20CS811 PROJECT PHASE II

SMART DETECTION OF CAR DEFECTIVE PARTS WITH RECOMMENDATIONS

A PROJECT REPORT

Submitted by

VETRIVELM **111720102172**

THAMMIREDDIGARIHEMANTHKRISHNA **111720102160**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

R.M.K. ENGINEERING COLLEGE

(An Autonomous Institution)

R.S.M. Nagar, Kavaraipettai-601 206

MARCH 2024

R.M.K. ENGINEERING COLLEGE

(An Autonomous Institution)

R.S.M. Nagar, Kavaraipettai-601 206

BONAFIDE CERTIFICATE

Certified that this project report **"Smart Detection of Car Defective Parts with Recommendations"** is the bonafide work of **VETRIVEL M (111720102172)** and **THAMMIREDDIGARI HEMANTH KRISHNA (111720102160)** who carried out the **20CS811 Project Phase II** work under my supervision**.**

SIGNATURE

Dr. T. Sethukarasi, M.E., M.S. Ph.D.,
Professor and Head
Department of Computer Science and Engineering
R.M.K. Engineering College
R.S.M. Nagar, Kavaraipettai,
Tiruvallur District– 601206.

SIGNATURE

Ms. Rohini S, M.E., (Ph.D.,)
Supervisor
Assistant Professor
Department of Computer Science and Engineering
R.M.K. Engineering College
R.S.M. Nagar, Kavaraipettai,
Tiruvallur District–601206.

Submitted for the Project Viva–Voce held on …… at **R.M.K. Engineering College**, Kavaraipettai, Tiruvallur District– 601206.

INTERNALEXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We earnestly portray our sincere gratitude and regard to our beloved **Chairman Shri. R. S. Munirathinam,** our **Vice Chairman, Shri. R. M. Kishore** and **our Director, Shri. R. Jyothi Naidu,** for the interest and affection shown towards us throughout the course.

We convey our sincere thanks to our **Principal**, **Dr. K. A. Mohamed Junaid,** for being the source of inspiration in this college.

We reveal our sincere thanks to our **Professor and Head of the Department, Computer Science and Engineering, Dr. T. Sethukarasi,** for her commendable support and encouragement for the completion of our project.

We would like to express our sincere gratitude for our Project Coordinator **Dr. C. Geetha, M.E., Ph.D., Professor** and Project Guide **Ms. Rohini S, M.E., (Ph.D.,) Assistant Professor** for their valuable suggestions towards the successful completion for this project in a global manner.

We take this opportunity to extend our thanks to all faculty members of Department of Computer Science and Engineering, parents and friends for all that they meant to us during the crucial times of the completion of our project.

TABLE OF CONTENTS

ABSTRACT

Vehicles can sustain damage from accidents, collisions, natural disasters, and regular wear and tear. These issues impact various stakeholders like car manufacturers, insurers, rental companies, and individual owners. This project focuses on developing an integrated system for car damage detection utilizing multiple Convolutional Neural Networks (CNNs) within a Flask web application framework. The system integrates the advanced CNN techniques which precisely identify and categorize damage types, including dents, scratches, and deformations, from input images. Additionally, QR code generation and scanning functionalities are incorporated to encode and retrieve detailed information about the detected damages, enhancing data accessibility and organization. The user interface, facilitated by Flask, provides a user-friendly platform for image uploads, and damage report views. This has the potential to streamline the detection process and enhance customer satisfaction across the automotive industry.

Keywords – Convolutional Neural Network (CNN), Car Damage Detection, Flask Web Application, QR Code Generation.

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	EXPANSION
1	AI	ARTIFICIAL INTELLIGENCE
2	CNN	CONVOLUTIONAL NEURAL NETWORK
3	QR Code	QUICK RESPONSE CODE
4	UML	UNIFIED MODELING LANGUAGE

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Despite the growing demand for efficient and accurate methods of assessing vehicle damage, current manual inspection processes are often time-consuming, subjective, and prone to errors. This inefficiency leads to delays in repair approvals, increased insurance costs, and customer dissatisfaction. Therefore, there is a pressing need for an automated and reliable car damage detection system that utilizes advanced technologies such as computer vision and machine learning to swiftly and accurately identify and assess damage to vehicles. Car damage detection can be a challenging task, especially when the damage is minor or hard to see, making it crucial in various applications. The primary problem addressed by this project is significant, as inaccurate or delayed damage assessments can lead to increased costs for insurers, longer processing times for claims, and customer dissatisfaction. The goal of this project is to develop a system that can accurately and reliably detect the damages using multiple CNNs. The system should be able to detect damage of all types and severities, even in challenging conditions such as poor lighting or occlusion.

1.2 Literature Survey

The research paper titled “Vehicle Damage Severity Estimation for Insurance Operations Using In-The-Wild Mobile Images” authored by DimitriosMallios, Li Xiaofei, Niall McLaughlin, Jesus Martinez Del Rincon, Clare Galbraith, Rory Garland,They investigates the current landscape of automatic car damage assessment, a critical concern for the insurance industry. Existing approaches, primarily leveraging computer vision techniques, have made strides in damage detection but lack a comprehensive solution. Addressing this gap, a recent paper proposes an innovative pipeline utilizing photographs and structured data to estimate damage severity. Unique to this work is the incorporation of semantic segmentation for detailed identification of damaged car parts. The model's training and evaluation on a diverse dataset of real-world insurance claims demonstrate its potential in providing accurate and practical solutions for cost estimation. This research not only signifies a departure from conventional methods but also contributes to the advancement of automatic car damage assessment, offering a promising direction for future developments at the intersection of computer vision and insurance claims processing.**[1]**

The research paper titled “Vehicle-damage-detection segmentation algorithm based on improved mask RCNN” authored by Q. Zhang, X. Chang, and S. B. Bian,The main result is a methodology to estimate quickly and easily repair costs of vehicles involved in road accidents. Real-world accidents analyzed in this paper are Crashworthiness Data System (NASS CDS Database)- field research teams located across a country study about 5000 crashes a year; Audaplus estimation system which used data about costs of the vehicle parts and the time necessary to replace it based on manufacturer's information. This study has developed a retrospective methodology to estimate easily repair costs of vehicles involved in road accidents with the front zone involved. Using residual deformation measurements based on Tumbas and Smith's protocol, it is viable to estimate deltaV and absorbed energy for the vehicle involved in an accident. **[2]**

The research paper titled “CarDD: A New Dataset for Vision-based Car Damage Detection” authored by Xinkuang Wang, Wenjing Li, Zhongcheng Wu,In this paper, The Automatic car damage detection in the car insurance industry has drawn significant attention, but the absence of accessible high-quality datasets hampers the development of effective models. Addressing this gap, this paper introduces Car Damage Detection (CarDD), the first large-scale public dataset tailored for vision-based car damage detection and segmentation. CarDD comprises 4,000 high-resolution car damage images, featuring over 9,000 well-annotated instances across six damage categories. The paper outlines the image collection, selection, and annotation processes, providing a statistical dataset analysis. The authors conduct extensive experiments on CarDD using state-of-the-art deep learning methods for various tasks, offering comprehensive analyses to underscore the unique challenges in car damage detection. CarDD, as a benchmark dataset, is poised to advance research in the field, providing a valuable resource for enhancing the robustness and effectiveness of automatic car damage detection systems in real-world applications.**[3]**

The research paper titled “Car Damage Assessment Based on VGG Models” authored by Phyu Mar Kyu, KuntpongWoraratpanya, This paper addresses the surge in insurance claims correlating with the proliferation of the automobile industry, focusing on simultaneous claims and claims leakage mitigation. Employing advanced artificial intelligence, specifically VGG16 and VGG19 deep learning algorithms, the study aims to detect and assess car damage in real-world datasets. Utilizing domain-specific pre-trained CNN models initially trained on ImageNet, the authors fine-tune and apply transfer learning to adapt to specific tasks, enhancing system accuracy. Results indicate that transfer learning and regularization outperform fine-tuning, achieving 94.56% accuracy for VGG16 and 94.35% for VGG19 in

damaged detection, 74.39% for VGG16 and 76.48% for VGG19 in damage localization, and 54.8% for VGG16 and 58.48% for VGG19 in damage severity. This study underscores the efficacy of deep learning approaches, emphasizing the significance of transfer learning and regularization techniques for improved car damage assessment in insurance industries.**[4]**

The research paper titled "A Very Deep Transfer Learning Model for Vehicle Damage Detection and Localization" authored by NajmeddineDhieb, Hakim Ghazzai, Hichem Besbes, Yehia Massoud, In response to the significant issue of claims leakage causing substantial financial losses for insurance companies, this paper addresses the problem by leveraging advanced Artificial Intelligence (AI) and deep learning technologies. Focusing on inefficient claims processing, fraud, and poor decision-making as the root causes of these losses, the study proposes automated deep learning architectures for vehicle damage detection and localization. Combining instance segmentation and transfer learning techniques, the proposed solution aims to automatically identify and locate damages in vehicles, classify severity levels, and visualize the results. The research showcases the effectiveness of the proposed transfer learning solution, utilizing the Inception-ResnetV2 pre-trained model, in outperforming another pre-trained model, VGG16, particularly in features extraction and damage detection/localization. This work contributes to the application of AI and deep learning in the insurance industry for more accurate and efficient claims processing.**[5]**

The research paper titled "Car Damage Detection and Assessment Using CNN" authored by AtharvaShirode, TejasRathod, ParthWanjari, AparnaHalbe, If the vehicle is insured, an insurance agent will go to the customer's home to investigate and prepare a report. Book review is a time-consuming process. However, thanks to significant advances in deep learning algorithms, it can be used to solve these problems in the insurance industry. Two CNN models are used in the proposed solution. VGG16 is used to diagnose damage to the vehicle as well as the location and severity of the damage. Mask RCNN is used to isolate damaged areas. Both models provide reasonable estimates of vehicle damage, allowing insurance companies to submit insurance claims without wasting resources and time on manual checks.**[6]**

The research paper titled "Car Damage Identification and Categorization Using Various Transfer Learning Models" authored by Sruthy C M, Sandra Kunjumon, Nandakumar R, This study used the transfer learning based models Inception V3, Xception, VGG16, VGG19, ResNet50, and Mobile Net 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 a 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.**[7]**

The research paper titled “A large-scale car dataset for fine-grained categorization and verification” authored by Linjie Yang et al, This study used to the underexplored domain of car-related vision tasks, this paper endeavors to shed light on various problems and applications within the realm of cars that have received comparatively less attention from the vision community. The authors emphasize the existence of unexplored challenges and opportunities within the car-centric domain and introduce their ongoing initiative, the “CompCars” dataset. This large-scale dataset is designed to encompass diverse car views, internal and external parts, and rich attributes, presenting a cross-modality nature with both surveillance and web-based sets. The paper further showcases applications utilizing the dataset, including car model classification, verification, and attribute prediction, while discussing specific challenges inherent in car-related problems and proposing potential avenues for further research. **[8]**

1.3 SYSTEM REQUIREMENTS

1.3.1 Hardware Requirements

Any kind of internet connection like WIFI, modem data etcetera, to allow the browser interfaces to connect to the website. The website can be accessed through any devices like computer, laptop, tablet, etc.

- CPU: Inteli5 processor with 64-bit Operating System
- RAM:8 GB
- HARD DISK: 40 GB
- STORAGE: ITB Storage
- INTERNET: wireless adapter(Wi-Fi)

1.3.2 Software Requirements

Some of the software interfaces which you can use to access our website are:

- Operating system : Windows 11

- Coding Language : Python 3.7
- Software Tools Used : SQLite
- Web Development Framework: Flask
- Libraries Used : Pandas, qrcode, pyqrcode , Keras, Matplotlib, Numpy.

1.3.3 Feasibility Study

A feasibility study evaluates the practicality of implementing a system by analyzing key criteria. It measures the potential organizational benefits and applicability of an information system. This particular feasibility study aims to assess the viability of creating a car damage detection system. The study will scrutinize the project's technical, market, economic, social, and regulatory aspects. To determine feasibility, the study will adhere to the following steps.

a) Technical Feasibility:

Assessing the technical feasibility of a car damage detection system involves evaluating the availability of necessary technologies like computer vision, machine learning algorithms, and image processing techniques, determining whether further development is required. Additionally, it's crucial to gauge the accuracy and reliability of the technology in detecting various types of car damage across diverse environmental conditions and lighting scenarios. Integration with existing automotive systems or platforms must be considered to ensure seamless implementation, and scalability is paramount to accommodate a large volume of cars and different types of damage efficiently. These factors collectively determine the feasibility of deploying a robust and effective car damage detection system in real-world applications.

b) Operational Feasibility:

Operational feasibility of implementing a car damage detection system involves assessing several critical factors. Firstly, it's essential to gauge user acceptance among car owners, insurance companies, and automotive service providers to ensure willingness to adopt and utilize the system. Additionally, evaluating the feasibility of providing comprehensive training and ongoing support to users is crucial for effective utilization of the technology. Moreover, assessing the operational impact of implementing the system on existing processes and workflows within automotive repair shops or insurance companies is vital to anticipate any potential challenges or disruptions. By carefully considering these aspects, stakeholders can better determine the practicality and viability of integrating a car

damage detection system into existing operations.

c) Economic Feasibility:

In assessing the economic feasibility of implementing a car damage detection system, a thorough cost-benefit analysis is imperative to ascertain whether the benefits surpass the incurred costs. A crucial aspect of this analysis involves calculating the expected return on investment (ROI), factoring in variables such as reduced inspection time, enhanced accuracy in damage assessment, and potential cost savings for insurance companies. Additionally, estimating the budget and resource requirements for the development, deployment, and maintenance of the system over time is essential for informed decision-making. By evaluating these economic considerations comprehensively, stakeholders can determine the viability and potential long-term financial benefits of integrating a car damage detection system into their operations.

d) Social and Regulatory Feasibility:

In assessing the social and regulatory feasibility of implementing a car damage detection system, careful consideration must be given to privacy and data security concerns, particularly regarding the collection and storage of car images for damage detection purposes. Compliance with relevant regulations and standards in the automotive industry, including safety standards and data protection regulations, is imperative to ensure legal and ethical operation. Additionally, evaluating social acceptance is crucial to ascertain whether the implementation of the system aligns with societal norms and expectations surrounding car maintenance and repair processes. By addressing these factors comprehensively, stakeholders can navigate potential challenges and ensure the responsible and sustainable deployment of the car damage detection system.

By conducting a comprehensive feasibility study covering these aspects, stakeholders can make informed decisions about the viability of implementing a car damage detection system.

CHAPTER 2
SYSTEM ANALYSIS

2.1 Existing System

Existing approaches that attempt to directly estimate damage costs from images often encounter scalability challenges. These methods typically rely on training models to correlate visual features of damages with associated repair costs. However, these models require frequent re-training and adaptation as new car models are released, leading to significant maintenance overhead and potential inaccuracies. In contrast, our model adopts a more robust approach by separating the damage estimation task from the cost estimation task. By decoupling these processes, our model gains scalability and adaptability. The damage estimation component focuses solely on accurately identifying and characterizing damages in images, leveraging advanced techniques such as CNNs to achieve high accuracy and reliability. As a result, our model offers a more scalable and sustainable solution for estimating repair costs from images, ensuring robust performance across diverse automotive contexts without the need for constant re-training.

2.1.1 Disadvantages of Existing System

- Traditional car damage detection methods often rely on manual inspection by human experts, which can be time-consuming and prone to human error.
- These methods may not effectively detect hidden or internal damages, leading to incomplete assessments.
- Limited scalability and the need for specialized expertise further hinder the widespread application of traditional approaches.
- The Subjective nature of human judgment can result in inconsistent evaluations, leading to disputes and discrepancies in damage assessments.
- The Reliance on manual processes can slow down insurance claims and repair procedures, causing delays and inefficiencies in the overall assessment and repair workflow.

2.2 Proposed System

Our proposed system leverages advanced techniques, particularly Convolutional Neural Networks (CNNs), to detect car damage with exceptional efficiency and accuracy. By integrating QR codes, comprehensive information about identified damages is systematically stored, facilitating organized documentation and efficient management in the automotive inspection domain. Multilayer convolutional networks are seamlessly incorporated into existing CNNs, enhancing the model's ability to handle geometric transformations without additional supervision. These networks analyze distinct aspects of input images, optimizing the accurate detection of various damages. Customized and optimized layers leverage features extracted from preceding modules, ensuring robust performance and high accuracy in damage identification. The model is trained using a pre-processed dataset to achieve superior performance. Each identified damage is associated with a unique QR code, streamlining access to pertinent information.

- The utilization of advanced CNN architectures enables the system to achieve high levels of accuracy and precision in identifying different types of car damage, including dents, scratches, and structural issues. This accuracy contributes to improved decision-making in repair and maintenance processes.
- The system’s architecture is designed to scale seamlessly, allowing for the integration of additional features or modules as needed. This flexibility enables the adaptation of the system to evolving requirements and the incorporation of new technologies or detection techniques in the future.
- The system is designed to integrate seamlessly with existing automotive inspection workflows, minimizing disruption and ensuring a smooth transition to the new technology. This integration facilitates adoption and enhances the overall efficiency of inspection operations.

2.2.1 Advantages of Proposed System

- These systems enable swift and accurate identification of various forms of damage, including dents, scratches, and structural issues, facilitating prompt repair and maintenance actions.
- This comprehensive methodology facilitates a streamlined and efficient assessment process, enhancing the overall speed and accuracy of damage cost estimation.

- The use of advanced computer vision techniques minimizes the reliance on manual inspection, reducing the potential for human error and ensuring consistent and objective evaluations.
- This flexibility enables the adaptation of the system to evolving requirements and the incorporation of new technologies or detection techniques in the future.

CHAPTER 3

SYSTEM DESIGN

3.1 System Architecture

System architecture refers to the conceptual design that defines the structure, behavior, and interactions of a system. It encompasses the arrangement of components, modules, subsystems, and their relationships within a system.

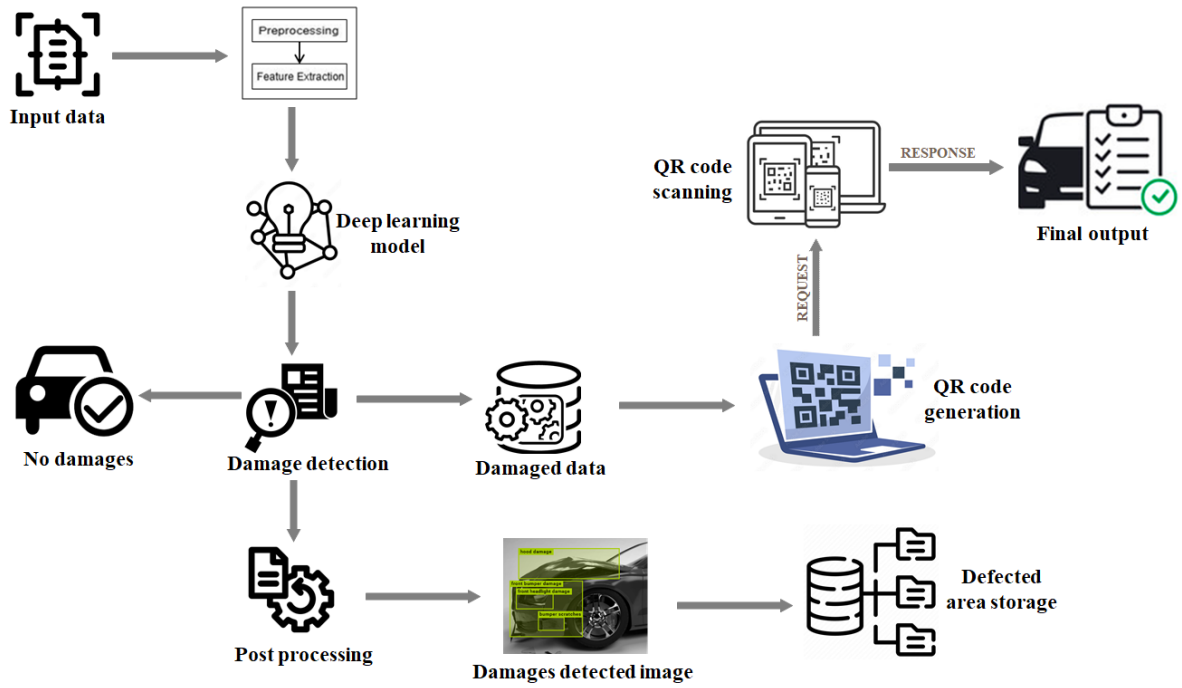

Figure 3.1 System Architecture

The proposed system architecture for Damage Detection involves collecting a labeled dataset, training multiple CNNs, developing a Flask web application for user interaction, deploying the trained models in the web application, and processing and displaying the results to users. The model is trained using a pre-processed dataset, emphasizing achieving superior performance. For each identified damage, a unique QR code is generated, streamlining access to pertinent information. The system supports real-time user interaction, enabling adjustments to settings while providing details on damage severity and classifications for efficient decision-making in automotive inspections.

3.2 UML Diagrams:

A UML diagram is a way to visualize systems and software using Unified Modeling Language (UML). These diagrams are also used to model workflows and business processes.

3.2.1 Use Case Diagram

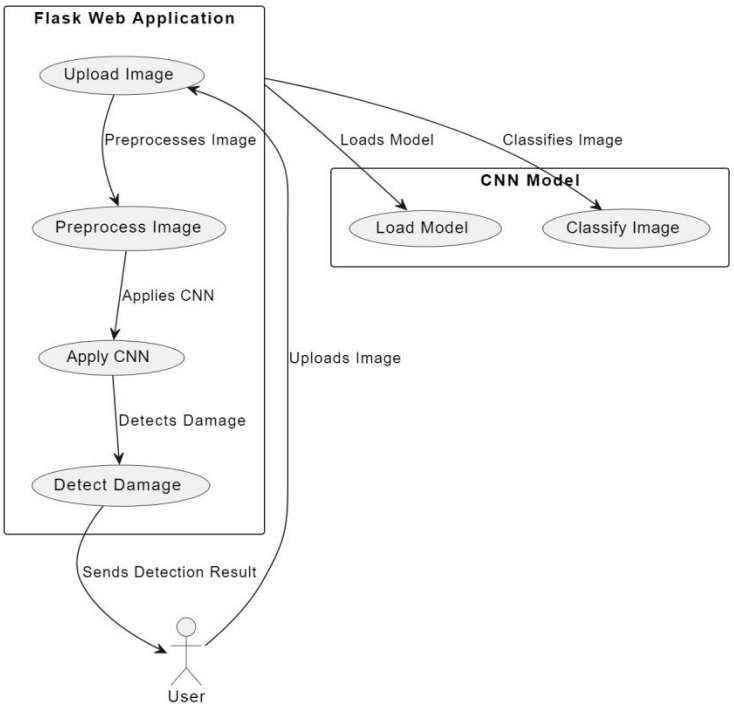

Figure 3.2.1 Use Case Diagram

This diagram represents the user's interaction with the Flask web application and the CNN model for car damage detection. The user uploads an image, which is then preprocessed, and the CNN model is applied to detect the damage. The detection result is sent back to the user. The Flask web application also loads the CNN model and classifies the image.

3.2.2 Class Diagram

In the field of software engineering, a class diagram within the Unified Modeling Language (UML) serves as a static structure diagram which explicates the allocation of information within each class, providing a comprehensive overview of the system's structural elements and their relationships.

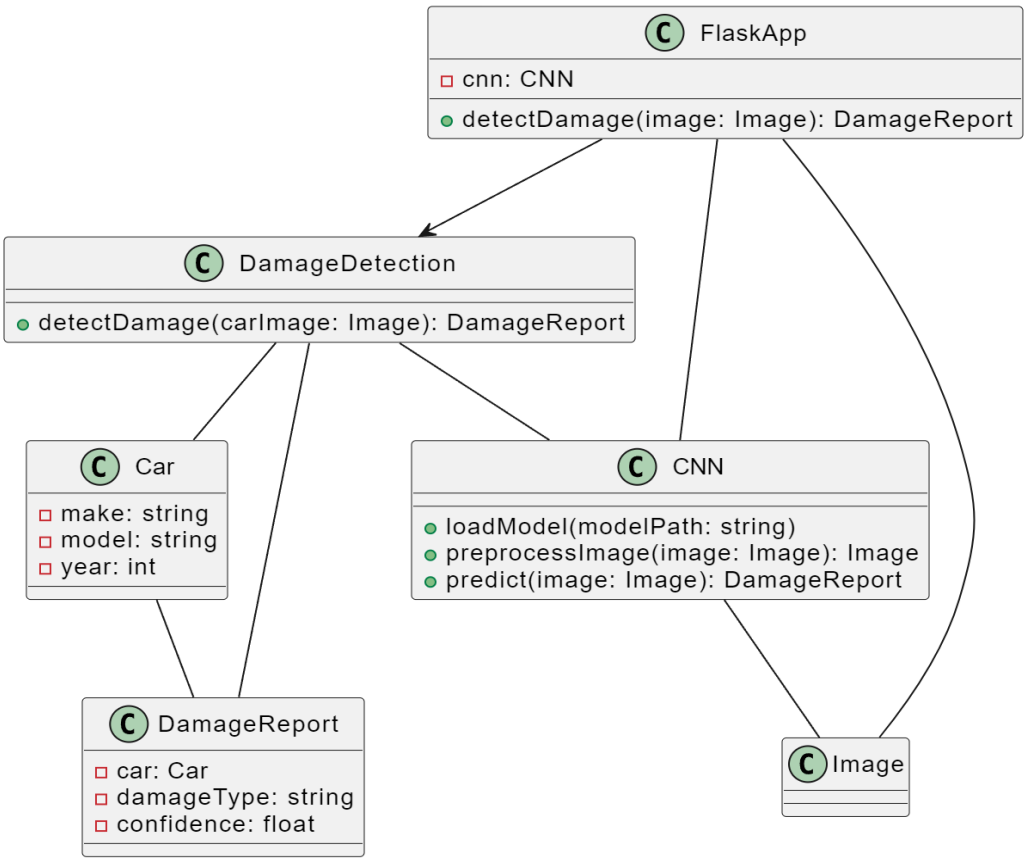

Figure 3.2.2 Class Diagram

This class diagram represents a vehicle collision detection using neural network (CNN) in a Flask web application. Vehicle class represents details of the vehicle such as make, model, and year. The Damage Detection class is responsible for detecting damage in the vehicle image and creating the Damage Report object. The CNN class is used to load pre-trained CNN models, pre-process traffic images, and predict damage and reliability. Flask App class plays the main role and uses CNN and Damage Detection class to detect damage in the traffic image.

3.2.3 Sequence Diagram

A Sequence diagram shows how the objects interact with others in a particular scenario of a use-case. This diagram describes the flow of events of a particular system.

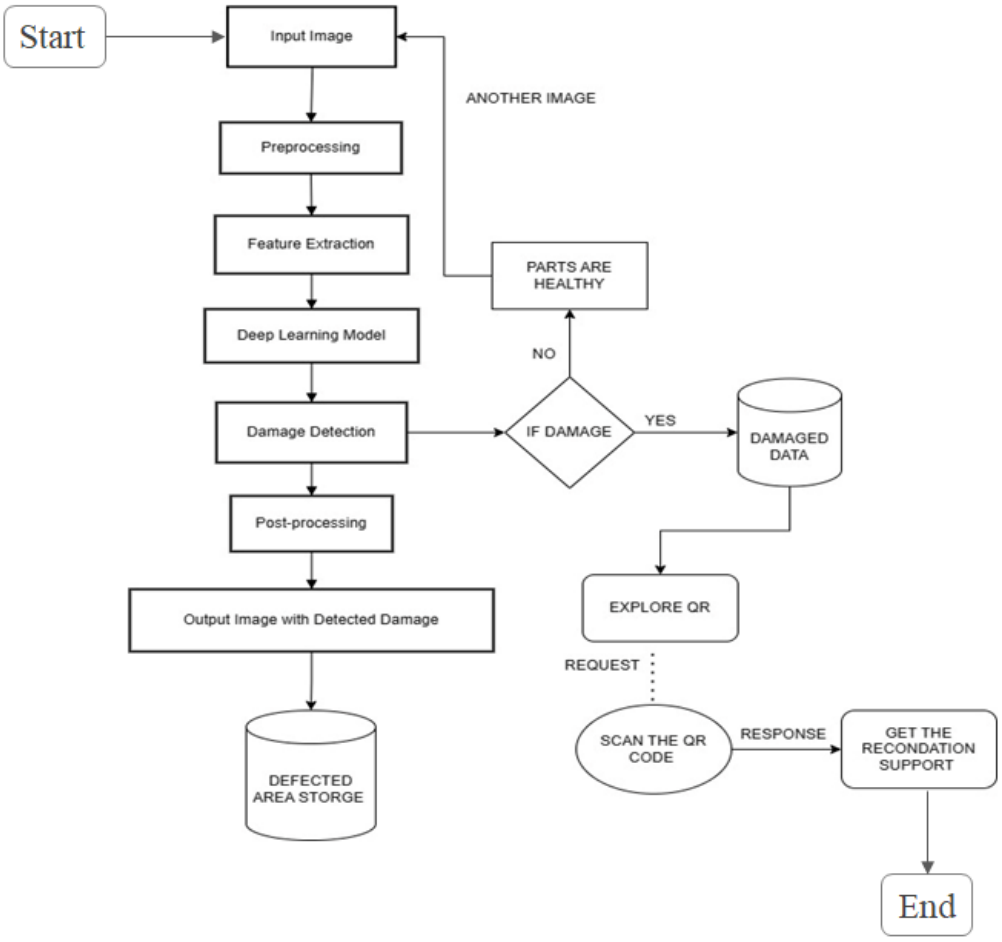

Figure 3.2.3 Sequence Diagram

This Diagram shows the flow of information and interaction between different components in the investigation of vehicle damage. The user uploads the image which processed for further feature extraction. The Damage Detection component uses CNN to predict damage types by loading pre-trained models and preliminary images and predicting damage types. It also retrieves vehicle details from vehicle to vehicle. For each detected damage, a unique QR code is generated which encoding those informations. Once damages are detected, the system assigns each damage a unique QR code that encapsulates relevant details, such as severity levels and classifications.

3.2.4 Activity Diagram

Activity diagrams are visual representations depicting stepwise workflows, actions, and activities within a system, incorporating features such as decision points, iterative processes, and concurrent actions. These diagrams provide a clear depiction of the overarching control flow, making them effective tools for understanding and communicating the dynamic aspects of a system's functionality.

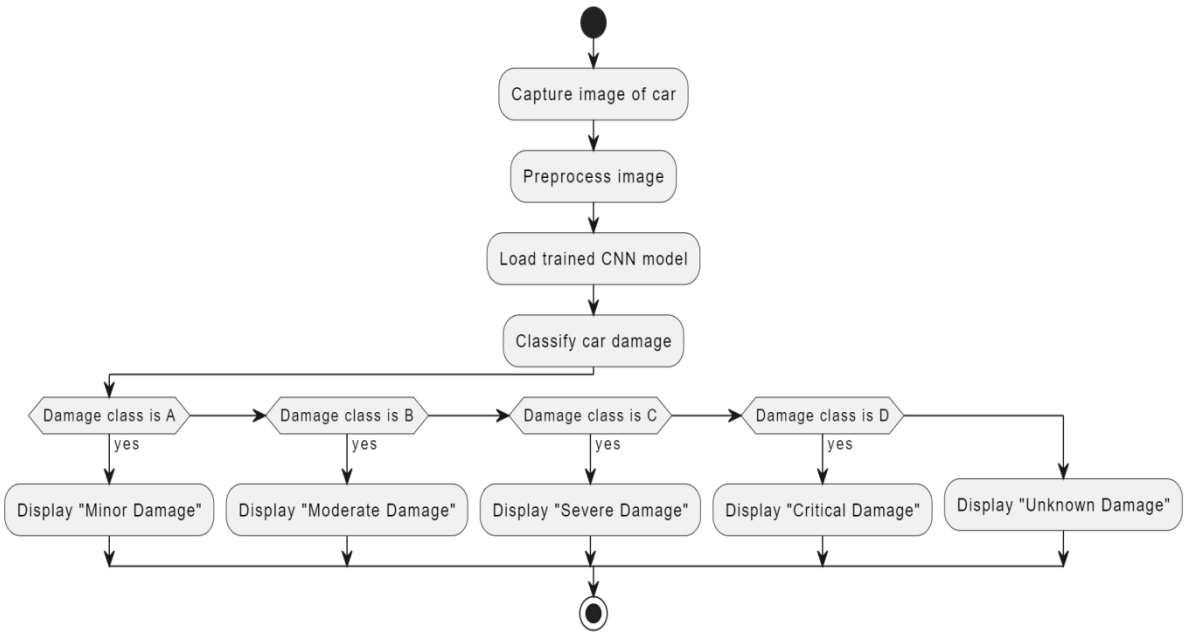

Figure 3.2.4 Activity Diagram

This activity diagram represents the process of car damage detection using a Convolutional Neural Network (CNN) with Flask web. The image of the car is captured and preprocessed. Then, a trained CNN model is loaded to classify the car damage into one of the five classes: A, B, C, D, or Unknown. Depending on the damage class, the corresponding message is displayed.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 Modules

Car damage detection system using Convolutional Neural Networks (CNN) with a Flask web app, you will need to set up different modules to handle various tasks. Here are the main modules you might find in such a system:

- **Dataset Preprocessing Module:** This module is responsible for preprocessing the dataset and preparing it for training the CNN model. It may include tasks such as data loading, data augmentation, and splitting the dataset into training and testing sets.
- **Feature Extraction:** Extract features from the preprocessed images using techniques such as edge detection, color histograms, and texture analysis Prepare the feature vector representation for training the CNN model.
- **CNN Model Training Module:** This module is used to define the architecture of the CNN model and train it on the preprocessed dataset. It involves defining the layers of the CNN, compiling the model, and fitting it to the training data.
- **Damage Detection:** Create a component that uses a trained CNN to identify and localize car damage, distinguishing between scratch, dent, and structural issues. Integrate algorithms for precise classification and highlighting, providing an intuitive visual representation for users to assess damage severity and extent.
- **Dataset Post-processing Module:** Post-processing in car damage detection involves additional steps or techniques applied after the initial detection model identifies potential damage in images. While the detection model locates and identifies vehicle damages, post-processing aims to refine and validate the results, enhancing accuracy, reliability, and interpretability for practical applications.
- **QR Code Generation and Scanning:** The QR Code module generates dynamic QR codes encoding details of detected damages during inspection. Scanning these codes provides quick access to detailed reports, offering instant insights into the nature and extent of each identified issue.
- **User Interface:** Design a user-friendly interface that allowing users to easily upload images for car damage detection. Display the results of the analysis in a clear and visually appealing manner, providing users with comprehensive information about the detected damages and their severity.

4.2 Module Description

4.2.1 Dataset Preprocessing Module:

This module focuses on preparing the car damage dataset for training the Convolutional Neural Network (CNN) model. It handles tasks such as data loading, resizing, normalization, and the division of data into training and testing sets. It also incorporates data augmentation techniques to enhance the diversity of the dataset, ensuring robust training of the CNN model.

4.2.2 CNN Model Training Module:

In this module, the architecture of the CNN model is defined, incorporating the extracted features from the previous module. The layers are customized and optimized to facilitate accurate detection of various types of car damage. The module also entails the training of the CNN model using the preprocessed dataset and the extracted features, with a focus on achieving high accuracy and robust performance in car damage identification.

4.2.3 Damage Detection:

Implement a component that identifies and localizes areas of damage within car images. Train the CNN model to differentiate between different types and extents of car damage, such as scratches, dents, and major structural damage. Integrate algorithms that can accurately classify and highlight the regions of the car that have been damaged. Provide an intuitive and informative visual representation of the detected damage areas for users to comprehend the severity and extent of the damage.

4.2.4 Dataset Post-processing Module:

Post-processing in car damage detection refers to the additional steps or techniques applied after the initial detection model has identified potential damage in an image or a set of images. While the primary objective of the detection model is to locate and identify damages in a vehicle, post-processing steps aim to refine, validate, or further analyze the detected results. This aims to improve the accuracy, reliability, and interpretability of the initial detection model's output, enhancing the utility of the system in practical applications.

4.2.5 QR Code Generation and Scanning:

In the QR Code Generation and Scanning module, dynamic QR codes are generated to encode specific information related to each detected damage during the inspection process.

This dynamic encoding ensures that comprehensive details about the damages are embedded within the QR codes. Scanning these QR codes facilitates quick and convenient access to detailed damage reports, providing stakeholders with instant insights into the nature and extent of each identified issue.

4.2.6 User Interface Module:

The User Interface (UI) module in car damage detection plays a crucial role in presenting visual outputs like annotated images and severity classifications. It facilitates real-time user interaction and adjustment of settings while providing information about damage severity and classifications. The UI module also integrates with external systems, enhancing the user experience for stakeholders like insurance assessors and auto repair professionals. Additionally, the inclusion of a QR code output further streamlines information access and retrieval, contributing to the overall efficiency and convenience of the system.

4.3. ALGORITHM

4.3.1 CONVOLUTIONAL NEURAL NETWORKS (CNNs):

The automotive industry faces the challenge of detecting and managing vehicle damage which can be solved by using Multiple Convolutional Neural Networks (CNNs) to create integration within the Flask web application framework. It is a deep learning method which uses convolutional and pooling layers to extract features from images or videos and then uses these features to classify or capture objects or scenes.

4.3.1.1 Convolutional layer

A traditional neural network consists of an input layer, a hidden layer, and an output layer. There are many neurons in each layer. Additionally, the neurons between two adjacent layers are fully connected. Therefore, it produces a large number of parameters. Excessive parameters will cause gradient dispersion. Convolution can reduce parameters in the neural network and improve learning efficiency. Therefore, the convolutional layer is the most important component of the convolutional neural network. Two-dimensional convolution is often used in image processing. The layers of a convolutional neural network (CNN) are convolution, pooling, flattening, and fully connected. The input picture and the filter, or kernel, are the two important components to take attention of before starting the Forward Pass from the Convolutional Layer.

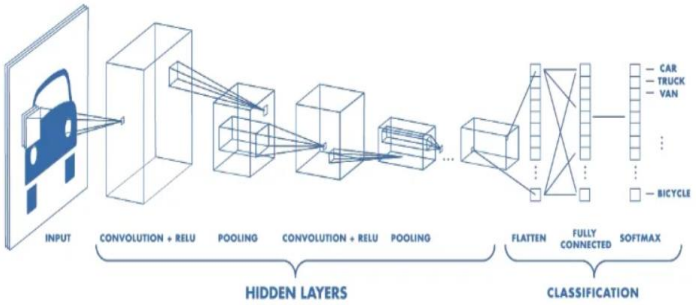

Figure 4.1 CNN architecture

A feature detector, which is a collection of weights that contains some of the most significant characteristics in the input picture, is another name for a kernel or filter. This layer undergoes a convolution operation using the kernels and the input picture, yielding an output that is also referred to as a feature map. The data is inputted into the convolutional layer, and the convolutional kernel in the convolutional layer and the input data are convoluted. The

input shape of the first convolutional layer is 50 × 12, the kernel size is 5 × 12, the kernel number is 32. Each convolution operation only processes a part of the input data. The convolutional kernel starts at the upper left corner of the input data, and the parameters of the convolutional kernel are multiplied by the parameters of the coverage area. Then, the multiplied values are summed.

4.3.1.2 Activation Function

In order to simulate the human nervous system, a neural network is proposed. The human nervous system responds only to some neurons. The activation function is also intended to activate some neurons. The activation of neurons is determined by the activation function. The activation function is divided into linear activation function and nonlinear activation function. Linear activation functions can only provide linear mapping capabilities and do not improve the expressive power of the network. Therefore, the nonlinear activation function has been widely used. Common nonlinear activation functions are sigmoid, tanh, and rectified linear unit (ReLU).

4.3.1.3 Pooling Layer

The role of the pooling layer is to reduce the size of the feature map. The pooling layer is similar to the convolutional layer, it also has a kernel. However, the pooling kernel does not contain any parameters. The pooling layer is divided into maximum pooling layer and mean pooling layer. The pooling operation starts at the upper left corner of the input data. For the maximum pooling, the maximum value of all the parameters in the coverage area is outputted. For the mean pooling, the mean value of all the parameters in the coverage area is outputted. The movement of the pooling kernel is the same as the movement of the convolutional kernel. The pooling layer is also called the down sampling layer, which can reduce the parameters in the feature map and improve the training efficiency of the model.

4.3.1.4 Fully Connected Layer

The fully connected layer can be seen as a “classifier” in the convolutional neural network. The convolutional layer maps the raw data to feature space of the hidden layer, and the fully connected layer maps the learned distributed feature to the sample tag space. Each neuron in the fully connected layer is connected to all neurons in the anterior layer. The output signal of the fully connected layer is passed to the objective function (loss function)

for classification. A dense layer is a classic fully connected neural network layer. The output shape of last fully connected layer is 4 in this CNN. This value is consistent with the number of categories.

4.3.2 Multiple CNN

Multiple CNN or Convolutional Neural Networks refers to the use of more than one neural network architecture in a system or application. Instead of relying on a single CNN, multiple networks can be used to solve different types of complex problems. The system offers solutions that improve the accuracy and integration of information and enable better control and decision-making during vehicle inspections. However, the entire concept is represented by the following function:

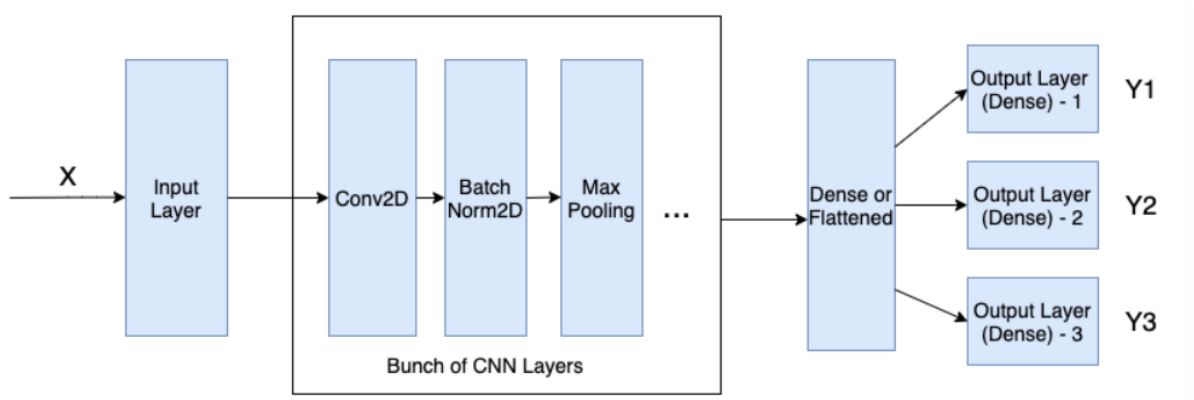

Figure 4.2 Multi Output CNN

As you can see in above diagram, CNN takes a single input `X` (Generally with shape (m, channels, height, width) where m is batch size) and spits out three outputs (here Y1, Y2, Y3 generally with shape (m, n classes) again m is batch size).

Output = CNN_1(Input)+ CNN_2(Input)+…+ CNN_n(Input)

Where CNN_1, CNN_2,..., CNN_n are separate CNN architectures and concept. It is input data. Combine the results of each CNN to get a better result, and use the power of each network for more complex and robust solutions.

4.3.3 YOLO (You Only Look Once)

YOLO (You Only Look Once) is a real-time object detection algorithm. It is a single-stage object detector that uses a convolutional neural network (CNN) to predict the bounding boxes and class probabilities of objects in input images. The YOLO algorithm divides the input image into a grid of cells, and for each cell, it predicts the probability of the presence of

an object and the bounding box coordinates of the object. The basic idea behind YOLO is to divide the input image into a grid of cells and, for each cell, predict the probability of the presence of an object and the bounding box coordinates of the object.

The process of YOLO can be broken down into several steps:

1. The input image is passed through a CNN to extract features from the image.
2. The features are then passed through a series of fully connected layers, which predict class probabilities and bounding box coordinates.
3. The image is divided into a grid of cells, and each cell is responsible for predicting a set of bounding boxes and class probabilities.
4. The output of the network is a set of bounding boxes and class probabilities for each cell.
5. The bounding boxes are then filtered using a post-processing algorithm called non-max suppression to remove overlapping boxes and choose the box with the highest probability.
6. The final output is a set of predicted bounding boxes and class labels for each object in the image.

One of the key advantages of YOLO is that it processes the entire image in one pass, making it faster and more efficient than two-stage object detectors such as CNN and its variants.

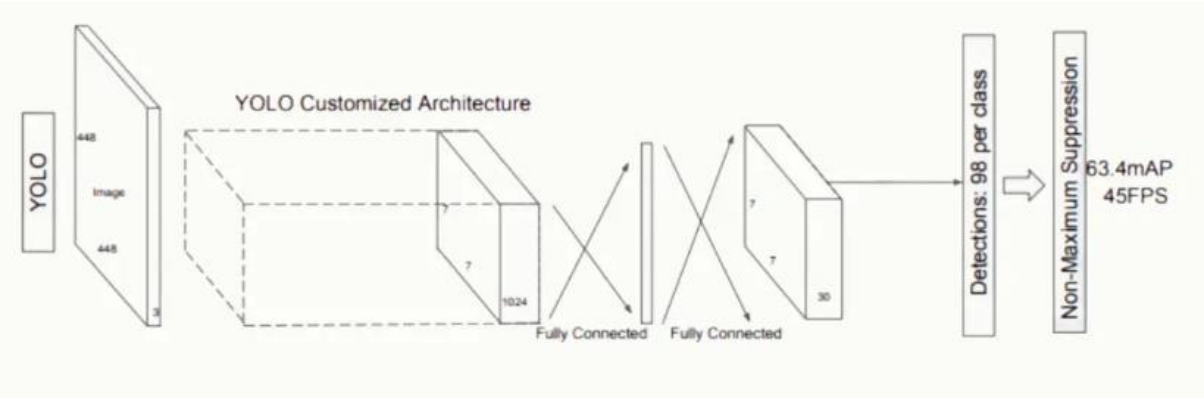

Figure 4.3 Structure of YOLO

The Efficient Det architecture is designed to be efficient in terms of computation and memory usage while also achieving high accuracy. Another important difference is the use of anchor-free detection, which eliminates the need for anchor boxes used in previous versions of YOLO. Instead of anchor boxes, YOLO v5 uses a single convolutional layer to predict the bounding box coordinates directly, which allows the model to be more flexible and adaptable to different object shapes and sizes.

4.4 TESTING

The primary aim of testing is error discovery. Testing is the systematic exploration of a work product to identify any conceivable faults or weaknesses. It encompasses the examination of components, sub-assemblies, assemblies, or a final product to verify their functionality. The process involves actively using the software to ensure that the software system aligns with its specified requirements, meets user expectations, and avoids unacceptable failures. Testing is a multifaceted activity, and various test types cater to distinct testing requirements, collectively contributing to the overall quality assurance of a product.

4.4.1 TESTING METHODS

S. No	Name of the test	Things to be tested
4.4.1.1	API testing	User's comfort with website should be tested.
4.4.1.2	Unit testing	Every page of the website should be tested.
4.4.1.3	Integration testing	Connection between the webpages is tested.
4.4.1.4	System testing	The functional flow of the webpages is tested.

4.4.1.1 API Testing

API testing is a software testing methodology that focuses on assessing the functionality, reliability, performance, and security of Application Programming Interfaces (APIs). APIs act as intermediaries facilitating communication and data exchange between different software components. In API testing, various scenarios are examined, including endpoint functionality, input validation, authentication, data consistency, error handling, performance under diverse conditions, security vulnerabilities, and overall integration with other system components.

The objective is to ensure that APIs operate as intended, deliver accurate responses, handle different inputs gracefully, and seamlessly integrate with other software elements, fostering the reliability and efficiency of the entire system. API testing can span multiple levels, including unit and integration testing, depending on the nature of the APIs being tested.

4.4.1.2 Unit Testing

Unit testing involves designing test cases to validate the internal program logic, ensuring that the individual software units function correctly. It checks all decision branches and internal code flow, and it's performed after completing each unit before integration. This structural testing relies on knowledge of construction, is invasive, and focuses on specific business processes, applications, or system configurations. Unit tests verify that each path of a business process adheres to documented specifications, with well-defined inputs and expected results.

1. Test strategy and approach:
    - Field testing will be performed manually and functional tests will be written in detail.
2. Test objectives:
    - All field entries must work properly.
    - Pages must be activated from the identified link.
    - The entry screen, messages and responses must not be delayed.

Features to be tested:

- Verify that the entries are of the correct format.
- No duplicate entries should be allowed.
- All links should take the user to the correct page.

4.4.1.3 Integration Testing

Integration tests are designed to verify if integrated software components function cohesively as a single program. The testing process is event-driven, focusing on the fundamental outcomes of screens or fields. Unlike unit testing, integration tests assess the combined performance of components, revealing any issues that may arise from their integration. It ensures that, although individual components pass unit tests successfully, their combination is correct and consistent.

- In software integration testing, multiple integrated software components are incrementally tested on a single platform to identify failures caused by interface defects.
- The primary goal of integration testing is to confirm that components or software applications interact seamlessly without errors, whether it involves components within a software system or software applications at a higher company level.

Test Results:

- All the test cases mentioned above passed successfully.
- No defects encountered.

4.4.1.4 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **Test Set Preparation:** Separate a portion of the dataset that was not used during training or validation for testing purposes. This ensures that the model's performance is evaluated on unseen data, providing a better indication of its generalization capabilities.
- **Input Data Preprocessing:** Preprocess the test images in the same way as the training images, including resizing, normalization, and any other necessary transformations.
- **Model Evaluation:** Use the preprocessed test images as input to the trained CNN model. The model generates predictions for each image, indicating whether it detects car damage or not.
- **Performance Metrics Calculation:** Calculate various performance metrics to assess the model's effectiveness, such as accuracy, precision, recall, and F1-score. These metrics provide insights into the model's ability to correctly classify damaged and undamaged cars.
- **Visual Inspection:** In addition to quantitative metrics, visually inspect the model's predictions on a subset of test images. This qualitative analysis helps identify any common patterns or types of errors made by the model.
- **Error Analysis:** Analyze misclassified images to understand the reasons behind incorrect predictions. This analysis can help identify areas for model improvement, such as the need for more diverse training data or adjustments to the model architecture.
- **Cross-validation:** In some cases, especially with limited data, cross-validation

techniques such as k-fold cross-validation can be employed to assess the model's stability and robustness across different subsets of the data.

Test Results:

- All the test cases mentioned above passed successfully.
- No defects encountered.

CHAPTER 5

RESULTS & DISCUSSION

In this study, we employ a sophisticated methodology harnessing the capabilities of multiple neural networks, specifically Convolutional Neural Networks (CNNs), to enhance the precision and resilience of car damage detection in images, ultimately contributing to the safety and efficiency of the automotive industry. The integrated system for car damage detection, employing multiple Convolutional Neural Networks (CNNs) within the Flask web application framework, has yielded promising results across various modules.

In the multiple damage detection, the stiffness of the damaged column was reduced by 20%. The multiple damage dataset includes five categories: “c0”, “c1c6”, “c1c11”, “c6c11”, and “c1c6c11”. “c0” means that there are no damaged columns; “c1c6” means that the damaged columns are column 1 and column 6; “c1c11” means that the damaged columns are column 1 and column 11; “c6c11” means that the damaged columns are column 6 and column 11; “c1c6c11” means that the damaged columns are column 1, column 6, and column 11. The data in the dataset were obtained from the finite element model. The parameters in the finite element model were set according to the damage types. Each sample in the dataset is a matrix where the m represents the nodes and n represents the data points. For example, the number of samples is 399 in each category, and the dataset contains a total of 1995 samples. After these samples are randomly sorted, 1500 samples were used for training and verification, and 495 samples were used as test sets.

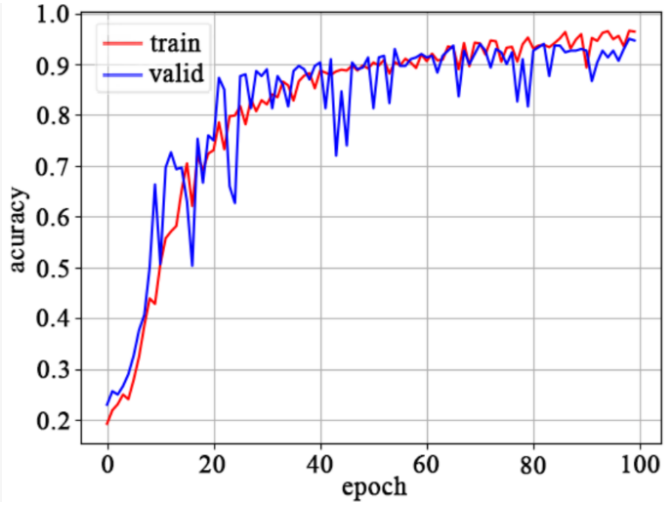

Figure 5.1 Training accuracy and valid accuracy

The training set and the valid set are inputted into the proposed convolutional neural network. Training accuracy and valid accuracy are shown in Figure 5.1. When the number of iteration epoch is 100, the training accuracy is 0.9900 and the valid accuracy is 0.9833. The detection accuracy has met the requirements. A detection model can be obtained by training the dataset. The test set is used to test the detection performance of the model.

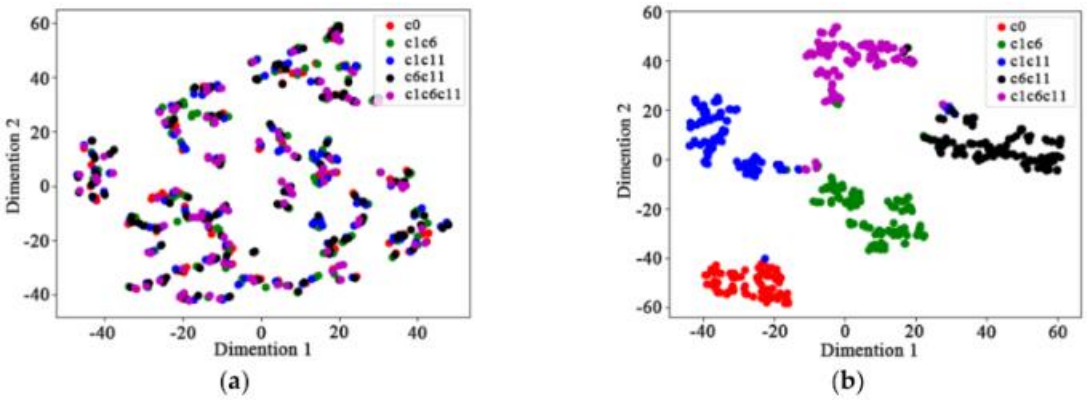

Figure 5.2 The Two-Dimensional display of data. (a) Raw Data (b) Output Data of CNN

The raw data and the output data of CNN are displayed in two dimensions, as shown in Figure 5.2. The two-dimensional display of raw data is also terrible. There is no distinction between the categories. However, the two-dimensional display of the output data of CNN is perfect. There is a clear distinction between the categories. It is not as effective as the two-dimensional display of single damage location detection; a small number of samples were misclassified. There are overlaps between different categories in the multiple damage detection. For instance, “c1c6c11” means that the damaged columns are column 1, column 6, and column 11. Other categories overlap with “c1c6c11” partially. So, it is more difficult to distinguish. Nevertheless, the recognition accuracy of “c1c6c11” is 92.9%. The results showed that the proposed convolutional neural network has a powerful ability to detect multiple damages. They successfully identifies and categorizes damages in input images with a high degree of accuracy, providing valuable insights for stakeholders like car manufacturers, insurers, rental companies, and individual owners.

CHAPTER 6
CONCLUSION

The Detection of defective parts using deep learning, coupled with QR code integration, presents a robust and innovative solution that revolutionizes the car inspection process. This system leverages cutting-edge technologies to significantly improve efficiency, ensuring swift and accurate identification of damages. The integration of QR codes adds a layer of streamlined documentation, facilitating comprehensive and organized management of car damages. This rich metadata not only aids in tracking the history of damages but also enables efficient communication between stakeholders involved in repair and maintenance activities. As a result, the system not only simplifies the inspection process but also fosters improved collaboration and decision-making among automotive professionals, ultimately enhancing the overall user experience and satisfaction.

“The art of repair is the art of honoring what once was”

REFERENCES

[1] DimitriosMallios, Li Xiaofei, Niall McLaughlin, Jesus Martinez Del Rincon, Clare Galbraith, Rory Garland, “Vehicle Damage Severity Estimation for Insurance Operations Using In-The-Wild Mobile Images”, IEEE Access (Volume: 11), 2023.

[2] Q. Zhang, X. Chang, and S. B. Bian, “Vehicle-damage-detection segmentation algorithm based on improved mask RCNN,” IEEE Access, vol. 8, pp. 6997–7004, 2020.

[3] Xinkuang Wang, Wenjing Li, Zhongcheng Wu, “CarDD: A New Dataset for Vision-based Car Damage Detection”, IEEE Transactions on Intelligent Transportation Systems (Volume: 24, Issue: 7, July 2023)

[4] Phyu Mar Kyu, KuntpongWoraratpanya, “Car Damage Assessment Based on VGG Models”, IEEE Access, 2020.

[5] NajmeddineDhieb, 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.

[6] AtharvaShirode, TejasRathod, ParthWanjari, AparnaHalbe, “Car Damage Detection and Assessment Using CNN”, IEEE Delhi Section Conference (DELCON), 2022

[7] 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.

[8] Linjie Yang et al., “A large-scale car dataset for fine-grained categorization and verification”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[9] K. Patil, M. Kulkarni, A. Sriraman, and S. Karande, “Deep learning based car damage classification”, in Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA), Dec. 2017, pp.

50–54.

[10] S. Ren, K. He, R. Girshick, and J. Sun. “Faster R-CNN: Towards real-time object detection with region proposal networks”, (IEEE Volume: 39 Issue: 6), In NIPS, 2015.

[11] UmerWaqas, NimraAkram, Soohwa Kim, Donghun Lee, JihoonJeon, “Vehicle Damage Classification and Fraudulent Image Detection Including Moiré Effect Using Deep Learning”, IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), 2020

[12] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In CVPR (IEEE) , 2016.

[13] Krizhevsky, I. Sutskever, and G. Hinton. “ImageNet classification with deep convolutional neural networks”, In NIPS, 2012.

[14] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection”, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 320–329

[15] G. Wang and S. Liang, “Ship object detection based on mask RCNN”, in Proc. Radio Eng., IEEE (IGARSS), 2018.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: Alarge-scale hierarchical image database”, in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2009, pp. 248–255

[17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection”, 2020, arXiv:2004.10934.

[18] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, “Deformable convolutional networks”, in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 764–773.

[19] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks”, in Proc. 36th Int. Conf. Mach. Learn., 2019, pp. 6105–6114.

[20] Tsung-Yi Lin, Piotr Doll'ar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature pyramid networks for object detection". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), pages 2117–2125, 2017.

[21] Andre Luckow et al., "Deep learning in the automotive industry: Applications and tools", 2016 IEEE International Conference on Big Data (Big Data), 2016.

[22] Baozhen Yao and Tao Feng, "Machine learning in automotive industry", 2018.

[23] Hiroya Maeda et al., "Road damage detection using deep neural networks with images captured through a smartphone", 2018.

[24] Christian Szegedy et al., "Rethinking the inception architecture for computer vision", Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[25] Kaiming He et al., "Deep residual learning for image recognition", Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.

[26] Kalpesh Patil et al., "Deep learning based car damage classification", 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017.

[27] Srimal Jayawardena, "Image based automatic vehicle damage detection", 2013.

[28] M. Oquab, L. Bottou, I. Laptev and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks", Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1717-1724, 2014.

[29] J. Krause, M. Stark, J. Deng and L. Fei-Fei, "3d object representations for fine-grained categorization", Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 554-561, 2013.

APPENDIX I - SOURCE CODE

```
import argparse
import io
import os
from PIL import Image
import torch
from flask import Flask, render_template, request, redirect
from keras_preprocessing import image
from keras.applications.imagenet_utils import preprocess_input
import numpy as np
from tensorflow.keras.models import load_model
from flask import Flask, render_template, request, redirect, flash, url_for
import urllib.request
from werkzeug.utils import secure_filename



import cv2
#from pyzbar.pyzbar import decode

app = Flask(_name_)

@app.route("/", methods=["GET", "POST"])
def predict():
    if request.method == "POST":
        if "file" not in request.files:
            return redirect(request.url)
        file = request.files["file"]
        if not file:
            return
img_bytes = file.read()
img = Image.open(io.BytesIO(img_bytes))
        print(img)
print()
```

```
        results = model(img)  # inference
        print(results)
results.render()  # updates results.ims with boxes and labels
Image.fromarray(results.ims[0]).save("static/images/image0.jpg")
        import random

        # Generate a random integer between a specified range
random_int = random.randint(1, 1000)
print("Random Integer:", random_int)

Image.fromarray(results.ims[0]).save("database/image0"+ str(random_int) +".jpg")
        filename=file.filename
        img2 = image.load_img(filename, target_size=(224, 224))
        x = image.img_to_array(img2)
        x = np.expand_dims(x, axis=0)
        x = preprocess_input(x)
        model2 = load_model('vgg_model.h5')
        preds = model2.predict(x)
        result = preds[0][0]
        image1 = Image.open('static/images/image0.jpg')
        image2 = Image.open('static/images/screenshot.jpg')
        image1_size = image1.size
        image2_size = image2.size
new_image = Image.new('RGB',(3*image2_size[0],image2_size[1]), (250,250,250))
new_image.paste(image1,(0,0))
new_image.paste(image2,(image1_size[0],0))
new_image.save("static/images/merged_image.jpg","JPEG")
        if result <preds[0][1]:
            print("messy")
        else:
            print("clean")

        return redirect("static/images/merged_image.jpg")
```

```
    return render_template("index.html")

@app.route('/bookappt')
def bookappt():
    return render_template('bookappt.html')

@app.route('/regform')
def regform():
    return render_template('regform.html')



@app.route('/qr')
# Function to decode QR codes in a video stream
def read_qr_code():
    # Open the video capture device (webcam)
    cap = cv2.VideoCapture(0)

    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()

        # Decode QR codes
decoded_objects = decode(frame)

        # Print decoded information
        for obj in decoded_objects:
print('Type:', obj.type)
print('Data:', obj.data)
print()
            # Draw bounding box around the QR code
            (x, y, w, h) = obj.rect
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

```
        # Display the frame with bounding boxes
        cv2.imshow('QR Code Scanner', frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release the capture
cap.release()
    cv2.destroyAllWindows()

    return "QR PROCESS IS GOING ON"
if _name_ == "_main_":

    parser = argparse.ArgumentParser(description="Flask app exposing yolov5 models")
parser.add_argument("--port", default=5000, type=int, help="port number")
args = parser.parse_args()

    model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)  # force_reload =
recache latest code
model.eval()

app.run(host="0.0.0.0", port=args.port)  # debug=True causes Restarting with stat


<!DOCTYPE html>

<html lang="en" >

<head>

<meta charset="UTF-8">

<title>Avail Service</title>
```

<link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/css/bootstrap.min.css'>

<link rel='stylesheet' href='https://fonts.googleapis.com/css?family=Muli'>

<link rel="stylesheet" href="/static/style1.css">

</head>

<body>

<!-- partial:index.partial.html -->

<h1 class="text-center">Need service? Book Appointment right now!</h1>

<form id="submitForm" action="/login" method="post" data-parsley-validate="" data-parsley-errors-messages-disabled="true" novalidate="" _lpchecked="1"><input type="hidden" name="_csrf" value="7635eb83-1f95-4b32-8788-abec2724a9a4">

<lSabel for="username">Email</lSabel>

<input type="text" class="form-control text-lowercase" id="email" required="" name="email" value="">

<lSabel for="username">Mobile no</lSabel>

<input type="number" class="form-control text-lowercase" id="password" required="" name="password" value="">

<lSabel for="username">Location</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required="" name="username" value="">

<lSabel for="username">Time to Allocate</lSabel>

<input type="range" min="1" max="24" value="4" oninput="rangeValue.innerText = this.value">

4 hrs

<lSabel for="username">Additional Description</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required="" name="username" value="">

<button class="btnbtn-primary btn-block" type="submit">Avail Service</button>

```
</form>
<p class="small-xl pt-3 text-center">
<span class="text-muted">Not a member?</span>
<a href="regform">Create Account</a>
</p>
</div>
</div>
</div>
</div>
</div>

<!-- partial -->

</body>
</html>

<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css" integrity="sha512-
```

```
iBBXm8fW90+nuLcSKlbmrPcLa0OT92xO1BIsZ+ywDWZCvqsWgccV3gFoRBv0z+8dLJg
yAHIhR35VZc2oM/gI1w==" crossorigin="anonymous" referrerpolicy="no-referrer" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<style>
    .bd-placeholder-img {
      font-size: 1.125rem;
      text-anchor: middle;
    }
    @media (min-width: 768px) {
      .bd-placeholder-img-lg {
        font-size: 3.5rem;
      }
    }
</style>
<link rel="stylesheet" href="/static/style.css">
<title>Car Damage</title>
</head>
<body>
<div>
<nav>
<input type="checkbox" id="check">
<label for="check" class="checkbtn">
<i class="fas fa-bars" id="bars"></i>
<i class="fas fa-times" id="cancel"></i>
```

```
</label>

<imgsrc="/static/logo.png" width="80">

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#services">Services</a></li>

<li><a href="#test">Demo</a></li>

</ul>

</nav>

<lSabel for="username">Mobile no</lSabel>

<input type="number" class="form-control text-lowercase" id="password" required="" name="password" value="">

</div>

<div class="form-group required">

<lSabel for="username">Location</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required="" name="username" value="">

</div>

<div class="form-group required">

<lSabel for="username">Time to Allocate</lSabel>

<div class="slider">

<input type="range" min="1" max="24" value="4" oninput="rangeValue.innerText = this.value">

<p id="rangeValue">4</p><p>hrs</p>

</div>

</div>
```

<lSabel for="username">Additional Description</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required="" name="username" value="">

<button class="btnbtn-primary btn-block" type="submit">Avail Service</button>

<section id="home" class="home" style="background: url('/static/bg.jpg'); background-position: center; background-size: cover; background-repeat: no-repeat;">

<h1>Car Damage Detection System Using Flask</h1>

<button>CLICK ME</button>

</section>

<section class="page2" id="services">

≪

≫

```
</section>
<section class="test" id="test">
<div class="heading2">
<hr>
<h1><b>Demo</b></h1>
</div>
<div class="yolo">
<form class="form-signin" method=post enctype=multipart/form-data>
<h1 class="h3 mb-5 font-weight-normal">Upload car image</h1>
<input type="file" name="file" class="form-control-file" id="inputfile">
<br/>
<button class="btnbtn-lg btn-primary btn-block" type="submit">Upload</button>
</form>
</div>
<div>



</div>
</section>
<script src="/static/script.js"></script>


@app.route('/bookappt')
def bookappt():
    return render_template('bookappt.html')
```

```
@app.route('/regform')
def regform():
    return render_template('regform.html')



@app.route('/qr')
# Function to decode QR codes in a video stream
def read_qr_code():
    # Open the video capture device (webcam)
    cap = cv2.VideoCapture(0)

    while True:
        # Capture frame-by-frame
        ret, frame = cap.read()

        # Decode QR codes
decoded_objects = decode(frame)

        # Print decoded information
        for obj in decoded_objects:
print('Type:', obj.type)
print('Data:', obj.data)
print()
            # Draw bounding box around the QR code
            (x, y, w, h) = obj.rect
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        # Display the frame with bounding boxes
        cv2.imshow('QR Code Scanner', frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
```

```
    # Release the capture
cap.release()
    cv2.destroyAllWindows()

    return "QR PROCESS IS GOING ON"
if _name_ == "_main_":

    parser = argparse.ArgumentParser(description="Flask app exposing yolov5 models")
parser.add_argument("--port", default=5000, type=int, help="port number")
args = parser.parse_args()

    model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)  # force_reload =
recache latest code
model.eval()

app.run(host="0.0.0.0", port=args.port)  # debug=True causes Restarting with stat


<lSabel for="username">Mobile no</lSabel>

<input type="number" class="form-control text-lowercase" id="password" required=""
name="password" value="">

</div>

<div class="form-group required">

<lSabel for="username">Location</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required=""
name="username" value="">

</div>

<div class="form-group required">

<lSabel for="username">Time to Allocate</lSabel>

<div class="slider">
```

<input type="range" min="1" max="24" value="4" oninput="rangeValue.innerText = this.value">

4 hrs

<lSabel for="username">Additional Description</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required="" name="username" value="">

<button class="btnbtn-primary btn-block" type="submit">Avail Service</button>

<script src="//code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.6/umd/popper.min.js" integrity="sha384-wHAiFfRlMFy6i5SRaxvfOCifBUQy1xHdJ/yoi7FRNXMRBu5WHdZYu1hA6ZOblgut" crossorigin="anonymous"></script>

<script src="//stackpath.bootstrapcdn.com/bootstrap/4.2.1/js/bootstrap.min.js" integrity="sha384-B0UglyR+jN6CkvvICOB2joaf5I4l3gm9GU6Hc1og6Ls7i6U/mkkaduKaBhlAXv9k" crossorigin="anonymous"></script>

<script type="text/javascript">

$('#inputfile').bind('change', function() {

```
        let fileSize = this.files[0].size/1024/1024; // this gives in MB
        if (fileSize> 1) {
          $("#inputfile").val(null);
alert('file is too big. images more than 1MB are not allowed')
          return
        }
        let ext = $('#inputfile').val().split('.').pop().toLowerCase();
if($.inArray(ext, ['jpg','jpeg']) == -1) {
          $("#inputfile").val(null);
alert('only jpeg/jpg files are allowed!');
        }
     });
</script>
</body>
</html>


<!DOCTYPE html>
<html lang="en" >
<head>
<meta charset="UTF-8">
<title>Avail Service</title>
<link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/twitter-
bootstrap/4.1.3/css/bootstrap.min.css'>
<link rel='stylesheet' href='https://fonts.googleapis.com/css?family=Muli'>
<link rel="stylesheet" href="/static/style1.css">
```

```
</head>

<body>

<!-- partial:index.partial.html -->

<div class="head pt-5">

<h1 class="text-center">Need service? Book Appointment right now!</h1>


<div class="container">

<div class="row">

<div class="col-md-5 mx-auto">

<div class="card card-body">


<form id="submitForm" action="/login" method="post" data-parsley-validate="" data-parsley-errors-messages-disabled="true" novalidate="" _lpchecked="1"><input type="hidden" name="_csrf" value="7635eb83-1f95-4b32-8788-abec2724a9a4">

<div class="form-group required">

<lSabel for="username">Email</lSabel>

<input type="text" class="form-control text-lowercase" id="email" required="" name="email" value="">

</div>

<div class="form-group required">

<lSabel for="username">Mobile no</lSabel>

<input type="number" class="form-control text-lowercase" id="password" required="" name="password" value="">

</div>
```

```
<div class="form-group required">

<lSabel for="username">Location</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required="" name="username" value="">

</div>

<div class="form-group required">

<lSabel for="username">Time to Allocate</lSabel>

<div class="slider">

<input type="range" min="1" max="24" value="4" oninput="rangeValue.innerText = this.value">

<p id="rangeValue">4</p><p>hrs</p>

</div>

</div>

<div class="form-group required">

<lSabel for="username">Additional Description</lSabel>

<input type="text" class="form-control text-lowercase" id="username" required="" name="username" value="">

</div>

<div class="form-group pt-1">

<button class="btnbtn-primary btn-block" type="submit">Avail Service</button>

</div>

</form>

<p class="small-xl pt-3 text-center">

<span class="text-muted">Not a member?</span>

<a href="regform">Create Account</a>
```

```
</p>

</div>

</div>

</div>

</div>

</div>


<!-- partial -->


</body>

</html>
```

APPENDIX II – SCREENSHOTS

The Detection of defective parts by developing an integrated system for car damage detection utilizing multiple Convolutional Neural Networks (CNNs) within a Flask web application framework. The Home page of this system shown in Figure 7.1

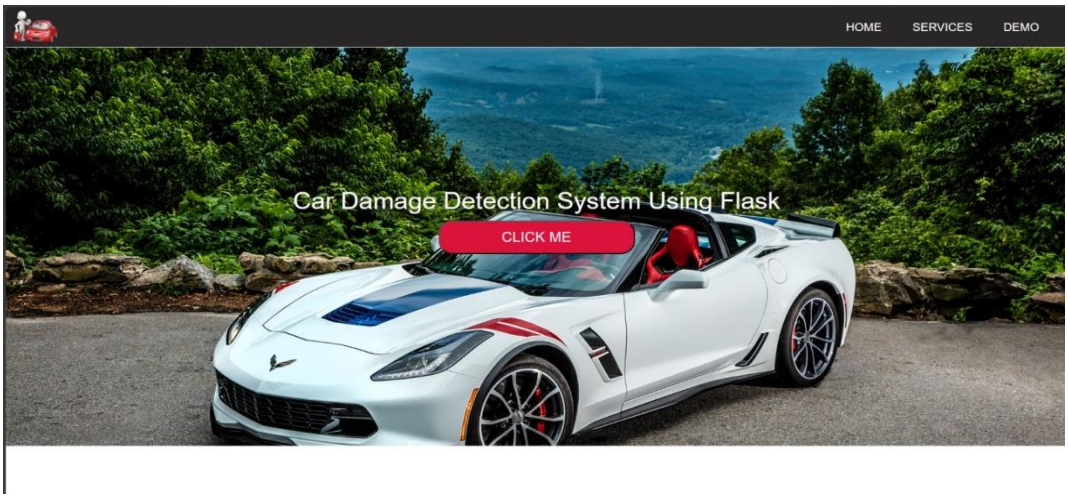

Figure 7.1 Home Page

Implementing image uploading for car damage detection involves a combination of web development, backend processing, and image recognition technologies. We used to upload the damaged car Images (shown in Figure 7.2), where they making process seamless and efficient.

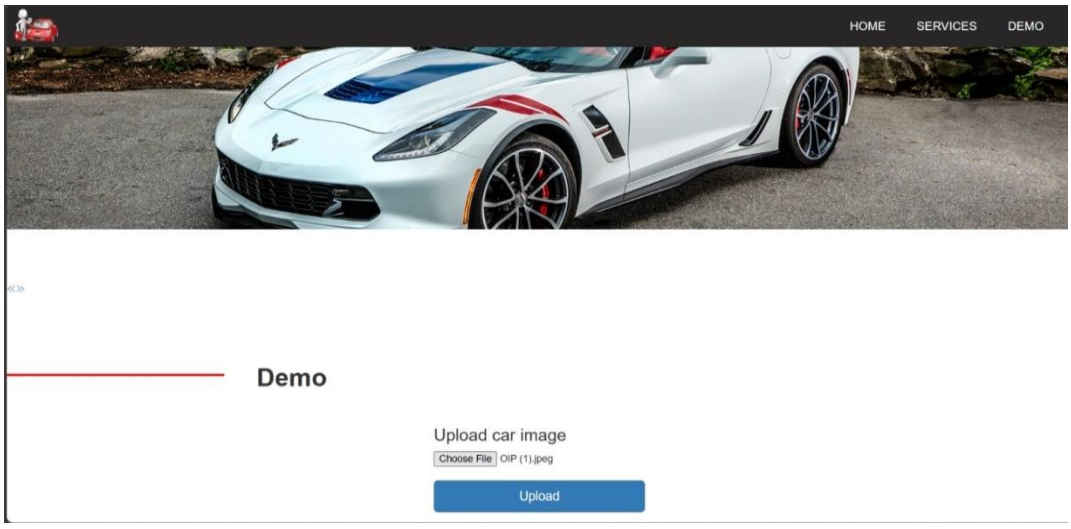

Figure 7.2 Uploading Input(Damaged Car Images)

Our trained CNN module used to identify and localize car damage, distinguishing between scratch, dent, and structural issues. Integrate algorithms for precise classification and highlighting, providing an intuitive visual representation for users to assess damage severity and extent. For each detected damage, a unique QR code is generated, encoding information such as damage type, location, severity, and timestamp. The Output is shown in Figure 7.3

Figure 7.3 Output

A damaged vehicle typically exhibits visible signs of impairment, including dents, scratches, crumpled body panels, broken lights, or misaligned components. (Figure 7.4) These damages can vary in severity, ranging from minor cosmetic issues to significant structural damage, impacting the vehicle's functionality and safety.

Figure 7.4 Damaged Vehicle

It is in a condition where it exhibits no visible signs of impairment or structural damage. Non-damaged vehicles typically undergo routine maintenance to ensure optimal performance and safety. They are considered to be in good condition (Figure 7.5) and suitable for regular use on the road**.**

Figure 7.5 Non Damaged Vehicle

The inclusion of a QR code output further streamlines information access and retrieval, contributing to the overall efficiency and convenience of the system. Figure 7.6 shows the detection of damages.

Figure 7.6 Damage Detection