

# **INTELLIGENT VEHICLE DAMAGE ASSESSMENT AND COST ESTIMATOR FOR INSURANCE COMPANIES**

## **ABSTRACT**

The motor insurance sector loses a lot of money as a result of leakage claims. The gap between the amount actually paid for claims and the amount that would have been paid had all of the best practises in the industry been followed is known as underwriting leakage. These results have been reached using both testing and visual assessment. However, they do delay the processing of claims. By reducing loss adjustment costs, improvements in the First Notice of Loss and the speed with which claims are examined and evaluated might save a lot of money in the automobile insurance claims process. Car damage is automatically identified and classified using advanced picture analysis and pattern recognition technology, a method for automatically locating the damaged area by comparing photos of the automobile from before and after an accident. This project's proposed a CNN model that can recognise a car's damage area. If users upload images, the model can evaluate damage (be it a dent or scratch from an object), and it can also estimate the extent of damage. Insurance firms can handle claims more efficiently as a result. When accepting a car loan, particularly one for a used vehicle, lenders may also consider this model.

# **1. INTRODUCTION**

## **1.1 PROJECT OVERVIEW**

Vehicles are significantly rising in today's globe. Because there are more cars on the road, accidents happen more frequently because individuals are driving them at high speeds. When an accident occurs, the people file a claim with their auto insurance for the necessary funds to repair the car, because to inaccurate claims, the corporation behaves improperly and doesn't make payments now. This occurs as a result of claims leakage, which is the discrepancy between the sums secured by the firm and the sums that it should have secured in accordance with the claims. Even if the car's damage is easily seen, the claim procedure will take longer than usual in accordance with company policy. Despite the company's best efforts, there is a delay in the claims procedure. Differentiate the suggested approach to perhaps speed up the process of assessing automotive damage. Instead of taking hours to accomplish automotive damage detection if it were visually inspected, a system may perform it in a minute by just providing a picture of a damaged vehicle. The system can determine the analysis of the damage, the position of the damage, and the degree of the damage using machine learning and computer vision.

## **1.2 PURPOSE**

Today's world is seeing a substantial increase in automobiles. Because there are more automobiles on the road and more people are driving them at high speeds, accidents happen more frequently. When an accident happens, the parties involved submit a claim with their auto insurance to obtain the money needed to repair the vehicle since, according to false claims, the company acts inappropriately and withholds payments.

## **2. LITERATURE REVIEW**

### **1. EXISTING PROBLEM**

**1.TITLE: Convolutional Neural Networks for vehicle damage detection, 2021**

**AUTHOR NAME: R.E. Ruitenbeek**

Vehicle damage is becoming an increasing liability for shared mobility providers. The high number of driver handovers necessitates the use of an accurate and quick inspection system capable of detecting minor damage and categorising it. To address this, a damage detection model is created that locates vehicle damages and categorises them into twelve groups. To improve detection performance, multiple deep learning algorithms are used, and the effect of various transfer learning and training strategies is evaluated. The final model, which was trained on over 10,000 damage photos, can detect minor defects in a variety of environments, including water and dirt. A performance evaluation using domain experts reveals that the model performs comparably. Furthermore, the model is tested in a specially designed light street, demonstrating how strong reflections complicate detection performance.

**2.1.2. TITLE: Deep Learning Based Car Damage Detection, Classification and Severity**  
**AUTHOR NAME: Ritik Gandhi1, 2021**

Because it is a manual procedure, resolving a claim in the accident insurance sector takes time, and there is a gap between the ideal and real settlement. We are using deep learning models to not only speed up the process, but also to deliver better customer service and boost insurance company profitability. In this paper, we use multiple pre trained models such as VGG 16, VGG 19, Resnet50, and DENSENET to choose the top performing models. We first use the Resnet50 model to determine whether or not the automobile is damaged, and if it is, we utilise the WPOD-net model to identify the licence plate. The YOLO model is used to detect the affected region. Finally, the damage severity is implemented using the DENSENET model. We discovered that transfer learning outperforms fine-tuning after applying multiple models. Furthermore, we present a framework that incorporates all of this into a single application, assisting in the automation of the insurance sector.

### **2.1.3. TITLE: Car Damage Assessment to Automate Insurance Claim, 2022**

**AUTHOR NAME: Siddhant Gole**

Car damage inspection is an essential stage in claim sanctioning, and the procedure is frequently delayed and erroneous, resulting in claim leakages. Our task is to create a web application connected with a deep learning model that receives user input in the form of photographs of damaged automobiles and assesses the damage to provide a cost report that the firm can use to approve the first reimbursement. To detect and localise the damaged regions, the model employs the MASK R-CNN algorithm in conjunction with Faster RCNN. The device also includes a security module that detects and stores the vehicle's licence plate, body type, and logo data for verification. Our goal is to develop a system that can detect damaged parts of a car using images and generate a cost analysis report that the company can use to sanction the insurance amount. The task would be to develop an end-to-end system for detecting and classifying types of damage via images, as well as to implement a car number plate, body type, and logo detection system to verify car details.

#### **2.1.4. TITLE: Using Machine Learning Models To Compare Various Resampling Methods In Predicting Insurance Fraud, 2021**

**AUTHOR NAME: Ruixing Ming**

Insurance fraud is one of the most prevalent kinds of fraud. In particular, the cost of automotive insurance fraud is significant for property insurance companies and has a long-term influence on insurance businesses' pricing tactics. And Car insurance fraud detection has become required in order to reduce insurance prices. Although predictive models for detecting insurance fraud are widely used in practise, there are few published research on the use of machine learning algorithms to identify insurance fraud, most likely due to a lack of available data. Evaluate 13 machine learning approaches in this paper using real-world data. Predicting insurance fraud has become a big difficulty due to the uneven datasets in this domain. Because our data consists primarily of "non-fraud claims" with a minor number of "fraud claims." As a result, classification models predict fraud poorly; thus, the current study seeks to propose an approach that improves machine learning algorithms' results by using re-sampling techniques, such as Random over Sampler, Random under Sampler, and hybrid methods, to address the issue of unbalanced data.

### **2.1.5. TITLE: Evaluation of deep learning algorithms for semantic segmentation of car parts, 2021**

**AUTHOR NAME: Kitsuchart Pasupa**

One of the most significant operations in the auto insurance industry is the evaluation of accident-damaged vehicles. Currently, each fundamental component must be manually examined. It is believed that in the future, a smart device will be able to do this evaluation more effectively. We analysed and compared five deep learning algorithms for semantic segmentation of automobile parts in this work. Mask R-CNN served as the baseline reference method, while the other algorithms were HTC, CBNet, PANet, and GCNet. These five algorithms were used to do instance segmentation runs. HTC's ResNet-50 algorithm was the best for segmentation on various types of cars such as sedans, trucks, and SUVs. On our initial data set, it attained a mean average accuracy of 55.2 when distinct labels were allocated to the left and right sides, and 59.1 when a single label was assigned to both sides. Furthermore, the models from each method were verified for robustness by running them on photos of components in a real-world setting with varying weather conditions such as snow, frost, fog, and different lighting situations. When left and right sides were assigned different labels, GCNet achieved a mean performance under corruption,  $mPC = 35.2$ , and a relative degradation of performance on corrupted data, compared to clean data ( $rPC$ ), of 64.4%, and  $mPC = 38.1$  and  $rPC = 69.6\%$  when left and right sides were considered the same part.



## **2.2 REFERENCES**

- [1]. R.E. Ruitenbeek, Convolutional Neural Networks for vehicle damage detection, 2021
- [2]. Ritik Gandhi1Deep Learning Based Car Damage Detection, Classification and Severity, 2021
- [3]. Siddhant Gole, Car Damage Assessment to Automate Insurance Claim, 2022
- [4]. Ruixing Ming, Using Machine Learning Models To Compare Various Resampling Methods In Predicting Insurance Fraud, 2021
- [5]. Kitsuchart Pasupa, Evaluation of deep learning algorithms for semantic segmentation of car parts, 2021

## 2.3 PROBLEM STATEMENT DEFINITION

In existing system, the procedure of making an insurance claim for an automobile is laborious, and there is a delay before the first reimbursement is authorised. Insurance firms lose millions of dollars each year due to claim leakage as a result of the expansion of the vehicle sector and the daily rise in the number of accidents. The discrepancy between the company's actual spending and what they should have really spent is known as claim leakage. Ineffective claim processing, erroneous payments, human error such as a lack of quality control or poor customer service or even claim fraud may be to blame for this. Auditing closed claim files is the only way to find claim leakage.

### Problem Statement 1:



### Problem Statement 2:



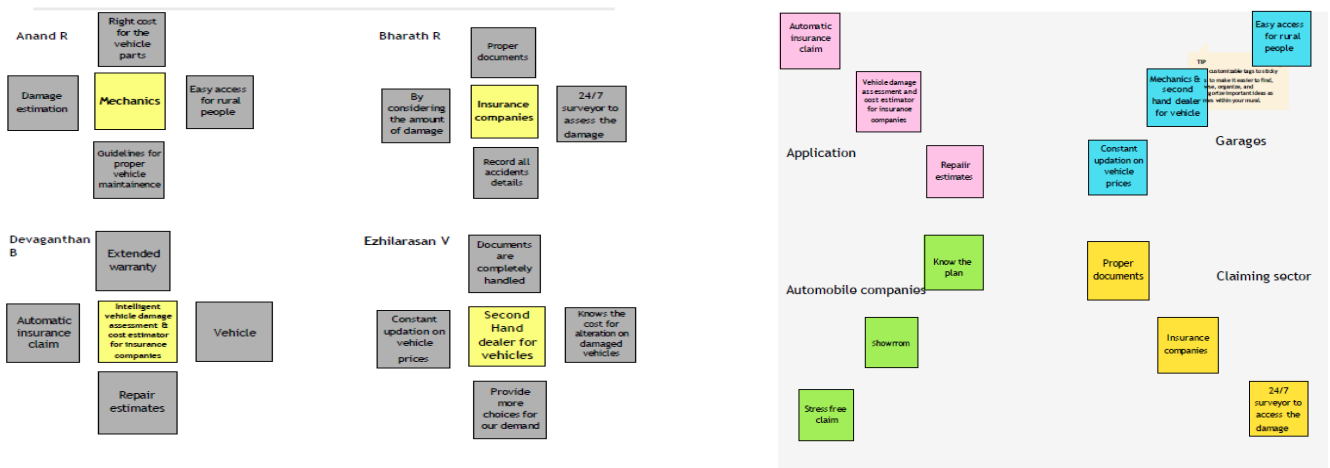
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Person whose car is damaged due to accident.	Calculate the damage assessment of my car.	I don't know how to calculate the price amount.	I don't know the process to calculate.	Confused
PS-2	Car owner	Find the percentage / level of damage in my car	I don't know the process to calculate the damage percentage	I don't know any application or website for the process.	Helpless and frustrated

### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 EMPATHY MAP CANVAS

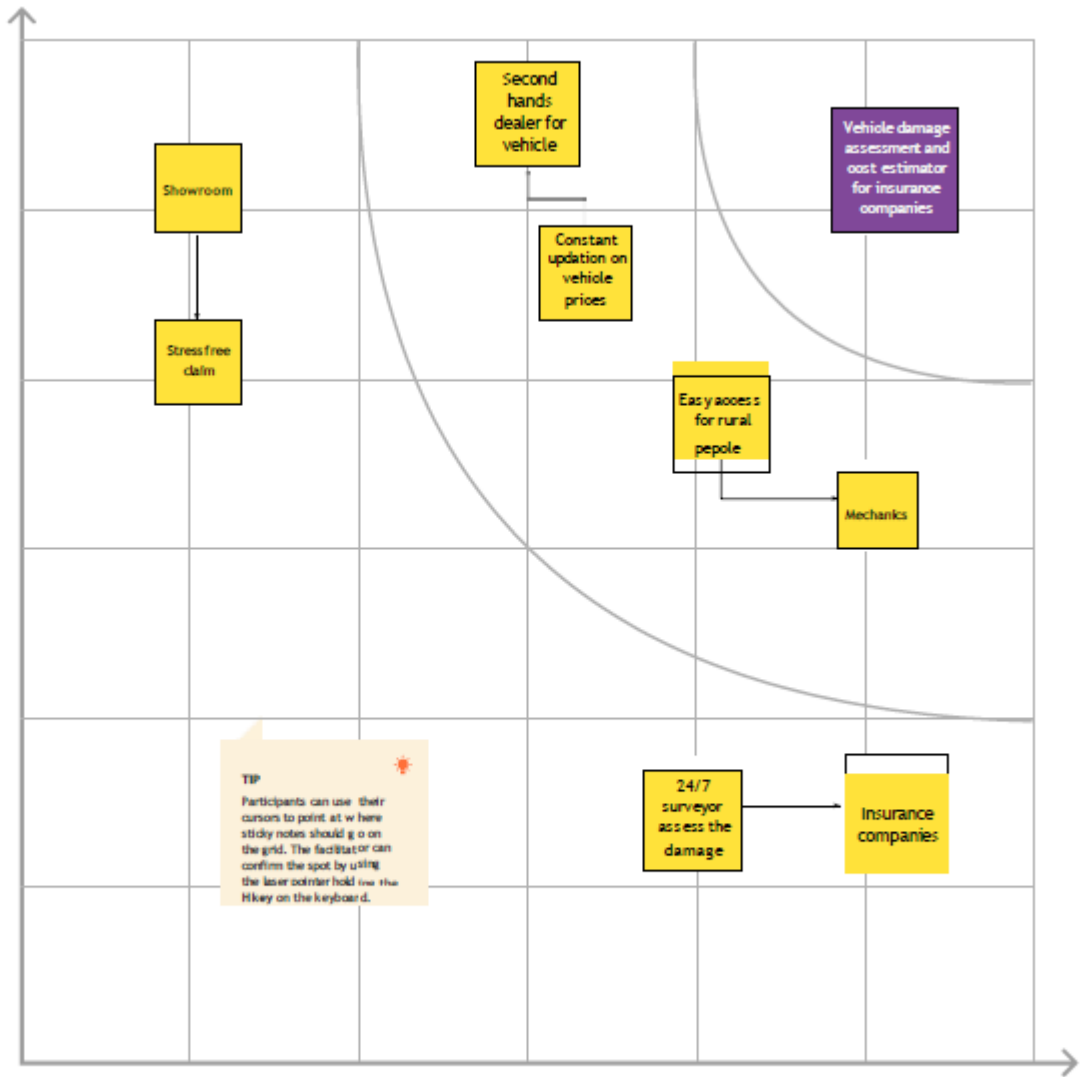


#### 3.2 IDEATION & BRAINSTORMING



Importance

If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?



2

Feasibility

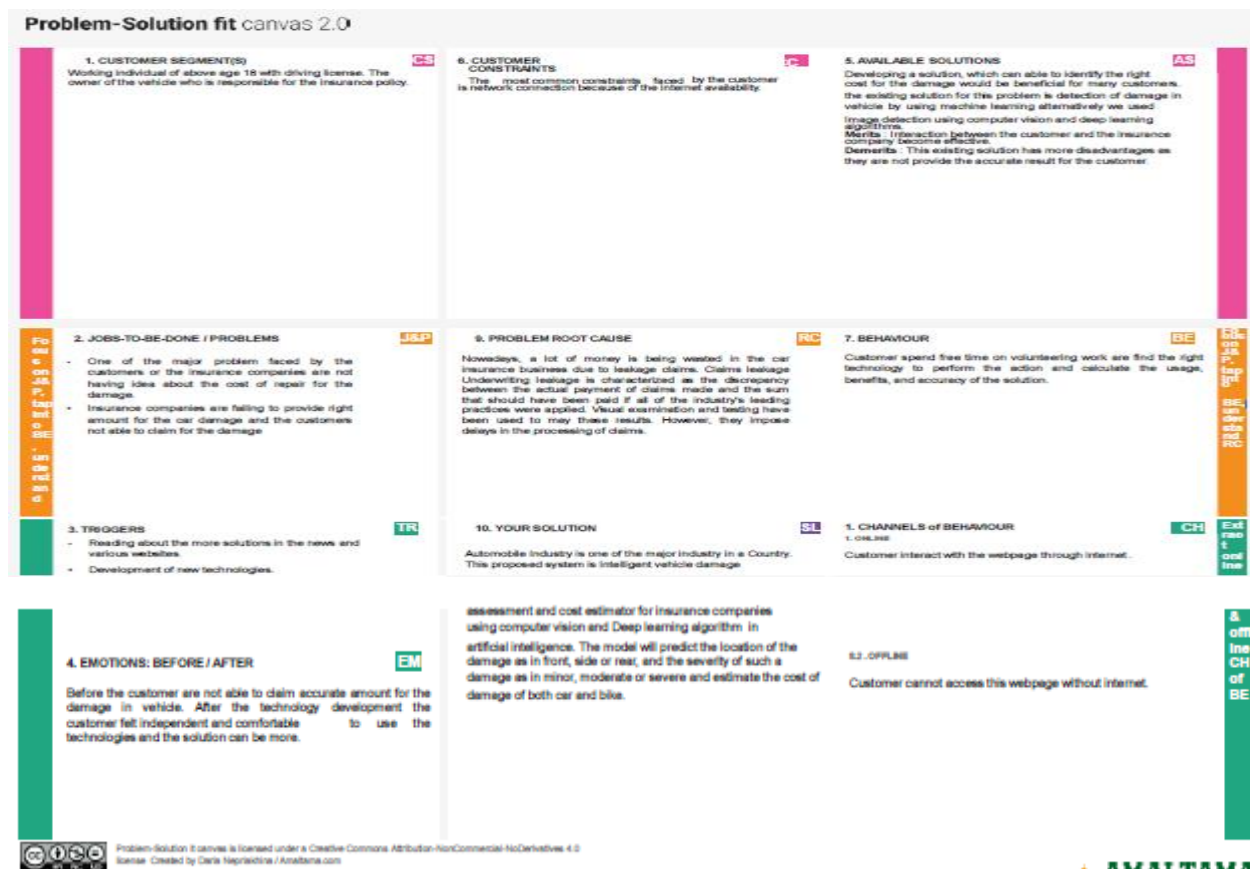
### 3.3 PROPOSED SOLUTION

The proposed approach collects photographs of a person's damaged automobile, then utilises those images as input for a deep learning model that use image processing to recognise the elements of the image and determine the percentage of the vehicles'' damage. After then, the images are separated into two groups: replace and repair. When the damage percentage is less than 80, the damaged part must be replaced; however, in the other case, the compensation amount is set depending on the damage percentage. Finally, it generates a comprehensive analysis report on the vehicle that is used to ask the insurance company for payment.

S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	<ul style="list-style-type: none"><li>❖ Insurance companies, 72.22% of small cases require the presence of damage fixers, which leads to high cost of risk investigation, and the leakage problem in the process of damage fixing is difficult to control.</li><li>❖ The accident party, the long waiting time at the accident site, the slow payment process, the unreasonable fixed price and other issues, to a certain extent, reduce customer satisfaction with the insurance company</li></ul>
2.	Idea / Solution description	<ul style="list-style-type: none"><li>❖ Deep neural networks have led to a series of breakthroughs for image classification. With the development of deep learning the process of computer vision has been greatly accelerated. Research on visual recognition is undergoing a transition from feature engineering to network engineering</li><li>❖ At the same time, the massive data of the automobile insurance industry provides abundant raw materials for network model training.</li></ul>
3.	Novelty / Uniqueness	<ul style="list-style-type: none"><li>❖ The dataset used consists of more set images of damaged vehicles with a single class named scratch.</li><li>❖ The images are precisely annotated with the area of damage where in the area of damage is highlighted</li></ul>
4.	Social Impact / Customer Satisfaction	<ul style="list-style-type: none"><li>❖ This model would help in reducing the cost of processing insurance claims and lead to greater customer satisfaction.</li><li>❖ It can identify damage to all passenger cars. Damage range includes exterior parts, while environment range encompasses rain, snow, dark, bright light, etc..</li></ul>

### 3.4 PROBLEM SOLUTION FIT

There is no systematic approach to receive a rapid answer from an insurance company. A week of waiting is required. The proposed solution should enable consumers to contact with the insurance provider and receive payments both online and offline. After uploading the damaged image and determining the extent of the damage, the user may obtain insurance only if the company approves the damaged image and the condition is more than 80%.



## 4. REQUIREMENT ANALYSIS

### 4.1 FUNCTIONAL REQUIREMENT

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User registration	Download the app Registration through Gmail Create an account Follow the instructions
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Interface	Good Interface to user to operate
FR-4	Accessing datasets	Details about user Details about vehicle Details about insurance companies
FR-5	Mobile application	AI and camera sensor in the field can be access by mobile application.

#### Framework Creation:

This approach provides a way for evaluating vehicle damage that insurance companies may utilise when processing claims. This module offered a framework for submitting a vehicle's damaged parts and requesting insurance from an organisation. The dataset needed to train the Damage Detection and it has prepared by an admin. In order to make the images useful for training, they were manually annotated; damages were categorised into 7 distinct types such as Door Dent, Bumper Dent, Body Scratch, Broken Windshield, Broken Glass, Broken Lights and Smash By modifying its settings and loading the learned dataset, the model was set up to train on user data.

#### Object Detection

Employ a specially trained CNN model utilising transfer learning on to identify the object. This model takes different forms of damage into account validation sets such as Bumper Dent, Bumper Scratch, Door Dent, Door Scratch, Glass Shattered, Head Lamp, Tail Lamp, Undamaged, etc. The classification of car damage severity is as follows: Minor Damage which typically involves slight damage to the vehicle that does not impede the vehicle to cause severe injuries. It includes the headlight scratches, dents and digs in the hood or windshield, from gravel or debris, scratches in the paint. Moderate Damage which deals with any kind of damage that impairs the functionality of the vehicle in any way is moderate damage. It involves large dents in hood, fender or door of a car. Even if the airbags are deployed during collision, then it comes under moderate damage. Severe Damage. These types of damages are a big threat to the human life.

**Damage Detection:**

To locate damaged areas in a picture and create a bounding box around each object found, object localization is used which combines object localisation and classification to provide a bounding box and a class for each item for object detection. Use CNN to generate a convolutional features map from an image to forecast the class and bounding box of an item. If the car is undamaged then it simply detects it and if it's a damaged one, then there are further localizations made models. The model shows accuracy on the validation set. To automate such a system, the easiest method would be to build a Convolution Neural Network model capable of accepting images from the user and determining the location and severity of the damage. The model is required to pass through multiple checks would first ensure that given image is that of a car and then to ensure that it is in fact damaged. These are the gate checks before the analysis begins. Once all the gate checks have been validated, the damage check will commence. The model will predict the location of the damage as in front, side or rear, and the severity of such damage as in minor, moderate or severe.

**Claim Insurance**

The procedure of claiming insurance is done by persons who are in need. For access to the company's insurance, the user must register and authenticate. After that, users may access their insurance information and submit an insurance claim request. The request for an insurance claim can be viewed and approved by the insurance company. Once the damaged image has been uploaded and the degree of the damage has been determined, the user may receive insurance only if the firm accepts the damaged image and the condition is greater than 80%.



## **4.2 NON FUNCTIONAL REQUIREMENTS**

### **Usability**

The system shall allow the users to access the system with pc using web application. The system uses a web application as an interface. The system is user friendly which makes the system easy

### **Availability**

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

### **Scalability**

Scalability is the measure of a system's ability to increase or decrease in performance and cost in response to changes in application and system processing demands.

### **Security**

A security requirement is a statement of needed security functionality that ensures one of many different security properties of software is being satisfied.

### **Performance**

The information is refreshed depending upon whether some updates have occurred or not in the application. The system shall respond to the member in not less than two seconds from the time of the request submittal. The system shall be allowed to take more time when doing large processing jobs. Responses to view information shall take no longer than 5 seconds to appear on the screen.

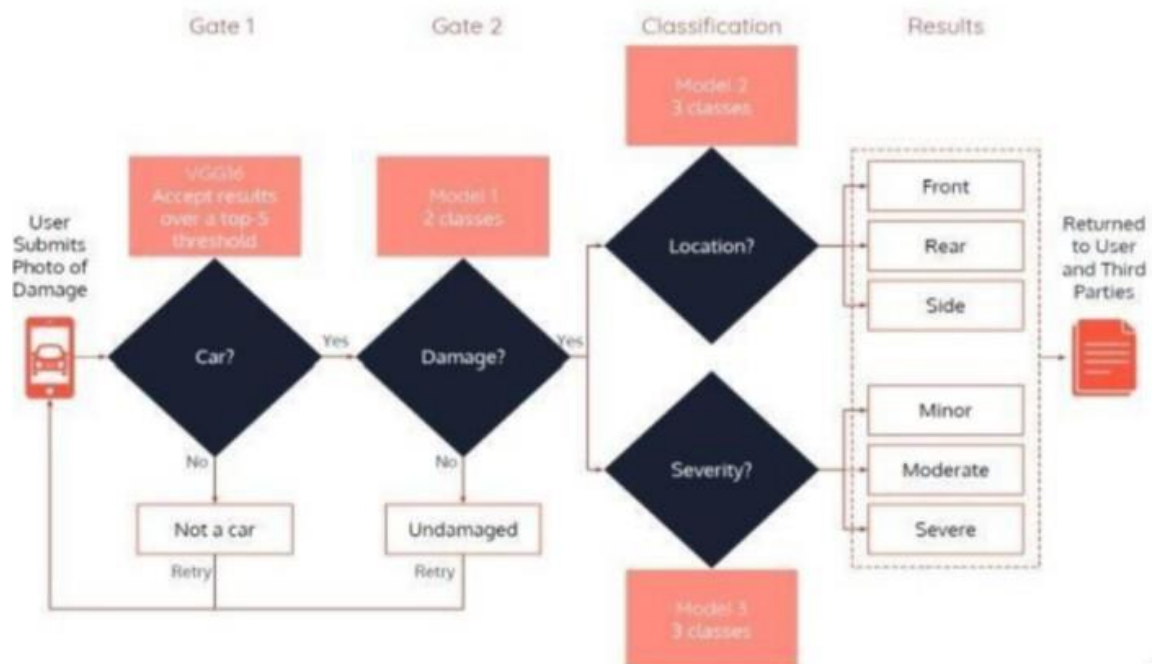
### **Reliability**

The system has to be 100% reliable due to the importance of data and the damages that can be caused by incorrect or incomplete data. The system will run 7 days a week. 24 hours a day.

## 5. PROJECT DESIGN

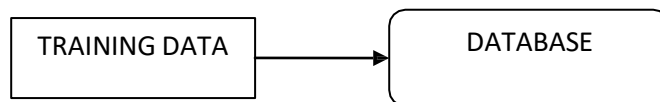
### 5.1 DATA FLOW DIAGRAMS

A two-dimensional diagram explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must identify external inputs and outputs, determine how the inputs and outputs relate to each other, and explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.



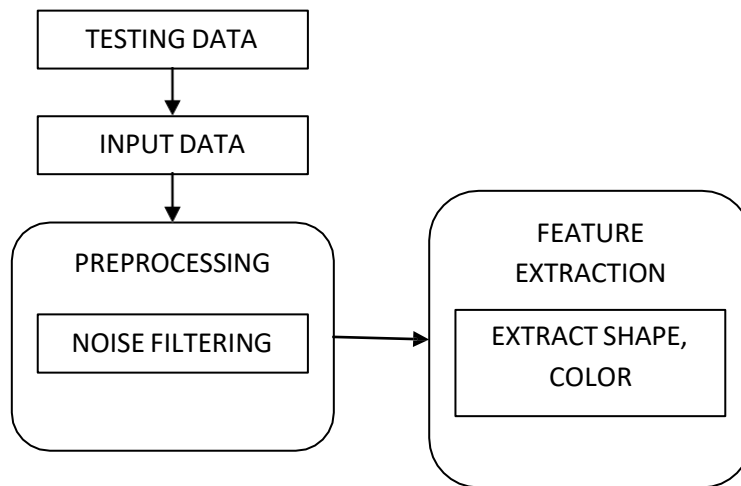
## LEVEL 0

The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



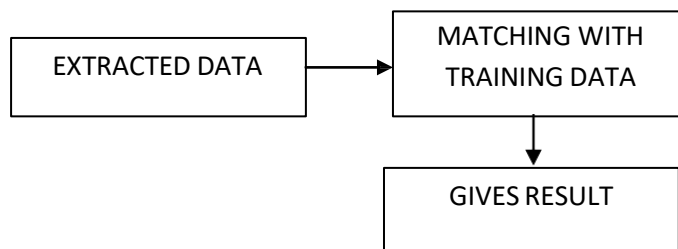
## LEVEL 1

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.



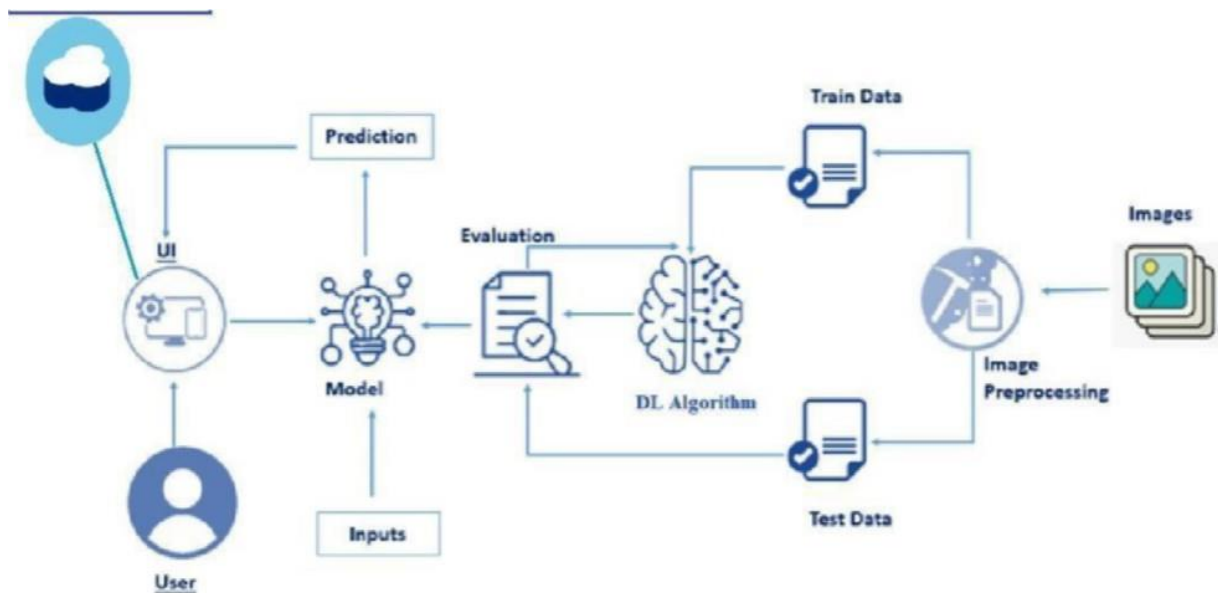
## LEVEL 2

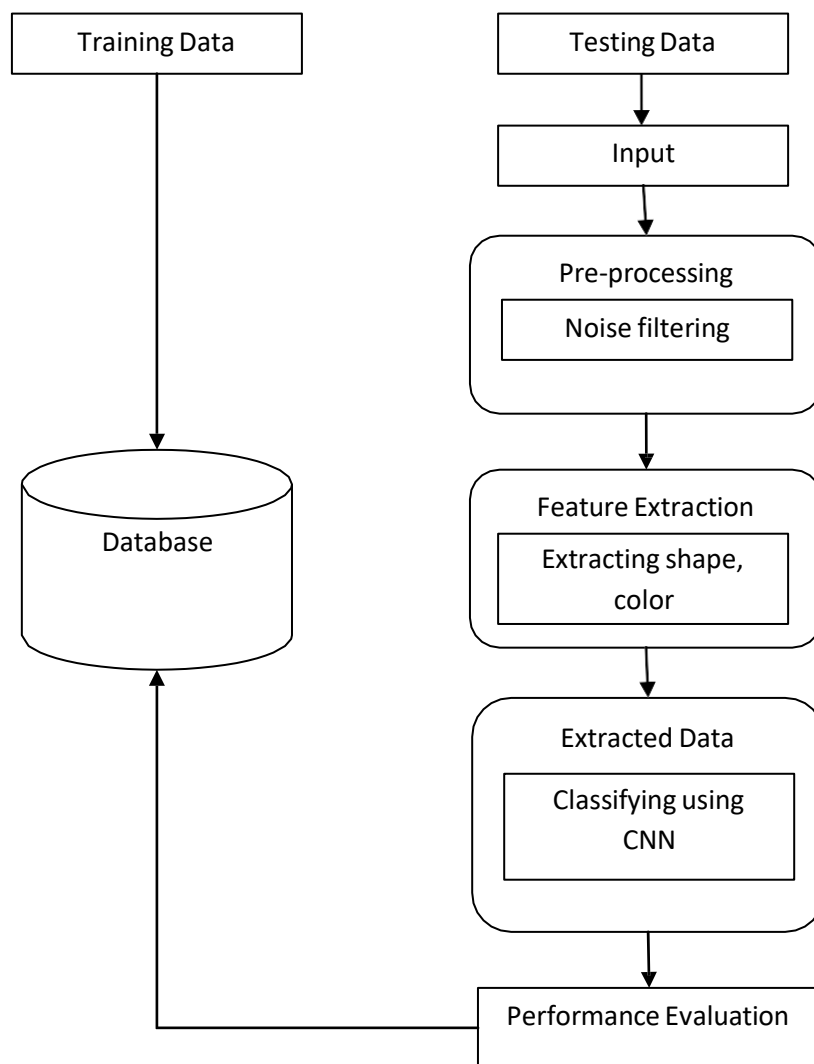
A Data Flow Diagram (DFD) tracks processes and their data paths within the business or system boundary under investigation. A DFD defines each domain boundary and illustrates the logical movement and transformation of data within the defined boundary. The diagram shows 'what' input data enters the domain, 'what' logical processes the domain applies to that data, and 'what' output data leaves the domain. Essentially, a DFD is a tool for process modelling and one of the oldest.



## 5.2 SOLUTION & TECHNICAL ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them. It can provide a plan from which products can be procured, and systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).





### 5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
Customer Details	Login	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
Customer Uses	Dashboard	USN-3	As a user, I can register for the application through Facebook.	I can register & access the dashboard with Facebook Login	Low	Sprint-2
Customer Options	Detail about banks	USN-4	As a user, I can register for the application through Gmail	I can register & access the Details through gmail.	Medium	Sprint-2
Customer Must do	Camera scanner	USN-6	As a user must scan the damage vehicle in camera, it scans the entire damage.	I can scan the entire vehicle In camera .	High	Sprint-1
Customer Value	Details about cost based on damage	USN-7	It gives the estimation cost based on the damage.	I can get the estimated cost price.	Medium	Sprint-1
Customer Care Executive	Good Customer support	USN-8	We have good customer support to the user to apply the insurance	I can get good customer support.	Low	Sprint-2
Administrator	To finish the customer Work	USN-9	We will finish the customer needs in good manner without any failure.	I can finish my work without any failures.	High	Sprint-1



## 6. PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN - 1	As a user, I can register for the application by entering my details of name, email, cars etc. verifying my Gmail account and creating new account with password	7	HIGH	TM-1,4
Sprint-1	Login	USN -2	As a user, entering my email, and password, and confirming my password, I can login to my account.	7	HIGH	TM-1,4
Sprint-1	Dashboard	USN-3	As a user, I can clearly see data, point, graphs, charts and trends of my previous activity and global activity related to my views	2	LOW	TM-1,4
Sprint-2	Details about insurance company	USN-4	As a user, I can register for the Application through Gmail and account id.	8	MEDIUM	TM-2,3
Sprint-1	repeated logins and logout	USN-5	As a user, I can log in and view my dashboard at my demand on any time	4	HIGH	TM-1,4
Sprint-2	Webpage	USN-6	As a user, I must enter all details of car, accident, capture images of my vehicle and upload it into the web portal.	12	HIGH	TM-2,3
Sprint 3	Details about estimated cost based on damage	USN-7	As a user I must receive a detailed report of the damages present in the vehicle and the Cost estimated.	20	HIGH	TM-1,2
Sprint 4	Provide friendly and efficient	USN-8	As a user, I need to get support from developers in case of	10	MEDIUM	TM-1,2,3

## 6. PROJECT PLANNING & SCHEDULING

### 6.1 SPRINT PLANNING & ESTIMATION

### 6.3 REPORTS FROM JIRA














Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN - 1	As a user, I can register for the application by entering my details of name, email, cars etc. verifying my Gmail account and creating new account with password	7	HIGH	TM-1,4
Sprint-1	Login	USN -2	As a user, entering my email, and password, and confirming my password, I can login to my account.	7	HIGH	TM-1,4
Sprint-1	Dashboard	USN-3	As a user, I can clearly see data, point, graphs, charts and trends of my previous activity and global activity related to my views	2	LOW	TM-1,4
Sprint-2	Details about insurance company	USN-4	As a user, I can register for the Application through Gmail and account id.	8	MEDIUM	TM-2,3
Sprint-1	repeated logins and logout	USN-5	As a user, I can log in and view my dashboard at my demand on any time	4	HIGH	TM-1,4
Sprint-2	Webpage	USN-6	As a user, I must enter all details of car, accident, capture images of my vehicle and upload it into the web portal.	12	HIGH	TM-2,3
Sprint 3	Details about estimated cost based on damage	USN-7	As a user I must receive a detailed report of the damages present in the vehicle and the Cost estimated.	20	HIGH	TM-1,2
Sprint 4	Provide friendly and efficient	USN-8	As a user, I need to get support from developers in case of	10	MEDIUM	TM-1,2,3

	customer support and sort out the queries.		queries and failure of service Provided by chat-box, mail or call.			
Sprint 4	overview the entire process and act as a bridge between user and developer	USN-9	As a Team member, We need to satisfy the customer needs in an efficient way and make sure any sort of errors are fixed	10	HIGH	TM-1,2,3

## 6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	-	-
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	-	-
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	-	-

## 6.3 Reports from JIRA

<p>TO DO 1 ISSUE</p> <p>Overview the entire process and act a bridge between user and developer</p> <p> JAC-14</p>	<p>IN PROGRESS 4 ISSUES</p> <p>Design a Webpage for the Project</p> <p> JAC-2 </p> <p>Repeated login and logout</p> <p> JAC-9</p> <p>Provide friendly and efficient customer support and sort out the quries</p> <p> JAC-13</p>	<p>DONE 7 ISSUES </p> <p>Design User Interface</p> <p> JAC-3  </p> <p>Design the Login form with gmail confirmation</p> <p> JAC-4 </p> <p>Detials about insurence companies</p> <p> JAC-10 </p> <p>Input Data's</p>
---	--	---

## 7. CODING & SOLUTIONING

### DATABASE SCHEMA

#### 1. Database Schema

A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields, data types, and the relationships between these entities. Schemas commonly use visual representations to communicate the architecture of the database, becoming the foundation for an organization's data management discipline. A database schema is considered the "blueprint" of a database which describes how the data may relate to other tables or other data models. However, the schema does not actually contain data. key benefits of database schemas include:

- Access and security: Database schema design helps organize data into separate entities, making it easier to share a single schema within another database.
- Organization and communication: Documentation of database schemas allow for more organization and better communication among internal stakeholders.
- Integrity: This organization and communication also helps to ensure data validity.

## 8. TESTING

### 1. TEST CASES

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

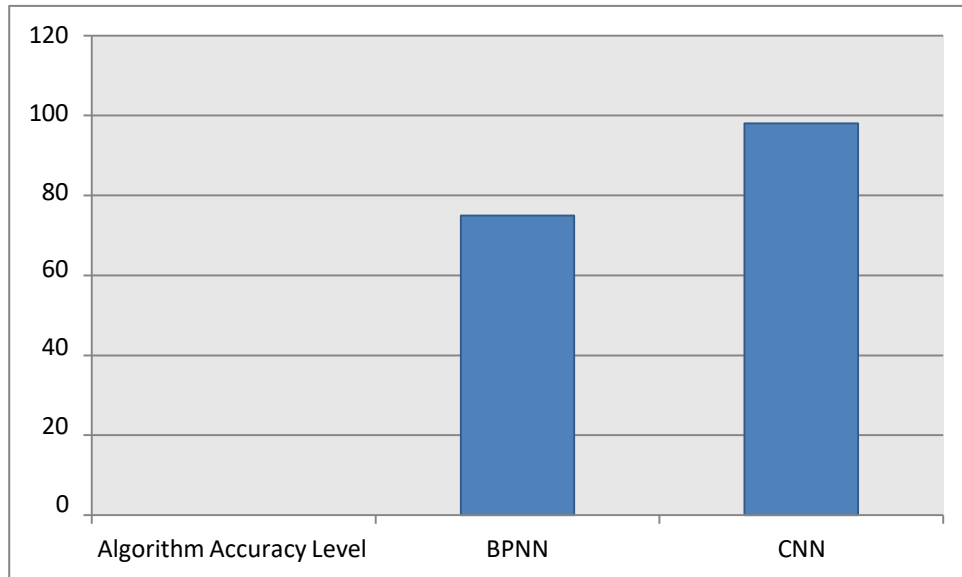
S.NO	Scenario	Input	Excepted output	Actual output
1	User login	User name and password	Login	Login success.
2	Upload Image	Upload damaged vehicle image as a input	Detecting object and analyze for claim insurance	Details are stored in a database.

## **8.2 USER ACCEPTANCE TESTING**

This sort of testing is carried out by users, clients, or other authorised bodies to identify the requirements and operational procedures of an application or piece of software. The most crucial stage of testing is acceptance testing since it determines whether or not the customer will accept the application or programme. It could entail the application's U.I., performance, usability, and usefulness. It is also referred to as end-user testing, operational acceptance testing, and user acceptance testing (UAT).

## 9. RESULTS

### 9.1 PERFORMANCE METRICS



## **10. ADVANTAGES & DISADVANTAGES**

### **ADVANTAGE**

- Digitalized claim process makes easy to use
- Give the accurate result of the damaged vehicle
- Helps the insurance company to analyze the damaged vehicle and also payment process.

### **DISADVANTAGE**

- It will take more time to claim the insurance in manual process
- Because of incorrect claims, the company behaves badly and doesn't make payments currently.
- Poor customer support



## **11. CONCLUSION**

In this research proposal, a neural network-based solution for automobile detection will be used to address the issues of automotive damage analysis and position and severity prediction. This project does several tasks in one bundle. The method will unquestionably assist the insurance firms in conducting far more thorough and systematic analyses of the vehicle damage. Simply sending the system a photograph of the vehicle, it will evaluate it and determine whether there is damage of any type, where it is located, and how severe it is.

## **12. FUTURE SCOPE**

In future work, need to use several regularisation methods with a big dataset in our next work. Anticipate the cost of a car damaged component more accurately and reliably if we have higher quality datasets that include the attributes of a car (make, model, and year of production), location data, kind of damaged part, and repair cost. This study makes it possible to work together on picture recognition projects in the future, with a focus on the auto insurance industry. The study was able to accurately validate the presence of damage, its location, and its degree while eliminating human bias. These can be further enhanced by adding the on the fly data augmentation approaches.

### 13. APPENDIX SOURCE CODE

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import
Dense,Convolution2D,MaxPooling2D,Flatten
from tensorflow.keras.preprocessing import image
import numpy as np
import keras
from matplotlib import pyplot as plt
import io
import urllib.request
#Location Classification
data_level_train = ImageDataGenerator(rescale =
1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,
vertical_flip=True) data_level_test =
ImageDataGenerator(rescale=1./255)
#Training Data
Location_train =
data_level_train.flow_from_directory(r"C:\Users\
ELCOT\Desktop\IBM\Final project\Car
damage\body\training",
target_size=(76,76),batch_size=30,class_mode="categorical")
#Testing Data
Location_test =
data_level_test.flow_from_directory(r"C:\Users\ELCOT\
Desktop\IBM\Final project\Car damage\body\validation",
target_size=(76,76),batch_size=30,class_mode="categorical")
Found 979 images belonging to 3 classes.
Found 171 images belonging to 3 classes.
#Damage Assessment
data_damage_train = ImageDataGenerator(rescale =
1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True,
vertical_flip=True)
data_damage_test =
ImageDataGenerator(rescale=1./255)
#Training Data
Damage_train =
data_damage_train.flow_from_directory(r"C:\Users\ELCOT\D
esktop\IBM\Final project\Car damage\level\training",
target_size=(76,76),batch_size=30,class_mode="categorical")
```

### *#Testing Data*

```
Damage_test =  
data_damage_test.flow_from_directory(r"C:\Users\ELCOT\Desktop\IBM\  
Final project\Car damage\level\validation",  
target_size=(76,76),batch_size=30,class_mode="categorical")  
Found 979 images belonging to 3 classes.  
Found 171 images belonging to 3 classes.
```

### *#Location CNN*

```
location = Sequential()  
location.add(Convolution2D(32,(3,3),activation='relu',input_shape=(76,76,3)))  
location.add(MaxPooling2D(pool_size=(2, 2)))  
location.add(Flatten())  
location.add(Dense(500,activation='relu'))  
location.add(Dense(300,activation='relu'))  
location.add(Dense(3,activation='softmax'))
```

### *#Damage CNN*

```
damage = Sequential()  
damage.add(Convolution2D(32,(3,3),activation='relu',input_shape=(76,76,3)))  
damage.add(MaxPooling2D(pool_size=(2, 2)))  
damage.add(Flatten())  
damage.add(Dense(500,activation='relu'))  
damage.add(Dense(300,activation='relu'))  
damage.add(Dense(3,activation='softmax'))
```

### *#Compiling the models*

#### *#location model*

```
location.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])
```

#### *#Damage model*

```
damage.compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])
```

### **Training the model**

#### *#Location model training*

```
#s_p_e => (979/30) == 33
```

```
#v_s => (171/30) == 6
```

```
Location_Train =
```

```
location.fit(Location_train,steps_per_epoch=33,epochs=30,validation_data=Location_test,validation_steps=6)
```

Epoch 1/30

33/33 [=====] - 63s 1s/step  
- loss: 2.2914 -

accuracy: 0.3626 - val\_loss: 1.1260 - val\_accuracy: 0.4269

Epoch 2/30

33/33 [=====] - 32s

971ms/step - loss: 1.0917

1. accuracy: 0.4280 - val\_loss: 1.0687 - val\_accuracy: 0.4211

Epoch 3/30

33/33 [=====] - 39s 1s/step

- loss: 1.0791 -

accuracy: 0.4127 - val\_loss: 1.0865 - val\_accuracy: 0.4211  
Epoch 4/30  
33/33 [=====] - 33s  
982ms/step - loss: 1.0585  
2. accuracy: 0.4423 - val\_loss: 1.0706 - val\_accuracy: 0.4211  
Epoch 5/30  
33/33 [=====] - 32s  
973ms/step - loss: 1.0499  
3. accuracy: 0.4494 - val\_loss: 1.1198 - val\_accuracy: 0.4386  
Epoch 6/30  
33/33 [=====] - 34s 1s/step  
- loss: 1.0111 -  
accuracy: 0.4974 - val\_loss: 1.0905 - val\_accuracy: 0.4795  
Epoch 7/30  
33/33 [=====] - 33s  
979ms/step - loss: 1.0145  
4. accuracy: 0.5158 - val\_loss: 1.0914 - val\_accuracy: 0.4854  
Epoch 8/30  
33/33 [=====] - 33s  
993ms/step - loss: 0.9759  
5. accuracy: 0.5312 - val\_loss: 1.1071 - val\_accuracy: 0.5439  
Epoch 9/30  
33/33 [=====] - 33s  
987ms/step - loss: 0.9711  
6. accuracy: 0.5301 - val\_loss: 1.1298 - val\_accuracy: 0.4854  
Epoch 10/30  
33/33 [=====] - 32s  
979ms/step - loss: 0.9440  
7. accuracy: 0.5669 - val\_loss: 1.0939 - val\_accuracy: 0.4678  
Epoch 11/30  
33/33 [=====] - 33s  
995ms/step - loss: 0.9361  
8. accuracy: 0.5526 - val\_loss: 1.2094 - val\_accuracy: 0.4854  
Epoch 12/30  
33/33 [=====] - 33s  
988ms/step - loss: 0.9123  
9. accuracy: 0.5761 - val\_loss: 1.1241 - val\_accuracy: 0.4971  
Epoch 13/30  
33/33 [=====] - 35s 1s/step  
- loss: 0.8780 -  
accuracy: 0.5996 - val\_loss: 1.2161 - val\_accuracy: 0.4971  
Epoch 14/30  
33/33 [=====] - 33s  
994ms/step - loss: 0.8684  
10. accuracy: 0.5986 - val\_loss: 1.1626 - val\_accuracy: 0.4503  
Epoch 15/30  
33/33 [=====] - 38s 1s/step  
- loss: 0.8609 -  
accuracy: 0.6210 - val\_loss: 1.3011 - val\_accuracy: 0.4795  
Epoch 16/30  
33/33 [=====] - 34s 1s/step  
- loss: 0.8447 -

accuracy: 0.6139 - val\_loss: 1.3229 - val\_accuracy: 0.5029  
Epoch 17/30  
33/33 [=====] - 32s  
972ms/step - loss: 0.8191  
11.accuracy: 0.6170 - val\_loss: 1.2998 - val\_accuracy: 0.4269  
Epoch 18/30  
33/33 [=====] - 32s  
974ms/step - loss: 0.8148  
12.accuracy: 0.6415 - val\_loss: 1.3024 - val\_accuracy: 0.5088  
Epoch 19/30  
33/33 [=====] - 33s  
990ms/step - loss: 0.7798  
13.accuracy: 0.6599 - val\_loss: 1.2260 - val\_accuracy: 0.4795  
Epoch 20/30  
33/33 [=====] - 35s 1s/step  
- loss: 0.7823 -  
accuracy: 0.6415 - val\_loss: 1.2835 - val\_accuracy: 0.5029  
Epoch 21/30  
33/33 [=====] - 33s  
976ms/step - loss: 0.7301  
14.accuracy: 0.6864 - val\_loss: 1.3868 - val\_accuracy: 0.4795  
Epoch 22/30  
33/33 [=====] - 32s  
961ms/step - loss: 0.7282  
15.accuracy: 0.6915 - val\_loss: 1.2598 - val\_accuracy: 0.5263  
Epoch 23/30  
33/33 [=====] - 33s  
986ms/step - loss: 0.6653  
16.accuracy: 0.7222 - val\_loss: 1.3745 - val\_accuracy: 0.4971  
Epoch 24/30  
33/33 [=====] - 33s 1s/step  
- loss: 0.6918 -  
accuracy: 0.7120 - val\_loss: 1.3745 - val\_accuracy: 0.5146  
Epoch 25/30  
33/33 [=====] - 33s  
995ms/step - loss: 0.6724  
17.accuracy: 0.7120 - val\_loss: 1.4253 - val\_accuracy: 0.5088  
Epoch 26/30  
33/33 [=====] - 34s 1s/step  
- loss: 0.6219 - accuracy: 0.7324 - val\_loss: 1.3387 -  
val\_accuracy: 0.5322  
Epoch 27/30  
33/33 [=====] - 32s  
975ms/step - loss: 0.6533  
18.accuracy: 0.7324 - val\_loss: 1.4048 - val\_accuracy: 0.4561  
Epoch 28/30  
33/33 [=====] - 38s 1s/step  
- loss: 0.6526 -  
accuracy: 0.7242 - val\_loss: 1.3891 - val\_accuracy: 0.4854  
Epoch 29/30

```

33/33 [=====] - 39s 1s/step
- loss: 0.6031 -
accuracy: 0.7426 - val_loss: 1.4374 - val_accuracy: 0.5380
Epoch 30/30
33/33 [=====] - 43s 1s/step
- loss: 0.5996 - accuracy: 0.7497 - val_loss: 1.4803 -
val_accuracy: 0.5146
#plotting the accuracy and loss of Location Model
#Accuracy
plt.plot(Location_Train.history['accuracy'])
plt.plot(Location_Train.history['val_accuracy'])
plt.title('Level model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
#Loss
plt.plot(Location_Train.history['loss'])
plt.plot(Location_Train.history['val_loss'])
plt.title('Level model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
#Damage model training
#s_p_e => (979/30) == 33
#v_s => (171/30) == 6
Damage_Train =
damage.fit(Damage_train,steps_per_epoch=33,epochs=30,validation_data=Damage_test,validation_steps=6)
Epoch 1/30
33/33 [=====] - 47s 1s/step
- loss: 1.5142 -
accuracy: 0.3820 - val_loss: 1.1120 - val_accuracy: 0.3977
Epoch 2/30
33/33 [=====] - 48s 1s/step
- loss: 1.0369 -
accuracy: 0.4637 - val_loss: 1.2644 - val_accuracy: 0.3977
Epoch 3/30
33/33 [=====] - 45s 1s/step
- loss: 1.0091 -
accuracy: 0.4883 - val_loss: 1.2730 - val_accuracy: 0.4152
Epoch 4/30
33/33 [=====] - 45s 1s/step
- loss: 0.9801 -
accuracy: 0.5005 - val_loss: 1.0369 - val_accuracy: 0.4971
Epoch 5/30
33/33 [=====] - 36s 1s/step
- loss: 0.9522 -
accuracy: 0.5189 - val_loss: 1.0641 - val_accuracy: 0.4795
Epoch 6/30

```

33/33 [=====] - 39s 1s/step  
- loss: 0.9307 -  
accuracy: 0.5495 - val\_loss: 1.0548 - val\_accuracy: 0.4678  
Epoch 7/30  
33/33 [=====] - 38s 1s/step  
- loss: 0.9024 - accuracy: 0.5812 - val\_loss: 1.0847 -  
val\_accuracy: 0.4620  
Epoch 8/30  
33/33 [=====] - 64s 2s/step  
- loss: 0.9150 -  
accuracy: 0.5434 - val\_loss: 1.1425 - val\_accuracy: 0.4737  
Epoch 9/30  
33/33 [=====] - 69s 2s/step  
- loss: 0.8955 -  
accuracy: 0.5853 - val\_loss: 1.1233 - val\_accuracy: 0.4561  
Epoch 10/30  
33/33 [=====] - 67s 2s/step  
- loss: 0.8571 -  
accuracy: 0.6180 - val\_loss: 1.1477 - val\_accuracy: 0.4620  
Epoch 11/30  
33/33 [=====] - 70s 2s/step  
- loss: 0.8678 -  
accuracy: 0.5914 - val\_loss: 1.1796 - val\_accuracy: 0.4503  
Epoch 12/30  
33/33 [=====] - 65s 2s/step  
- loss: 0.8404 -  
accuracy: 0.6016 - val\_loss: 1.1327 - val\_accuracy: 0.4678  
Epoch 13/30  
33/33 [=====] - 69s 2s/step  
- loss: 0.8085 -  
accuracy: 0.6415 - val\_loss: 1.1858 - val\_accuracy: 0.4795  
Epoch 14/30  
33/33 [=====] - 56s 2s/step  
- loss: 0.8483 -  
accuracy: 0.6078 - val\_loss: 1.1295 - val\_accuracy: 0.4211  
Epoch 15/30  
33/33 [=====] - 46s 1s/step  
- loss: 0.8118 -  
accuracy: 0.6231 - val\_loss: 1.2709 - val\_accuracy: 0.4561  
Epoch 16/30  
33/33 [=====] - 45s 1s/step  
- loss: 0.7777 -  
accuracy: 0.6364 - val\_loss: 1.2445 - val\_accuracy: 0.5029  
Epoch 17/30  
33/33 [=====] - 59s 2s/step  
- loss: 0.7613 -  
accuracy: 0.6558 - val\_loss: 1.3651 - val\_accuracy: 0.4152  
Epoch 18/30  
33/33 [=====] - 48s 1s/step  
- loss: 0.7729 -  
accuracy: 0.6517 - val\_loss: 1.3303 - val\_accuracy: 0.4854  
Epoch 19/30



33/33 [=====] - 39s 1s/step  
- loss: 0.7196 - accuracy: 0.6803 - val\_loss: 1.3915 -  
val\_accuracy: 0.4737  
Epoch 20/30  
33/33 [=====] - 44s 1s/step  
- loss: 0.7726 -  
accuracy: 0.6721 - val\_loss: 1.3091 - val\_accuracy: 0.4503  
Epoch 21/30  
33/33 [=====] - 47s 1s/step  
- loss: 0.7135 -  
accuracy: 0.6956 - val\_loss: 1.4631 - val\_accuracy: 0.4737  
Epoch 22/30  
33/33 [=====] - 42s 1s/step  
- loss: 0.6661 -  
accuracy: 0.7160 - val\_loss: 1.5048 - val\_accuracy: 0.3977  
Epoch 23/30  
33/33 [=====] - 50s 2s/step  
- loss: 0.6671 -  
accuracy: 0.7232 - val\_loss: 1.5726 - val\_accuracy: 0.4327  
Epoch 24/30  
33/33 [=====] - 47s 1s/step  
- loss: 0.6569 - accuracy: 0.7191 - val\_loss: 1.4662 -  
val\_accuracy: 0.4503  
Epoch 25/30  
33/33 [=====] - 47s 1s/step  
- loss: 0.6579 -  
accuracy: 0.6987 - val\_loss: 1.2992 - val\_accuracy: 0.3743  
Epoch 26/30  
33/33 [=====] - 47s 1s/step  
- loss: 0.6463 -  
accuracy: 0.7232 - val\_loss: 1.4134 - val\_accuracy: 0.4971  
Epoch 27/30  
33/33 [=====] - 46s 1s/step  
- loss: 0.6001 -  
accuracy: 0.7692 - val\_loss: 1.6205 - val\_accuracy: 0.4444  
Epoch 28/30  
33/33 [=====] - 53s 2s/step  
- loss: 0.6009 -  
accuracy: 0.7508 - val\_loss: 1.3528 - val\_accuracy: 0.4094  
Epoch 29/30  
33/33 [=====] - 50s 1s/step  
- loss: 0.5835 -  
accuracy: 0.7549 - val\_loss: 1.4194 - val\_accuracy: 0.4561  
Epoch 30/30  
33/33 [=====] - 47s 1s/step  
- loss: 0.4999 - accuracy: 0.8069 - val\_loss: 1.7530 -  
val\_accuracy: 0.3743

*#plotting the accuracy and loss of Damage model*

*#Accuracy*

```
plt.plot(Damage_Train.history['accuracy'])
plt.plot(Damage_Train.history['val_accuracy'])
plt.title('Damage model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

*#Loss*

```
plt.plot(Damage_Train.history['loss'])
plt.plot(Damage_Train.history['val_loss'])
plt.title('Damage model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

*#Saving the Location Model*

```
location.save("Location.h5")
```

*#Saving the Damage Model*

```
damage.save("Damage.h5")
```

**Testing the Model**

```
Location_train.class_indices
```

```
{'00-front': 0, '01-rear': 1, '02-side': 2}
```

```
Damage_train.class_indices
```

```
{'01-minor': 0, '02-moderate': 1, '03-severe': 2}
```

*#Location Model Testing*

```
img = image.load_img(r"C:\Users\ELCOT\Desktop\IBM\Final  
project\
```

```
Front.jpg",target_size=(76,76))
```

```
x = image.img_to_array(img)
```

```
x = np.expand_dims(x,axis=0)
```

```
pred_prob = location.predict(x)
```

```
class_name = ['00-front', '01-rear', '03-side']
```

```
pred_id = pred_prob.argmax(axis=1)[0]
```

```
class_name[pred_id]
```

```
1/1 [=====] - 5s 5s/step
```

```
'00-front'
```

*#Damage Model Testing*

```
img = image.load_img(r"C:\Users\ELCOT\Desktop\IBM\Final  
project\
```

```
Front.jpg",target_size=(76,76))
```

```
x = image.img_to_array(img)
```

```
x = np.expand_dims(x,axis=0)
```

```
pred_prob = damage.predict(x)
```

```
class_name = ['01-minor', '02-moderate', '03-severe']
```

```
pred_id = pred_prob.argmax(axis=1)[0]
```

```
class_name[pred_id]
```

```
1/1 [=====] - 1s 1s/step
```

```
'02-moderate'
```

## **GITHUB & PROJECT DEMO LINK**

<https://github.com/IBM-EPBL/IBM-Project-20174-1659714163.git>

<https://github.com/IBM-EPBL/IBM-Project-20174-1659714163.git>