Vehicle Damage Detection and Classification Using Image Processing

Abstract: Vehicle have an impact on people's daily safety, and because there are so many different types and sizes of materials, it can be challenging to distinguish and detect the conditions around the vehicle. In this project, we looked into the matter of car damage classification and detection, which insurance providers can utilize to quickly automates the handling of vehicle insurance disputes. Deep convolutional networks can be used to detect car damage and with recent developments in computer vision, which are largely attributable to the implementation of quick, scalable, and entire trainable CNN. We manually gathered and annotated pictures of numerous online sources that showed various kinds of car damage. By analyzing the deep learning-based YOLO (you only look once) and OpenCV series target detection method, a recognition approach that relies on YOLOS is provided to achieve timely and efficient identification of the damage in the vehicle. The COCO dataset's base weights are used to train the model. 35-90 epochs are used to process the photos. The region of damage is highlighted in the final image using a color splash technique after processing. The approach would increase customer satisfaction while assisting in lowering the cost of processing insurance claims. Vendors of automobiles can do away with the labor-intensive manual damage assessment process. Additional vehicles will be priced accurately and transparently, along with any necessary repairs. It is also able to decrease misleading vehicle insurance claims.

Keywords: Vehicle Damage

I. INTRODUCTION

The world today is already unimaginable without automobiles, and there are now daily increases in the number of vehicles on the road. This inherently carries possible risks, and some drivers' lack of control can result in small to severe harm to other people's property and health while driving.

Therefore, as time goes on, vehicle insurance firms receive more insurance claims. Manually evaluating numerous claims would no longer be able to handle them quickly enough for express claim processing, leading to claim leaks. Simple terms, claim leaks is the word used to describe money wasted due to claims processing mistakes that ultimately come from the current manual and automated systems' inefficiency. Claim value mainly depends on the nature of the damage and portion of the vehicle, and that we require an automation approach for the process of filing an insurance claim, that can accurately analyse and identify damage and prevent claim leakage.

In this proposed study, we propose an automated system that would analyse the damaged vehicle and find the severity of the damage. To identify and analyse different kinds of damage in the minor and main portions of the car, we use a deep learning-based object detection approach. Any sort of damage is possible, including bumper dents, door dents, window breaks, headlight and redlight breakage, scratches, and smashes.

A single neural network is used on entire image by the YOLO algorithm. The model then image is split into regions, delivering bounding boxes and forecast probabilities for every section. The expected instances are used to weight these created bounding boxes. The newest and possibly most significant member of YOLO series of object recognition models is YOLOS - You Only Look At One Sequence & OpenCV methods. It is used specifically in the context of vehicle damage recognition in proposed system. On the dataset of damaged vehicles, classifiers are trained.

There aren't many publicly accessible datasets for vehicle damaged photographs with labelling as automotive damaged assessment is a specialised field. The tough thing of model development is to use tiny datasets. When the tiny dataset is insufficient to train a Classification model, like our case, This issue can be resolved by manually gathering and annotating data from the Online platform and using it as data augmentation.In this project we perform annotation

The key concern is also cutting down on model training time. It can take a traditional Network model a lot of time to undertake digital image classification tasks and find the appropriate weight values over numerous forward and backward

iteration. Leveraging GPUs, one such operation can take days or even weeks to finish. However, pre-trained CNN models can greatly reduce on model training time,

Pre-trained CNN models could be customized and utilised as a feature extractor. But, because the diversity is intense, these frameworks are quite difficult to comprehend. In order to focus on the effectiveness of different hyper-parameters and explore ideas to modify them, Jeffrey proposed a method.

Similar to this, when the evaluation algorithm is applied towards the correct predicted values, the learning method of kepoch based model training evaluation can yield the finest learner parameters. In order to tackle the overfitting conflict and the impact of hyper-parameters, normalisation can be adjusted in this way.

We encountered that transfer learning functions most effectively. We proposed an operational pipeline that would first extract the vehicle's damaged area from every image before passing it on to the training classification model in order to enhance the system's robustness. As previously mentioned, we also use YOLO-v5 for detecting objectives.

II. LITRETURE SURVEY

This study introduces a deep learning-based algorithm for identifying car appeal damage and a way for evaluating the algorithm's model. An approach using computer vision is presented, with an evaluation system. A damage detection approach is developed using the specific target object identification of the four primary types of damage: scraped, deform, crack, and rupture. Mask R-CNN is employed as the fundamental framework. newly developed bounding box regression To mutually acquire the ambiguity of bounding box regression and locating, the Institute of Megvii's loss KLloss approach is applied.

In order to prevent overfitting and understand more basic characteristics, this study proposed models that were pre-trained on a vast and varied dataset. utilising CNN models that have been pre-trained on the ImageNet dataset, among other methods. Additionally, a pipeline for recognizing the damage in vehicles is proposed by combining the processes of recognition and classification in order to identify the location of damage using state-of-the-art YOLO object detector. In this study, VGG16 and VGG19,OpenCV method three deep learning-based algorithms, were used for car damage assessment and detection in datasets. The algorithms locate the damaged car component, identify its place, and then analyse its severity. Notice how domain-specific pre-trained CNNs were trained over an ImageNet dataset-perform first, and then fine-tune them. Applying transfer learning to already-trained VGG models after that. According to their findings, VGG19 performs better than VGG16. Observed that the outcomes of using transfer learning and L2 regularisation can be more effective than those of fine-tuning after analysing and putting models into practise.

On a smaller Dataset, this study's application of the Mask RCNN technique has also shown useful results. They employ the "detectron2" pre-trained model of Facebook's free and open-source Library. In the paper, vehicle damaged areas are detected and segmented using the Mask RCNN approach. Reduced segmentation and slower detection speed are problems that arise as a result of the complication of automated damage recognition and detection. In the context of this study, the Mask RCNN is used, and a model for identifying and segmenting a vehicle's damaged area following an accident is suggested.

This paper suggested a method for producing reliable features by precisely finding the faults and using YOLO to detect the damage zones as well as OpenCv techniques. By fusing data from many sensors, Gontscharov.al attempts to tackle the issue of vehicle damages. By using a higher resolution vision system with many cameras, Keyence Vision suggested an industrial solution for hail damage to cars.

III. PROPOSED SYSTEM

The proposed approach was created with the purpose of identify different types damages present in the image of car and classify the damage based on severity of the damage and ensure the insurance firms with accurate damage assessment.

1. System Design

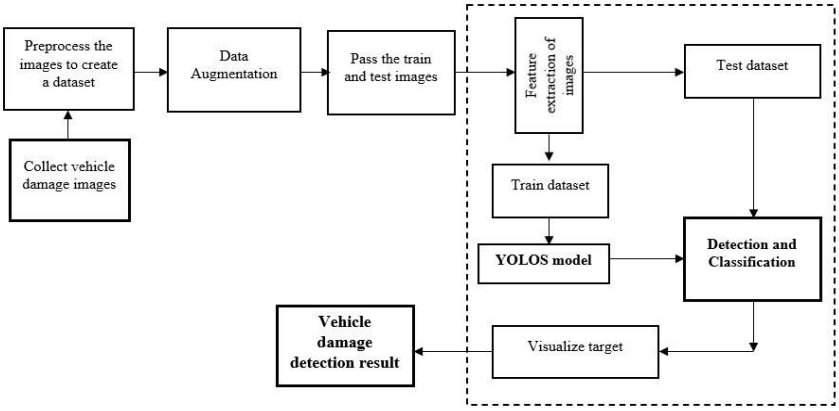

Fig 1 System flow representation

2. Assembling Our Dataset

For the classification of vehicle damage, there are relatively few publicly available datasets that are not very good. We constructed our own dataset by gathering images using Internet sites. Since it is fairly difficult to apply typical computer vision techniques without a vast and varied dataset. As indicated in Table 1, we manually selected and classified images into seven forms of classification that are usually encountered.

Classes	Train size	Aug. train size	Test size
Dent	150	450	30
Scratch	112	330	22
Large Dent	146	430	25
Glass shatter	104	310	25
Head light broken	57	150	14
Red light broken	39	110	11
Severe Damage	256	768	30
Mild Scratch	347	1010	50
Dislocation	115	300	20
Tear	200	630	30
Front side	200	550	35
Door side	150	400	20
Rear side	100	250	15

Table 1 Dataset Description

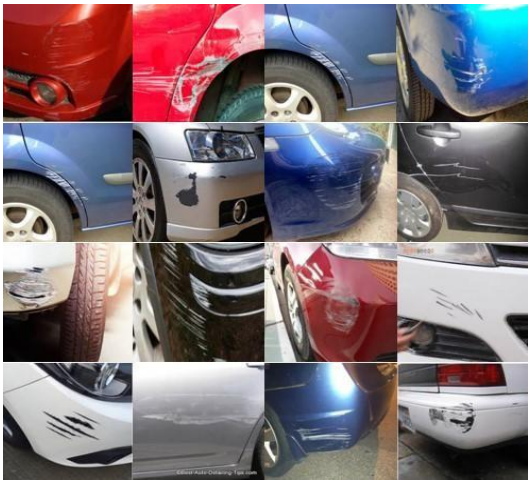

Fig 2 Different Damage Type Samples

We also gathered pictures that does not belongs to any of the class of damages. Fig 2 displays some example pictures from our dataset, with each representing a distinct sort of damages (dent, scratch, dislocation, glass shatter, head light broken, red light broken, severe damage, mild scratch and tear, on front, door and rear side). We manually labelled various types of damaged with bounding boxes for the purpose of classification. We have used Visual Object Tagging Tool (VoTT) which is a open source annotation tool which is used for image annotation. And is developed by Microsoft.

3. Data Augmentation

For developing additional, altered data from the original set, a technique known as data augmentation can be utilized to purposely increase the volume of a training set. Using DA is an excellent idea if you just want to avoid overfitting, the training dataset seems to be too small, or whether you need to get more efficiency out of your model. We can make a number of modifications to the original data. Such as shape alterations, Chromaticity manipulations, Core filtration, Erasing randomly and Image fusion.

By using Augmentor, Albumentations, Imgaug and Imgaug Libraries, We perform data augmentation to enhance the dataset size in order to address some conflicts, such as overfitting, that may arise when one training dataset on a smaller dataset. By applying brightness between -25 percent and -25 percent, exposure between -20 percent and -20 percent, and horizontal flip, we increased the dataset to around 3x. We divided the data randomly into a 4:1 ratio for classification, using 80% of the data to train and 20% for test.

4. Transfer Learning

Transfer learning, it has a great reduction on identification tasks when we got limited dataset, can be used to avoid overfitting on short datasets rather than building the CNN models from scratch. We employ Auto feature extractor from transformers to perform feature extraction to construct the data loader that will be used for train and to address our dataset with torchvision adoption of the COCO dataset genre.

5. YOLOS model

There are three key components to the YOLO detection.

- Backbone: A CNN model serves as the YOLO backbone, pooling picture pixels to create features at various levels of resolution. The Backbone is often trained and tested using an ImageNet-like classifying dataset.
- Neck: Before forwarding the outputs from the Convolutional feature layer to the classification head, the YOLO neck combines and blends them.
- Head: This segment of the model is responsible for class and bounding box predictions. It is directed by the three loss functions for category, enclosure, and objectness.

A variant of CNN models serves as the framework for extracting features from photos in all other YOLO models. Every one of the YOLO approaches develop the neck of the object recognition model on the CNN backbone, despite the fact that they have different opinions on how exactly this CNN is constructed.

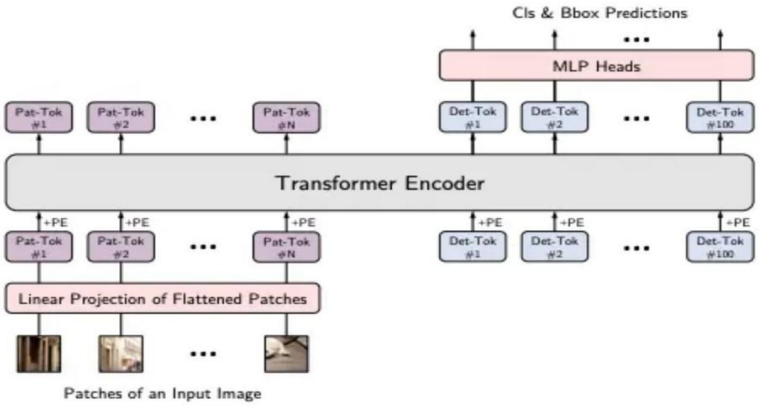

Fig 3 The YOLOS Network Architecture

Here we are using YOLOS model architecture. Instead of the conventional set of different tokens use for NLP, YOLOS examines portions of a picture to create "patch tokens." On the right, there really are 100 detect tokens which are trainable embeddings that every search for a certain object in a picture.

Additionally, YOLOS features a detector section of the networks that links a produced detection representations into predictions for classes and boxes. As a "You Only Look Once" model, YOLOS is characterised by the fact that it only looks at the sequence of picture patches once. Besides that, there are no other similarities between the YOLOS architecture and earlier YOLO Models.

By importing a significant amount of functionality from transformers and pytorch-lightning, we built up the training configuration. We begin by importing the complete model specification. We selected the 117 megabytes yolos-small model. Yolos-tiny, a 24 mb model, is another option for quicker training and inference but with lower accuracy. Then, we enclose our model in a pl.LightningModule training loop. In the end, we initialize our model and train the model with 30-40 epochs.

6. Testing

The proposed method was to make the classification on the vehicle damage severity to assess the damage for vehicle insurance claim. We performed damage detection and classification of vehicle from our model using Yolov5. It provides a clear visual representation of the vehicle's damage. The region of damage is highlighted in the final image using a bounding box over the region after processing.

III. RESULTS AND ANALYSIS

Despite the modest size of the dataset employed, the results were fairly accurate. To ensure that the dataset format is compatible with the coco dataset format of the model using the JSON file data, we first run the coco identification method through random pictures from the valid dataset. The outcomes are displayed below. Fig 4 depicts coco detection algorithm.

Fig 4 Samples of valid dataset with annotation.

Then, in order to check the detection process, we selected a random image from the test dataset. The outcomes were particularly good. The image samples are is displayed below. Following successful completion, we divided the damage into ten types. Dent, Scratch, Large Dent, Glass Shatter, Head Light, Red Light, Severe Damage, Mild Scratch, Dislocation, and Tear. we classify location into three categories Front side, Door side and Rear side. Because there are fewer photos in the training dataset, as seen, the precision and recall of the "Head light broken" and "Red light broken" classes is comparably lower.

Fig 5 Results of vehicle damage detection and classification.

7. Code

```
import cv2
import numpy as np

def detect_car_damage(image_path, output_path="output_detected_damage.jpg"):
    """
    Enhanced car damage detection using traditional computer vision techniques
    (Headless version without imshow displays)

    Args:
        image_path (str): Path to the input image
        output_path (str): Path to save the output image
    """
    # Load image
    image = cv2.imread(image_path)
    if image is None:
        raise FileNotFoundError(f"Image not found at: {image_path}")

    original = image.copy()
```

```
# Resize while maintaining aspect ratio
height, width = image.shape[:2]
max_dim = 640
if height > width:
    new_height = max_dim
    new_width = int(width * (max_dim / height))
else:
    new_width = max_dim
    new_height = int(height * (max_dim / width))
image = cv2.resize(image, (new_width, new_height))
original = cv2.resize(original, (new_width, new_height))

# Convert to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply CLAHE for better contrast
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
enhanced = clahe.apply(gray)

# Apply Gaussian Blur
blurred = cv2.GaussianBlur(enhanced, (5, 5), 0)

# Edge Detection using Canny with automatic thresholding
sigma = 0.33
v = np.median(blurred)
lower = int(max(0, (1.0 - sigma) * v))
upper = int(min(255, (1.0 + sigma) * v))
edges = cv2.Canny(blurred, lower, upper)

# Morphological operations to enhance edges
kernel = np.ones((3, 3), np.uint8)
dilated = cv2.dilate(edges, kernel, iterations=1)
closed = cv2.morphologyEx(dilated, cv2.MORPH_CLOSE, kernel)

# Find contours
contours, _ = cv2.findContours(closed.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Filter and process contours
damage_detected = False
for contour in contours:
    area = cv2.contourArea(contour)
    if area > 300:  # Minimum area threshold
        x, y, w, h = cv2.boundingRect(contour)

        # Calculate solidity to filter out irregular shapes
        hull = cv2.convexHull(contour)
        hull_area = cv2.contourArea(hull)
        solidity = float(area) / hull_area if hull_area > 0 else 0
```

```
            if solidity > 0.2:  # Filter very concave shapes
                damage_detected = True
                cv2.rectangle(original, (x, y), (x + w, y + h), (0, 0, 255), 2)

                # Add text label
                label = f"Damage ({area}px)"
                cv2.putText(original, label, (x, y-10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

    # Add overall result text
    result_text = "Damage Detected!" if damage_detected else "No Significant Damage Found"
    text_color = (0, 0, 255) if damage_detected else (0, 255, 0)
    cv2.putText(original, result_text, (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX, 0.8, text_color, 2)

    # Save results
    cv2.imwrite(output_path, original)
    print(f"✅ Results saved to: {output_path}")
    print(f"🔍 Detection Result: {result_text}")

if __name__ == "__main__":
    # Example usage
    detect_car_damage("C:/Users/Admin/Downloads/car2.jpg")  # Your image path
```

8. Output

IV. CONCLUSION

In order to offer an effective way for automating automobile damage insurance claims, we combine data and AI in this article. In this work, we present a method for classifying and identifying vehicle damage. We employed deep learning techniques for the damage categorization process and manually gathered a variety of Online datasets by web crawling on search engines Such as Google.

We achieved the present state of the art in car damage categorization by ease by blending transfer learning using transformers for feature extraction. Using the YOLO architecture, we are indeed effective in locating the damaged area and identify the type of the damage and severity the of the car. Despite the fact that our dataset is very tiny in comparison to other deep learning datasets, successful results were obtained.

By,
S.SHYAM SHANKAR - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)
G.SIVA KUMAR - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)
SB.GOKUL - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)
P.DHANUSH - (B.TECH ARTIFICIAL INTELLIGENCE & DATA SCIENCE- III YEAR)