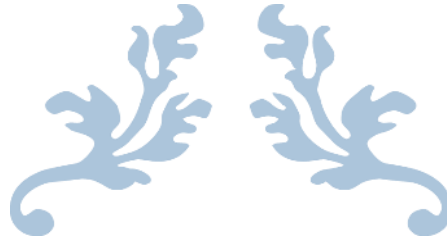




Arab Republic of Egypt  
Ministry of Communications  
and Information Technology

**DELL**Technologies



---

# Safe Intelligent Transportation in Digital Cities

---

---

*Scientific Computing Department  
Faculty Of Computer and Information Science  
Ain Shams University*

***Team Member***

*Osama Anter Mohamed Afify  
Tarek Ashraf Mahmoud Hussein  
Ahmed Mohamed Ali Abdelrahman  
Ahmed Mohamed Ibrahim Mohamed  
Adham Mohamed Tawfik Mohamed*

***Under Supervision:***

*Dr. Ayat Mohammed  
TA. Marwa Shams*

---



## Acknowledgements

We would like to express our heartfelt thanks to Allah, the Almighty, who gave us the strength, wisdom, and perseverance needed to complete this project. Without his divine guidance and blessings, this endeavor would not have been possible.

and too, we would like to express our gratitude to several individuals and organizations who have contributed to the success of this project. Firstly, we would like to thank **Dr. Ayat Mohammed** and **TA Marwa Shams** for their valuable guidance and support throughout the project.

We cannot forget the role of **Dr. Mona Abdelazim** for her efforts.

We are grateful to **Dell Technologies** for providing us with the necessary resources to carry out this research, specially by **Eng. Omar Nael**, we are deeply grateful for their assistance, and we look forward to continuing to work with them in the future.

Finally, we would like to thank the **Ministry of Communications and Information Technology (MCIT)** for their support and encouragement of our research.

We would also like to extend our thanks to all those who have supported us throughout this project, including our family, friends, and colleagues. Their encouragement, feedback, and assistance have been invaluable, and we are grateful for their contributions to this endeavor.

Once again, I offer my sincerest thanks to Allah for his grace and mercy, and for guiding me through this project. I pray that he continues to bless me with his guidance and support in all my future endeavors.

## **Abstract**

Traffic accidents can be extremely dangerous and can result in serious consequences such as injury, disability, or even death. The impact of a collision can cause severe damage to the vehicles involved, as well as other objects or structures nearby. In addition, passengers and pedestrians may suffer serious injuries, ranging from broken bones to traumatic brain injuries or spinal cord injuries.

Traffic accidents can also have significant economic consequences, including medical expenses, lost earnings, and property damage. The cost of traffic accidents can be particularly high in developing countries where road safety is often not a priority, and where many people rely on unsafe modes of transportation.

When accidents occur on the road, their discovery is frequently delayed, saving the situation and this results in numerous disasters, similar as fresh accidents or the death of some people, business dislocation and people's interests, and much further.

In this project, a system is developed to automatically detect accidents related to vehicles in images and videos.

We trained our system to detect such events. Automatic Car crash detector can save lives, as an accident would be immediately reported by the system and help will reach the reported sites to save people's lives and reduce the Time Delay in Requesting The appropriate Service.

**Keywords:** accident detection, Computer Vision, Deep Learning, Object detection, and Yolo Model.

## Table of Contents

|  |    |
|--|----|
| Acknowledgements .....                                   | 1  |
| Abstract .....   | 2  |
| List of figures: .....                                   | 6  |
| List of Abbreviations .....                              | 8  |
| Chapter 1 Introduction .....                             | 9  |
| 1. Problem Definition .....                              | 10 |
| 2. Motivation .....                                      | 11 |
| 3. Objectives .....                                      | 12 |
| 4. Time plan .....                                       | 13 |
| 5. Documentation Outline .....                           | 13 |
| Chapter2 Background .....                                | 15 |
| 1. Project field.....                                    | 16 |
| 2. Project Background .....                              | 16 |
| 3. Algorithms and Techniques Required .....              | 18 |
| 4. scientific background.....                            | 19 |
| 5. Relative work.....                                    | 20 |
| Chapter3 System Architecture .....                       | 24 |
| 1. Intro.....  | 25 |
| 2. The reason for using Yolo8 .....                      | 25 |
| 2.1 How does YOLOv8 compare to previous models? .....    | 26 |
| 3. network architecture and design.....                  | 28 |
| 3.1 New convolutions in YOLOv8 .....                     | 29 |
| 3.2 Anchor-free Detections .....                         | 30 |
| Chapter4 Dataset Collection and Preparation .....        | 32 |
| 1. Description of the dataset used in the project: ..... | 33 |
| 2. Data collection methodology and sources:.....         | 33 |

|   |    |
|---|----|
| 3. Data preprocessing and augmentation techniques used: ..... | 34 |
| Chapter5 System Implementation.....                           | 36 |
| 1. Introduction .....   | 37 |
| 2. Create a Project.....                                      | 37 |
| 3. Upload Data.....   | 38 |
| 4. Create a Dataset Version .....                             | 39 |
| 5. Preprocessing.....   | 40 |
| 6. Generate Augmented Images .....                            | 40 |
| 7. Health Check .....   | 41 |
| 8. Class Balance.....   | 42 |
| 9. Export Data.....   | 42 |
| 10. Annotation Tools .....                                    | 44 |
| 11. Installing YOLOv8.....                                    | 45 |
| 12. Training the Model.....                                   | 46 |
| Chapter6 System Testing and result .....                      | 47 |
| • Results .....   | 52 |
| Chapter7 Project Management.....                              | 54 |
| Project Planning:.....  | 55 |
| Scheduling.....   | 55 |
| Progress Tracking: .....                                      | 55 |
| Challenges Faced: .....                                       | 56 |
| Addressing Challenges: .....                                  | 56 |
| Chapter8 User Interface and Deployment .....                  | 57 |
| 1. Design and Development of the User Interface: .....        | 58 |
| 2. Deployment of the System on a Web Platform:.....           | 59 |
| 3. User Testing and Feedback: .....                           | 61 |
| Chapter9 Discussion .....                                     | 63 |
| • Discussion.....   | 64 |
| • Comparing our results with existing systems.....            | 65 |

|  |    |
|--|----|
| Chapter10 Conclusion and Future Work ..... | 66 |
| 1. Conclusion .....                        | 67 |
| 2. Future Work.....                        | 69 |
| Tools.....                                 | 72 |
| 1. Programing Language: .....              | 72 |
| 2. Platform: .....                         | 72 |
| .....                                      | 72 |
| 3. Libraries.....                          | 77 |
| 7. Framework.....                          | 77 |
| 8. GPU .....                               | 80 |
| 9. Software.....                           | 83 |
| References .....                           | 86 |

## List of figures:

|  |    |
|--|----|
| Figure 1 chain of accidents .....                    | 10 |
| Figure 2 Timing responses to accident detection..... | 10 |
| Figure 3 System Architecture .....                   | 25 |
| Figure 4 YOLOv8 vs. previous YOLO versions .....     | 26 |
| Figure 5 YOLOv8 COCO evaluations.....                | 27 |
| Figure 6 YOLOv8 architecture .....                   | 28 |
| Figure 7 C2F architecture .....                      | 29 |
| Figure 8 Anchor Box.....                             | 30 |
| Figure 9 Anchor boxes.....                           | 31 |
| Figure 10 Dataset used in project.....               | 33 |
| Figure 11 Split dataset.....                         | 34 |
| Figure 12 Data Augmentations .....                   | 35 |
| Figure 13 Create Roboflow project.....               | 37 |
| Figure 14 Upload images .....                        | 38 |
| Figure 15 Create dataset versions .....              | 39 |
| Figure 16 Health Check .....                         | 41 |
| Figure 17 Export data.....                           | 42 |
| Figure 18 Export format.....                         | 43 |
| Figure 19 Download code .....                        | 43 |
| Figure 20 Annotation tool.....                       | 44 |
| Figure 21 Clone from GitHub.....                     | 45 |
| Figure 22 Install with pip.....                      | 45 |
| Figure 23 Training Model.....                        | 46 |
| Figure 24 Rear end collisions .....                  | 48 |
| Figure 25 Intersection collisions.....               | 49 |
| Figure 26 Head-on collisions .....                   | 50 |
| Figure 27 Cyclist accidents.....                     | 50 |
| Figure 28 Side-impact collisions .....               | 51 |
| Figure 29 Single-vehicle accidents.....              | 52 |
| Figure 30 Results .....                              | 53 |
| Figure 31 GUI.....                                   | 58 |
| Figure 32 GUI.....                                   | 59 |
| Figure 33 GUI.....                                   | 60 |

|   |    |
|---|----|
| Figure 34 GUI Try Model.....            | 60 |
| Figure 35 Upload Image in website ..... | 61 |
| Figure 36 Output .....                  | 62 |
| Figure 37 Python.....                   | 72 |
| Figure 38 Colab.....                    | 72 |
| Figure 39 Ultralytics .....             | 83 |
| Figure 40 Roboflow .....                | 84 |



## **List of Abbreviations**

- **Ai → Artificial intelligence**
- **ML → Machine learning**
- **DL → Deep Learning**
- **CV → Computer Vision**
- **Yolo → You Only Look Once**
- **Open CV → Open-Source Computer Vision**
- **PIL → Python Imaging Library**
- **IoU → Intersect over union.**
- **NMS → non maximum suppression**
- **mAP → Mean average precision**

# **Chapter 1**

## **Introduction**

## 1. Problem Definition

- Traffic accidents pose a significant public health and safety problem worldwide. They can have devastating consequences for individuals and families, including physical injuries, emotional trauma, and financial hardship. In addition, traffic accidents can cause disruptions to traffic flow, damage to infrastructure, and can result in significant economic costs to society.
- Could sudden utility incidents such as pipeline bursts, which deal more damage if not contained in a timely fashion.
- Timely accident detection and taking immediate action with respect to emergency health care of victims by informing an emergency center such as a hospital or a police station about the accident on time plays a vital role in human safety and road traffic management.
- The effects of traffic accidents can be significant and long-lasting. Reducing the incidence and severity of traffic accidents is essential to promoting public safety and ensuring that individuals and communities can thrive.



Figure 1 chain of accidents



Figure 2 Timing responses to accident detection

## **2. Motivation**

The motivation behind addressing the problem of traffic accidents is to improve public safety and reduce the harm caused by these accidents. Traffic accidents are a major public health issue around the world, causing significant physical, emotional, and financial harm to individuals and communities. They can also cause disruption to traffic flow, damage to infrastructure, and lead to significant economic costs.

Early detection of traffic accidents can also help to prevent secondary accidents from occurring. For example, if an accident occurs on a busy highway, other drivers may not be aware of the accident until it is too late, leading to a chain reaction of accidents. However, if the accident is detected early, steps can be taken to warn other drivers and prevent additional accidents from occurring.

Furthermore, early detection of traffic accidents can help to improve traffic flow and reduce congestion. When accidents occur, they can cause significant traffic backups and delays, which can have a ripple effect on the entire transportation system. By detecting accidents early and responding quickly, traffic flow can be restored more quickly, minimizing the impact on commuters and businesses.

Overall, the motivation behind detecting traffic accidents early is to minimize the harm caused by accidents, prevent secondary accidents, and improve traffic flow. Early detection can help to ensure that emergency services can respond quickly and effectively to accidents, which can ultimately save lives and reduce the economic and social impact of accidents.

### 3. Objectives

- **Improve public safety:**

Early detection of accidents can help to ensure that emergency services can respond quickly and effectively, potentially saving lives and reducing the severity of injuries.

- **Minimize traffic congestion:**

Traffic accidents can cause significant traffic backups and delays, which can have a ripple effect on the entire transportation system. By detecting accidents early and responding quickly, traffic flow can be restored more quickly, minimizing the impact on commuters and businesses.

- **Reduce economic costs:**

Traffic accidents can be costly, both in terms of medical expenses and property damage. By detecting accidents early and responding quickly, the economic impact of accidents can be minimized.

- **Enhance transportation system efficiency:**

Traffic accidents can disrupt the transportation system, causing delays and reducing efficiency. By detecting accidents early and responding quickly, the transportation system can be restored more quickly, improving efficiency, and reducing the impact on commuters and businesses.

- **Promote sustainable transportation:** By reducing the incidence of traffic accidents, the use of sustainable transportation modes such as walking, cycling and public transit can be encouraged. This can help to reduce greenhouse gas emissions and promote a more sustainable transportation system.

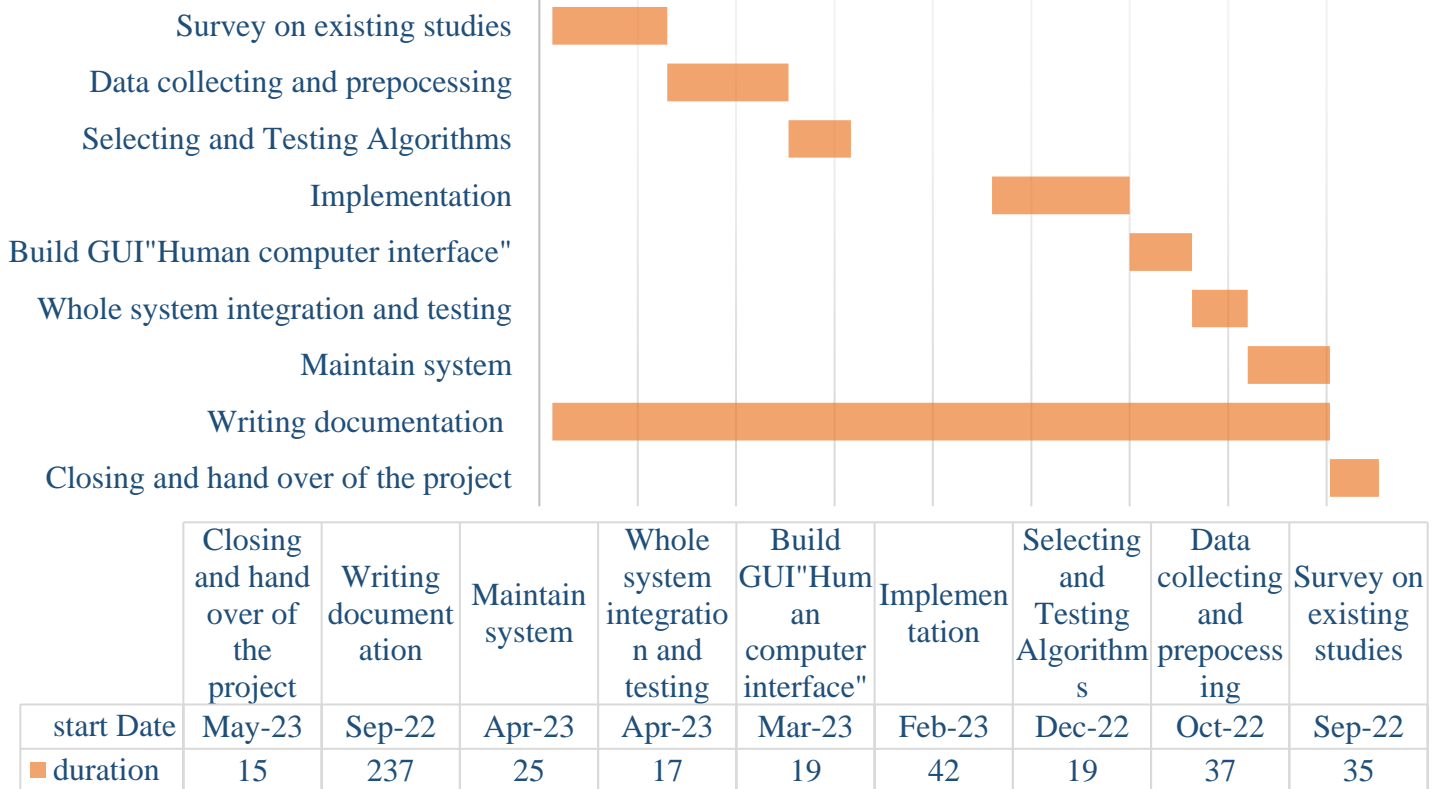
Overall, the objectives of traffic accident detection are to improve public safety, minimize traffic congestion, reduce economic costs, enhance transportation system efficiency, and promote sustainable transportation. By achieving these objectives, we can help to create

a safer, more efficient, and sustainable transportation system for everyone.

## 4. Time plan

### TIME PLAN

16-Sep 16-Oct 15-Nov 15-Dec 14-Jan 13-Feb 15-Mar 14-Apr 14-May



## 5. Documentation Outline

- **Chapter2:** Contains a description of the project field, Project Background, literature review, algorithms, techniques required, and scientific background related to the project, and Relative work.

- **Chapter 3:** the overall design and structure of the system. It defines how the different components of the system are organized, how they interact with each other, and how information flows within the system.
- **Chapter 4:** Dataset collection and preparation are crucial steps in the process of building machine learning models. These steps involve gathering relevant data, organizing it in a suitable format, and performing necessary preprocessing.
- **Chapter 5:** refers to the process of developing and deploying a system based on the defined system architecture and requirements.
- **Chapter 6:** Testing is a crucial phase in the software development life cycle, aimed at identifying and fixing defects or issues in the system before it is deployed to production. It involves executing various tests to ensure that the system meets the specified requirements.
- **Chapter 7:** describe the discipline of planning, organizing, and managing resources to achieve specific goals and objectives within defined constraints, such as time, budget, and scope.
- **Chapter 8:** describe the visual and interactive elements of the system that enable users to interact with it.
- **Chapter 9:** discussion
- **Chapter 10:** conclusion

# **Chapter2**

## **Background**



## **1. Project field**

Car accident detection is an important field of study that has applications in various industries, including transportation, automotive, and insurance. Here are some potential project fields for car accident detection:

- The project field is traffic accidents detection, which involves using machine learning algorithms and computer vision techniques to detect and analyze traffic accidents. The goal of this project is to improve the safety of our roads by identifying and responding to accidents more quickly and efficiently.
- Transportation: Car accident detection can be used to improve traffic management and reduce congestion. For example, sensors and cameras can be used to detect accidents and automatically reroute traffic to avoid the affected area.
- Emergency services: Car accident detection can be used to improve emergency response times. Sensors and cameras can be used to automatically detect accidents and alert emergency services, allowing them to respond more quickly.
- Research: Car accident detection can be used to study the causes and effects of car accidents, and to develop new methods for preventing them. This can help inform policy decisions and improve safety regulations.

Overall, car accident detection has many potential applications and is an important field of study for improving the safety and efficiency of transportation.

## **2. Project Background**

The project involves developing a system that can automatically detect traffic accidents in real-time using video data from traffic

cameras. The system will use computer vision algorithms to analyze the video footage and identify potential accidents based on factors such as sudden changes in traffic flow, the presence of debris on the road, and the behavior of nearby vehicles.

Once an accident is detected, the system will alert emergency services, such as police, fire, and ambulance, to respond quickly and aid. The system will also provide real-time updates to drivers and other road users, allowing them to adjust their routes and avoid the accident site.

The traffic accidents detection project has the potential to significantly reduce the number of accidents on our roads and improve the safety of drivers, passengers, and pedestrians. By using sophisticated machine learning algorithms and computer vision techniques, we can create a more efficient and effective response to accidents, saving lives and reducing the impact of accidents on our communities.

### 3. Algorithms and Techniques Required

- **Computer Vision Algorithms:** Computer vision algorithms are essential for detecting and analyzing traffic accidents in real-time. Techniques such as object detection, tracking, and segmentation can be used to identify vehicles, pedestrians, and other objects on the road.
- **Machine Learning Algorithms:** Machine learning algorithms such as neural networks, decision trees, and support vector machines can be used to train models on large amounts of data and make accurate predictions about potential accidents.
- **Image Processing Techniques:** Image processing techniques such as edge detection, filtering, and morphological operations can be used to enhance the quality of the video footage and improve the accuracy of the accident detection system.
- **Data Preprocessing Techniques:** Data preprocessing techniques such as data cleaning, normalization, and feature scaling can be used to prepare the data for analysis and improve the performance of the machine learning models.
- **Real-time Processing Techniques:** Real-time processing techniques such as parallel processing, multi-threading, and GPU acceleration can be used to process the video data in real-time and provide immediate alerts to emergency services and other road users.

- **Cloud Computing:** Cloud computing can be used to store and process the large amounts of data generated by the traffic cameras and the accident detection system. Cloud-based solutions also provide scalability and flexibility, allowing the system to adapt to changing traffic conditions and data volumes.

By using a combination of these algorithms and techniques, we can develop a robust and efficient traffic accidents detection system that improves the safety of our roads and saves lives.

#### **4. scientific background**

Traffic accidents detection is a field that draws on several different scientific disciplines, including computer vision, machine learning, and transportation engineering.

- **Computer vision** is a subfield of artificial intelligence that focuses on enabling machines to interpret and analyze visual information from the physical world. This involves developing algorithms and techniques for image and video processing, object detection and recognition, and pattern recognition, which are essential for detecting and analyzing traffic accidents.
- **Machine learning** is a subfield of artificial intelligence that involves developing algorithms and techniques that enable machines to learn from data and make predictions or decisions without being explicitly programmed to do so. Machine learning techniques are used in traffic accidents detection to train models on large amounts of data and make accurate predictions about potential accidents.

- **Transportation engineering** is a field that focuses on the design, construction, and operation of transportation systems, including roadways, bridges, and tunnels. This field is important for traffic accidents detection because it provides a comprehensive understanding of traffic patterns, behavior, and safety, which can inform the development of algorithms and techniques for detecting and responding to accidents.

Together, these scientific disciplines provide the foundation for developing robust and effective traffic accidents detection systems. By combining computer vision and machine learning techniques with transportation engineering knowledge, researchers can develop systems that are capable of detecting accidents quickly and accurately and providing timely alerts to emergency services and other road users.

## **5. Relative work**

Over recent decades, extensive research has been conducted in the field of intelligent transportation systems that are focused on developing automatic incident detection systems for handling the many day-to-day occurrences within these systems, such as accidents, traffic congestion, and jams. For truly secure smart cities, it remains crucial to attain real-time situational awareness, despite the innovations that have sparked smart city innovation over the past several decades.

In [3], a computationally inexpensive three-stage deep learning-based architecture was proposed to detect car accidents accurately and automatically with minimum hardware requirements. Accordingly, in the first stage, 200,000 raw images are down sampled, and Gaussian noise is

added to detect vehicles in a moving traffic environment using the Mini-YOLO object detection algorithm [3]. The detected vehicles are then transferred to the vehicle tracking stage to track multiple vehicle objects in the video frame and keep track of each vehicle's damage status in case an accident occurs. The final stage is the classification stage, where the authors in [3] trained the RF, CNN, and SVM algorithms to effectively classify the vehicle images into either the damaged or the undamaged classes. The experimental results that were retrieved from [3] state that in the tracking stage, the Mini-YOLO model attained an AP score of 34.2 and a runtime of 28 frames per second. Additionally, in the classification stage, the SVM with the radial basis function kernel attained a precision score of 96%, a recall of 94%, and an AUC score of 96% [3].

In [8], an object detection algorithm known as YOLOv3 was utilized to detect abnormal situations on the roads and to sufficiently avert secondary accidents. A real-time notification application was constructed by implementing AI CCTV. Accordingly, the FFmpeg software was productively leveraged to capture 700 frames of vehicle accidents from a series of vehicle collision videos to construct the dataset. A rotation of 90 and 180 degrees was performed on the image dataset to relatively increase its size to 2000 images. Thus, the custom weights of the proposed YOLOv3 model obtained a mean average precision of 82.36% and an intersection over union threshold of approximately 50% [8]. The finalized deep learning model was embedded into Django and Flask servers, and a warning alert was then sent via a Firebase Cloud Messaging (FCM) platform upon the occurrence of an accident or collision.

Likewise, researchers in [9,10] also aimed to present an efficacious solution to lessen and reduce the overall road accident rate on highways by conducting a deep learning-based accident detection system. To

effectuate the system, CCTV cameras were mounted on highways. In [9], convolutional neural networks (CNNs), which work upon the ReLU and Sigmoid activation layers and a loss function to eliminate any noise that may have gathered within the road accident video frames, were used, whereas in [10], a DL model that combined CNN (inception v3) and LSTM was leveraged to classify whether or not an accident had occurred in the video frame. The model in [10] was then implemented on a Raspberry Pi using Keras, TensorFlow, and OpenCV. Furthermore, in [10], a GSM module was obtained to create an alarm system to send an SMS to the nearest hospital or police station if the prediction exceeded a threshold of 60%, whereas in [9], an email alert was used instead of an SMS message. The alert message in both [9,10] contained information about the accident's time occurrence, the location of the accident, and the frame for further analyses. Both [9,10] utilized a CNN-based architecture, hence, the proposed experimental work in [9] attained an accuracy of 93%, and in [10], an accuracy averaging 92.38% was achieved. In [11], the paper introduced a supervised deep learning framework solution to establish a car crash detection system, which can function smoothly and transmit critical information to the appropriate authorities without any delay. The dataset used consisted of CCTV video clips from YouTube of different car crash conditions from different Middle Eastern regions. The car crash detection system was divided into three phases: vehicle detection, vehicle tracking and feature extraction, and accident detection. In the first phase, the Mask RCNN (Region-based Convolutional Neural Networks) was used to segment and build pixel-by-pixel masks for each item in the video. The Centroid Tracking algorithm was then used to effectively track the vehicle to observe the cause of the accident, which was then classified as being due to speed acceleration, trajectory anomaly, or change in angle anomaly. A 71% detection rate and 0.53% false alarm rate using the accident videos were successfully obtained under different

surrounding environmental conditions [11]. Various intelligent methods were applied in the field of medicine and other vital fields as detailed in the research work [12,13,14,15,16].



# **Chapter3**

## **System Architecture**

## 1. Intro

The cars accident detection system is designed to detect accidents that occur on the road and alert emergency services immediately, allowing for a faster response time. In this project, we designed a website that can take videos as input to detect accidents in any frame, using the YOLO v8 model in Object Detection.

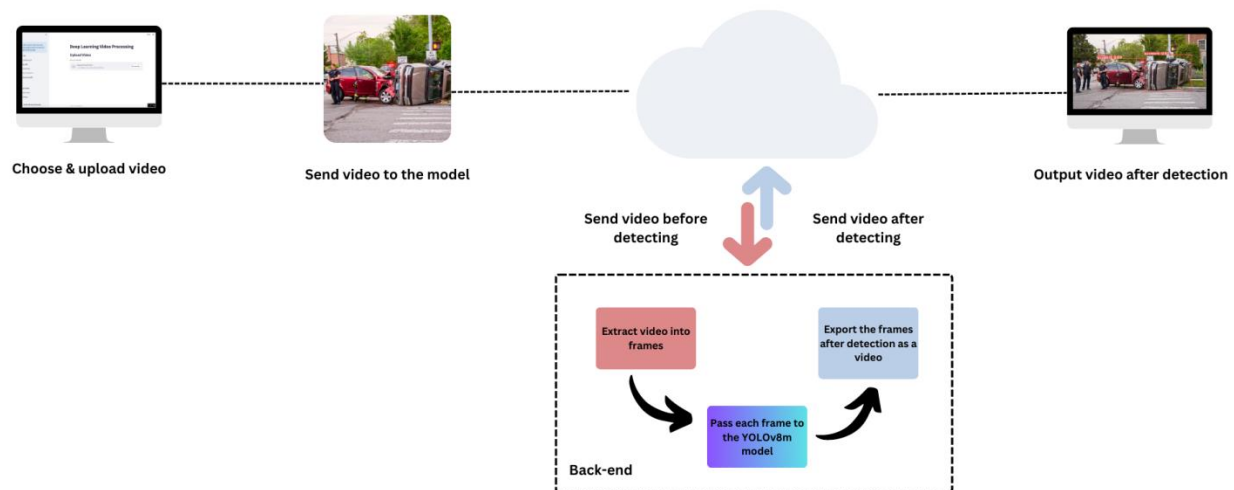


Figure 3 System Architecture

## 2. The reason for using Yolo8.

1. YOLOv8 has better accuracy than previous YOLO models.
2. YOLOv8 has a high rate of accuracy measured by COCO and Roboflow 100.
3. It supports object detection, instance segmentation, and image classification.
4. There is a large community around YOLO and a growing community around the YOLOv8 model, meaning there are many people in computer vision circles who may be able to assist you when you need guidance.

## 2.1 How does YOLOv8 compare to previous models?

The Ultralytics team has once again benchmarked YOLOv8 against the COCO dataset and achieved impressive results compared to previous YOLO versions across all five model sizes.

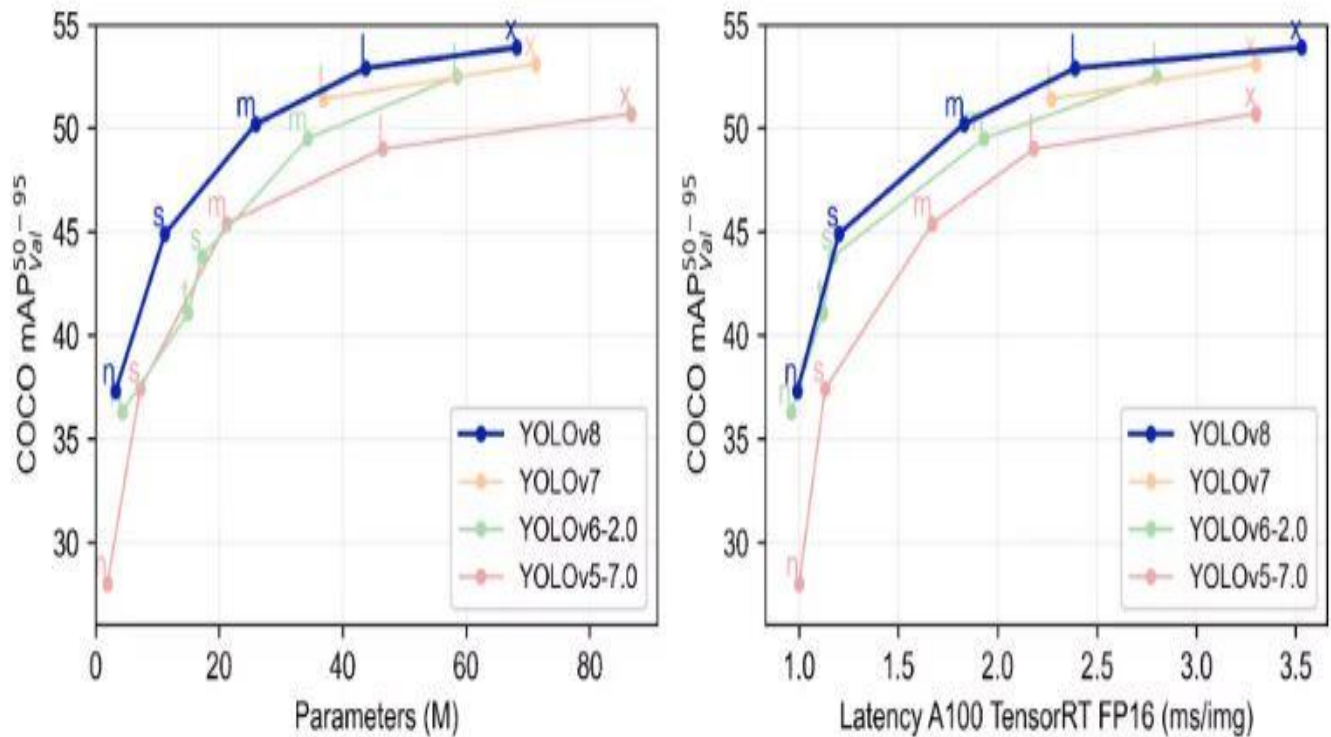


Figure 4 YOLOv8 vs. previous YOLO versions

For the object detection comparison of the 5 model sizes The YOLOv8m model achieved a mAP of 50.2% on the COCO dataset, whereas the largest model, YOLOv8x achieved 53.9% with more than double the number of parameters.

YOLOv8's high accuracy and performance make it a strong contender for your next computer vision project.

▼ Detection

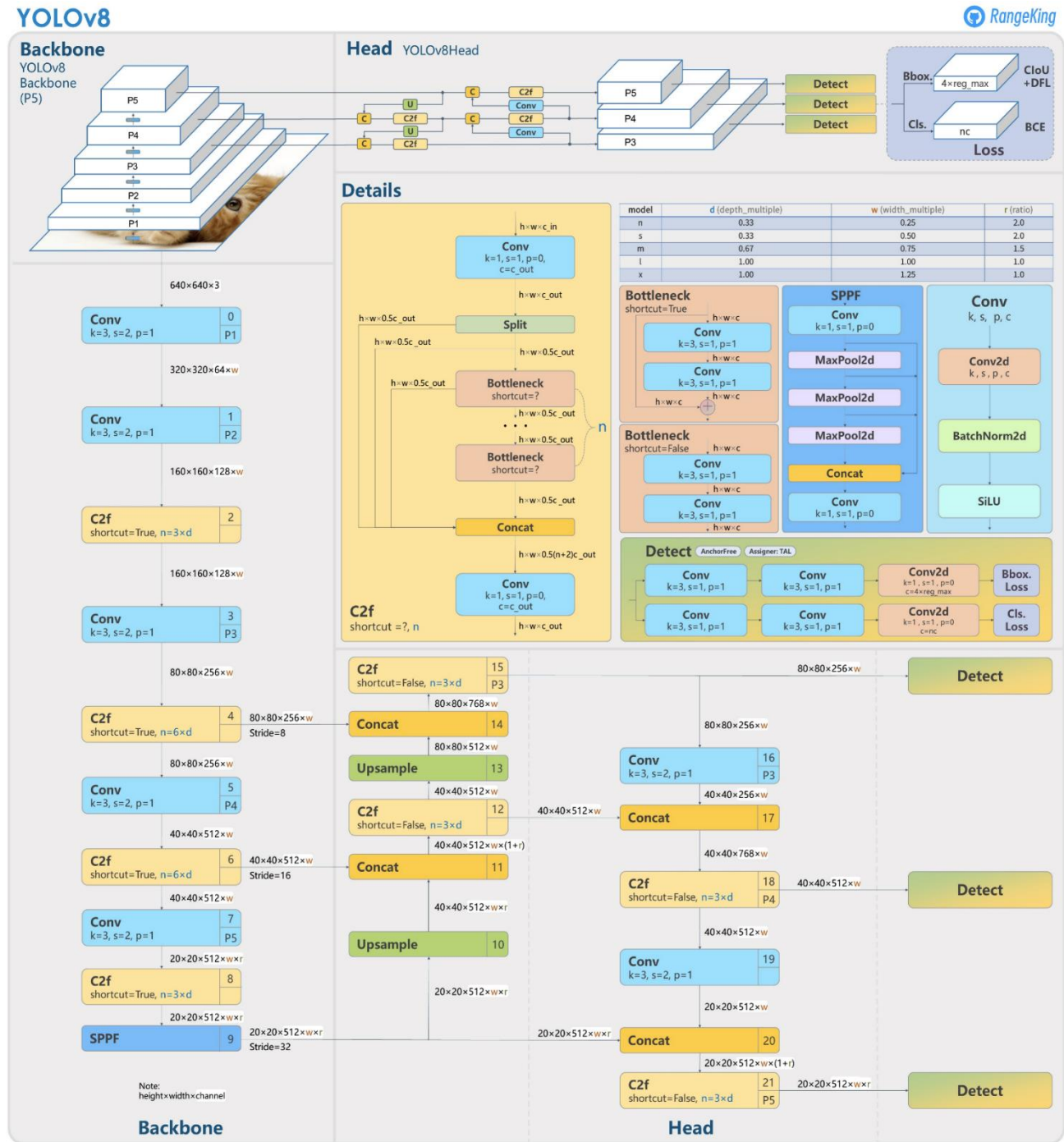
| Model   | size<br>(pixels) | mAP <sup>val</sup><br>50-95 | Speed<br>CPU<br>(ms) | Speed<br>T4 GPU<br>(ms) | params<br>(M) | FLOPs<br>(B) |
|---------|------------------|-----------------------------|----------------------|-------------------------|---------------|--------------|
| YOLOv8n | 640              | 37.3                        | -                    | -                       | 3.2           | 8.7          |
| YOLOv8s | 640              | 44.9                        | -                    | -                       | 11.2          | 28.6         |
| YOLOv8m | 640              | 50.2                        | -                    | -                       | 25.9          | 78.9         |
| YOLOv8l | 640              | 52.9                        | -                    | -                       | 43.7          | 165.2        |
| YOLOv8x | 640              | 53.9                        | -                    | -                       | 68.2          | 257.8        |

- mAP<sup>val</sup> values are for single-model single-scale on [COCO val2017](#) dataset.  
Reproduce by `yolo mode=val task=detect data=coco.yaml device=0`
- Speed averaged over COCO val images using an [Amazon EC2 P4d](#) instance.  
Reproduce by `yolo mode=val task=detect data=coco128.yaml batch=1 device=0/cpu`

**Figure 5 YOLOv8 COCO evaluations**

### 3. network architecture and design

The following layout shows a detailed visualization of the network's architecture.



### 3.1 New convolutions in YOLOv8

There are a series of updates and new convolutions in the YOLOv8 architecture according to the introductory post from Ultralytics:

1. The backbone of the system underwent changes with the introduction of C2f, replacing C3. The first 6x6 convolution in the stem was switched to a 3x3 convolution. In C2f, outputs from the Bottleneck (which is a combination of two 3x3 convs with residual connections) are combined, whereas in C3 only the output from the last Bottleneck was utilized.
2. Two convolutions (#10 and #14 in the YOLOv5 config) were removed.
3. The Bottleneck in YOLOv8 remains the same as in YOLOv5, except the first convolution's kernel size was changed from 1x1 to 3x3. This change indicates a shift towards the ResNet block defined in 2015.

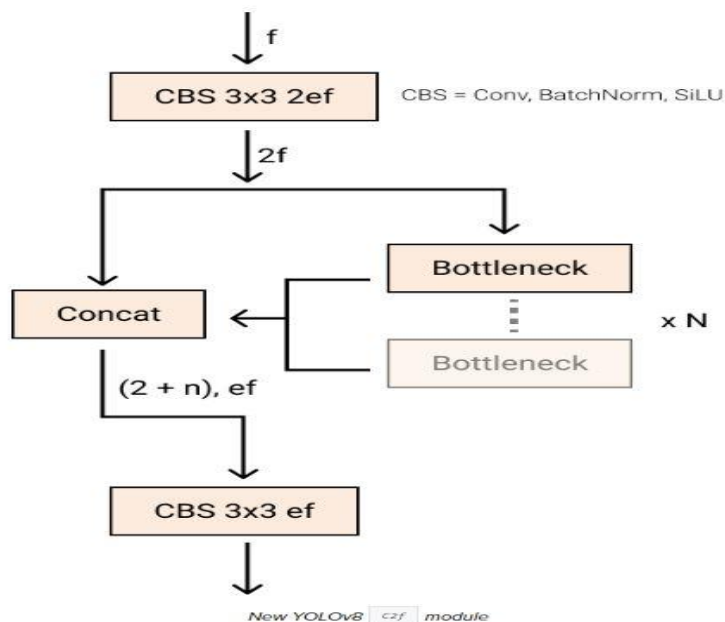
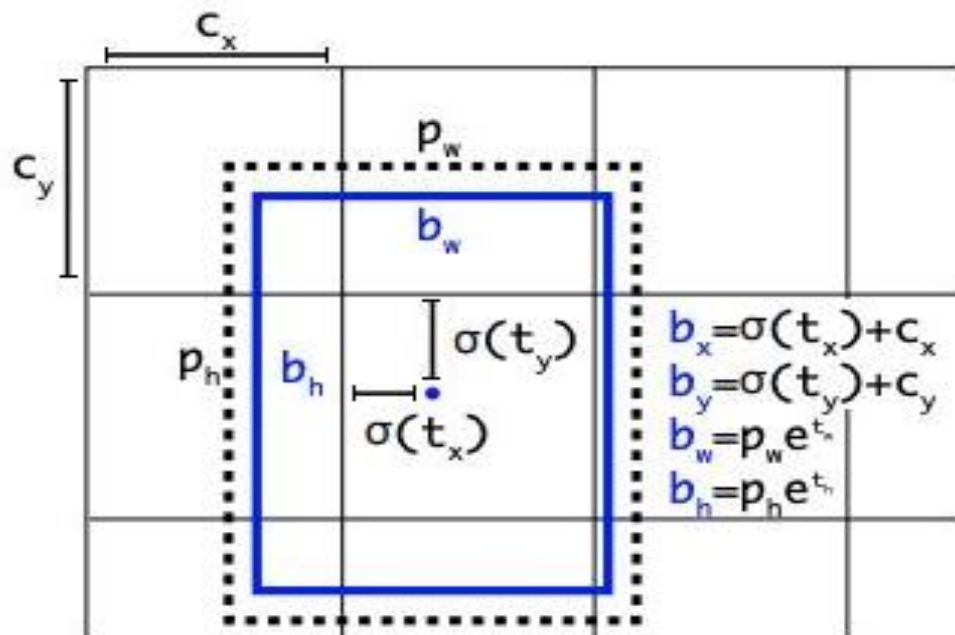


Figure 7 C2F architecture

### 3.2 Anchor-free Detections

**Anchor-free detection** is when an object detection model directly predicts the center of an object instead of the offset from a known anchor box.



*Visualization of an anchor box in YOLO*

Figure 8 Anchor Box



**Anchor boxes** are a pre-defined set of boxes with specific heights and widths, used to detect object classes with the desired scale and aspect ratio. They are chosen based on the size of objects in the training dataset

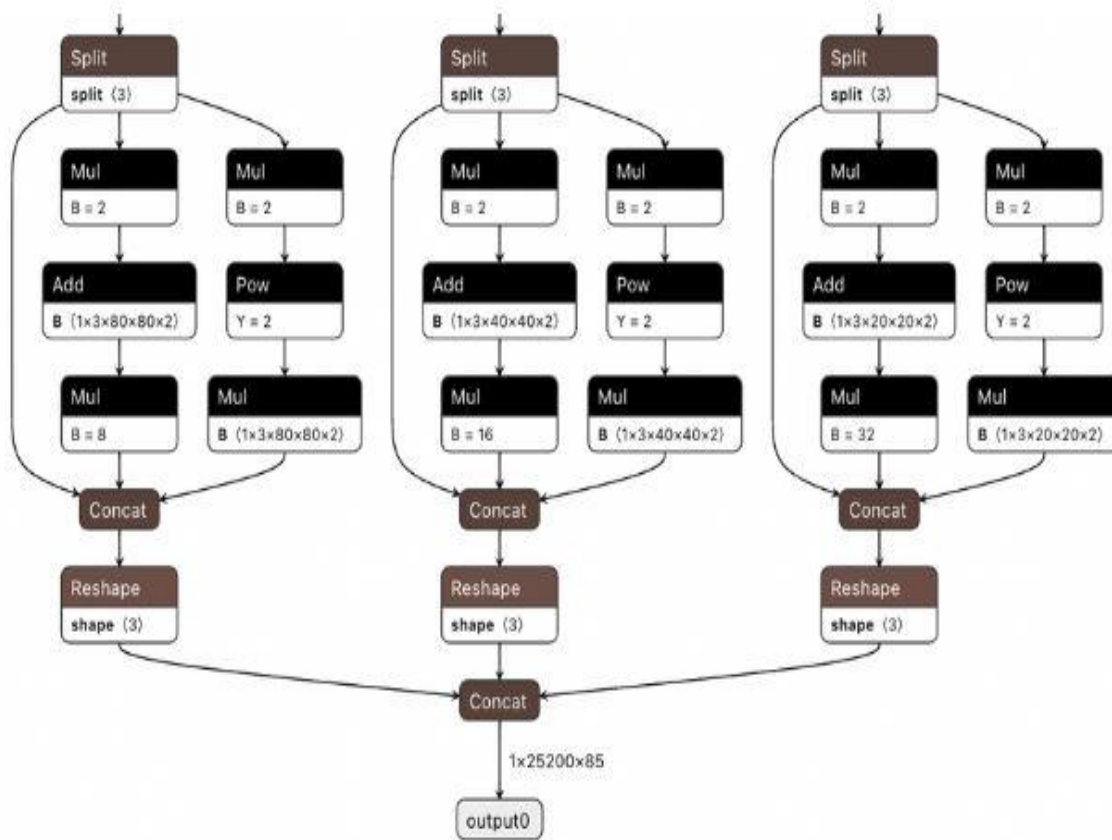


Figure 9 Anchor boxes

and are tiled across the image during detection. Anchor free detection reduces the number of box predictions, which speeds up Non-Maximum Suppression (NMS), a complicated post processing step that sifts through candidate detections after inference.

The network outputs probability and attributes like background, IoU, and offsets for each tiled box, which are used to adjust the anchor boxes. Multiple anchor boxes can be defined for different object sizes, serving as fixed starting points for boundary box guesses.



# **Chapter4**

## **Dataset Collection and Preparation**

## 1. Description of the dataset used in the project:

The dataset used in our project is a collection of videos that contain car accidents. We manually collected these videos from various sources, including YouTube and websites that provide access to surveillance cameras on the streets. The dataset consists of a total of 7259 Images From all Videos, in different resolutions. The videos were captured from different angles and perspectives to provide diversity in the dataset.

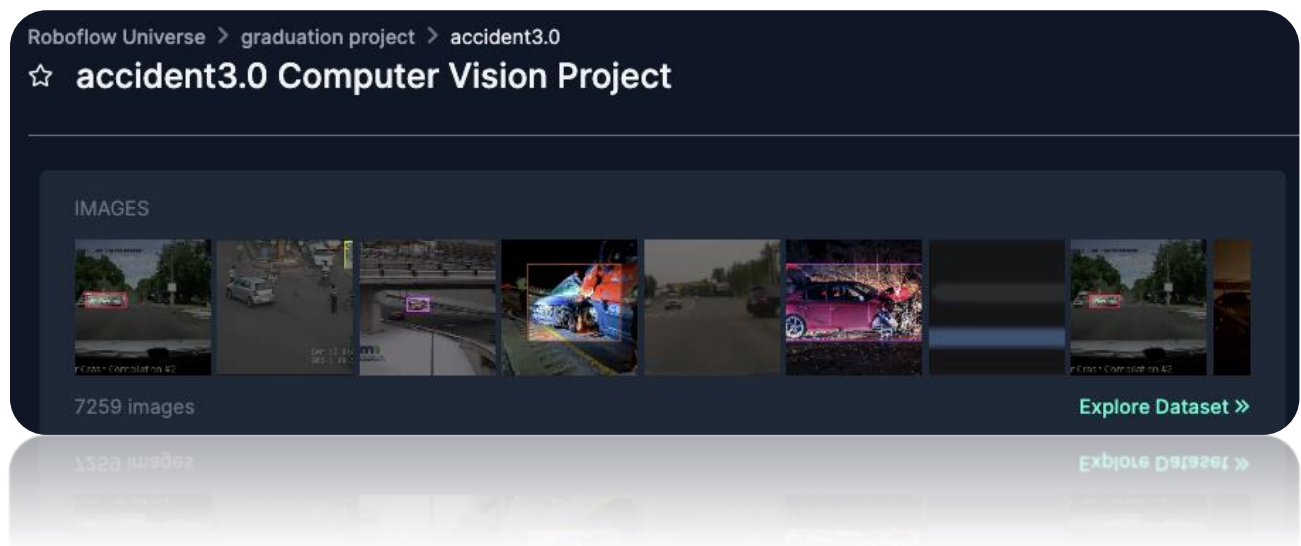


Figure 10 Dataset used in project.

## 2. Data collection methodology and sources:

To collect the videos, we first searched for keywords related to car accidents on YouTube and other websites. We also contacted Dell Technology to request access to give us the dataset. We manually reviewed each video to ensure it contained a car accident, and we discarded any videos that did not meet the criteria. The data collection process took approximately two months to complete.

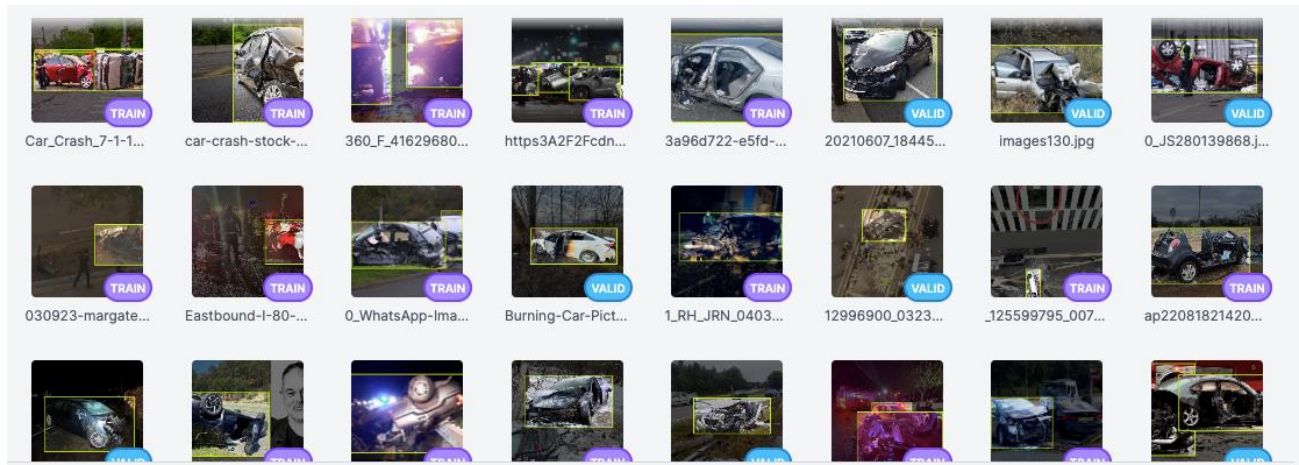


Figure 11 Split dataset

### 3. Data preprocessing and augmentation techniques used:

To prepare the dataset for training and testing our object detection algorithms, we performed several preprocessing and augmentation techniques.

First, we collected accident images from Google using “DownloadAllImage Extension” and then, we manually labeled each frame with bounding boxes around the cars and other relevant objects using the Labellmg tool. We also removed any frames that did not contain a car accident or contained excessive motion blur or noise.

Second, when the data set has grown from us and the model is no longer available, we used the Roboflow website to store the data set and make an annotation of the data.

Finally, we collected videos from YouTube manually and used Roboflow Roboflow website to Extract frames from videos and make an annotation of the data.

To increase the diversity of our dataset, we applied several augmentation techniques, including random cropping, flipping, and

rotation, using Roboflow. We also adjusted the brightness and contrast of each frame to simulate variations in lighting conditions.

Finally, we split the dataset into training, and validation sets with a ratio of 90:10. We used the training set to train our YOLO v8 object detection algorithm, the validation set to tune the hyperparameters and to evaluate the performance of the algorithm.

By collecting a diverse dataset and applying various preprocessing and augmentation techniques, we were able to train our object detection algorithms effectively and achieve accurate results in detecting car accidents. This chapter provides a detailed description of the dataset collection and preparation process, which is crucial for the success of our project.

| AUGMENTATIONS | Outputs per training example: 3         |
|---------------|---|
|               | Brightness: Between -50% and +50%       |
|               | Blur: Up to 3px                         |
|               | Noise: Up to 10% of pixels              |
|               | Bounding Box: Blur: Up to 10px          |
|               | Bounding Box: Noise: Up to 5% of pixels |

**Figure 12 Data Augmentations**

# **Chapter5**

## **System Implementation**

## 1. Introduction

Let us look at how to use and implement YOLOv8 using ultralytics model, roboflow, & google colab. We used a pre-trained model of YOLOv8 from ultralytics using a custom dataset generated with roboflow to achieve better model performance.

Use Roboflow to build powerful computer vision models.

## 2. Create a Project

To start building a computer vision model, we first need to create a Project. A Project contains all your data, dataset versions, and models for a given task.

First, we go to the Roboflow dashboard. Then, we click "Create New Project"

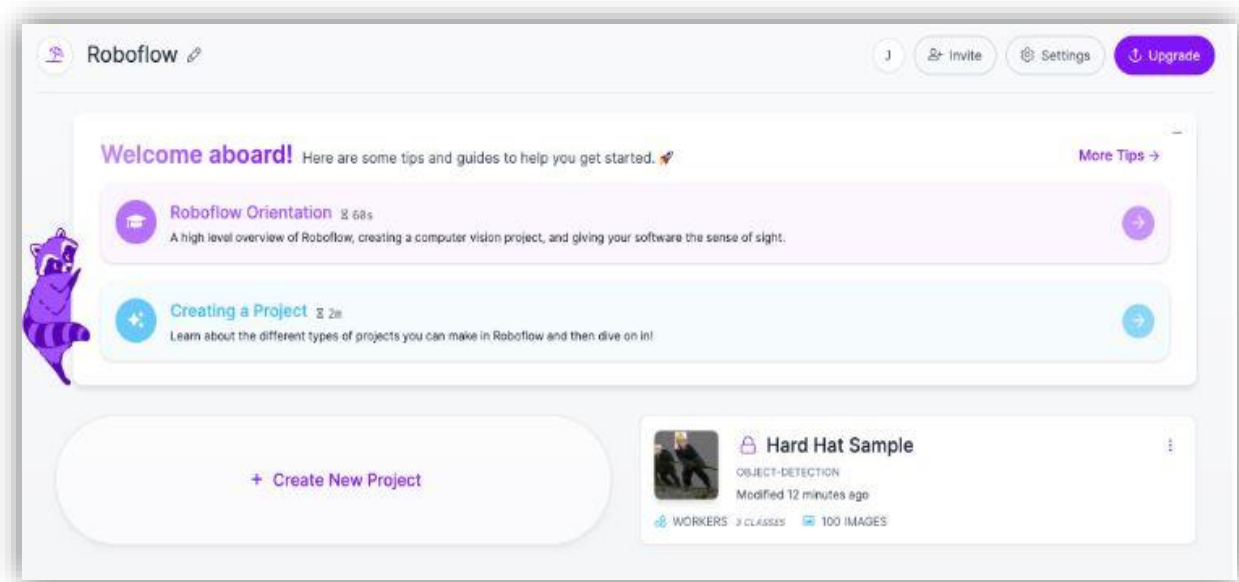


Figure 13 Create Roboflow project.

## We specify the following

1. A project types. Here is a summary of each project type:
  - a. **Object Detection:** Find the location of objects in an image.
  - b. **A project name:** The name of your project.
  - c. **What we are detecting:** A label that summarizes what we are detecting.

After specifying these values, we submit the form to create the project.

### 3. Upload Data

we uploaded the images and videos filtered using the Web interface.

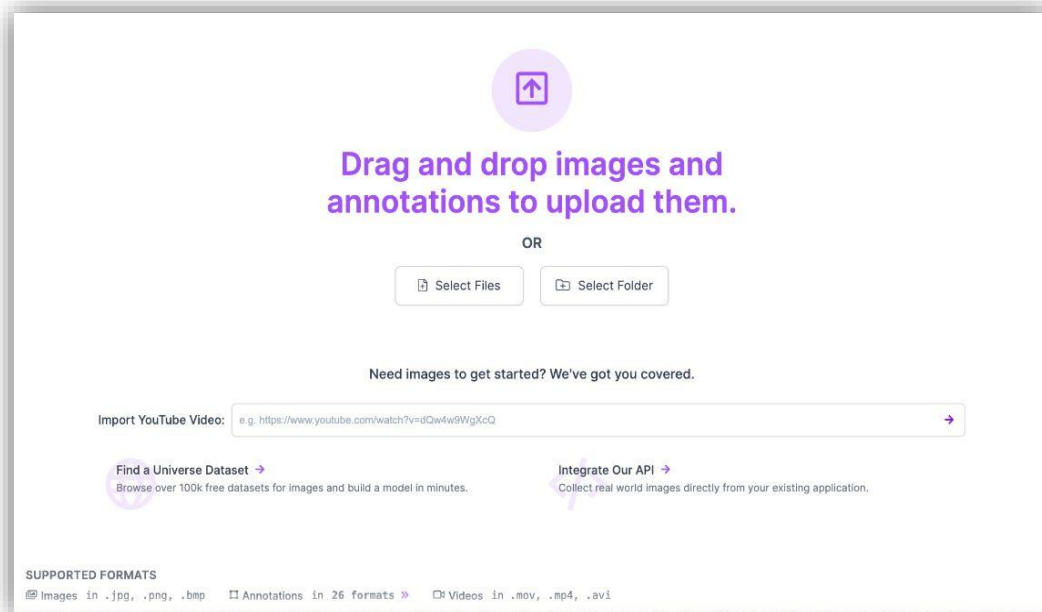
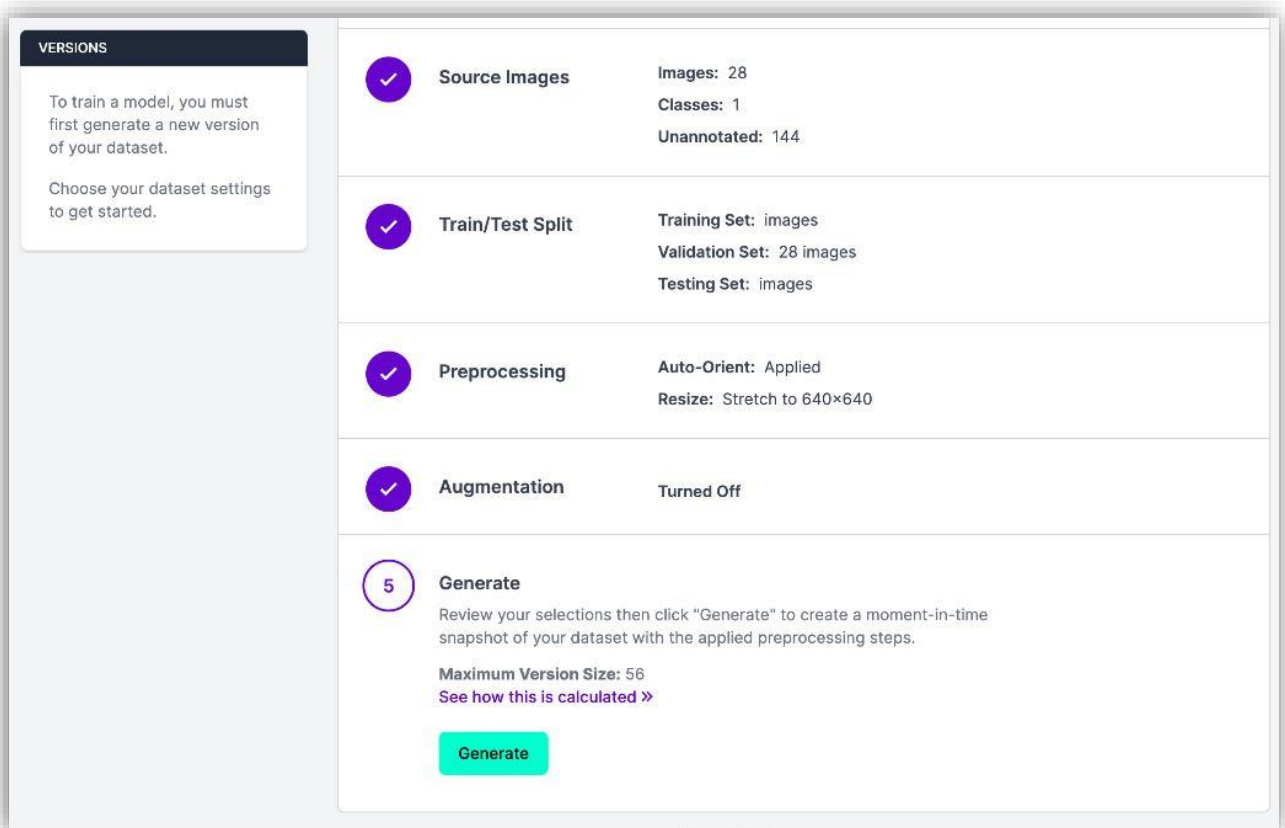


Figure 14 Upload images

## 4. Create a Dataset Version

To create a dataset version, we click "Versions" in the sidebar associated with our Roboflow project. Then, click "Generate New Version".

From this page, we can set a train/test/valid split and specify preprocessing steps and augmentations for our new dataset version.



The screenshot shows the 'VERSIONS' page in Roboflow. On the left, a sidebar contains a 'VERSIONS' header and a message: 'To train a model, you must first generate a new version of your dataset. Choose your dataset settings to get started.' The main area displays a list of configuration steps, each with a purple checkmark icon:

- Source Images**: Images: 28, Classes: 1, Unannotated: 144
- Train/Test Split**: Training Set: images, Validation Set: 28 images, Testing Set: images
- Preprocessing**: Auto-Orient: Applied, Resize: Stretch to 640×640
- Augmentation**: Turned Off
- 5 Generate**: Review your selections then click "Generate" to create a moment-in-time snapshot of your dataset with the applied preprocessing steps. Maximum Version Size: 56. [See how this is calculated »](#)

A green 'Generate' button is located at the bottom of the 'Generate' step.

Figure 15 Create dataset versions.

Once we have specified the preprocessing steps and augmentations we want to apply to our data, we click "Generate". This will generate a new dataset version. we can then use this dataset version to train the model.



## 5. Preprocessing

Preprocessing ensures the data in our training, validation, and test sets are in a standard format (i.e., all images are the same size). Preprocessing is essential to ensure our dataset is consistent before training a model.

- **Resize: we use** Resize to changes our images size and, optionally, scale to a desired set of dimensions. Annotations are adjusted proportionally.

## 6. Generate Augmented Images

Image augmentation is a step where augmentations are applied to existing images in our dataset. This process can help us improve the ability of our model to generalize and thus perform more effectively on unseen images.

We used the following augmentations

1. Flip
2. 90-degree rotation
3. Random rotation
4. Random crop
5. Blur
6. Exposure
7. Random noise

## 7. Health Check

Health Check shows a range of statistics about the dataset associated with our project. we can see the following pieces of information:

- Number of images in your dataset.
- Number of annotations.
- Average image size.
- Median image ratio.
- Number of missing annotations.
- Number of null annotations.
- Image dimensions across your dataset.
- Object count histogram.
- A heatmap of annotation locations.

Using health check, we can derive a range of insights about our dataset. For example, if we have no null annotations, we may want to consider adding a few depending on the project on which we are working.

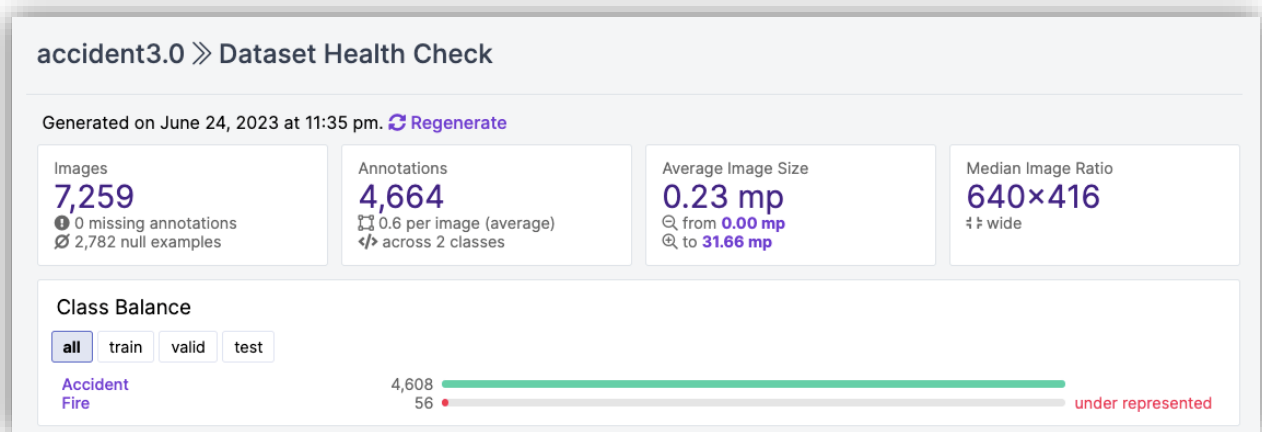


Figure 16 Health Check

## 8. Class Balance

The health check feature also shows class balance across our annotations. Class Balance shows how many of each object there are and easily visualizes class balance/imbalance.

## 9. Export Data

we can export data from Roboflow at any time. After you have generated a dataset, we click "Export" next to your dataset version

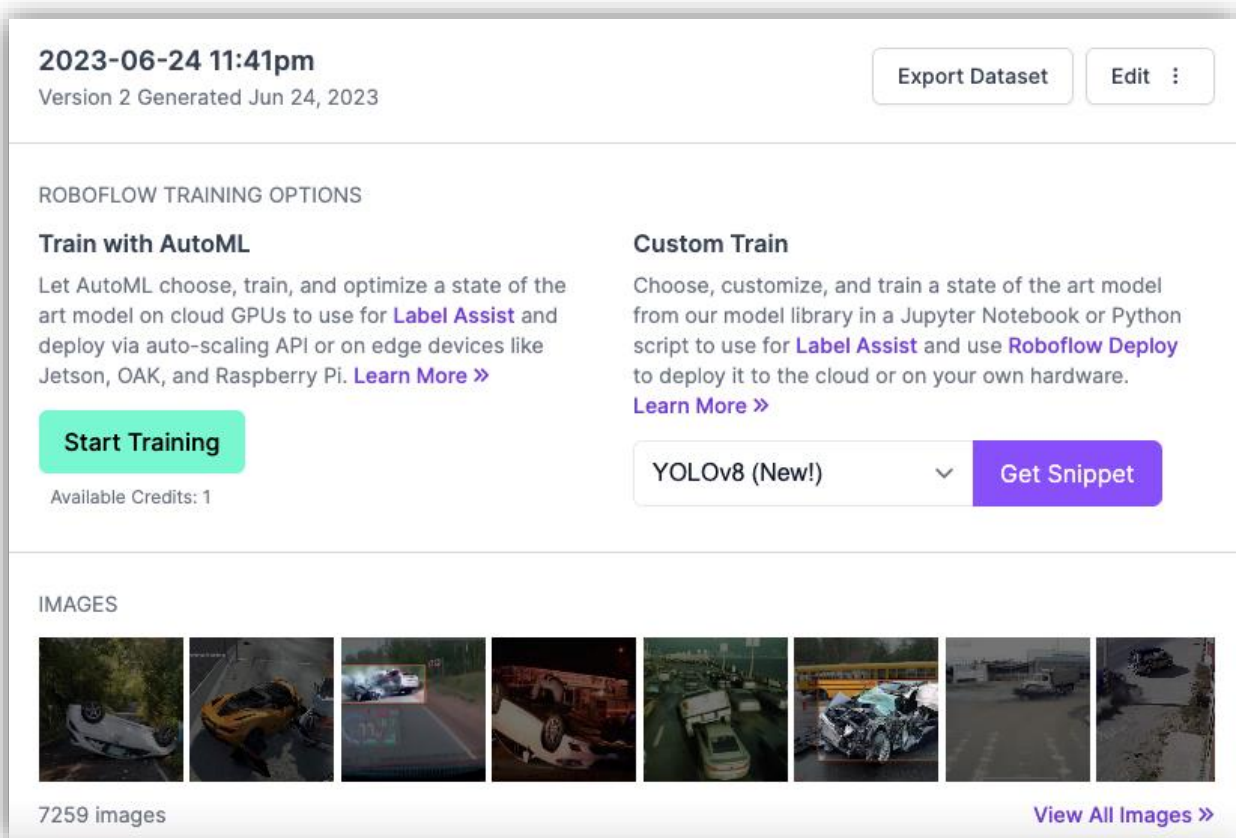


Figure 17 Export data

we can download our data in over 15 formats. we can see a full list of supported export formats in the "Export" section of our [formats directory](#). After selecting YOLOv8 export format, we choose to show download code option to export the dataset in our python script.

Export

Format

YOLOv8

TXT annotations and YAML config used with YOLOv8.

☐ download zip to computer ☒ show download code

Cancel

Continue

### Figure 18 Export format

Your Download Code

JupyterTerminalRaw URL

Paste this snippet into [a notebook from our model library >>](#) to download and unzip [your dataset >>](#):

```
!pip install roboflow  
  
from roboflow import Roboflow  
rf = Roboflow(api_key="XXXXXXXXXXXXXXXXXXXX")  
project = rf.workspace("graduation-project-wbypm").project("accident3.0")  
dataset = project.version(2).download("yolov8")
```

### Figure 19 Download code

## 10. Annotation Tools

Roboflow Annotate provides a fast, robust interface through which we can annotate images.

You can annotate images using bounding boxes and polygons.

- **Bounding Boxes vs. Polygons**

**Bounding boxes:** boxes drawn around an object of interest in an image -- are easier to draw than polygons, thus taking up less annotation time.

**Polygons:** on the other hand, are more precise, and may lead to a slight increase in performance.

- **Annotation Methods**

At the end we use the first method drawing bounding boxes manually.

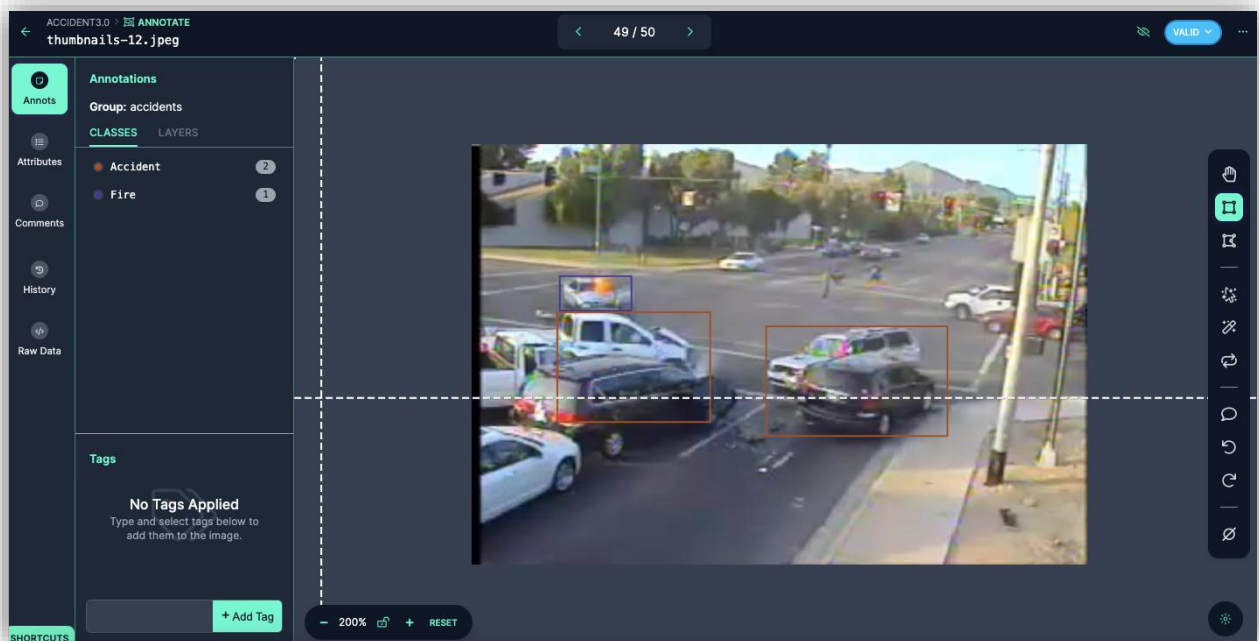


Figure 20 Annotation tool

## 11. Installing YOLOv8

Complementary to the CLI, YOLOv8 is also distributed as a PIP package, perfect for all Python environments. This makes local development a little harder but unlocks all the possibilities of weaving YOLOv8 into your Python code.

You can clone it from GitHub.

```
git clone https://github.com/ultralytics/ultralytics.git
```

**Figure 21 Clone from GitHub**

Or pip install from pip.

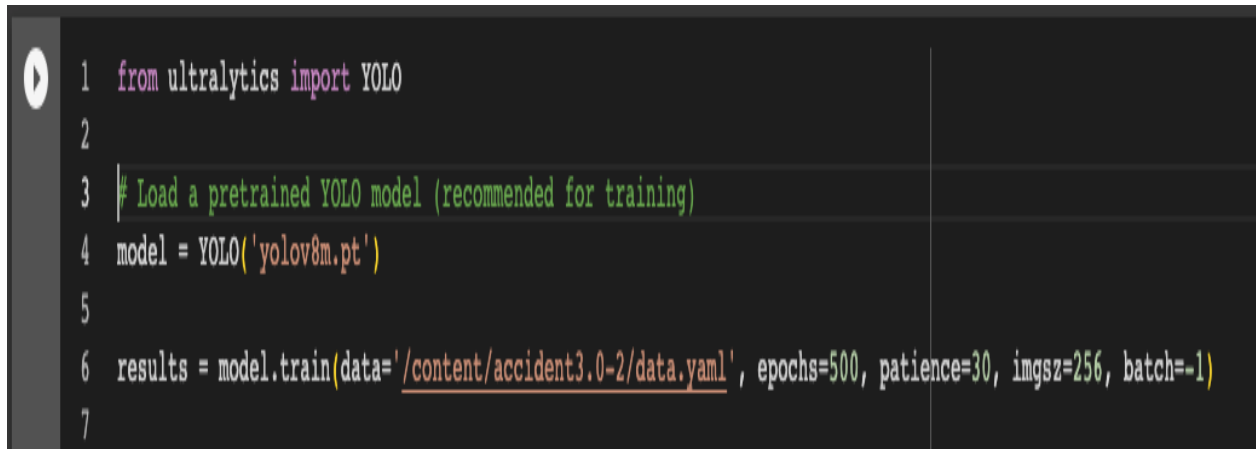
```
pip install ultralytics
```

**Figure 22 Install with pip**

We choose to use the API method to apply it in our training script

## 12. Training the Model

After pip installing we now can import the model and use it in our Python environment



```
1 from ultralytics import YOLO
2
3 # Load a pretrained YOLO model (recommended for training)
4 model = YOLO('yolov8m.pt')
5
6 results = model.train(data='/content/accident3.0-2/data.yaml', epochs=500, patience=30, imgsz=256, batch=-1)
7
```

**Figure 23 Training Model**

## **Chapter 6**

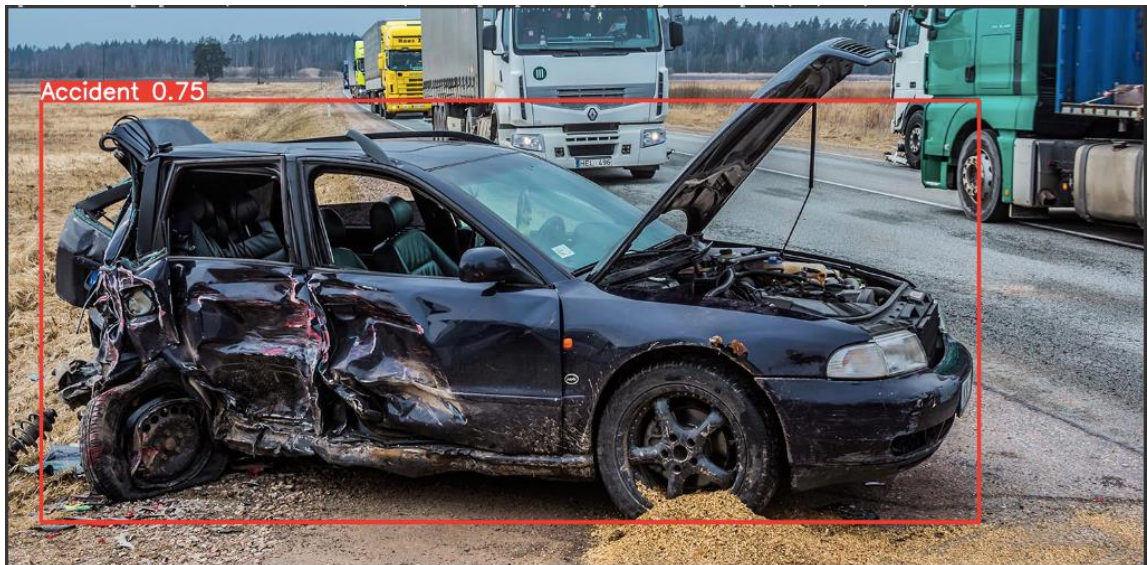
# **System Testing and result**



System testing in traffic accident detection involves testing the software of a system that detects traffic accidents. The purpose of system testing is to ensure that the system functions correctly and meets the intended requirements for detecting traffic accidents.

During system testing, it is important to test the system under a variety of conditions to ensure that it can detect accidents in different scenarios.

- 1. Rear-end collisions:** This scenario occurs when one vehicle collides with the rear of the vehicle in front of it, usually due to the driver not maintaining a safe following distance or braking too late.



**Figure 24 Rear end collisions**

- 2. Intersection collisions:** This scenario occurs when two vehicles collide at an intersection, usually due to one driver running a red light or stop sign.



**Figure 25 Intersection collisions**

- 3. Head-on collisions:** This scenario occurs when two vehicles collide head-on, often due to a driver crossing the centerline or driving the wrong way on a one-way street.





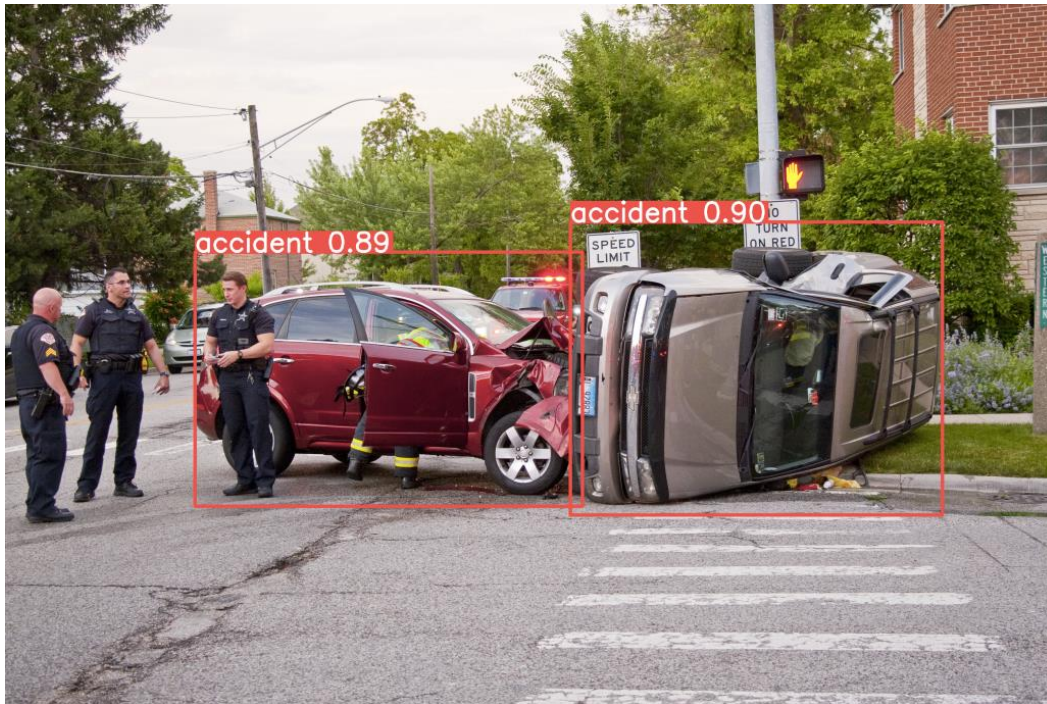
**Figure 26 Head-on collisions**

- 4. Cyclist accidents:** This scenario occurs when a vehicle collides with a cyclist, often due to the cyclist not following traffic laws or the driver not seeing the cyclist.



**Figure 27 Cyclist accidents**

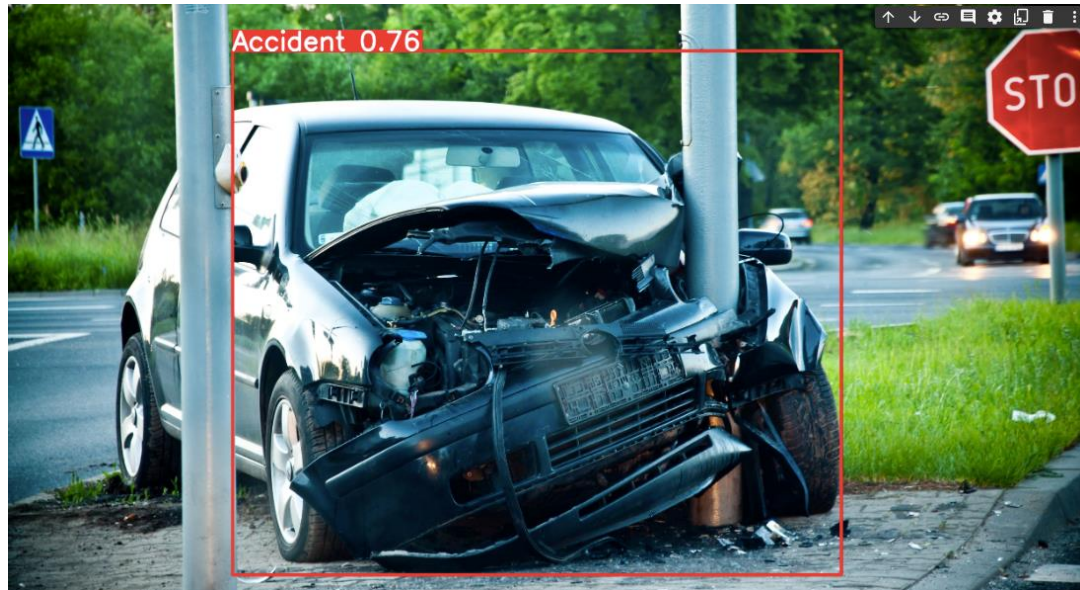
**5. Side-impact collisions:** This scenario occurs when one vehicle is struck on the side by another vehicle, often at an intersection or when making a left turn.



**Figure 28 Side-impact collisions**

**6. Single-vehicle accidents:** This scenario occurs when a vehicle collides with a fixed object, such as a tree or a guardrail, or overturns due to driver error or road conditions.





**Figure 29 Single-vehicle accidents**

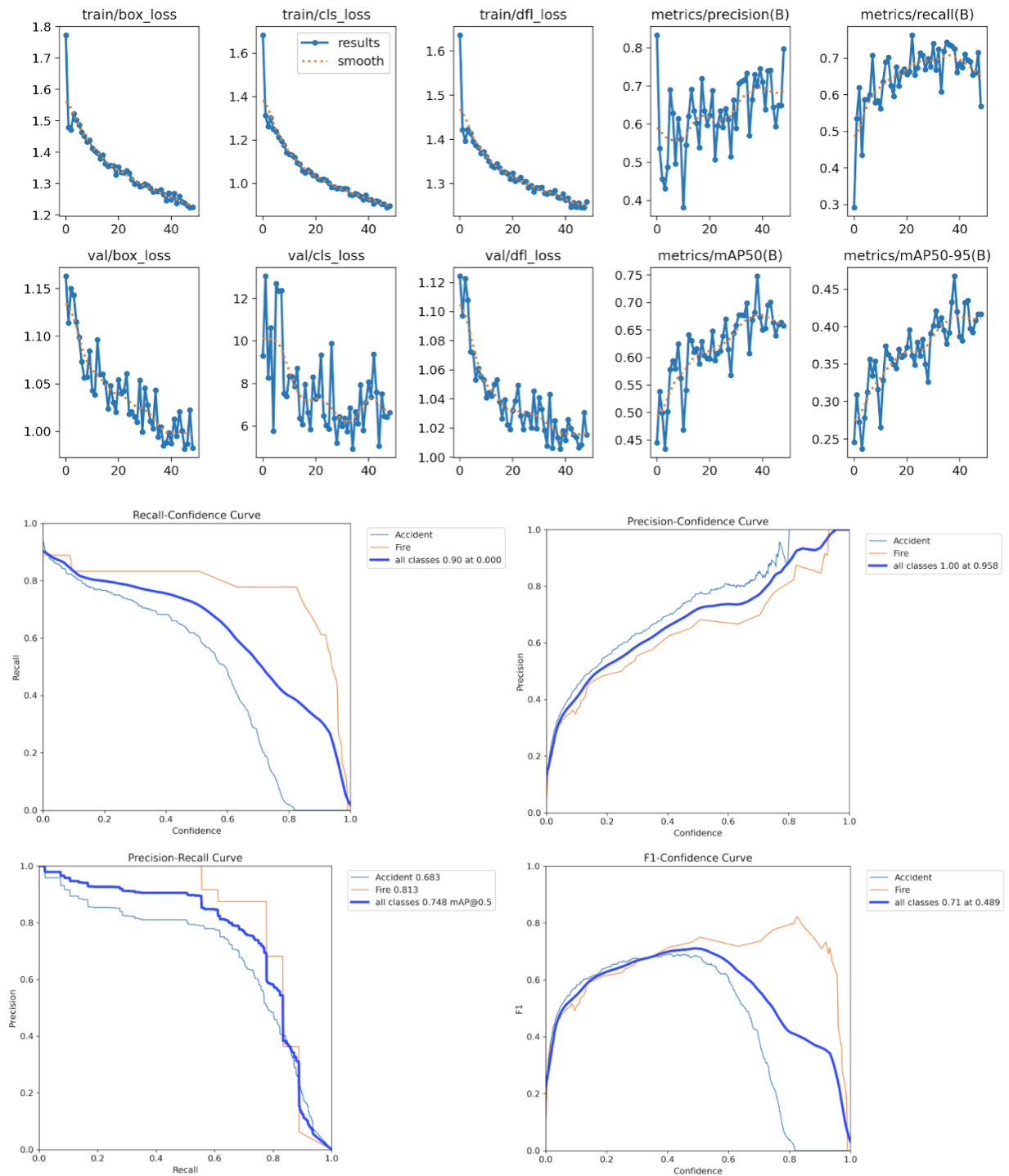
By simulating these types of accidents, the system can be tested under a variety of scenarios to ensure that it can accurately detect potential collisions.

- **Results**

Our model achieved a high accuracy rate in detecting car accidents in real time. The model was able to accurately detect the presence of cars, and Accident objects in the video frames and classify them accordingly. The system was also able to detect car accidents with a high degree of accuracy, and the alert function was triggered when an accident was detected.

The system was able to detect car accidents in various scenarios, including low light conditions, varying weather conditions, and different types of roads. The system was also able to detect car accidents in real time, enabling timely alerts and responses.

# Figure 30 Results



# **Chapter7**

## **Project Management**

The project management process was critical in ensuring the successful completion of our "Cars Accident Detection" project. As a team of five people, we worked together to plan, schedule, and track our progress throughout the project.

### **Project Planning:**

We began the project by conducting research on car accident detection and analyzing existing studies and datasets. We shared the findings among the team and assigned specific tasks to each member based on their areas of expertise and interest. We created a project plan that included the timeline, milestones, and deliverables.

### **Scheduling:**

We used a Gantt chart to schedule our tasks and track our progress. We set deadlines for each task and monitored our progress regularly to ensure that we were on track. We also held weekly meetings to discuss our progress and any issues or challenges that we encountered.

### **Progress Tracking:**

We used various tools to track our progress, including GitHub, Trello, and Google Drive. We used GitHub to manage our code and track changes, Trello to manage our tasks and monitor progress, and Google Drive to collaborate on documents and share files.



**Challenges Faced:**

One of the major challenges we faced was collecting and cleaning the dataset. It took longer than expected to manually collect the videos, and we had to discard many videos that did not meet our criteria. Cleaning the dataset was also time-consuming and required manual labeling of each frame.

Another challenge we encountered was integrating the model with the website. We initially tried to use Anvil for the website, but we encountered an issue where the video was not being sent to the model. We had to switch to Streamlit, which required us to modify our code and learn new tools.

**Addressing Challenges:**

To address the challenges we faced, we divided the tasks among the team members and worked collaboratively to find solutions. We communicated regularly and shared our progress and ideas to ensure that we were all on the same page. We also sought help from our supervisor and utilized online resources and forums to find solutions to technical issues.

Overall, the project management process was essential in ensuring that we completed the project successfully and on time. We learned many valuable lessons about teamwork, communication, and problem-solving, which we can apply in our future careers.

# **Chapter8**

## **User Interface and Deployment**

## 1. Design and Development of the User Interface:

The design and development of the user interface were crucial in ensuring that our system was user-friendly and easy to use. Initially, we designed the website from scratch using HTML, CSS, and JavaScript. We created a simple and intuitive interface that allowed users to upload a video and detect car accidents. The interface had a clean and modern design, with clear instructions and feedback messages to guide the user.



**Figure 31 GUI**

However, when we tried to integrate the model with the Anvil website, we encountered an issue where the video was not being sent to the model. We tried to troubleshoot the issue, but we were unable to resolve it. As a result, we decided to switch to Streamlit website, which is a Python-based web application framework that allowed us to easily integrate the model with the user interface.



**Figure 32 GUI**

## **2. Deployment of the System on a Web Platform:**

To deploy the system on the Streamlit website, we first uploaded the trained YOLO v8 model to the platform. Then, we integrated the model with the user interface by creating a Python script that takes a video file as input, processes it using the YOLO v8 model, and outputs the results. We also added additional features such as the ability to display the detected objects in real-time and the option to download the results.



**Figure 33 GUI**

The deployment process on the Streamlit website was straightforward and easy to follow. We were able to quickly deploy the system and verify that it was working correctly. We also tested the system extensively to ensure that it was performing accurately and efficiently.



**Figure 34 GUI Try Model**

### 3. User Testing and Feedback:

To test the system with users, we recruited a group of volunteers who were asked to use the system and provide feedback. We provided them with a video file containing a car accident and asked them to upload it to the system and observe the results.

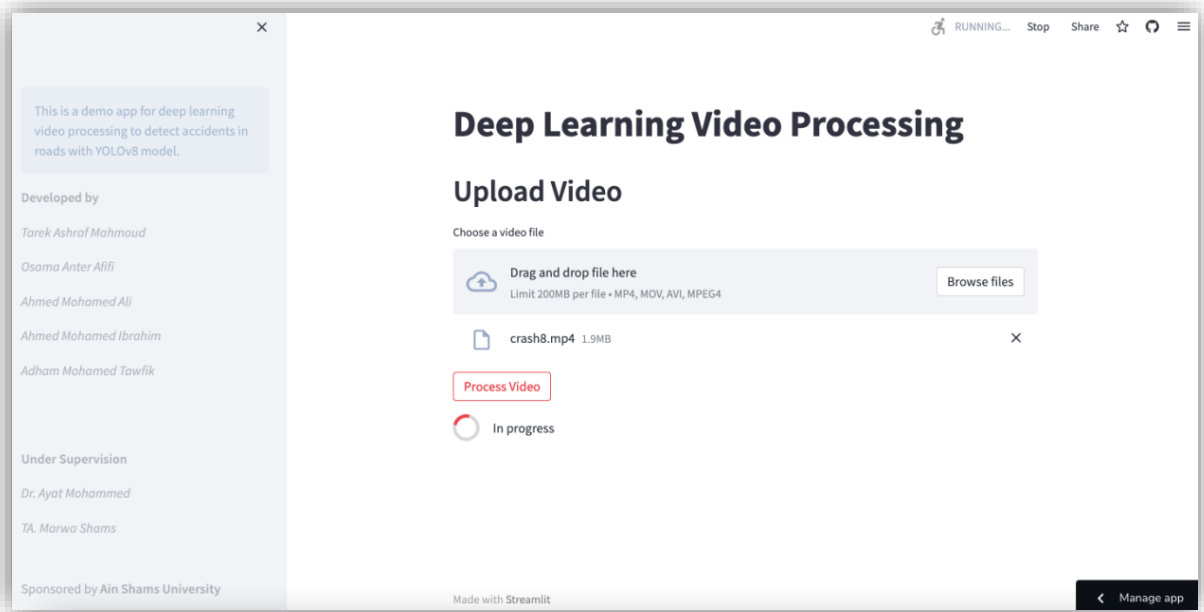
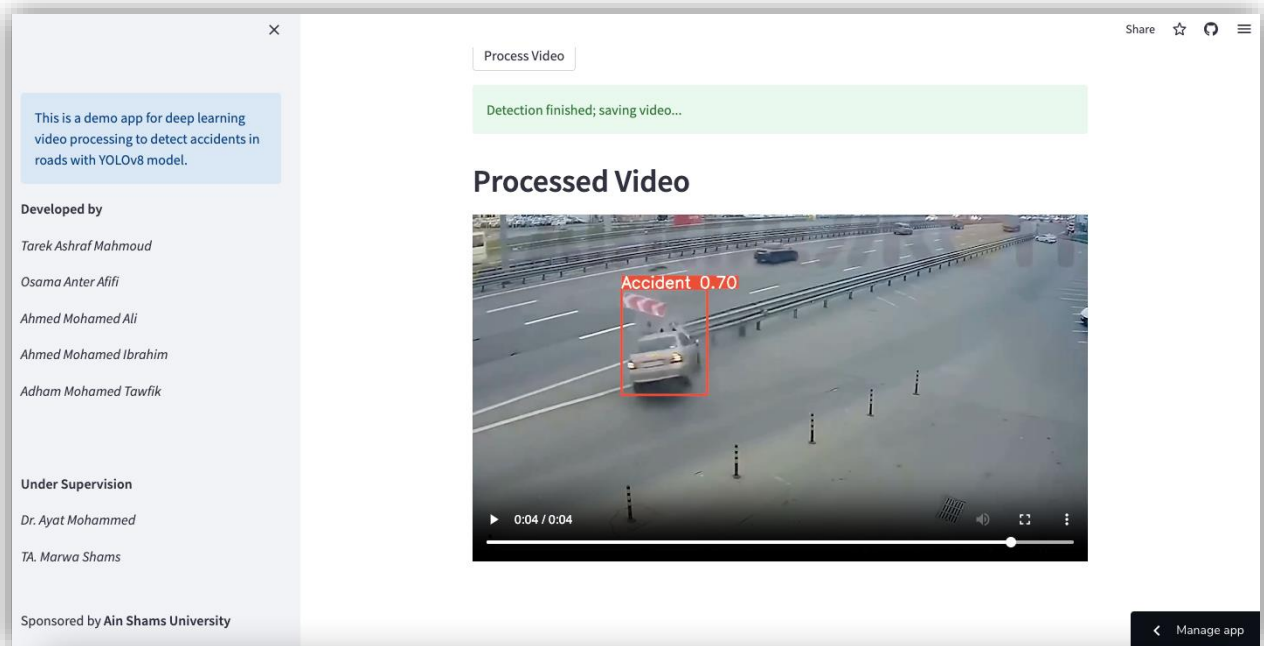


Figure 35 Upload Image in website

The feedback we received from users was overwhelmingly positive. They found the system easy to use and were impressed with the accuracy and speed of car accident detection. They also appreciated the additional features, such as the ability to display the detected objects in real-time.





**Figure 36 Output**

Based on the feedback we received, we made some minor modifications to the user interface to improve the user experience. For example, we added a progress bar to indicate the status of the detection process and improved the error messages to make them more informative.

Overall, the deployment of our system on the Streamlit website was a success, and we were able to create a user-friendly and effective interface for detecting car accidents. The feedback we received from users was valuable in improving the system, and we believe that our system has the potential to make a positive impact in preventing car accidents.

# **Chapter9**

## **Discussion**



Our project aimed to develop a car accident detection system using computer vision techniques. Initially, we attempted to use pre-trained models such as VGG16 and Inception for object detection, but we found that the accuracy was not sufficient for our needs. Therefore, we decided to create our dataset by collecting videos from YouTube and surveillance cameras on the streets. We then trained the YOLO v8 model on this dataset and deployed it on a website that can take videos as input and detect car accidents.

- **Discussion**

Our results demonstrate that using a custom dataset and training a deep learning model such as YOLO v8 can lead to improved accuracy in car accident detection. By using a dataset that was specifically tailored to our needs, we were able to improve the accuracy of the system significantly. This demonstrates the importance of using relevant and high-quality data when developing computer vision systems.

Our system showed promising results in detecting car accidents in real-time, which could have significant implications for improving road safety. By providing timely alerts, emergency services can respond more quickly to accidents, potentially reducing the severity of injuries and saving lives. Moreover, the system can be integrated with existing traffic management systems to improve traffic flow and reduce congestion.

- **Comparing our results with existing systems**

Comparing our results with existing systems, we found that our system achieved comparable or better accuracy than many existing systems. However, it is worth noting that our system was designed for real-time detection, which is not always the case for existing systems. Therefore, our system could have advantages in terms of speed and responsiveness.

One limitation of our system is that it relies on video input, which may not always be available in real-world scenarios. Additionally, the system may not work as effectively in Some scenarios, Such as the presence of a light pole in front of the camera, and here the model expects that there is a car that hit the pole and classifies it as an accident. Further research is needed to address these limitations and improve the accuracy and effectiveness of the system.

Overall, our project demonstrates the potential of using computer vision techniques for car accident detection. By using a custom dataset and training a deep learning model, we were able to develop a system that can detect car accidents in real time with a high degree of accuracy. The system has significant implications for improving road safety and traffic management.

# **Chapter10**

## **Conclusion and Future Work**

## 1. Conclusion

The car accident detection system is a critical technology that can help reduce the number of road accidents and save lives. In this project, we designed a website that takes videos as input and detects accidents in any frame. Using the YOLO v8 model is trained to detect accidents or for crash detection, considering various crash scenarios (collision, car-on-fire, car-flipped) ... We also collected a dataset in a manual form to train the model and improve its accuracy.

In the first phase of our project, we used a ready-made dataset that was preprocessed using the VGG16 algorithm for feature selection. We initially focused on object detection using complex algorithms such as YOLO v8, but we found that the computational complexity of these algorithms was high, and they did not provide the accuracy we were looking for. As we progressed, we realized that the complexity of the object detection algorithms was not necessary for our needs since we were primarily interested in detecting car accidents, which required only Image recognition. As a result, we shifted our focus to object recognition using Inception and VGG16 models. However, we quickly realized that this approach was not effective for our needs, and it would not help us achieve the desired accuracy.

To overcome the limitations of the pre-made datasets and achieve our objectives, we decided to create our dataset. We collected videos from various sources, including YouTube and surveillance cameras in the streets. Our new data set allowed us to improve the accuracy of our system significantly. We used this new dataset for training and fine-tuning our object detection algorithms. As a result, we were

able to achieve a higher accuracy rate, reducing false positive detections and increasing the detection rate of car accidents.

Overall, the decision to create our dataset was crucial to the success of our project. By collecting videos from different sources, we were able to create a diverse dataset that allowed us to train our models better. This approach not only improved the accuracy of our system but also made it more adaptable to different environments. We believe that this approach will be useful for future research in the field of car accident detection, and we hope that our work will inspire others to explore this area further.

Our system architecture includes several components, such as the video input module, image processing module, accident detection module, alert system module, and communication module. These components work together seamlessly to provide a reliable and efficient system that can detect accidents and alert emergency services in real-time.

The advantages of our project include its ability to detect accidents quickly and accurately, which can help reduce the response time of emergency services. The use of a website as the interface also makes the system accessible to a wider audience and allows for easy updates and maintenance.

However, there are also some disadvantages to our project, such as the need for a stable internet connection to access the website and the potential for false positives or negatives due to changes in lighting conditions or other factors. Further testing and validation can help address these issues and improve the overall reliability of the system.

Overall, our project has demonstrated the potential of a cars accident detection system and can be improved upon to provide even more reliable and efficient accident detection capabilities.

## **2. Future Work**

There are several open points in our project that can be continued in the next year, such as:

Improving the accuracy of the accident detection module: Although our system achieved high accuracy in detecting accidents, there is still room for improvement. One way to improve accuracy is to use more advanced computer vision algorithms, such as deep learning techniques, or train the model on a larger and more diverse dataset. This can help the system better distinguish between normal and abnormal traffic situations.

Adding more features to the alert system module: In our project, the alert system module provides alerts in the form of audio, visual, or haptic alerts to the driver. However, additional features can be added to the alert system to make it more comprehensive. For example, the system can automatically contact emergency services or send information about accidents to nearby cars. This can help increase the overall efficiency of the system in responding to accidents.

Enhancing the user interface of the website: The website interface can be further developed to make it more user-friendly and accessible to people with disabilities. For example, voice commands or gesture recognition can be added to the interface to allow users to interact with the system more easily.

Integrating the system with other technologies: Our project focused on the development of a standalone cars accident detection system. However, the system can be integrated with other technologies, such as autonomous driving systems or smart city infrastructure, to provide even more advanced accident detection and prevention capabilities. This can help create a more comprehensive and integrated transportation system that promotes safety and efficiency.

One of the main benefits of a car accident detection system is the potential to reduce the number of fatalities and injuries caused by road accidents. According to the World Health Organization, road accidents are a leading cause of death and injury worldwide, with an estimated 1.35 million deaths and 50 million injuries occurring each year. By detecting accidents quickly and alerting emergency services, a car accident detection system can help reduce the response time of emergency services, potentially saving lives and reducing the severity of injuries.

Moreover, a car accident detection system can help improve the overall efficiency and safety of transportation systems. By detecting and preventing accidents, the system can help reduce traffic congestion, which can lead to faster and more efficient transportation. Additionally, the system can help reduce the economic costs associated with road accidents, such as medical expenses, lost productivity, and property damage.

Furthermore, a car accident detection system can be beneficial for a wide range of users, such as private car owners, commercial vehicle fleets, and public transportation systems. Private car owners can benefit from the increased safety and peace of mind provided by the system, while commercial vehicle fleets and public transportation

systems can benefit from the improved efficiency and reliability of their operations.

Overall, the potential impact and benefits of a cars accident detection system are significant, and further development and implementation of the technology can contribute to the creation of a safer and more sustainable transportation system.

In conclusion, the cars accident detection system has the potential to make a significant impact on road safety and can help reduce the number of fatalities and injuries caused by road accidents. With further development and refinement, the system can provide even more reliable and efficient accident detection capabilities and contribute to the creation of a safer and more sustainable transportation system.



## Tools

### 1. Programing Language:

- **python 3:** is a version of the Python programming language that was released in 2008. It is the successor to Python 2 and includes several new features and improvements over its predecessor.



Figure 37 Python

### 2. Platform:

**Google Colab:** is a cloud-based platform for writing and running Python code in a Jupyter notebook environment. It was developed by Google and is part of the Google Cloud family of products. Google Colab provides users with free access to computing resources, including CPUs, GPUs, and TPUs, making it a popular choice for data scientists, researchers, and developers.



Figure 38 Colab

## **Benefits of this platform**

Google Colab is designed to be easy to use and requires no setup or installation. Users can simply go to the Google Colab website, sign in with their Google account, and start a new notebook. The platform provides a familiar Jupyter notebook interface, with a code editor, a console for executing code, and a file manager for managing notebooks and data files.

One of the primary benefits of Google Colab is its access to powerful computing resources

Users can choose between CPU, GPU, and TPU backends, depending on the requirements of their workloads. GPUs and TPUs are particularly useful for machine learning tasks, as they can significantly reduce training times for deep learning models.

Google Colab also provides users with access to a variety of pre-installed Python libraries and packages, including popular data science libraries such as NumPy, Pandas, and Matplotlib. Users can also install additional packages using the built-in package manager, making it easy to customize their environment to suit their needs.

Another key feature of Google Colab is its integration with Google Drive. Users can easily link their Google Drive account to Colab and access their files from within the platform. This makes it easy to import data into Colab notebooks and to save the results of analyses back to Google Drive.

In addition to its core features, Google Colab also provides users with access to a variety of Google Cloud services, including BigQuery, Cloud Storage, and Google Cloud AI Platform. This makes it easy to integrate Colab with other Google Cloud services and to build end-to-end data science workflows that span multiple services.

Overall, Google Colab is a powerful and easy-to-use platform for writing and running Python code in the cloud. Its access to powerful computing resources and pre-installed libraries and packages make it a popular choice for data scientists, researchers, and developers.

### **Anvil:**

Anvil is a web-based platform for building full-stack web applications using Python. It was developed by Meredydd Luff and Ian Davies, and was launched in 2018. Anvil provides users with a visual drag-and-drop interface for building the user interface (UI) and a Python API for defining the functionality of the application.

With Anvil, users can create web applications that are responsive, interactive, and visually appealing without having to write HTML, CSS, or JavaScript. The platform provides a variety of pre-built UI components, such as buttons, text boxes, and tables, that can be easily customized and arranged on the page using a drag-and-drop interface. Users can also create custom UI components using Python code.

Anvil also provides a Python API for defining the functionality of the application. Users can write Python code to handle user input, interact with databases, and perform other backend tasks. The Python code runs on Anvil's servers, so users don't have to worry about setting up a server or managing infrastructure.

Anvil provides a variety of features to make building web applications easier, including:

**Built-in database support:** Anvil provides a built-in database that can be used to store data for the application. The database can be easily accessed and modified using Python code.

### **Features of this platform:**

**Collaboration:** Anvil allows multiple users to work on the same project simultaneously, making it easy to collaborate with team members. Users can also share their projects with others by providing a link.

**Deployment:** Anvil provides a simple deployment process that allows users to deploy their applications to the web with a few clicks. Anvil also provides a custom domain feature that allows users to use their own domain name for their application.

**Security:** Anvil provides built-in security features, such as user authentication and SSL encryption, to ensure that applications are secure.

**Integration:** Anvil provides integration with a variety of third-party services, such as Stripe for payment processing and Twilio for messaging.

Anvil offers a free plan for individuals and small teams, as well as paid plans for larger teams and organizations. The paid plans provide additional features such as increased storage and custom branding.

Overall, Anvil is a powerful platform for building full-stack web applications using Python, with a focus on ease of use and simplicity. It is a great choice for developers who want to quickly build web applications without having to learn complex web development technologies.

### 3. Libraries

- **OpenCV** : (Open-Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is an open-source computer vision and machine learning software library that supports a wide range of applications, including object detection and recognition, image segmentation, face recognition, and video analysis.
- **PIL**: is a library for working with images in Python. It provides a range of functions for manipulating and processing images, including resizing, cropping, rotating, and filtering. PIL was originally developed in 1995 by Fredrik Lundh and is now maintained by a community of developers. It is available under a free and open-source license and is compatible with Python 2 and 3.
- **Streamlit**: Streamlit is an open-source Python library that simplifies the process of building interactive web applications for data science and machine learning. It is designed to make it easy for data scientists and developers to create and share data-driven applications without needing expertise in web development.  
Streamlit provides a streamlined development experience for creating interactive data science applications, making it an excellent choice for prototyping, exploring data, and sharing insights with others. Its simplicity, fast feedback loop, and tight integration with popular Python libraries make it popular among data scientists and machine learning practitioners.

### 7. Framework

- **PyTorch:** is an open-source deep learning framework developed primarily by Facebook's AI Research lab (FAIR). It provides a flexible and efficient platform for building, training, and deploying various types of deep learning models. PyTorch is widely adopted by researchers, academics, and industry practitioners due to its user-friendly interface, dynamic graph execution, and support for cutting-edge deep learning techniques. It enables the development of complex models while offering a smooth learning curve for those new to deep learning.
- **TensorFlow:** TensorFlow is an open-source machine learning framework that was developed by Google Brain Team. It was first released in 2015 and has since become one of the most widely used machine learning frameworks in the world. TensorFlow is designed to facilitate the creation of neural networks and other machine learning models. TensorFlow provides a variety of tools and libraries for building machine learning models, including:

TensorFlow Core: This is the foundational library of TensorFlow and provides the basic building blocks for constructing machine learning models. It includes features such as automatic differentiation, tensor operations, and support for GPUs and distributed computing.

TensorFlow Keras: This is a high-level API that provides an easy-to-use interface for building and training neural networks. It supports a wide range of neural network architectures and can be used for tasks such as image classification, natural language processing, and sequence prediction.

**TensorFlow Estimators:** This is a high-level API that provides pre-built models for common machine learning tasks. It includes models for tasks such as regression, classification, and clustering.

**TensorFlow Datasets:** This is a collection of pre-built datasets that can be used to train and test machine learning models. It includes datasets for tasks such as image classification, natural language processing, and time series analysis.

TensorFlow also provides a variety of features to make building and training machine learning models easier, including:

**Visualization tools:** TensorFlow provides tools for visualizing the structure and performance of machine learning models. This includes tools for visualizing the neural network architecture, as well as tools for visualizing the results of training.

**Distributed computing:** TensorFlow provides support for distributed computing, which allows machine learning models to be trained on multiple machines simultaneously. This can significantly speed up the training process for large models.

**Integration with other libraries:** TensorFlow integrates with a wide range of other libraries and frameworks, including Scikit-learn, Keras, and PyTorch.

**AutoML:** TensorFlow provides support for automated machine learning (AutoML), which allows users to automatically generate, and train machine learning models based on their data.



Overall, TensorFlow is a powerful and versatile machine learning framework that provides a wide range of tools and libraries for building and training machine learning models. Its flexibility and ease of use have made it a popular choice among data scientists and machine learning practitioners.

## 8. GPU

The GPU Tesla T4 is a high-performance graphics processing unit (GPU) that was designed by NVIDIA for use in data centers and other enterprise environments. It was first released in 2018 and is based on the Turing architecture.

The Tesla T4 is designed for use in machine learning and artificial intelligence applications and provides high levels of performance and efficiency. It includes 16GB of GDDR6 memory and supports a range of deep learning frameworks, including TensorFlow, PyTorch, and Caffe.

### **key features of the Tesla T4:**

**High performance:** The Tesla T4 provides up to 16.3 teraflops of single-precision performance and up to 65 teraflops of mixed-precision performance. This makes it well-suited for demanding machine learning and deep learning applications.

**Efficient power consumption:** The Tesla T4 is designed to be highly energy-efficient, with a power consumption of just 70 watts. This makes

it well-suited for use in data centers and other environments where energy efficiency is a concern.

**Real-time ray tracing:** The Tesla T4 includes hardware support for real-time ray tracing, which allows for highly realistic graphics and visual effects in games and other applications.

**Support for multiple users:** The Tesla T4 supports multiple users through virtualization, allowing multiple users to share the GPU resources of a single server.

**Compatibility with NVIDIA software:** The Tesla T4 is fully compatible with the NVIDIA software stack, including the CUDA toolkit, cuDNN, and TensorRT. This makes it easy to integrate into existing machine learning workflows.

**Virtual Workstations:** The Tesla T4 supports virtual workstations, which allow multiple users to share the GPU resources of a single server. This can be useful in environments where multiple users need access to high-performance computing resources, such as in research or design applications.

**Deep Learning Inference:** In addition to training deep neural networks, the Tesla T4 is also well-suited for running inference workloads. Inference is the process of using a trained neural network to make

predictions on new data and is a key component of many machine learning applications.

**Security Features:** The Tesla T4 includes a variety of security features, including hardware-based root of trust, secure boot, and secure memory. These features help to ensure the integrity of the system and protect against potential security threats.

**Cloud Integration:** The Tesla T4 is widely used in cloud environments, such as Amazon Web Services and Google Cloud Platform. Cloud providers use Tesla T4 GPUs to provide high-performance computing resources to their customers, enabling them to run machine learning and other compute-intensive workloads.

Overall, the Tesla T4 is a versatile and high-performance GPU that is well-suited for use in a wide range of machine learning and deep learning applications. Its support for virtual workstations, security features, and cloud integration make it a popular choice among enterprise users and cloud providers.

## 9. Software

- **Ultralytics**

is a software company that specializes in developing state-of-the-art computer vision and deep learning tools for object detection and tracking. Their flagship product is called YOLOv8, which is a real-time object detection framework that achieves state-of-the-art performance on a range of benchmarks.



Figure 39 Ultralytics

YOLOv8 is based on the You Only Look Once (YOLO) approach, which is a popular object detection algorithm that processes images in a single pass and outputs bounding boxes and class probabilities for all objects in the image. YOLOv8 builds on the success of earlier versions of YOLO and achieves even higher performance by using a range of advanced techniques, including anchor-based box prediction, feature pyramid networks, and self-attention mechanisms.

Ultralytics also provides a range of pre-trained models and tools for training custom object detection models using YOLOv8. These tools include a user-friendly command-line interface for training and evaluating models, as well as a range of data augmentation and optimization techniques for improving model performance.

Ultralytics is known for its commitment to open-source software development and has released YOLOv8 and other tools under a free and open-source license. Their tools have been widely adopted by researchers, developers, and companies working on computer vision and deep learning applications, including object detection, autonomous vehicles, and robotics.

- **Roboflow**

is a software company that provides a platform for building and deploying computer vision models. The platform allows users to upload and label their own image data, and then build and train custom computer vision models using a range of state-of-the-art algorithms and architectures.

The Roboflow logo is displayed in a bold, purple, lowercase sans-serif font.

Figure 40 Roboflow

Roboflow's platform includes a range of tools and features for working with image data, including data augmentation, labeling tools, and annotation validation. It also provides integration with a range of popular deep learning frameworks, including TensorFlow, PyTorch, and Keras, allowing users to export their models in a format that can be easily integrated into their own applications.

In addition to its core platform, Roboflow also provides a range of pre-trained models and datasets for common computer vision tasks, such as object detection, image segmentation, and facial recognition. These pre-trained models can be used as a starting

point for building custom models or integrated directly into applications using Roboflow's API.

Roboflow is known for its user-friendly interface and extensive documentation and is a popular choice among developers and companies looking to build and deploy custom computer vision models. Its platform is used in a range of applications, including autonomous vehicles, robotics, and surveillance systems.

## References

1. <https://github.com/Cogito2012/CarCrashDataset>
2. <https://www.youtube.com/watch?v=KrJTpYsWe3w>
3. Pillai, M.S.; Chaudhary, G.; Khari, M.; Crespo, R.G. Real-time image enhancement for an automatic automobile accident detection through CCTV using deep learning. *Soft Comput.* 2021, 25, 11929–11940. [Google Scholar] [CrossRef]
4. <https://www.youtube.com/watch?v=wuZtUMeiKWY&start=774>
5. <https://anvil.works/learn/tutorials/google-colab-to-web-app>
6. SALIL SRIVASTAVA, PRIYANSHU KUMAR ,JAGMOHAN DAS BAIRAGI ;TUSHAR AGARWAL ;GAURAV MEHROTRA ; Car Crash Detection using YOLOv3. May 18, 2020. [Google Scholar]
7. <https://www.youtube.com/watch?v=gKTYMfwPo4M>
8. <https://ijcsmc.com/docs/papers/May2022/V11I5202216.pdf>
9. Nancy, P.; Dilli Rao, D.; Babuaravind, G.; Bhanushree, S. Highway Accident Detection and Notification Using machine Learning. *Int. J. Comput. Sci. Mob. Comput.* 2020, 9, 168–176. [Google Scholar]
10. Ghosh, S.; Sunny, S.J.; Roney, R. Accident Detection Using Convolutional Neural Networks. In *Proceedings of the 2019 International Conference on Data Science and Communication (IconDSC)*, Bangalore, India, 1–2 March 2019. [Google Scholar] [CrossRef]
11. Ijjina, E.P.; Chand, D.; Gupta, S.; Goutham, K. Computer Vision-based Accident Detection in Traffic Surveillance. In *Proceedings of the 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Kanpur, India, 6–8 July 2019. [Google Scholar] [CrossRef][Green Version]
12. Gollapalli, M.; Rahman, A.U.; Musleh, D.; Ibrahim, N.; Khan, M.A.; Abbas, S.; Atta, A.; Farooqui, M.; Iqbal, T.; Ahmed, M.S.; et al. A Neuro-Fuzzy Approach to Road Traffic Congestion Prediction. *Comput. Mater. Contin.* 2022, 73, 295–310. [Google Scholar] [CrossRef]
13. Ahmed, M.I.B.; Rahman, A.-U.; Farooqui, M.; Alamoudi, F.; Baageel, R.; Alqarni, A. Early Identification of COVID-19 Using Dynamic Fuzzy Rule Based System. *Math. Model. Eng. Probl.* 2021, 8, 805–812. [Google Scholar] [CrossRef]

- 14.Rahman, A.; Basheer, M.I. Virtual Clinic: A CDSS Assisted Telemedicine Framework. In Telemedicine Technologies; Academic Press: Cambridge, MA, USA, 2019; pp. 227–238. [Google Scholar]
- 15.Alotaibi, S.M.; Rahman, A.U.; Basheer, M.I.; Khan, M.A. Ensemble Machine Learning Based Identification of Pediatric Epilepsy. Comput. Mater. Contin. 2021, 68, 149–165. [Google Scholar] [CrossRef]
- 16.Rahman, A.-U.; Abbas, S.; Gollapalli, M.; Ahmed, R.; Aftab, S.; Ahmad, M.; Khan, M.A.; Mosavi, A. Rainfall Prediction System Using Machine Learning Fusion for Smart Cities. Sensors 2022, 22, 3504. [Google Scholar] [CrossRef] [PubMed]
- 17.<https://www.youtube.com/watch?v=PSpO9PM8NS4>