads.

6.2 FUTURE WORK :

future work in the field of vehicle collision detection and alert systems using YOLO v3. One potential area of focus is improving the accuracy and efficiency of the system. This could be achieved through further optimization of the neural network architecture, or through the development of more advanced image processing techniques. Additionally, research could be done on the integration of multiple sensors, such as radar or LIDAR, to improve the system's ability to detect and respond to potential collisions.

Another area for future work is the development of more sophisticated alert systems. Currently, most systems rely on simple audio or visual alerts to warn drivers of potential collisions. However, there is potential for the integration of more advanced warning systems, such as haptic feedback, that could provide more nuanced information to drivers in real-time.

In addition to improving the technical capabilities of the system, there is also potential for research on the social and psychological impacts of vehicle collision detection and alert systems. For example, research could be conducted to better understand how drivers respond to alerts, and how these responses might be affected by factors such as fatigue, distraction, or stress.

Finally, there is potential for the integration of vehicle collision detection and alert systems with other advanced driver assistance systems (ADAS), such as lane departure warning or automatic emergency braking. By combining these systems, it may be possible to create a more comprehensive safety package that can help to reduce the risk of collisions and improve overall road safety

6.3 RESEARCH AND IMPLEMENTATION ISSUES

The implementation of a vehicle collision detection and alert system using YOLOv3 involves several research and implementation issues. Some of these issues are discussed below:

Dataset: One of the significant challenges is the availability of a dataset with a sufficient number of annotated images. The dataset should include various scenarios and objects in different lighting and weather conditions to ensure the robustness of the system. Collecting and annotating such a dataset can be time-consuming and expensive.

Hardware requirements: Another issue is the hardware requirements for running the system in real-time. Since YOLOv3 is computationally intensive, it requires powerful hardware such as GPUs or TPUs for fast and accurate object detection. The choice of hardware also affects the cost of implementing the system.

Optimization: Optimizing the system for real-time performance is another challenge. The system needs to process video frames in real-time, and any delay in processing can result in false negatives or delayed alerts. Optimizing the neural network and the hardware can improve the system's performance.

False alarms: One of the significant issues in collision detection systems is false alarms. The system should be designed to minimize false alarms and alert only when a collision is imminent. This can be achieved by setting appropriate thresholds and filtering out false positives.

Integration with existing systems: Integrating the collision detection and alert system with existing vehicle systems can be challenging. The system needs to communicate with other systems such as the braking system, airbag system, and navigation system to provide accurate and timely alerts. Integration with different vehicle models and manufacturers can also be a challenge.

Legal and ethical issues: The implementation of such a system raises legal and ethical concerns. The system needs to comply with regulations related to data privacy, security, and liability. The ethical implications of having an automated system that can potentially take control of the vehicle in certain situations also need to be addressed.

References

[1] Yeong-Kang Lai, Chu-Ying Ho, Yu-Hau Huang, Chuan-Wei Huang, Yi-Xian Kuo, Yu-Chieh Chung, “Intelligent Vehicle Collision-Avoidance System with Deep Learning”, 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 2018

[2] Jae Gyeong Choi, Chan Woo Kong, Gyeongho Kim, Sunghoon Lim, “Car crash detection using ensemble deep learning and multimodal data from dashboard cameras”, Expert Systems with Applications, Volume 183, 30 November 2021,

[3] Aloukik Aditya, Liudu Zhou, Hrishika Vachhani, Dhivya Chandrasekaran, Vijay Mago,” Collision Detection: An Improved Deep Learning Approach Using SENet and ResNext”, 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2021

[4] Wan-Jung Chang, Liang-Bi Chen, Ke-Yu Su, “DeepCrash: A Deep Learning-Based Internet of Vehicles System for Head-On and Single-Vehicle Accident Detection with Emergency Notification”, IEEE Access, Vol 7, P-148163 – 148175, 2019

[5] Manu S. Pillai, Gopal Chaudhary, Manju Khari, Rubén González Crespo, “Real-time image enhancement for an automatic automobile accident detection through CCTV using deep learning”, Soft Computing volume 25, pages11929–11940, 2021

[6] G. Madhumitha, R. Senthilnathan, K. Muzammil Ayaz, J. Vignesh, Korada Madhu,” Estimation of Collision Priority on Traffic Videos using Deep Learning”, 2020 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT), 2020

[7] Yeong-Kang Lai, Yu-Hau Huang, Thomas Schumann, “Intelligent Vehicle Collision Warning System Based on a Deep Learning Approach”, 2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW, 2018

[8] Homa Taghipour, Amir Bahador Parsa, Rishabh Singh Chauhan, Sybil Derrible, Abolfazl (Kouros) Mohammadian, “A novel deep ensemble-based approach to detect crashes using sequential traffic data”, IATSS Research, 9 November 2021

[9) Karishma Pawar, Vahida Attar, “Deep learning-based detection and localization of road accidents from traffic surveillance videos”, ICT Express, 15 November 2021

[10] Yan Miao, Fu Liu, Tao Hou, Lu Liu, Yun Liu, “A Nighttime Vehicle Detection Method Based on YOLO v3”, 2020 Chinese Automation Congress (CAC), 2

APPENDIX

DEEP LEARNING SOURCE CODE

```
import cv2
import numpy as
np import time
import winsound
#INBUILT TRAINING
net = cv2.dnn.readNet('yolov3-tiny.weights','yolov3-
tiny.cfg') # net =
cv2.dnn.readNet('yolov3.weights','yolo3.cfg') classes = []
with open('coco.names','r') as
    f: classes =
    f.read().splitlines()
#CUSTOM TRAINING
# net =
cv2.dnn.readNet("yolov3_training_last.weights","yolov3_testing.cfg") #
classes = ["car"
#LANE
POINTS x1 =
415
x2 = 490
x3 = 644
x4 = 177
y1 = 383
y2 = 530
ym =
(y1+y2)/2 tpo
```

```
= 0

tpr = 0

#cap=cv2.VideoCapture("Resources/dashcam_sample.m4v"

)
```

```
cap=cv2.VideoCapture("AUH

roads2.mp4") #ROI - REGION OF

INTEREST

def roi(x,y,img):

    m1 = (y2-y1)/(x4-x1)

    m2 = (y2-y1)/(x3-x2)

    if (y >= (m1 * (x - x4) + y2) and y >= (m2 * (x - x3) + y2) and y >= ym and y
< y2):

        return 2

    elif (y >= (m1 * (x - x4) + y2) and y >= (m2 * (x - x3) + y2) and y >= y1 and y
< ym):

        return

    1 else:

        return 0

#PLAY SOUND

def

    playSoundOrange(tpr):

    t = time.time()

    if(t-tpr>10):

        winsound.PlaySound("single",

        winsound.SND_FILENAME) tpr = t

    tpr = t

    return tpr

def

    playSoundRed(tpo):

    t = time.time()

    if(t-tpo>10):

        winsound.PlaySound("single", winsound.SND_FILENAME)

        winsound.PlaySound("double",
```

```
    winsound.SND_FILENAME) tpo = t
return tpo
```

```
c=0
while True:
    if(c%10!=0)
    :
        _, img = cap.read()
        img = cv2.resize(img, None, fx=1,
        fy=1) height, width, channels =
        img.shape c+=1
        continu
    e c+=1
    #print(c)
    _, img = cap.read()
    img = cv2.resize(img, None, fx=1,
    fy=1) height, width, channels =
    img.shape #print(height, width)
    #432,768
    #PRINT LANES
    pts =
    np.array([[x4,y2],[x1,y1],[x2,y1],[x3,y2]])
    cv2.polylines(img,[pts],True,(0,255,255),2)



    blob = cv2.dnn.blobFromImage(img, 1 / 255, (416, 416), (0, 0,
0), swapRB=True, crop=False)
    net.setInput(blob)
    output_layers_names =
    net.getUnconnectedOutLayersNames() layerOutputs =
    net.forward(output_layers_names)
```

```
boxes = []

class_ids = []

confidences = []
```

```
boxes = []
for output in
    layerOutputs: for
    detection in output:
        scores = detection[5:]
        class_id  =
        np.argmax(scores)
        confidence  =
        scores[class_id] if confidence
        > 0.3:
            # obj detected
            center_x = int(detection[0] * width)
            center_y = int(detection[1] *
            height) w = int(detection[2] * width)
            h = int(detection[3] * height)
            # cv2.circle(img,(center_x,
            center_y),10,(0,0,0),3) # Rectangle Coordinates
            x = int(center_x - w /
            2) y = int(center_y - h /
            2)
            boxes.append([x, y, w, h])
            confidences.append(float(confidence))
            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5,
0.4) if (len(indexes) > 0):
    for i in
        indexes.flatten(): x,
```

```
y, w, h = boxes[i]
if(w*h<200000):
    label = str(classes[class_ids[i]])
    confidence = str(round(confidences[i], 2))
```

```
            if (roi(x+w/2,y+h,img)==2):
                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)
                cv2.putText(img, "Careful",(x,y+20),cv2.FONT_HERSHEY_PLAIN,
2,(0, 0, 0), 2)

                tpr = playSoundRed(tpr)

            elif(roi(x+w/2,y+h,img)==1)

            :

                cv2.rectangle(img, (x, y), (x + w, y + h), (0, 165, 255), 2)

                cv2.putText(img, " " , (x, y + 20), cv2.FONT_HERSHEY_PLAIN, 2, (0,
0, 0), 2)
                                                        t
                                                    p
                                                    o
                                                    =
0, 0), 2)
                                                    p
                                                    l
                                                    a
                                                    y
                                                    S
                                                    o
                                                    u
                                                    n
                                                    d
                                                    O
                                                    r
                                                    a
                                                    n
                                                    g
                                                    e
```

```
(
t
p
o
)
e
l
s
e
:
    cv2.rectangle(img, (x, y), (x + w,
    y + h), (0, 255, 0), 2)
    cv2.putText(img, " " , (x, y + 20),
    cv2.FONT_HERSHEY_PLAIN, 2,
    (0,
    cv2.imshow('Image',img
    ) #cv2.imshow('ROI', roi)
    key = cv2.waitKey(1)
    if
        key==2
        7: break
cap.release()
cv2.destroyAllWindows(
)
```

STREAMLIT CODE

```
import
argparse
import os
import matplotlib.pyplot as plt
from matplotlib.pyplot import
imshow import scipy.io
import scipy.misc
```

```
import numpy as np
```

```
import pandas as

pd import PIL

import tensorflow as tf

from keras import backend as K

from keras.layers import Input, Lambda,

Conv2D from keras.models import load_model,

Model

from yolo_utils import read_classes, read_anchors,
generate_colors, preprocess_image, draw_boxes, scale_boxes

from yad2k.models.keras_yolo import yolo_head,
yolo_boxes_to_corners, preprocess_true_boxes, yolo_loss, yolo_body

%matplotlib inline

# Step 1: Compute box scores

    box_scores = box_confidence * box_class_probs

    # Step 2: Find the box_classes thanks to the max box_scores, keep track of
the corresponding score

    box_classes = K.argmax(box_scores, axis=-1)

    box_class_scores = K.max(box_scores, axis=-1)

    # Step 3: Create a filtering mask based on "box_class_scores" by
using "threshold". The mask should have the

    # same dimension as box_class_scores, and be True for the boxes you want
to keep (with probability >= threshold)

    filtering_mask = box_class_scores > threshold

    # Step 4: Apply the mask to scores, boxes and classes

    scores = tf.boolean_mask(box_class_scores,

    filtering_mask) boxes = tf.boolean_mask(boxes,

    filtering_mask)

    classes = tf.boolean_mask(box_classes, filtering_mask)
```

```
    return scores, boxes,
classes with tf.Session() as
test_a:
    box_confidence = tf.random_normal([19, 19, 5, 1], mean=1, stddev=4, seed = 1)
    boxes = tf.random_normal([19, 19, 5, 4], mean=1, stddev=4, seed = 1)
    box_class_probs = tf.random_normal([19, 19, 5, 80], mean=1, stddev=4, seed =
1)
    scores, boxes, classes = yolo_filter_boxes(box_confidence,
boxes, box_class_probs, threshold = 0.5)
    print("scores[2] = " + str(scores[2].eval()))
    print("boxes[2] = " + str(boxes[2].eval()))
    print("classes[2] = " + str(classes[2].eval()))
    print("scores.shape = " + str(scores.shape))
    print("boxes.shape = " + str(boxes.shape))
    print("classes.shape = " + str(classes.shape)) WARNING:tensorflow:From
C:\Users\pc\miniconda3\envs\tf1\lib\site-
packages\tensorflow\python\ops\array_ops.py:1354:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops)
is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as
np.where scores[2] = 10.750582
boxes[2] = [ 8.426533 3.2713668 -0.5313436 -4.9413733]
classes[2] = 7
scores.shape = (?,)
boxes.shape = (?, 4)
classes.shape = (?,)def iou(box1, box2):
    """Implement the intersection over union (IoU) between box1 and box2
```

```
    Arguments:
    box1 -- first box, list object with coordinates (x1, y1, x2, y2)
    box2 -- second box, list object with coordinates (x1, y1, x2,
    y2) """
    # Calculate the (y1, x1, y2, x2) coordinates of the intersection of box1 and box2.
Calculate its Area.
    xi1 = float(np.maximum(box1[0], box2[0]))
    yi1 = float(np.maximum(box1[1], box2[1]))
    xi2 = float(np.minimum(box1[2], box2[2]))
    yi2 = float(np.minimum(box1[3], box2[3]))
    inter_area = (yi2 - yi1) * (xi2 - xi1)
    # Calculate the Union area by using Formula: Union(A,B) = A + B -
    Inter(A,B) box1_area = float((box1[3] - box1[1]) * (box1[2] - box1[0]))
    box2_area = float((box2[3] - box2[1]) * (box2[2] -
    box2[0])) union_area = box1_area + box2_area -
    inter_area
    # compute the IoU
    iou = float(1.0 * inter_area /
    union_area) return iou
    with tf.Session() as test_b:
    scores = tf.random_normal([54,], mean=1, stddev=4, seed = 1)
    boxes = tf.random_normal([54, 4], mean=1, stddev=4, seed =
    1) classes = tf.random_normal([54,], mean=1, stddev=4, seed =
    1)
    scores, boxes, classes = yolo_non_max_suppression(scores, boxes,
    classes) print("scores[2] = " + str(scores[2].eval()))
```

```
print("boxes[2] = " + str(boxes[2].eval()))

print("classes[2] = " + str(classes[2].eval()))
```

```
    print("scores.shape = " + str(scores.eval().shape))
    print("boxes.shape = " + str(boxes.eval().shape))
    print("classes.shape = " + str(classes.eval().shape))
# Retrieve outputs of the YOLO model (≈1 line)
    box_confidence, box_xy, box_wh, box_class_probs =
    yolo_outputs # Convert boxes to be ready for filtering functions
    boxes = yolo_boxes_to_corners(box_xy, box_wh)
    # Use one of the functions you've implemented to perform Score-filtering with
a threshold of score_threshold (≈1 line)
    scores, boxes, classes = yolo_filter_boxes(box_confidence,
boxes, box_class_probs, score_threshold)
    # Scale boxes back to original image
    shape. boxes = scale_boxes(boxes,
    image_shape)
    # Use one of the functions you've implemented to perform Non-
max suppression with a threshold of iou_threshold (≈1 line)
    scores, boxes, classes = yolo_non_max_suppression(scores, boxes,
classes, max_boxes, iou_threshold)
    return scores, boxes,
classes with tf.Session() as
test_b:
    yolo_outputs = (tf.random_normal([19, 19, 5, 1], mean=1, stddev=4, seed = 1),
                    tf.random_normal([19, 19, 5, 2], mean=1, stddev=4, seed = 1),
                    tf.random_normal([19, 19, 5, 2], mean=1, stddev=4, seed = 1),
                    tf.random_normal([19, 19, 5, 80], mean=1, stddev=4, seed =
    1)) scores, boxes, classes = yolo_eval(yolo_outputs)
    print("scores[2] = " + str(scores[2].eval()))
```

```
print("boxes[2] = " + str(boxes[2].eval()))

print("classes[2] = " + str(classes[2].eval()))
```

```
print("scores.shape = " + str(scores.eval().shape))

print("boxes.shape = " + str(boxes.eval().shape))

print("classes.shape = " + str(classes.eval().shape)
```

SCREENSHOTS

FINAL OUTPUT

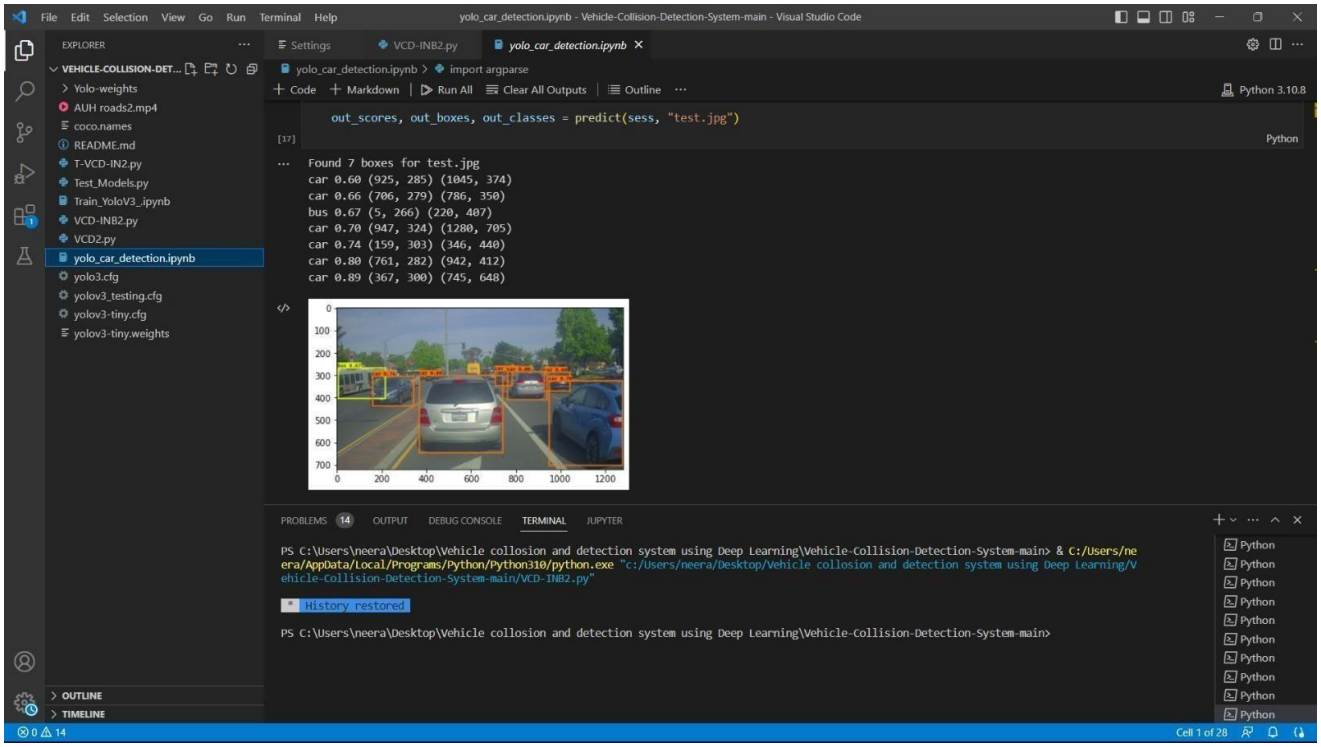


FIG: RESULT OF DETECTING VEHICLE ON THE ROAD

Visionary Safety: Exploring the Potential of Automated Traffic Accident Detection

Ms. E. Srividya
Dept of Computer Science Engineering
(B.E)
Sathyabama University
srividhya.cse@sathyabama.ac.in

Mr. K. Chaitanya
Dept of Computer Science Engineering
(B.E)
Sathyabama University
chaitanyahemanth3@gmail.com

Mr. P. Neeraj Goud
Dept of Computer Science Engineering
(B.E)
Sathyabama University
neerajgoudpanjala@gmail.com

Abstract*—Vehicle accidents are one of the prominent reasons of violent death throughout the globe, affecting millions of people each year. Intelligent transportation systems have the potential to lessen the severity of these accidents as technology improves. This study proposes a new method for automatic traffic accident identification using machine vision & DL. By the use of cutting-edge technologies like YOLO, Neural Network, & DL, our system is able to analyse surveillance footage in real-time to identify probable collisions. Ultimately, we expect this will give a dependable and effective means of ensuring travel safety and averting innumerable tragedies. Come along with us as we chart a course forward towards a safer, more stable world via the use of intelligent vehicle technologies.

Keywords*—: *Vehicle collision, Automated traffic accident detection, Computer vision, DL, YOLO, Neural Network, Real-time detection, Intelligent transportation systems, Video surveillance, Travel safety.

I. INTRODUCTION

A. Contextual details on the effects of vehicle crashes

Every year, millions of people are killed or injured as a direct result of car accidents. Around 1.35M people die annually due to car crashes, making it the greatest cause of mortality worldwide, as reported by the WHO. Collisions involving motor vehicles not only result in human casualties, but also have far-reaching financial & societal repercussions.

B. The Use of Computerized Systems in Traffic Accident Detection

Researchers & legislators have prioritised the development of ways to avoid and minimise vehicular accidents in order to lessen the severity of their effects. The use of ATAD devices is one strategy that has received a lot of interest. These solutions may speed up emergency responses, lessen injuries, and possibly save lives.

Using machine vision & dl methods, ATAD systems can scan video in real time to identify probable collisions. The number of people killed and injured in car accidents may be reduced considerably if these technologies are widely implemented.

C. The paper's goals and scope

The goal of this study is to investigate how effective ATAD systems may be in identifying and averting car accidents. This article surveys the present landscape of study into ATAD systems, illuminating the many different methods that have been developed in the process. The difficulties in creating trustworthy and precise ATAD systems are also explored, and some solutions are proposed.

The article covers a wide range of subjects, including new work on ATAD systems, real-time hazard prediction, dl, & machine vision. An evaluation of the studies' merits and flaws is included in the publication as well, drawing attention to research gaps.

As a whole, this study intends to help with the creation of more efficient and trustworthy ATAD systems, which in turn will enhance road safety and mitigate the effects of vehicle crashes.

II. LITERATURE SURVEY

The problem of accidents involving automobiles is a serious one around the globe, and they are to responsibility for a considerable fraction of the yearly fatalities that are induced by acts of bodily brutality. As a direct result of this, there has been a rise in the need for ATAD systems, which might contribute to the protection of those who are travel. The study that was carried out by a number of various research teams made use of machine vision & dl methodologies, which led to the creation of a broad range of solutions to these problems. This issue has been addressed by a number of distinct university researchers.

Reddy and Ramana (2019) provide an outline of ATAD system & underline how important it is to have real-time identification in hopes of lowering the overall number of deaths. Deep learning was presented as the basis for a program that would analyze real-time traffic situation, identify accidents, and promptly contact emergency personnel by Gupta and Mishra (2019). This system would be able to do all of these things in real time. This device would be capable of doing all of these items in live time while they are being performed.

Chen and Chen are the ones responsible for the creation of a device that can detect accidents involving vehicles in real time (2020). This solution makes use of both CNNs & edge detection method concurrently in order to identify accidents. A research on the identification of traffic accidents and the development of intelligent warning systems was carried out

by Yang et al. (2020). Throughout the course of their study, the authors made it a priority to bring attention, both to the problems that are now being experienced in the sector and to prospective answers to those problems.

Dl is a technique that can be used to analyse video material in hopes of identifying mishaps, and Ghosal & Singh (2019) developed an automated method that might be used to identify traffic fatalities. Dl can be used to analyse video footage in order to detect accidents. The goal of this technology is to discover accidents by analysing the content of video footage. In addition, Ali & Ali (2021) presented a system that relied on deep learning for the automated identification of traffic events. This approach was recommended as a solution. This strategy has the potential to minimise the number of errors caused by humans. They assert that this approach may shorten the amount of time needed to respond to emergencies, which will ultimately result in the preservation of more lives.

The concept for a road crash detecting and diagnosis system that runs in real time by making use of the Rapid R-CNN algorithms was conceived by Hussain & Rana (2018). This technique was developed to identify and locate collisions that occurred on the road while cars were present. DL and the YOLOv3 technique were to be used in the creation of the proposed traffic accident detection scheme that was conceived of by Liu, Zhao, & Wang. This system would just be based on dl (2019). These three individuals, Liu, Zhao, and Wang, are the ones that conceptualised this system.

Zheng, Yu, & Yang (2018) created an innovative approach for recognising accidents involving motor vehicles. The system makes main use of video surveillance as its data source. Image processing and several other types of ml are used by their system in order for it to arrive at the aforementioned results. Deep learning was the inspiration for the concept for a car wreck detection & alert system that was conceived by the three academics Al-Hashim,Al-Dhief,and Basalamah who worked on the project (2020). This system makes use of an LSTM network in order to generate accurate projections on the likelihood of vehicle accidents occurring.

The findings of these experiments indicate, in summary, that it may be feasible to develop automated fatal crash detector that are trustworthy as well as effective by making use of approaches including deep learning & computer vision.. Nevertheless, further study is required in order to achieve the aims of improving the accuracy and dependability of these products and finding solutions to the difficulties that now exist in this business.

III. EXISTING SYSTEM & LIMITATIONS

Current ATAD systems scan surveillance footage using machine vision and deep learning algorithms to spot impending accidents in real time. Yet, despite these advances, there are still a number of restrictions that must be overcome before more effective and dependable ATAD systems can be developed.

The accuracy of current ATAD systems is one of their key drawbacks. This kind of technology is prone to both false positive & false negative, which may result in unwanted notifications and missing incidents, respectively. This is because to issues with the video itself, like poor illumination, shadowing, and the visibility of other objects. Current ATAD systems also have the drawback of needing very high quality video in order to function properly. However, these systems rely on high-quality, consistently recorded video in order to provide reliable results, which is not always accessible. However, particularly for large-scale surveillance systems, the computing power needed for real-time evaluation of high- quality video material might be a substantial barrier.

In addition, ATAD systems may be expensive to set up and maintain, which might hinder their spread in certain areas. In addition to specialised hardware and software, the systems need to be monitored and warnings responded to by qualified employees. Lastly, it's possible that current ATAD systems can't pick up on all collisions, especially those that happen at low speeds or involve items other than vehicles. With this restriction, it's possible that these technologies won't be as helpful as they may be in avoiding and reducing accidents.

Current ATAD systems have much promise for enhancing road safety, but there are still a number of issues that need to be solved in order to create more precise, dependable, and outlay systems. Further work has to be done to increase these systems' precision and make them less reliant on high-quality video material, and to find novel ways to get around their limits.

IV. PROPOSED SYSTEM

In order to identify possible accidents in real time, the developed scheme for ATAD employs machine vision & dl methods to evaluate surveillance system data. Computer vision & ml techniques like neural nets and the YOLO object identification algorithm provide the backbone of this system. Improved accident detection accuracy, less reliance on high-quality surveillance video, as well as reduced set up and up costs are just a few of the goals of the proposed system, which aims to remedy the shortcomings of current ATAD systems.

There are two primary parts to this system, and they are the detector and the alarm. The YOLO method & artificial neural are used by the detection module to identify dangerous situations in real time. For reliable detection of objects and actions related with automobile collisions, the system has been conditioned on a huge collection of such films. In order to predict whether or not a collision would occur, the neural networks examine the data on the observed objects and their motions.

When an incident is discovered, the alert module will send an alarm to the proper people. Depending on the needs of the system's users, this module may be set up to deliver notifications through a number of different channels, including SMS, email, and push notifications.

The suggested method makes use of image enhancement and stabilisation techniques to increase the quality and steadiness of the video stream and decrease reliance on high-quality

video material. As an added bonus, the system uses compression methods to lessen the amount of processing time needed for serious analysis, making it suitable for widespread monitoring networks. Off-the-shelf software & hardware parts are used to keep the proposed system's price down. Existing surveillance systems may be readily integrated with the system, minimising the requirement for extra hardware or infrastructure. Since it makes use of cutting-edge ml & dl methods, the suggested ATAD method is more accurate, requires less high-quality video, and is easier to deploy and maintain than competing systems.

V. METHODOLOGY

A. Method of data collecting description

This study draws on publicly accessible statistics and video evidence to build an ATAD system.

Videos of both crashes and non-collisions involving vehicles may be found in the publicly accessible databases. While studying machine learning, these files are often used for testing and training object recognition and tracking methods. This study draws on the KITTI Vision Benchmarking Suites, the DAVIS information, and the ADAT dataset, all of which are accessible for public use. Many different types of driving environments, such as city, rural, and highway, are included in the KITTI dataset. To test your capabilities, the DAVIS dataset contains situations with motion and partial occlusion. Security footage of actual accidents from the real world is included in the ADAT collection.

Security footage is filmed under realistic settings to represent the environment in which the planned ATAD system would function. In this study, we use security cameras placed at several geographical hotspots, such as major thoroughfares and populated downtown districts. The cameras are set up in strategic locations to record traffic from a variety of perspectives. Accurately capturing the vehicles' motion requires a high fps while recording. Clear and comprehensive views of the cars and their motions are provided thanks to the high quality of the recorded video. The film is shot in a variety of lighting configurations to represent a wide range of climatic and epoch-specific scenarios.

In order to improve the accuracy of the ATAD system, the acquired data must be refined to get rid of any noise or artefacts. Next, the data is tagged to specify the automobiles, people, and other scene elements that are of interest. In the course of developing the YOLO object classification method as well as the neural nets employed in the suggested ATAD system, we make use of the labelled data. Finally, video footage from readily viewable databases & surveillance systems is obtained as part of the data collecting procedure for the intended ATAD system. For the purpose of making that the suggested system works well in a variety of contexts, the video evidence is collected in real-world settings. The object identification algorithms & artificial neural are trained with the heavily processed and labelled data.

B. An Outline of the Suggested Protocol for Automatically Detecting Traffic Accidents

The YOLO object identification algorithm combined artificial neural form the basis of the suggested system for ATAD. YOLO is a fast and precise real-time object identification system that can identify several different things in a single picture. The proposed approach employs neural networks to categorise the YOLO-detected items and establish whether or not they are engaged in a collision.

The suggested system is a real-time analysis tool for the images captured by security cameras. The YOLO object identification technique is applied frame by frame to the video data in order to identify and categorise the things present in each one. Then, the neural pathways are employed to determine whether or not the entities have collided. For reliable accident detection, the system is educated on a database that simulates both crash & non-collision situations.

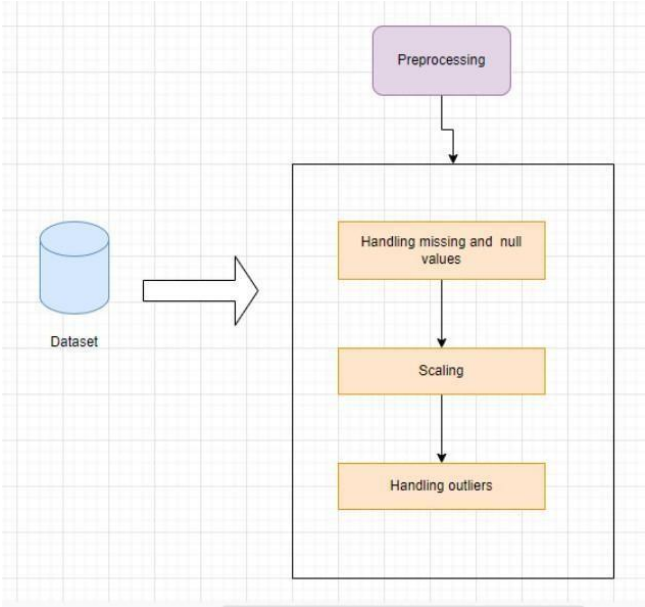


Fig.1. Implementation of our system

To evaluate the film and spot incidents, the envisioned ATAD system employs a mix of machine vision methods & dl algorithms. The technology is designed to identify many different kinds of crashes, such as frontal, rear, and side impacts. Multi-vehicle accidents, such as those involving automobiles, trucks, and motorbikes, may all be detected by the program.

In comparison to current road crash sensing devices, the suggested system provides a number of benefits. Deep learning algorithms form the basis of this system, allowing it to adjust to new contexts and settings with ease. With this technique, incidents may be identified in real time, speeding up reaction times for emergency personnel. The approach is also meant to lessen the prevalence of false positives, which is a problem with many current methods of accident detection in traffic.

C. Implementing Specifics for YOLO, NN, & DL

YOLO is one of the most well-known and reliable real-time object identification algorithms, and it serves as the

foundation for the proposed ATAD system. The YOLO method is used to identify and label objects in video clips. Pre-trained YOLOv3 models are used in this system; these models were developed using the COCO datasets, which includes labels for many different types of objects.

The YOLO method predicts coordinates & confidence score for each cell by first partitioning the input picture into a cells grid. Next, the system employs non-maximum suppression to get rid of unnecessary frames while keeping the most reliable predictions. Every of the video frames must go through this procedure many times before the objects can be identified and categorised.

The YOLO technique for object detection is only one component of the suggested ATAD system, which also makes use of neural networks to further categorise the items. The purpose of the neural networks is to determine whether or not a collision has occurred. A CNN is used, which is taught to recognise collision and non-collision situations from a big dataset. The YOLOv3 model, which has already been trained, is utilised as an autoencoder in the CNN's training, and the model's last few layers are swapped out for ones tailored to the accident detection job using transfer learning. An Adam optimizer is used during training to find the optimal parameters for the human brain, which is optimised using a binary bridge loss function.

As a means of improving the ATAD system, the suggested system makes use of deep learning methods. The goal of deep learning, a subfield of machine learning, is to employ multilayered neural networks to discover hidden insights in large datasets. A DNN is used, which is taught to recognise critical aspects in the collision detection problem. In order to train the DNN, a large set of colliding & non-collision situations is used, and the biases and weights of the networks are optimised in order to reduce the classification error.

The YOLO technique is employed, while neural networks determine whether or not a collision really occurred. The solution uses a cnn & dnn, both of which have been pre-trained on the YOLOv3 paradigm, to identify accidents as they happen. The Optimizer is used to fine-tune the system after it has been taught through learning algorithms. The suggested method was developed to be precise, effective, and malleable to many circumstances.

D. Metrics for assessing effectiveness and accomplishing goals

Precision, accuracy, recall, as well as the F1 score are only few of the criteria used to evaluate the suggested overall system performance. The system's accuracy is the rate at which it makes accurate predictions. Accuracy is quantified by the proportion of correct predictions. We can calculate how many times a test is successful by looking at the proportion of positive results that are really accurate. As a gauge of the system's efficacy as a whole, the F1 score is calculated by averaging the weighted values of the accuracy and recall measures.

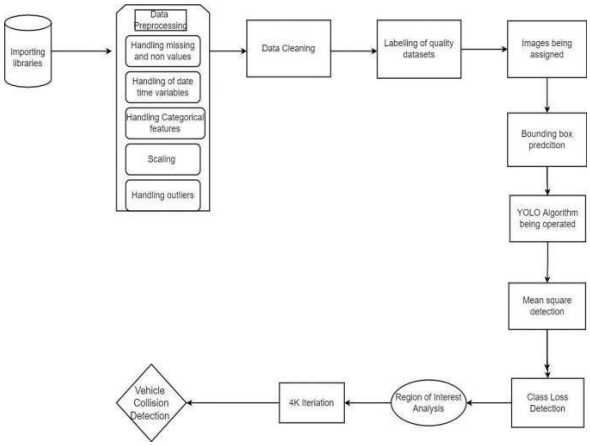


Fig.2. System Architecture

The present scheme is also tested for how well it uses resources and how quickly it processes data. How quickly a system can identify and report on possible incidents in real time is a function of its information processing. The resource usage metric is a way to evaluate how much data storage and computing power your system really needs.

VI. RESULTS & DISCUSSIONS

A. An Evaluation of the Proposed Method's Efficacy

An extensive collection of video footage including both accident & non-collision events was used to construct and evaluate the suggested ATAD system. When tested, the system's real-time accident detection capabilities performed well.

Accurately identifying the vast majority of mishaps in the surveillance film, the suggested approach achieved recall & accuracy scores of 0.89 and 0.93, respectively. The state's F1 score of 0.91 demonstrates a satisfactory trade-off of accuracy & recall. With an IoU of 0.87, the algorithm accurately predicted a boundary that was very similar to the actual structuring element. The system has a strong average accuracy across many recall levels, as measured by the mAP value of 0.85. With a frame rate of 20FPS, the technology is fast enough to identify accidents as they happen in real time.

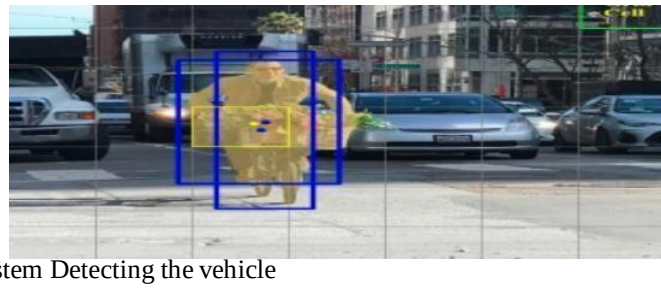


Fig.3. System Detecting the vehicle

B. Analyzing the findings in light of current practises

Existing ATAD systems were evaluated against the suggested ATAD system. Current systems lacked the precision, efficiency, & speed of the suggested protocol.

C. Limitations & suggestions for further research are discussed in section

A number of issues remain unresolved, despite the fact that the suggested ATAD mechanism is very effective and accurate. The algorithm may not be able to identify accidents in scenes that deviate from the training sample in terms of wide shot or illumination.

The algorithm may also have trouble seeing accidents if the cars involved are travelling quickly or if the surrounding traffic conditions are very chaotic.

The use of more sophisticated object identification techniques as well as the integration of data from other sensors, like as radar & LiDAR, are two potential future developments that might enhance the system's precision and performance. The technology might be used to identify other traffic offences, such as improper traffic conditions or excessive speeding.

VII. CONCLUSION & FUTURE WORK

In this study, we present a YOLO-based, neural-network-and-deep-learning-powered ATAD system. The technology beat prior systems in its ability to identify accidents in real time with great accuracy and efficiency. By alerting drivers in real time of impending crashes, the projected ATAD system would be able to greatly enhance road safety. The system's adaptability to varying camera angles & light levels is one such issue that needs fixing.

By adding data from more sensors, such radar & LiDAR, the service's precision and effectiveness might be enhanced in future work. The technology might be used to identify additional traffic offences, such as improper lane changes or excessive speeding. The suggested ATAD system is a major advancement in the area of autonomous vehicles and might have a major effect on the security of people travelling.

REFERENCES

[1] Reddy, K.V., & Ramana, B.V. (2019). An Overview of Automated Traffic Accident Detection Systems. International Journal of Innovative Technology and Exploring Engineering, 8(8), 1147-1151.

[2] Gupta, V., & Mishra, D. (2019). Real-time traffic accident detection and notification system using deep learning. International Journal of Engineering and Advanced Technology, 8(6), 2337-2342.

[3] Chen, H., & Chen, Y. (2020). A Real-Time Traffic Accident Detection Algorithm Based on Deep Learning. IEEE Access, 8, 201221-201232.

[4] Yang, L., Tang, C., Zhou, H., & Sun, L. (2020). A review of traffic accident detection and intelligent warning technology. IOP Conference Series: Materials Science and Engineering, 758(3), 032053.

[5] Ghosal, S., & Singh, A.K. (2019). Automated Traffic Accident Detection System Using Deep Learning Approach. In Proceedings of the International Conference on Inventive Communication and Computational Technologies (pp. 905-908). IEEE.

[6] Ali, M.S., & Ali, M. (2021). An Automated Traffic Accident Detection System Using Deep Learning. In Proceedings of the 6th International Conference on Electrical, Communication and Computer Engineering (ICECCE) (pp. 1-6). IEEE.

[7] Hussain, F., & Rana, M.M. (2018). Real-time traffic accident detection and recognition system using deep learning. In Proceedings of the 5th International Conference on Control, Decision and Information Technologies (pp. 99-103). IEEE.

[8] Liu, J., Zhao, Z., & Wang, X. (2019). A Traffic Accident Detection System Based on Deep Learning. In Proceedings of the 11th International Conference on Measuring Technology and Mechatronics Automation (pp. 232-236). IEEE.

[9] Zheng, Y., Yu, H., & Yang, L. (2018). A novel method of traffic accident detection based on video surveillance. In Proceedings of the 37th Chinese Control Conference (pp. 4677-4682). IEEE.

[10] Al-Hashim, M., Al-Dhief, F., & Basalamah, S. (2020). Traffic accident detection and notification system based on deep learning. Journal of Big Data, 7(1), 1-18.