**Project Report**

**On**

# Car Damage Detection



*Submitted*

*In partial fulfilment*

*For the award of the Degree of*

# PG-Diploma in Artificial Intelligence

**(C-DAC, ACTS (Pune))**

|  |  |
| --- | --- |
| **Guided By:** | **Submitted By:** |
| Prof. Saurabh Panwar. | Gaurav Kumar Singh (240340128009)  Mitali Giri (240340128016)  Satyasish Patranabish (240340128029)  Snehal Awadan (240340128033)  Swapnil Burud (240340128034) |

**Centre for Development of Advanced Computing(C-DAC), ACTS(Pune-411008)**

## Acknowledgment

Our journey towards achieving our destination of designing “**Car** **Damage Detection**” has finally come to a fruitful culmination. Our efforts and wholehearted cooperation of each and everyone has ended on a successful note. During this journey, we faced numerous unforeseen problems and unknown challenges. However, at these junctures, a few enterprising people stepped in and guided us in the right direction. We would like to sincerely acknowledge those who have supported and encouraged us during this tough journey. Many people met us during this endeavor and enriched us with their support and knowledge, personally and professionally, which resulted in the project being better than it could have been without them.

We sincerely thank our Guide Mr. Saurabh Panwar was guiding us throughout this project. Also, our warm thanks to C-DAC ACTS, Pune, which provided us this opportunity to carry out, this prestigious Project and enhance our learning in various technical to carry out, this prestigious Project and enhance our learning in various technical fields.

We thank them for providing us the reinforcement, confidence and most importantly the track for the project whenever we needed it.

Gaurav Kumar Singh (240340128009)

Mitali Giri (240340128016)

Satyasish Patranabish (240340128029)

Snehal Awadan (240340128033)

Swapnil Burud (240340128034)

## ABSTRACT

The **Car Damage Detection** project aims to address the challenge of accurately and efficiently detecting and classifying car damages from images. The purpose of this project is to develop a robust system that can be utilized in applications such as insurance claim processing, and car inspection, thereby reducing the time and labor involved in manual damage assessment.

By leveraging advanced deep learning techniques, this project seeks to enhance the accuracy, speed, and consistency of damage detection compared to traditional methods. The system is intended to support real-time analysis, enabling swift decision-making and streamlined workflows. Ultimately, this project aspires to contribute to the broader field of computer vision and artificial intelligence by providing a scalable solution that can be adapted to various automotive and inspection-related contexts, ensuring higher operational efficiency and improved user experience.

**Keywords**: Multi-class detection, YOLOv8, Streamlit, Ultralytics, PIL, OpenCV, Python, Insurance Claim process, Real-time analysis

## Table of Contents

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Title** | **Page No.** |
|  | **Front Page**  **Acknowledgment**  **Abstract** | **I**  **II**  **III** |
| **1**  **1.1**  **1.2** | **Introduction**  Introduction  Objective | **01**  **02** |
| **2** | **Literature Review** | **03-04** |
| **3** | **Methodology/Techniques** |  |
| **3.1** | Dataset | **05** |
| **4** | **Implementation** |  |
| **4.1 4.2 4.3 4.4**  **4.5**  **4.6** | Model Development  Data Augmentation  Model Training  Prediction  Model Testing  Observation | **12**  **13**  **14**  **15**  **16**  **17** |
| **5** | **Result and Analysis** |  |
| **5.1** | Results Analysis | **18** |
| **6**  **6.1**  **6.2** | **Conclusion**  Conclusion  Future Enhancement | **19**  **20** |
| **7** | **References** | **21** |

# Chapter 1 Introduction

## 1.1 Introduction

In recent years, the automotive industry has seen significant advancements in technology, particularly in the areas of automation and artificial intelligence. One of the critical applications of these technologies is in the detection and classification of car damages, which plays a vital role in insurance claim processing, car inspections, and fleet management. Traditional methods of damage assessment are labor-intensive, time-consuming, and often subjective, leading to inconsistencies and inefficiencies.

The **Car Damage Detection** project aims to address these challenges by developing an automated system capable of accurately and efficiently detecting and classifying car damages from images. YOLO (You Only Look Once) is a state-of-the-art object detection algorithm known for its speed and accuracy, making it an ideal choice for real-time applications. The latest iteration, YOLOv8, offers enhanced performance and greater flexibility, allowing for more precise damage detection.

This project involves collecting a comprehensive dataset of car images showing various types of damages, which are then labeled and organized for training, validation, and testing. Utilizing Google Colab for its GPU support, the YOLOv8 model is fine-tuned on this custom dataset, with training parameters optimized to achieve high accuracy and reliability. The system is designed to support real-time damage detection, providing immediate results and reducing the time and labor involved in manual assessments.

The successful implementation of this project has the potential to revolutionize the way car damages are assessed, providing significant benefits to insurance companies, car inspectors, and fleet managers. By automating the damage detection process, this system can lead to faster claim processing, more consistent inspections, and overall operational efficiencies, ultimately contributing to the broader adoption of AI technologies in the automotive industry.

## 1.2 Objectives

The objectives of the project work are as:

* The prime objective is to detect the damaged parts of the car images.
* Reduce the time and labor involved in manual damage assessment processes, making it faster and more consistent.
* Implement a system capable of performing real-time damage detection to enable swift decision-making in applications such as insurance claim processing and car inspections.

# Chapter 2 Literature Review

[1] Maleika Heenaye has deployed an application in this paper for the automatic detection and classification of vehicle damages, which can be used by insurance companies to process claims or by the police department to record accidents. Manually identifying the types and the severity of vehicle damage after an accident can be time-consuming. An automated damage detection application can help with insurance claims. Convolutional Neural Networks (CNN) have had great success in object classification. However, CNN has not been thoroughly investigated or applied for multiclass classifications of vehicle damages. In this paper, pre-trained CNN models, MobileNet, and VGG19 are adapted and used in transfer learning on the large-built dataset. This application achieved a MobileNet accuracy of 70% and a VGG19 accuracy of 50%.

[2] Najmeddine Dhieb proposes efficient and streamlined deep learning-based architectures for vehicle damage identification and localization in this paper. For feature extraction and damage identification, the proposed solution incorporates deep learning, instance segmentation, and transfer learning techniques. Its goal is to automatically detect vehicle damage, locate it, classify its severity levels, and visualize it by contouring its exact location.

[3] Umer Waqas considers the problem of car damage classification in this paper, where classifications include medium damage, huge damage, and no damage. For classification, the MobileNet model is proposed using deep learning techniques and transfer learning. Furthermore, moving toward automation comes with a variety of challenges; users can upload bogus images such as screenshots or take screenshots of computer screens, for example. To address this issue, a hybrid approach is proposed in which only authentic images are provided as input to an algorithm for damage classification. To detect fraudulent images, moiré effect detection and metadata analysis are used. Damage classification accuracy is 95%, and moiré effect detection accuracy is 99%.

[4] This study used the transfer learning-based models Inception V3, Xception, VGG16, VGG19, ResNet50, and MobileNet from Kera's library to train our model to predict damage and compare their efficacy. The proposed dataset is trained with these pre-trained models to achieve maximum accuracy and speed with minimal loss so that the model can be used to predict claims in real life. When compared to other models, MobileNet is more accurate and has a faster training time. The accuracy in forecasting damage and categorizing it into different types was 97.28%, which is significantly better than previous results in a similar test set.

# Chapter 3 Methodology and Techniques

## 3.1 Selection of Dataset:

For the Car Damage Detection project, we carefully selected a dataset that provides a comprehensive collection of labeled images representing various types of car damage. The dataset needed to be diverse regarding car models, damage types (e.g., scratches, dents, broken parts), and environmental conditions to ensure robust model training. We prioritized datasets that included high-resolution images and clear annotations, which are crucial for training a deep-learning model like YOLO. We also ensured the dataset was large enough to avoid overfitting and enable the model to generalize well to new, unseen data.

We have searched the required dataset from the following links:

1. Coco car detection dataset: <https://universe.roboflow.com/ayhan-gul-hgudf/car-damage-rlogo>
2. Car detection dataset: <https://universe.roboflow.com/car-damage-kadad/car-damage-images/dataset/3>
3. Damage Severity dataset: <https://universe.roboflow.com/mohit-classification-severity/damage-severity-6ya6c>
4. Car Damage: <https://universe.roboflow.com/ayhan-gul-hgudf/car-damage-rlogo>

### 3.1.1 Selected Dataset:

For the Car Damage Detection project, we selected the "**Car Damage**" dataset available on Roboflow, which offers a well-structured and labeled collection of images specifically focused on various types of car damage. This dataset was chosen due to its high-quality annotations, diverse range of damage types (such as dents, scratches, and broken parts), and coverage of different car models and environments. The availability of these features in the dataset makes it an excellent fit for training our YOLO model, ensuring it can accurately detect and classify damage in a variety of real-world scenarios.

**Selected dataset link**: <https://universe.roboflow.com/ayhan-gul-hgudf/car-damage-rlogo>

### 3.1.2 Selection of Algorithm

### We have so many algorithms to work with this project like,

**1) Faster R-CNN** (Region-Based Convolutional Neural Networks):

Faster R-CNN is one of the most popular object detection algorithms. It combines a Region Proposal Network (RPN) with a Fast R-CNN detector. The RPN generates region proposals, and the Fast R-CNN classifies these proposals and refines their bounding boxes.

**2) SSD** (Single Shot MultiBox Detector):

SSD detects objects in images by using a single deep neural network. It divides the image into a grid and applies bounding boxes of different aspect ratios at each grid point.

**3) RetinaNet:**

RetinaNet addresses the issue of class imbalance in object detection using Focal Loss, which down-weights the loss for well-classified examples. This makes it effective for detecting objects in cases where there is a large imbalance between classes.

**4)Mask R-CNN:**

An extension of Faster R-CNN that also outputs a binary mask for each detected object, making it suitable for both object detection and instance segmentation.

### 5)YOLO (You Only Look Once):

### It is a highly efficient object detection algorithm that performs object detection in real time by framing it as a single regression problem.

Among all the algorithms, we have selected the **YOLO** algorithm for our project.

**3.1.3 Selected algorithm:**

**YOLO** (You Only Look Once) was chosen for the vehicle damage detection project primarily due to its unique combination of speed and accuracy. Unlike traditional methods that use multiple stages, YOLO employs a single convolutional neural network (CNN) to simultaneously predict bounding boxes and class probabilities from an entire image.

### The image is divided into a grid, with each cell responsible for detecting objects whose centre falls within it. YOLO is known for its speed and accuracy, making it suitable for real-time applications.

### Over the years, several versions of YOLO have been released, including YOLOv1 through YOLOv8, each introducing improvements in architecture and performance, such as anchor boxes in YOLOv2 and multi-scale detection in YOLOv3, enhancing its effectiveness for a wide range of object detection tasks.

Additionally, YOLOv8, the latest iteration, has further improved on its predecessors with enhanced precision, making it capable of detecting even small and subtle damages with high accuracy. This balance of speed and accuracy, along with YOLO's proven track record in various object detection tasks, makes it the ideal choice for this project.

### 3.1.2 Data Labelling:

Data labeling is crucial for this car damage detection project because it ensures that the YOLO model can accurately identify and localize damages within images. Even though the goal is not to classify the types of damage, precise labeling provides the model with the necessary information to detect the exact location of damages on a car. Each labeled image includes bounding boxes that highlight the areas where damage is present, allowing the model to learn the patterns and features associated with car damage. This detailed labeling is essential for the model to perform well in real-world scenarios, where it needs to detect damage across various conditions and car types.

**Tools For Image labeling:**

There are many tools present for image labeling such as “LabelImg, Roboflow, VGG Image Annotator (VIA), Labelbox, and SuperAnnotate”, etc.

Among them, we have used the “**LabelImg**” tool.

**How to work with the LabelImg tool:**

1. **Step 1: Installation**

**i) Option 1: Installation via Python (pip)**

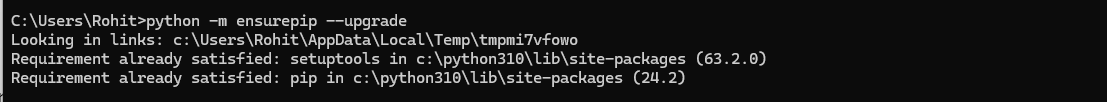
1. **Install Python**:

Ensure Python is installed on your system. You can download it from [python.org](https://www.python.org/downloads/).

1. **Install pip**:

Pip usually comes pre-installed with Python. If not, you can install it using:

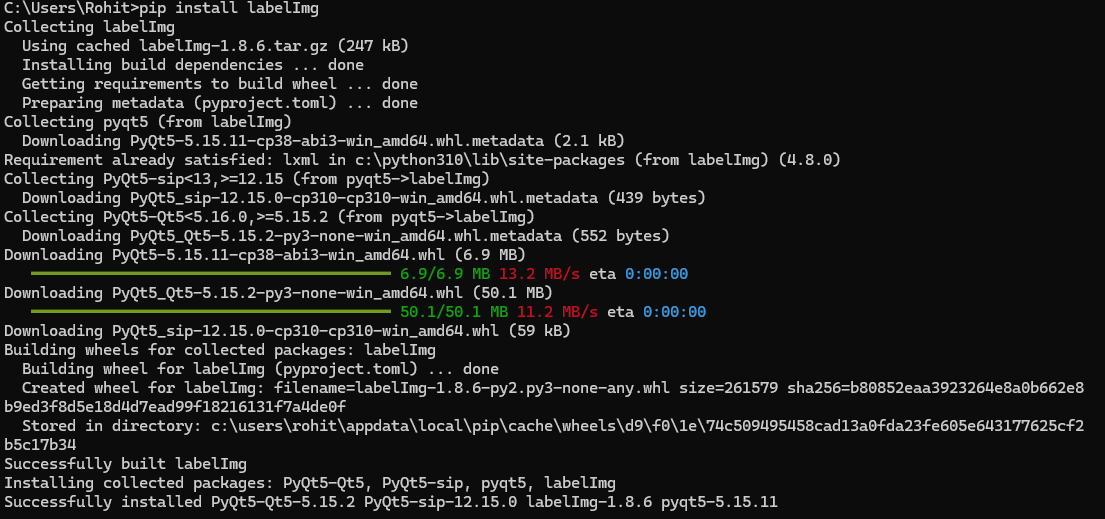
“python -m ensurepip –upgrade”



1. **Install LabelImg**:

Open a terminal or command prompt and run:

“pip install labelImg”

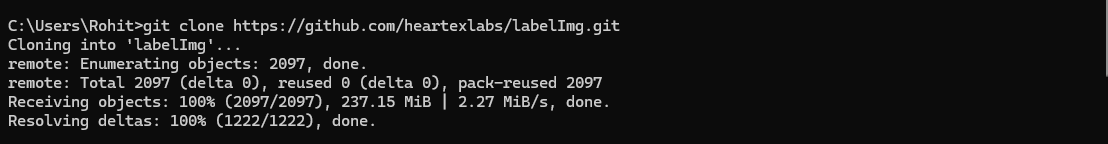


**ii) Option 2: Installation via GitHub (Manual Installation)**

1. **Clone the Repository**:

Open a terminal or command prompt and run:

“git clone <https://github.com/heartexlabs/labelImg.git>”



1. Navigate to the LabelImg directory:

“cd labelImg”



1. **Install Dependencies**:

Install the required Python dependencies using:

For Linux: “pip install -r requirements/requirements-linux-python3.txt”

For Windows: “pip install -r requirements/requirements-windows-python3.txt”

1. **Build and Launch**:

“pyrcc5 -o libs/resources.py resources.qrc”



1. Run LabelImg:

“python labelImg.py”



**Step 2: Launch LabelImg**

Once installed, you can launch LabelImg by simply typing labelImg in your terminal/command prompt, or by double-clicking the executable file if you installed it manually.

After that, it will open LabelImg’s GUI like this:

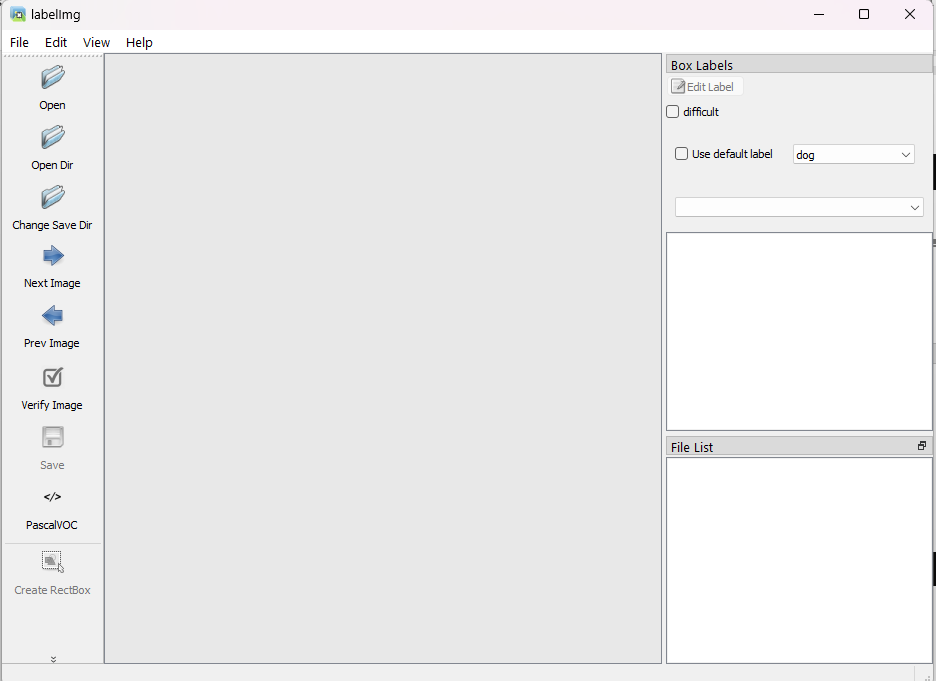


Fig: LabelImg GUI

**Step 3: Loading Images**

1. **Open Directory**:
   * Click on "Open Dir" on the left panel.
   * Navigate to the folder containing your images and select it. All images in this folder will be loaded for labeling.

**Step 4: Setting the Save Directory**

1. **Set the Save Directory**:
   * Click on "Change Save Dir".
   * Select or create a folder where your annotations (label files) will be saved.

**Step 5: Selecting the Annotation Format**

1. **Select the Format**:
   * Go to “**Menu 🡪 View 🡪 Auto Save Mode**” and select the desired annotation format.
   * For YOLO, ensure the YOLO format is selected.

**Step 6: Labeling the Images**

1. **Create Bounding Boxes**:

* Click on the "Create RectBox" button or press the W key.
* Click and drag on the image to draw a bounding box around the area of interest.

1. **Label the Bounding Box**:

* After drawing the box, a dialog box will prompt you to enter the label (e.g., "damage").
* Type the label and press OK.
* Repeat this process for all damage areas in the image.

1. **Navigation**:

* Use the arrow keys to move to the next or previous image.

**Step 7: Saving Annotations**

1. **Save the Annotations**:
   * Annotations are automatically saved if auto-save is enabled. If not, click "Save" or press “Ctrl+S” to save the annotation manually.
   * The annotations will be saved in the directory set earlier, typically in .txt files for YOLO format, with each file corresponding to an image.

**Step 8: Reviewing and Editing Labels**

1. **Review and Edit**:
   * You can review the labeled images by navigating through them using the arrow keys.
   * To edit a bounding box, select it and resize or move it as needed.
   * To delete a bounding box, select it and press Delete.

**Step 9: Close the Application**

1. **Exit**:
   * Once labeling is complete, you can close the application by clicking the X or selecting File > Exit.

**Step 10: Using Labeled Data**

1. **Use for Training**:
   * The labeled data can now be used for training your YOLOv8 model. The labels and bounding boxes saved in the .txt files correspond directly to the image files and are ready for use in model training.

**3.1.3 Organization of the data folders:**

This is one crucial step before putting the images to training. Before starting the training of our images, we need to organize our project files properly. First, we create a folder called "**data**" in our project directory.

Inside this "data" folder, we create two more folders named "**images**" and "**labels**." In each of these folders, create another folder named "**train**."

Finally, we copy all our images into the "train" folder within the "images" directory and place the corresponding annotation files (label files) into the "train" folder within the "labels" directory.

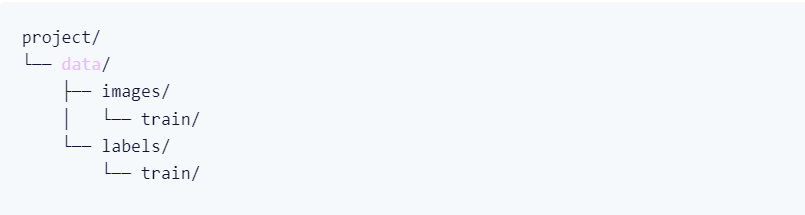
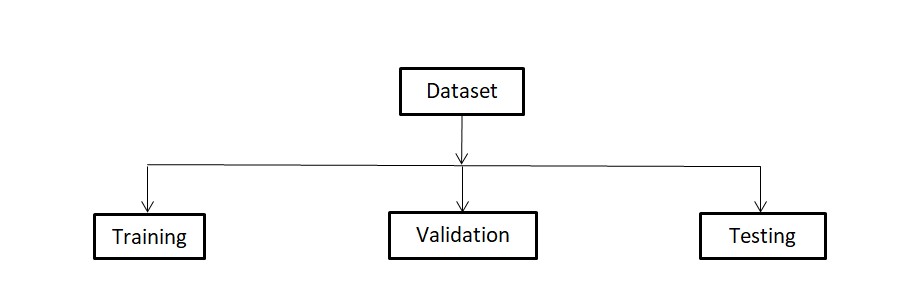


Fig: Project Folder Organization

### 3.1.4 Dataset splitting into Training and Validation set



Divide the dataset into-

70% -training set

20%- validation set

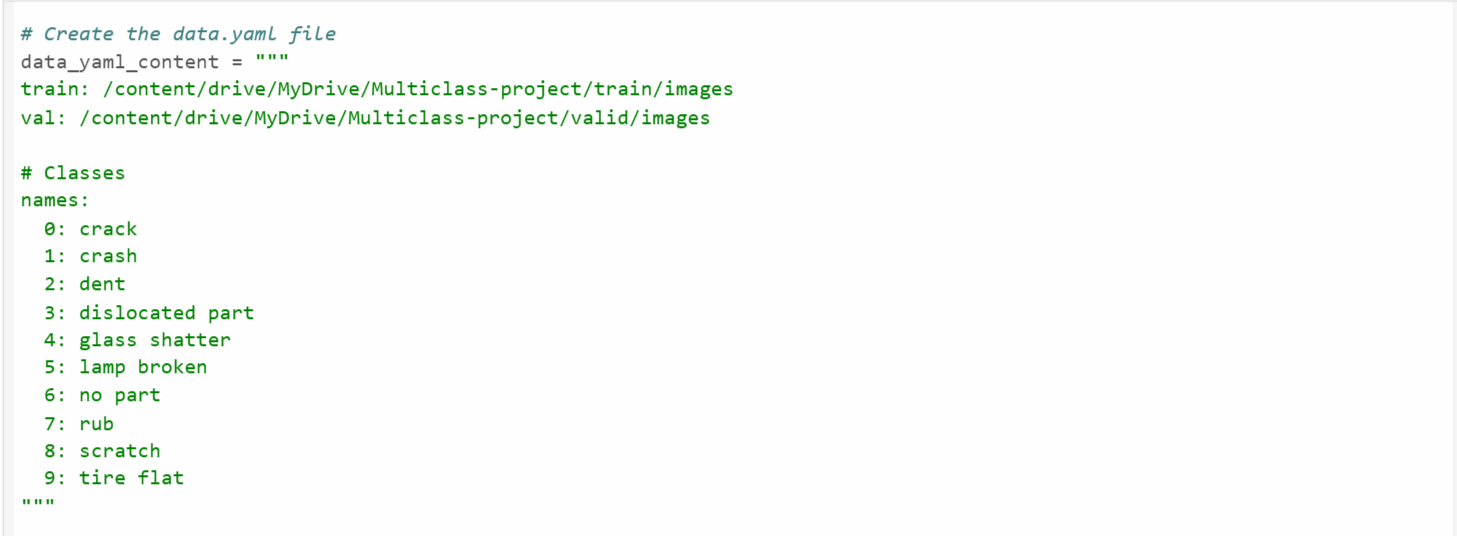
10%- test set

In the considered application domain, examples in the dataset for training an Object Detection system are composed of images that can represent multiple objects of interest. Then, we decided to split it by considering the number of objects for classes and not the number of available images.

**3.1.5 Creating data.yml file:**

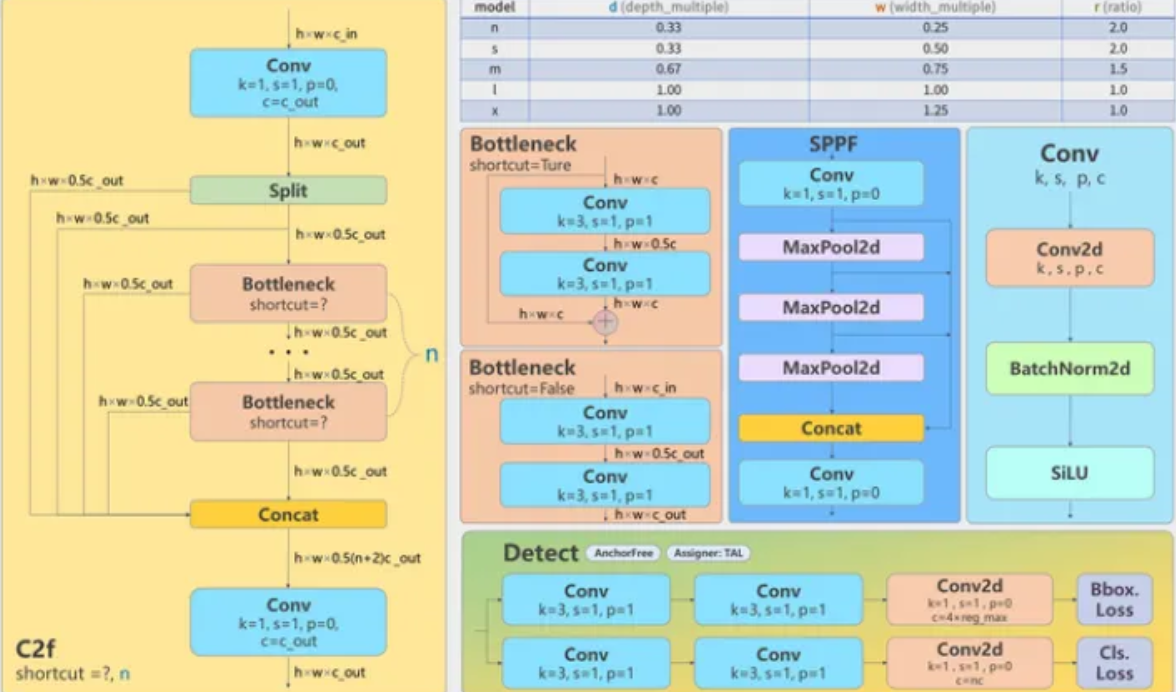
In YOLO (You Only Look Once), the data.yaml file plays a crucial role in configuring the model for training. It includes the paths to the training and validation datasets, specifies the number of classes the model needs to recognize, and lists the names of these classes. By centralizing this information in a single file, the data.yaml ensures that the training process is streamlined and consistent.

It allows YOLO to automatically understand the structure of the data, making it easier to manage and adjust as needed, especially when working with custom datasets for specific tasks like vehicle damage detection. This file is essential for setting up the training environment and ensuring that the model correctly interprets the dataset throughout the training process.



****

### 3.2.3 YOLOv8 Architecture



 **Backbone**:

* Function: The backbone is like the eyes of the model; it looks at the image and picks out important details.
* Activities:
  + At the start, it notices basic shapes and patterns like edges and textures.
  + As it goes deeper, it builds a detailed understanding of what's in the image.

 **Neck**:

* Function: The neck connects the backbone to the head, helping the model combine different pieces of information.
* *Activities*:
  + It mixes features from different parts of the image to detect objects of various sizes.
  + It also looks at the overall scene to improve the accuracy of object detection.

 **Head**:

* Function: The head is the decision-maker of the model, determining what objects are in the image and where they are.
* Activities:
  + It draws boxes around the objects it finds.
  + It gives each box a score to show how sure it is that the object is really there.
  + It organizes the objects by their categories, like cars, people, etc.

### 3.2.4 Model backbone – CSPDarkNet

The model backbone is the core feature extraction network within a deep learning model, typically used in tasks like object detection, image classification, and segmentation.

The YOLOv8 model is typically a variant of CSPDarknet (Cross Stage Partial Darknet). CSPDarknet is designed to enhance feature extraction efficiency by incorporating cross-stage connections that split and merge feature maps, allowing the network to achieve a better balance between computation and accuracy.

In YOLOv8, this backbone plays a crucial role in processing input images to extract detailed features at multiple scales, which are then used by the detection head to identify and localize car damages. The CSPDarknet backbone is particularly well-suited for tasks requiring real-time processing, making it ideal for applications like car damage detection where speed and accuracy are both essential.

## 3.2.5 Model neck – PANet

The model neck is a component in deep learning architectures, particularly in object detection models, that connects the backbone (feature extraction network) to the head (prediction layers). The neck typically serves to further refine and enhance the features extracted by the backbone, making them more suitable for the detection and classification tasks performed by the head.

The PAN structure in YOLOv8 helps aggregate features from both lower and higher levels, ensuring that the detection head receives a rich set of features that combine detailed spatial information with high-level contextual understanding. This makes the model more effective in detecting damages across different areas of the car, regardless of their size or position.

## 3.2.6 Model head-YOLOv8

In the YOLOv8 architecture, the model head is responsible for generating the final predictions based on the features extracted and refined by the backbone and neck. The head performs the actual object detection tasks, including predicting bounding boxes, object classes, and object confidence scores.

It consists of several key components that work together to generate final predictions based on the features processed by the backbone and neck. The head outputs bounding boxes that delineate the location and extent of detected car damages.

It also predicts class labels for these damages, categorizing them into predefined types, such as scratches or dents. Additionally, the head provides an objectness score for each bounding box, indicating the confidence that the detected area contains damage.

By leveraging multi-scale feature maps, the YOLOv8 head ensures accurate detection of damages across different sizes and resolutions, enabling the model to identify and localize car damage in diverse conditions effectively.

## 3.2.7 Annotations

**1)** **Bounding Boxes**: YOLOv8 uses boxes to mark the location of objects in an image. Each box is defined by its center’s position (x, y) and its width and height.

**2)** **Class Labels**: Every object in the image is given a label that tells the model what it is, like 'car,' 'person,' or 'dog.'

**3)** **Confidence Scores**: YOLOv8 includes a score for each detected object, showing how sure the model is that the object is there. A higher score means the model is more confident.

**4)** **Multiple Objects**: YOLOv8 can find several objects in one image. The format allows for multiple boxes and labels for each image, so it can identify everything it sees.

### 3.2.8. Transfer learning

Transfer learning is a machine learning technique where a model developed for a specific task is reused as the starting point for a model on a different but related task. Instead of training a model from scratch with a large dataset, transfer learning allows you to take a pre-trained model trained on a large and diverse dataset and fine-tune it for a new, related task.

Transfer learning is essential in the car damage detection project because it allows us to leverage pre-trained deep learning models, like YOLOv8, which have already been trained on vast datasets such as COCO. This approach significantly reduces the computational resources and time required to train a model from scratch, especially when dealing with limited labeled data specific to car damage.

By starting with a model that already understands general object detection, we can fine-tune it on our specific dataset, ensuring the model quickly adapts to accurately detect and localize car damages. This results in improved performance and faster deployment of the model in real-world applications.

# Chapter 4 Implementation

1. Hardware and Software Configuration:

**Hardware Configuration:**

* + GPU:20 GB

**Environment Setup:**

* + Python: 3.9 or 3.8
  + Ultralytics library
  + Streamlit
  + OpenCV

**Software Required:**

* **Anaconda**: It is a package management of software with free and open-source distribution of the Python and R programming language for scientific computations (data science machine learning applications, large-scale data processing, predictive analytics, etc.) that aims to simplify deployment.
* **Google Colab:** Google Colab is a cloud-based platform provided by Google that allows users to write, run, and share Python code in a Jupyter Notebook environment. It is particularly popular for machine learning, data analysis, and AI projects because it offers free access to powerful computing resources, including GPUs and TPUs, without the need to install anything locally. With Colab, users can collaborate in real time, making it an excellent tool for both education and research in data science and AI.
* **Jupyter Notebook**:

Jupyter is a web-based interactive development environment for Jupyter notebooks, code, and data.

Jupyter is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning.

Jupyterise xtensible and modular: write plug-ins that add new components and integrate with existing ones.

## 4.1 Model Development

For the car damage detection project, model development involves several key stages. Initially, the YOLOv8 model is selected for its real-time object detection capabilities and efficiency. The development process begins with the preparation of a high-quality, labeled dataset that includes diverse images of car damage.

This dataset is then split into training and validation sets. The YOLOv8 model is trained on the training set, where it learns to detect and localize car damage by identifying patterns and features within the images.

During training, hyperparameters are tuned to optimize performance, and the model is validated on the validation set to ensure it generalizes well to unseen data. After training, the model is evaluated on test data to assess its accuracy and robustness.

Finally, any necessary fine-tuning or adjustments are made based on performance metrics, and the model is integrated into an application or system for real-world deployment.

## 4.2 Data Augmentation:

Data augmentation is a technique used in machine learning and deep learning to artificially increase the size and diversity of a training dataset. This is achieved by applying various transformations to the original data, such as rotating, flipping, scaling, cropping, or adding noise to images.

The purpose of data augmentation is to improve the model's generalization ability by exposing it to a wider range of variations during training. By doing so, the model becomes more robust and performs better on unseen data, as it learns to recognize patterns and features under different conditions. Data augmentation is particularly useful in image classification, object detection, and other computer vision tasks where the available labeled data may be limited or imbalanced.

## 4.3 Predictions



**Output for the prediction:**

****

****

## 4.5 Observation

The observation for detecting images of the given car using the YOLOv8 model involves several key findings. Firstly, the model demonstrates a strong ability to accurately detect and localize visible damages on the car.

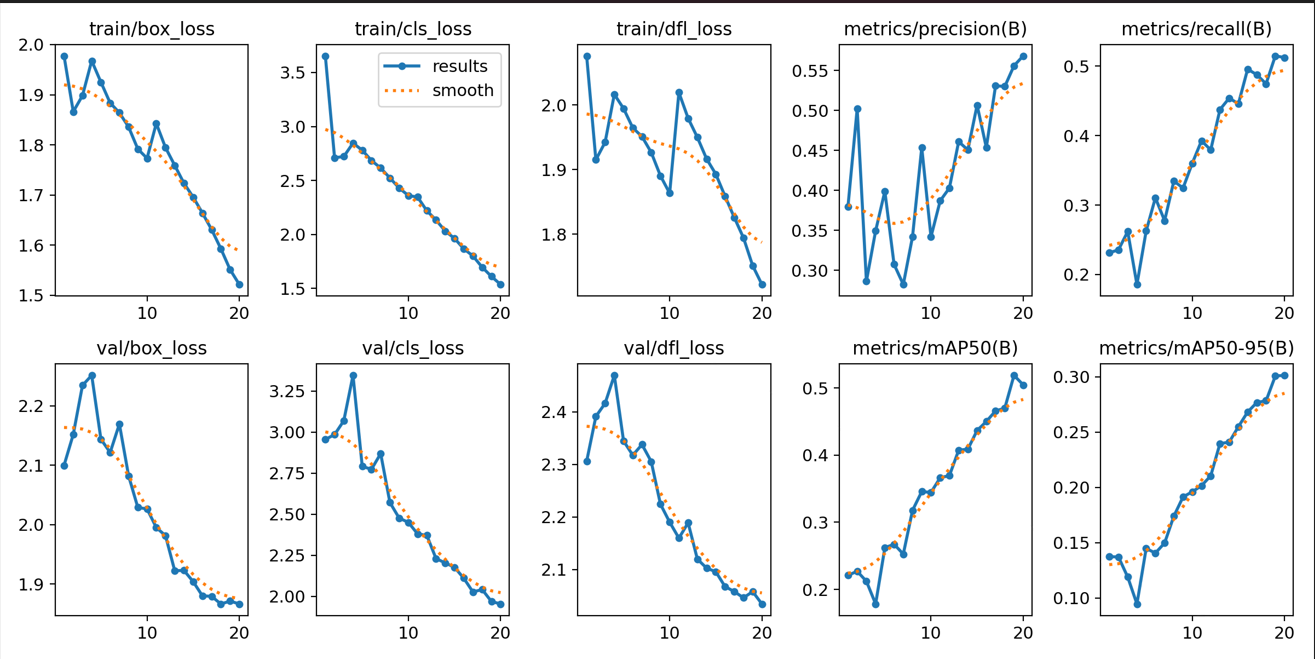
The detection is consistent across various angles and distances, indicating the model's robustness in handling different perspectives. However, the model's performance slightly decreases when detecting damages in low-light conditions or on darker-colored cars, where the contrast between the damage and the car surface is less pronounced.

Additionally, smaller or more subtle damages may not always be detected as reliably, suggesting that the model might benefit from further fine-tuning or training with more specific examples. Overall, the observations confirm the model's effectiveness for most standard car damage detection tasks, while also identifying specific areas for potential improvement in future iterations.

Top of Form

# Chapter 5 Result and Analysis

## 5.1 Result



**train/box\_loss**:

This plot represents the bounding box regression loss during training. It measures how well the predicted bounding boxes align with the ground truth boxes. The downward trend indicates that the model is learning to better predict the bounding boxes as training progresses.

**train/cls\_loss**:

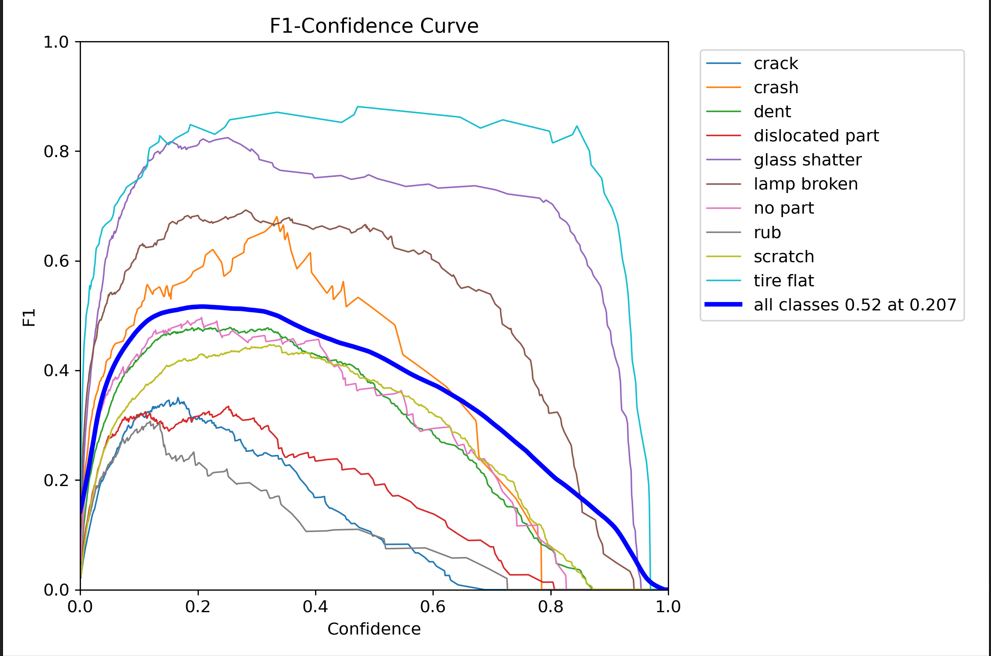
This plot shows the classification loss during training. It measures how well the model classifies the objects within the bounding boxes. The decreasing trend suggests that the model is improving its classification accuracy over time.

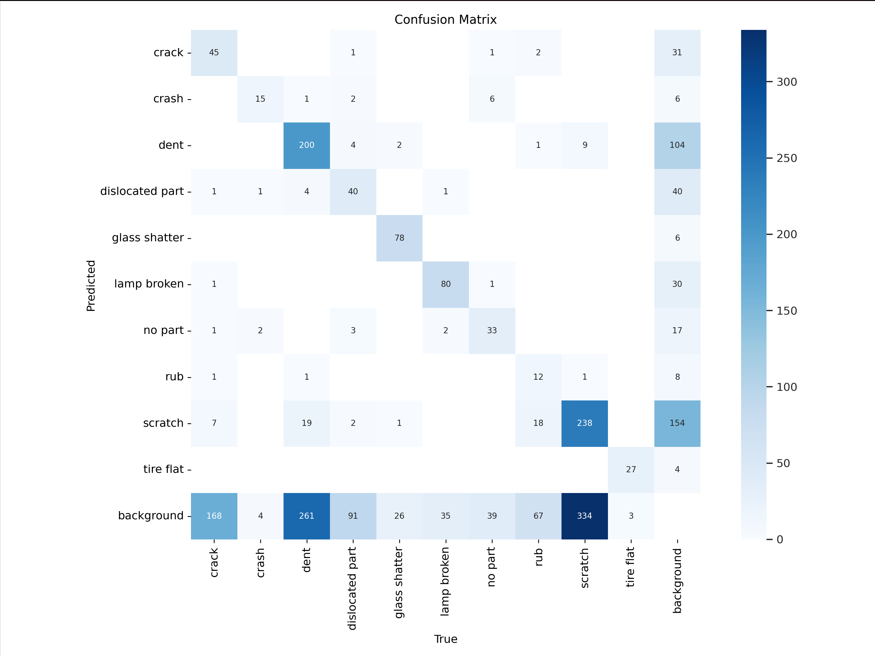
**train/dfl\_loss**:

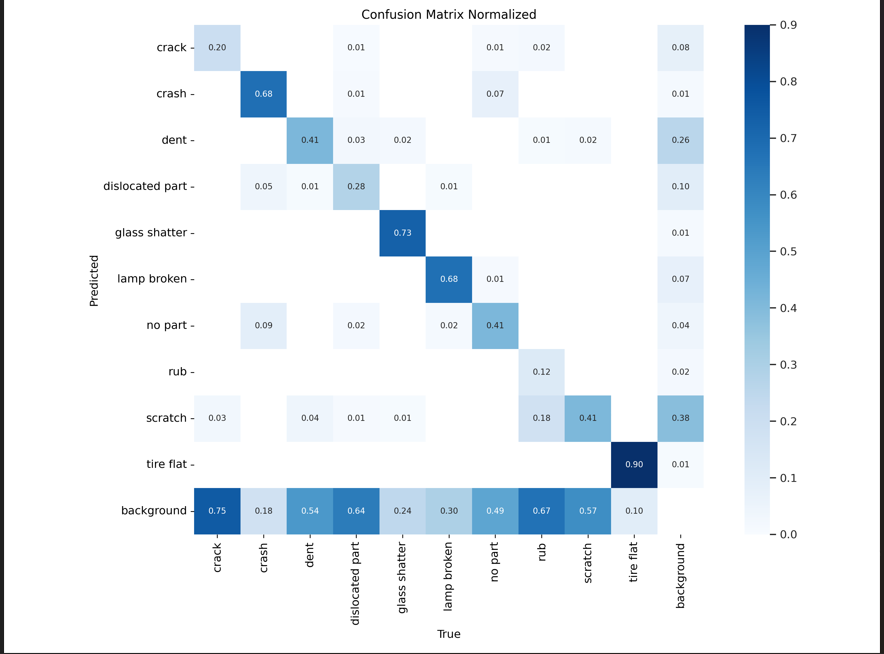
This represents the Distribution Focal Loss during training, which is used to improve the localization of bounding boxes. A decreasing trend indicates better localization of the object within the predicted bounding boxes.

**val/box\_loss**:

Similar to the train/box\_loss, but for the validation dataset. The consistent decrease indicates that the model is generalizing well to unseen data in terms of bounding box predictions.







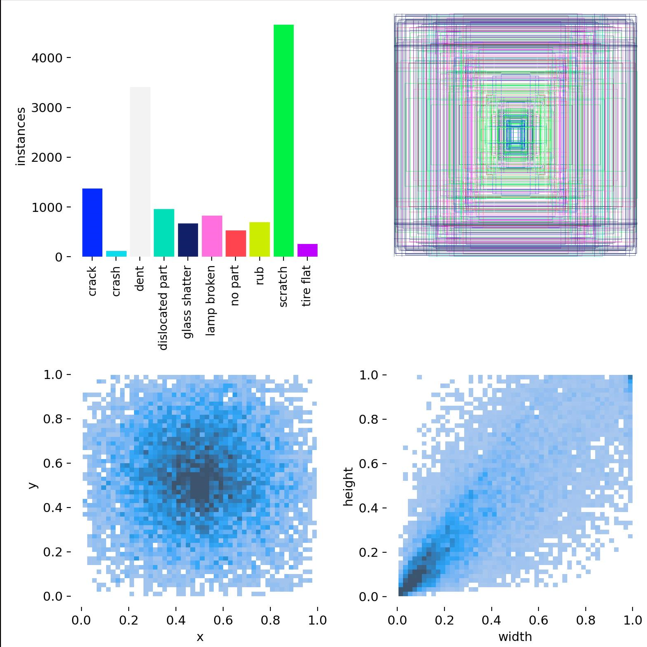


Fig: Label Correlation

**Front-end output:**

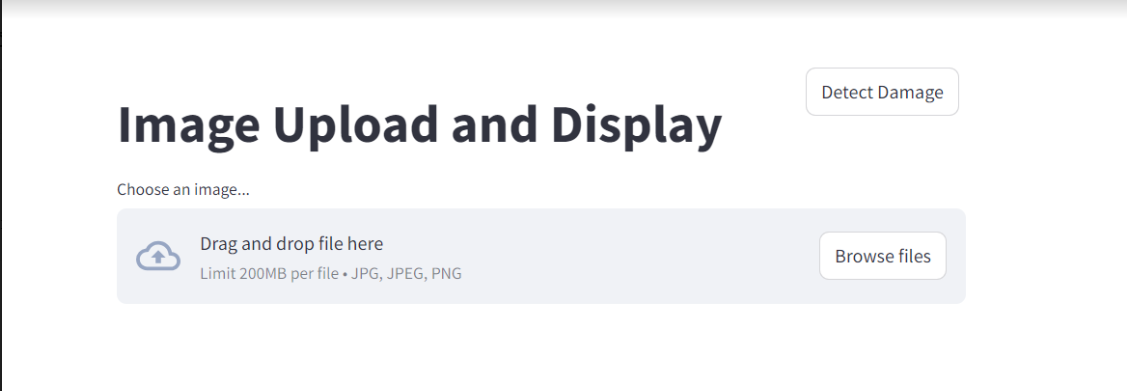
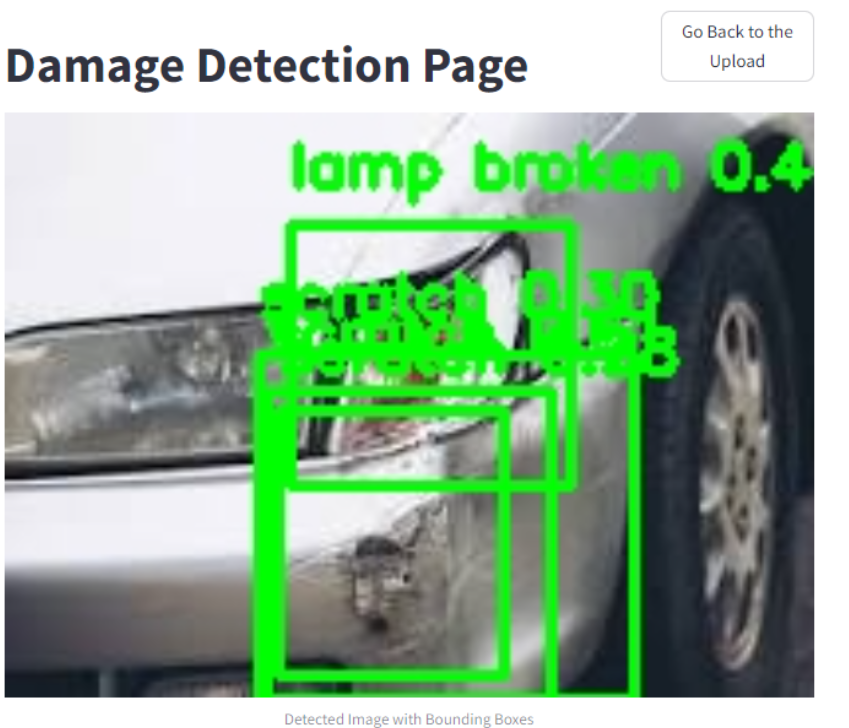
****

Fig: Project GUI





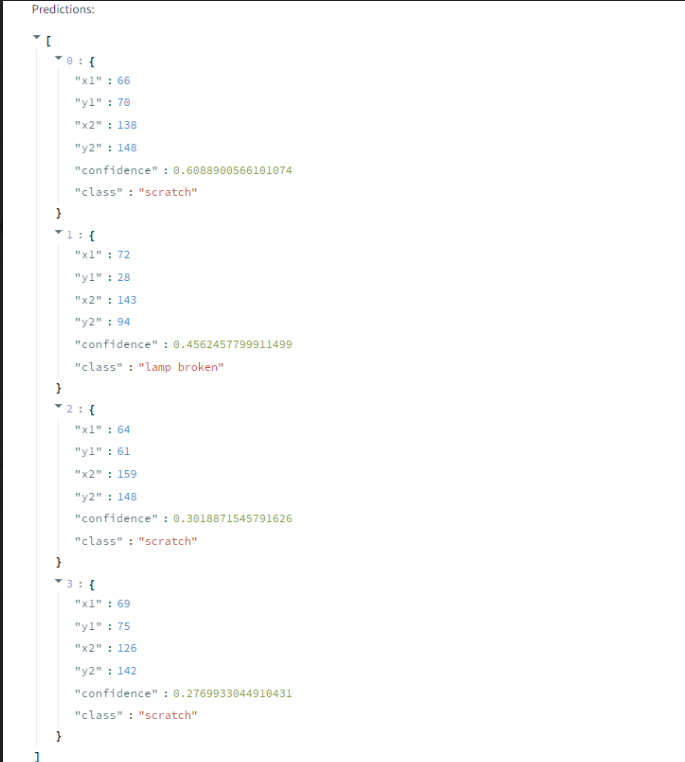
****

Fig: Coordinates of Bounding Box

­

## 5.2 Analysis

The car damage detection project utilizing YOLOv8 demonstrates promising results in both the training and validation phases. The analysis of the training metrics reveals a consistent decrease in losses (box, classification, and distribution focal loss), indicating that the model is effectively learning to predict the bounding boxes and classify the damages accurately. The gradual reduction in these losses suggests that the model is becoming more proficient in localizing the damaged areas within the images and distinguishing between different types of damage.

Moreover, the upward trends in precision and recall metrics underscore the model's growing capability to make accurate predictions. Precision's improvement shows that the model is reducing false positives, while the increasing recall indicates a decrease in false negatives, meaning the model is correctly identifying more instances of damage. This balance between precision and recall is crucial, especially in practical applications like insurance claims and vehicle inspections, where both accuracy and thoroughness are essential.

The validation metrics further corroborate the model’s robustness. The decreasing validation losses suggest that the model is generalizing well to unseen data, which is critical for real-world applications. Additionally, the improvement in mean Average Precision (mAP) across various Intersections over Union (IoU) thresholds reflects the model's strong performance across different levels of prediction accuracy and overlap with the ground truth.

# Chapter 6 Conclusion

## 6.1 Conclusion

The **Car Damage Detection** project demonstrates the significant potential of applying advanced deep learning techniques to the field of car damage assessment. By leveraging the YOLOv8 model, we have developed a robust and efficient system capable of accurately detecting car damage from images in real-time.

The YOLOv8-based model delivers high damage detection and classification accuracy, reducing the subjectivity and variability associated with human assessments. This ensures more reliable and consistent results.

The automated system significantly reduces the time and labor required for car damage assessment, making the process faster and more efficient. This is particularly beneficial for applications such as insurance claim processing, where quick and accurate damage evaluation is critical.

The developed model is scalable and can be adapted to various contexts within the automotive industry, including fleet management and car inspections. Its flexibility allows for easy integration with existing workflows and systems.

The real-time capabilities of the YOLOv8 model enable immediate damage detection, facilitating swift decision-making and enhancing operational efficiency.

This project contributes to the broader field of artificial intelligence and computer vision by demonstrating the effective use of YOLOv8 for a practical and impactful application. The findings and methodologies can serve as a foundation for further research and development in related areas.

## 6.2 Future Enhancement

**1) Integration with 3D Imaging:**

Integrating 3D imaging technologies, such as LIDAR or stereo cameras, could enhance the accuracy of damage detection by providing additional spatial information. This integration can help in better assessing damage on complex surfaces and in scenarios where 2D images might be insufficient.

**2)** **Real-Time Damage Detection in Autonomous Cars:**

Implementing the damage detection system into autonomous cars could allow for real-time monitoring and assessment of car conditions. This capability would enable proactive maintenance alerts and automatic reporting of damage after incidents, improving safety and reducing repair costs.

**3) Expansion to Different Car Types:**

Extending the model to detect damage across a broader range of car types (e.g., trucks, motorcycles, buses) could enhance its applicability and usefulness. This would require retraining the model with diverse datasets to ensure robust performance across different car categories.

**4) Development of a Mobile Application:**

Creating a mobile application that leverages the damage detection model could provide a user-friendly interface for end-users to easily assess car damage. Such an app could include features for uploading images, receiving damage assessments, and generating repair estimates.

# Chapter 7

# References

[1] Maleika Heenaye–Mamode Khan, Mohammad Zafir

Hussein Sk Heerah, Zuhairah Basgeeth, “Automated Detection of Multi-class Vehicle Exterior Damages using Deep Learning”, International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), 2021

[2] Najmeddine Dhieb, Hakim Ghazzai, Hichem Besbes, Yehia Massoud, “A Very Deep Transfer Learning Model for Vehicle Damage Detection and Localization”, 31st International Conference on Microelectronics (ICM), 2020

[3] Umer Waqas, Nimra Akram, Soohwa Kim, Donghun Lee, Jihoon Jeon, “Vehicle Damage Classification and Fraudulent Image Detection Including Moiré Effect Using Deep Learning”, IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2020

[4] Sruthy C M, Sandra Kunjumon, Nandakumar R, “Car Damage Identification and Categorization Using Various Transfer Learning Models”, 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021

For the architecture of yolov8: <https://medium.com/@juanpedro.bc22/detailed-explanation-of-yolov8-architecture-part-1-6da9296b954e>

For annotation of this project.

i)<https://yolov8.org/yolov8-annotation-format/#YOLOv8_Annotation_Format>

ii)<https://www.youtube.com/watch?v=m9fH9OWn8YM&list=PLb49csYFtO2FXGMZxqmPrw_0GPJnPR0Up&index=2&t=401s>

For frontend part: <https://youtube.com/playlist?list=PLfFghEzKVmjvuSA67LszN1dZ-Dd_pkus6&si=MyTyEFjYfJfJvY5c>