Car Damage Detection using Detectron2 and Mask R-CNN

Nandhana C Reghu
Master of Computer Applications
Amal Jyothi College of Engineering
Kottayam, Kerala
nandhanareghu8333@gmail.com

Ms. Navayamol T K
Master of Computer Applications
Amal Jyothi College of Engineering
Kottayam, Kerala
navyamolkt@amaljyothi.ac.in

Abstract: Car damage detection plays a pivotal role in the automotive industry, insurance sector, and vehicle maintenance services. Car damage inspections done by hand are labor-intensive and prone to human mistakes. In the paper, a cutting-edge instance segmentation model called Mask R-CNN is used to present a novel method of automating the identification of car damage. Both training and assessment are conducted using the COCO automobile damage detection dataset, which consists of annotated photos of damaged cars. The primary objective is to achieve accurate and efficient car damage detection, ultimately streamlining the assessment process across diverse automotive contexts. The architecture of Mask R-CNN, a deep learning model that can recognize and separate damaged regions in automotive photos, is examined in this work. The model is trained using a carefully curated dataset, and the impact of augmented data on model performance is investigated in detail. Results from the experiments underscore the effectiveness of this approach, particularly in terms of achieving high accuracy in car damage identification. The study shows the potential for effective computer vision applications and tools like the open-source deep learning framework Detectron2 in the automobile sector in addition to contributing to the field of automated car damage assessment.

Keywords-- Mask R-CNN, Car Damage Detection, Detectron2, Instance Segmentation, COCO Car Damage Dataset, Data Augmentation, gradio

I. INTRODUCTION

In today's fast-paced automotive industry, the automated detection of car damages is crucial for efficient assessment and repair processes. The manual inspection and assessment of car damages can be time-consuming, error-prone, and often subject to human biases. Integration of state-of-the-art technology with computer vision has emerged as a crucial answer to these problems. The paper explores the utilization of Mask R-CNN, an advanced instance segmentation model, in conjunction with Detectron2, an open-source deep learning framework developed by Facebook AI Research, to automate car damage identification. Through the use of Detectron2 for data preprocessing and evaluation, and model training on a well-defined dataset, this study highlights the potent combination of state-of-the-art technologies and cutting-edge computer vision approaches. The outcomes demonstrate the model's precision supported by captivating visualizations of successful damage detection, underscoring the transformative potential of technology in the automotive sector, with Detectron2 serving as a key element in this development in technology.

II. LITERATURE REVIEW

"Object Detection and Instance Segmentation: Mask R-CNN" by Kaiming He (2017) provide instance segmentation with the ability to precisely identify object instances and simultaneously define object boundaries. This marked a significant leap in the development of car damage detection models.

"COCO Car Damage Detection Dataset" by L. Plenka (2020) have played a vital role in benchmarking and advancing research in this area. Contains the images of cars with damages and their annotations in COCO (Common Objects in Context) format. These datasets are crucial for training deep learning models, including those based on Mask R-CNN and supported by Detectron2.

"Detectron2" an open-source deep learning framework developed by Facebook AI Research (2019) is a next generation library that provides state-of-the-art detection and segmentation algorithms. It is the replacement for both Mask R-CNN-benchmark and Detectron. It helps several computer vision research initiatives and Facebook production applications.

III. MOTIVATION

Addressing practical demands in the automobile sector is the driving force behind investigating vehicle damage detection using cutting-edge technologies such Mask R-CNN and Detectron2. Enhancing vehicle safety, expediting accident response times, and streamlining insurance claims are just a few of the useful advantages of this technology. Moreover, the power of computer vision and open-source frameworks like Detectron2 makes it accessible and adaptable for various applications, aligning with the current trends in artificial intelligence and computer vision.

IV. METHODOLOGY

The methodology employed in this research leverages advanced computer vision techniques, primarily Mask R-CNN, and utilizes the Detectron2 framework to implement car damage detection. The process can be summarized into the following key steps:

A. Data Collection:

An extensive dataset of car images with associated annotations in COCO format is collected. This dataset

consists of images depicting car with various types and extents of damage.

B. Data Preprocessing:

The gathered dataset is carefully preprocessed. It is divided into training, validation, and test sets, ensuring that the model is trained on a diverse range of examples.

C. Model Selection:

The Mask R-CNN model is chosen for its superior instance segmentation capabilities, allowing precise identification and localization of both car and damage. The Detectron2 framework, an open-source platform built on PyTorch, is employed to streamline model training and evaluation. Pre-trained weights on a sizable dataset with their annotations in COCO format are used to initialize the model.

D. Model Training and Evaluation:

During training, the model learns to differentiate between different car parts and damage types by minimizing the instance segmentation loss. On the validation set, the trained model is assessed using metrics such as average precision (AP), which quantifies the model's precision in identifying and classifying damage.

E. Inference and Damage Assessment:

The trained model is used for inference on new, unseen car images, both for damage detection and instance segmentation. A detailed evaluation is provided by the categorization of discovered damages and the accurate delineation of their locations.

F. Integration and User Interface:

The trained final model is seamlessly integrated with a user-friendly Gradio interface. Users upload car images, and the model quickly processes them, displaying annotated images that highlight and categorize damage areas.

G. Model Deployment:

The final trained model is used in real-world situations including handling insurance claims, maintaining cars, and responding to accidents. Additionally, it may be incorporated into mobile apps for examinations while on the move.

IV. BUILD MODEL

Building a model for car damage detection using the Detectron2 framework with Mask R-CNN. While model build, user use the algorithm. The steps involved are:

A. Importing Libraries

```
from pycocotools.coco import COCO
import skimage.io as io
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import detectron2
from detectron2.utils.logger import setup_logger
import numpy as np
import os, json, cv2, random
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog, DatasetMapper, build_detection_train_loader
```

Detectron2 is an advanced open-source computer vision library developed by Facebook AI Research (FAIR). The model_zoo is a module in this library that provides access to a collection of pre-trained models for various computer vision tasks.

COCO (Common Objects in Context) is a widely used dataset format for object detection, segmentation, and captioning tasks. The pycocotools library is a set of Python tools that provide support for working with datasets in the COCO format.

B. Training:

Create evaluator and augmentation

```
# create trainer with evaluation and augmentation
class Trainer(DefaultTrainer):
    @classmethod
    def build_evaluator(cls, cfg, dataset_name, output_folder=None):
        if output_folder is None:
            output_folder = os.path.join(cfg.OUTPUT_DIR, "evaluator")
        return COCOEvaluator(dataset_name, ("bbox", "segm"), output_folder)
    @classmethod
    def build_train_loader(cls, cfg):
        augs = [
            T.ResizeShortestEdge(cfg.INPUT.MIN_SIZE_TRAIN, cfg.INPUT.MAX_SIZE_TRAIN, cfg.INPUT.MIN_SIZE_TRAIN_SAMPLING),
            T.RandomFlip(prob=0.5, horizontal=True, vertical=False),
            T.RandomRotation([-90, 90]),
            T.RandomLighting(0.5)
        ]
        mapper = DatasetMapper(cfg, is_train=True, augmentations=augs)
        return build_detection_train_loader(cfg, mapper=mapper)
```

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("dataset_train",)
cfg.DATASETS.TEST = ("dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 2
cfg.SOLVER.BASE_LR = 0.001
cfg.SOLVER.MAX_ITER = 800
cfg.SOLVER.STEPS = []
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
cfg.TEST.EVAL_PERIOD = 200
import shutil
try:
  shutil.rmtree(cfg.OUTPUT_DIR)
except:
  print('no folder found, pass')
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = Trainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

'Trainer' class is designed to work with Detectron2 and provides methods for building an evaluator to assess the model's performance and creating a data loader with data augmentations for training.

C. Evaluation and Inference Val-Test:

```
cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7
predictor = DefaultPredictor(cfg)
evaluator = COCOEvaluator("dataset_val", output_dir="./output")
val_loader = build_detection_test_loader(cfg, "dataset_val")
print(inference_on_dataset(predictor.model, val_loader, evaluator))
```

```python
def inference(listImg):
    for img in listImg:
        im = io.imread(img)
        outputs = predictor(im)
        v = Visualizer(im[:, :, ::-1],
                       metadata=val_metadata_dicts,
                       scale=0.5,
        )
        out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
        plt.figure(figsize=(50,50))
        plt.subplot(1, 2, 1)
        plt.axis('off')
        plt.grid(False)
        plt.imshow(out.get_image()[:, :, ::-1])
        masks = outputs['instances'].pred_masks.cpu().numpy().astype('uint8')
        total_area = outputs['instances'].pred_boxes.area().sum()
        plt.text(20, 40, 'Number of Damaged: {}\n Damaged Area: {} px^2'.format(len(masks),
        total_area), fontsize = 40, color = 'white', bbox = dict(facecolor = 'red', alpha = 0.5))
        plt.subplot(1, 2, 2)
        plt.axis('off')
        plt.imshow(im)
        plt.show()
imgVal = [d['file_name'] for d in random.sample(val_dataset_dicts, 6)]
```

The function accepts a list of image file locations, applies the object detection model to each image, overlays the model's predictions, and shows the results along with details about the damage that was found. It is a visual evaluation method for identifying damage to a car.

V. RESULT

```
[10/23 08:11:14 d2.evaluation.fast_eval_api]: Evaluate annotation type *bbox*
[10/23 08:11:14 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.00 seconds.
[10/23 08:11:14 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[10/23 08:11:14 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.00 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.067
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.177
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.053
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.133
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.044
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.042
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.146
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.146
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.154
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.150
[10/23 08:11:14 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|  AP   |  AP50  |  AP75  |  APs  |  APm   |  APl  |
|:-----:|:------:|:------:|:-----:|:------:|:-----:|
| 6.743 | 17.666 | 5.255  | 0.000 | 13.300 | 4.414 |
```

Fig.1. Result of Evaluating annotation type “boundary box (bbox)”

```
[10/23 08:11:14 d2.evaluation.fast_eval_api]: Evaluate annotation type *segm*
[10/23 08:11:14 d2.evaluation.fast_eval_api]: COCOeval_opt.evaluate() finished in 0.01 seconds.
[10/23 08:11:14 d2.evaluation.fast_eval_api]: Accumulating evaluation results...
[10/23 08:11:14 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.00 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.097
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.209
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.101
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.155
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.062
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.067
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.171
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.171
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.200
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.150
[10/23 08:11:14 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|  AP   |  AP50  |  AP75  |  APs  |  APm   |  APl  |
|:-----:|:------:|:------:|:-----:|:------:|:-----:|
| 9.703 | 20.922 | 10.129 | 0.000 | 15.496 | 6.248 |
```

Fig.2. Result of Evaluating annotation type “segmentation (segm)”

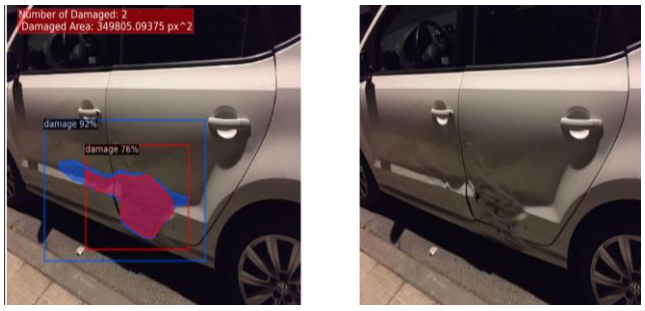

Fig.3. Result displaying of Inference Test.

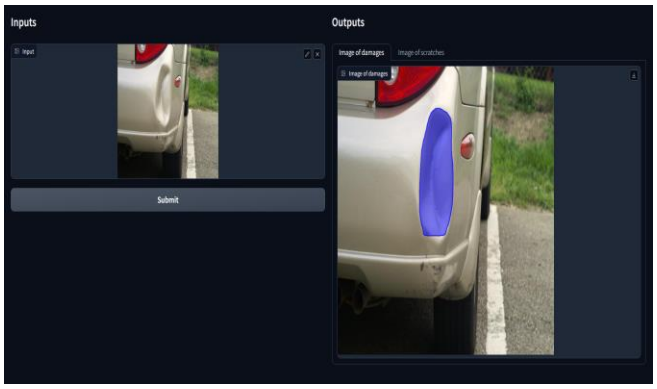

Fig.4. Result displaying Image with damage.

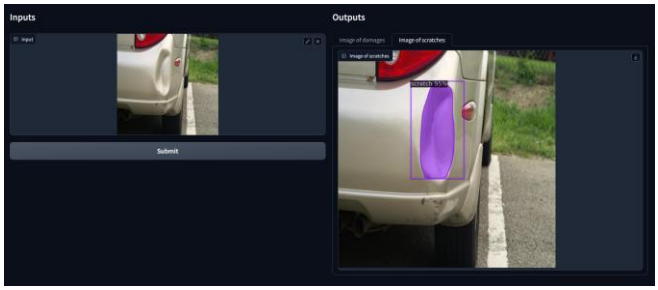

Fig.5. Result displaying Image with Scratches.

The implementation of Mask R-CNN with Detectron2 successfully identifies and highlights damaged areas in car images with impressive accuracy, ensuring precise damage assessment. In essence, it excels at pinpointing and categorizing damage and scratches in images, making it a powerful tool for enhancing decision-making and efficiency.

VI. CONCLUSION

The application of advanced computer vision techniques, specifically Mask R-CNN and the utilization of the Detectron2 framework, in the domain of car damage detection represents a significant leap forward in addressing crucial challenges within the automotive industry. To assure vehicle safety, accelerate accident reactions, and simplify insurance claims, it makes it possible to analyze automobile damage quickly and accurately. Additionally, it can improve user experiences by giving car owners the ability to evaluate damage and communicate with repair providers via mobile apps. With the increasing accessibility of computer vision and open-source frameworks, these innovations are within the reach of researchers, developers, and businesses, promising safer and more efficient solutions across various industries.

VI. REFERENCES

[1]. Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick: Mask R-CNN, ICCV 2017.

[2]. Q. Zhang, X. Chang and S. B. Bian, "Vehicle-Damage-Detection Segmentation Algorithm Based on Improved Mask RCNN," in IEEE Access, 2020.

[3]. K. He, G. Gkioxari, P. Dollar and R. Girshick, "Mask R-CNN", Proc. IEEE Int. Conf. Comput. Vis. (ICCV), pp. 2980-2988, Oct. 2017.
[4]. Abhishek, Allena Venkata Sai, and Sonali Kotni. "Detectron2 object detection & manipulating images using cartoonization." Int. J. Eng. Res. Technol. (IJERT) 10 (2021).
[5]. Veit, Andreas, et al. "Coco-text: Dataset and benchmark for text detection and recognition in natural images." arXiv preprint arXiv:1601.07140 (2016).
[6]. Hafiz, Abdul Mueed, and Ghulam Mohiuddin Bhat. "A survey on instance segmentation: state of the art." International journal of multimedia information retrieval 9.3 (2020): 171-189.