Damage Detection for Car Images

Can Erozer, Ozgur Sen

December, 2024

Abstract

Car-sharing companies need efficient solutions to identify vehicle damage after rentals, minimizing disputes and costs. Our project leverages advanced deep learning models, including segmentation tools and CNNs, to analyze post-rental images, overcoming challenges like background noise and varying image quality. This approach not only improves damage detection accuracy but also provides a scalable solution for the industry, with a fully reproducible codebase available for future developments.

Introduction

Car-sharing companies need a reliable way to determine whether a car has been damaged after a user returns it. If damage is detected, they can attribute the repair costs to the responsible user rather than bearing the expense themselves. Developing a model that analyzes photos taken at the end of each rental to assess vehicle damage would help car-sharing companies efficiently monitor the condition of their fleet and manage costs.

Our approach leverages state-of-the-art deep learning techniques to tackle this problem effectively. We utilize *Supervision by Roboflow* and *Segment Anything by Meta* for advanced image preprocessing to isolate key visual features. For training the damage detection model, we employ a Convolutional Neural Network (CNN), a proven architecture for image analysis tasks. Additionally, we integrate *Torch-Grad* to analyze predictions and refine model accuracy.

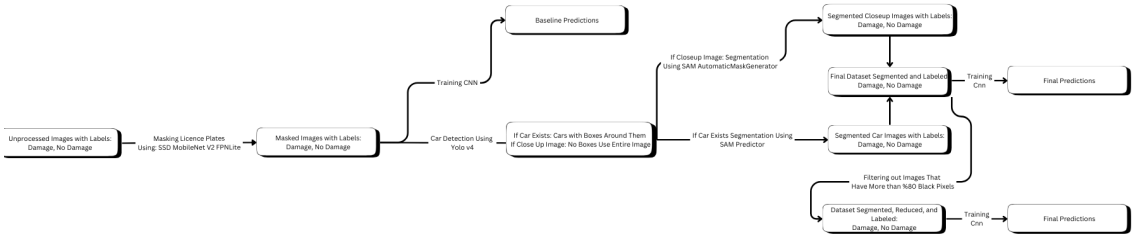
To ensure the reproducibility of our work, the complete codebase, datasets, and documentation are available in our GitHub repository: `https://github.com/ozgursen9/DS542_Final`. The repository includes clear instructions for setup and implementation, enabling others to replicate and build upon our results.

Related Work

This section provides an overview of the key models that we have utilized in the project.

1. **SSD MobileNet V2 FPNLite 320x320**[1]:
   A lightweight convolutional neural network designed for object detection, pre-trained on the COCO dataset. Its feature pyramid network (FPN) enhances its ability to detect objects of varying scales efficiently, which makes it suitable for tasks like detecting and masking licence plates.
2. **YOLOv4 (You Only Look Once, version 4)**[2]:
   A state-of-the-art real-time object detection model to detect objects and draw bounding boxes.
3. **Segment Anything Model (SAM)**[3]:
   A segmentation framework by Meta AI designed to handle both prompted and automatic segmentation tasks. SAM provides robust mask generation capabilities for various image analysis scenarios:
   - **SamPredictor**: For prompted segmentation, guided by specific inputs like bounding boxes.
   - **SamAutomaticMaskGenerator**: For generating masks across an entire image without specific prompts.

Approach (or Methodology)



Masking License Plates

When we asked the company in Turkey that provided us with this data set, they wanted us to mask the license plates of the cars. This is requested due to the risk of exposing personal information of the vehicle owners.

That is why before sharing this dataset to anyone, including putting it in our GitHub repository, we tried to find a successful model that does this task. But not every license plate detection model would be suitable for our dataset. This is because not every image contains a car image with a license plate. So, we needed a model that first detects if there

is a license plate on the car and, if it thinks there is one, then it should find the location of it along with providing its bounding box. There was no such custom-made model.

So, we found an existing model that gives a score based on its confidence in its license plate detection prediction. We used that score to detect if there is a license plate on the car. We set a threshold according to samples we made: In one of our samples there are car images with license plate and in the other one we have car images without license plate. According to the overall scores yielded upon running the model with these samples, we set our threshold. And then we made license plate detection using our model on the images that has a score above the threshold.

We were not able to do the masking on every car that have license plate on it. But, that wasn't our objective. Instead, our goal was to mask the majority of license plates of the cars accomodate in our dataset due to the ethical reasons and request of the dataset provider.

This model uses the TensorFlow and Easy OCR libraries. In order to train this model, we used a Kaggle dataset named "Car License Plate Detection" [1].

Here is an example of the masking we did:

Original Image

License Plate Detected

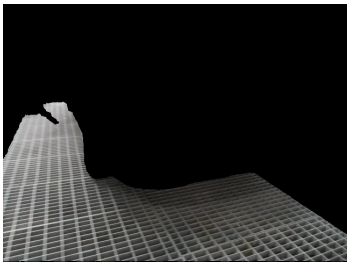
License Plate Masked

Vehicle Detection and Segmentation

If you take look at the "Baseline Model Prediction Visualizations" under the "Evaluation Results" section, you can notice that our baseline model makes miss-classifications due to the noises in the background. These noises are other cars that appear in the background or non-smooth surfaces that resemble to a damage. In order to get rid of the noises that cause our model to make wrong predictions, we decided to get rid of the background.

One way of doing this is segmenting the car object in the image. We decided to use the Meta AI's state-of-the-art segmentation model called Segment Anything (SAM). This model makes an instance segmentation which means it segments every distinct object on the image. So, with the assumption that the segmentation with the largest area corresponds to a car, we only segmented the object that has the largest segmentation area. Here is one interesting result when we gave some images to SAM:

Original Image



Bad example of segmentation

After seeing some results that are similar to this example above, we decided that we need to use a vehicle detection model that gives us a box that bounds the car. In order to do that, we used the yolov4 model. Here is an example of the output of the yolov4 model:

Figure 1: An Output of Yolov4 Model

After giving the coordinates of the bounding box of the car to SAM, it gave us a better segmentation result:

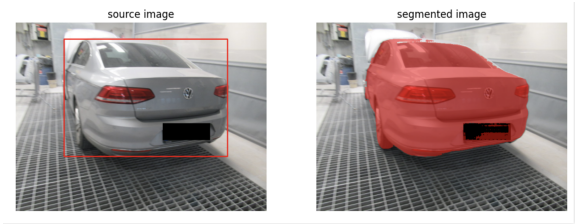


Figure 2: A Better Segmentation by SAM with box prompting

Elimination Process

Even though we used some precautions to get a good segmentation result, we still get some undesired segmentation results. But, this doesn't stem from the SAM model, but from the Yolov4 model. Yolov4 model successsfuly detects the cars in an image. And we only get the detection that has the highest score. There might be other cars in the background, and our vehicle detection model can detect all of the cars in the image. We made an assumption that the detection with the highest score is the detection that is the main object of the image: the car we want to do damage detection on. But, it turned out that our assumption doesn't hold all the time. Here is one example where our assumption is wrong:

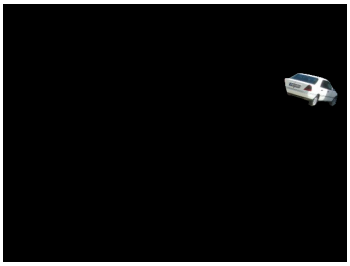


Figure 3: Here is the case where our assumption is incorrect

And training our CNN model with a lot images that looks like this resulted us to a get bad validation accuracy score (see the results and discussion for "model trained with segmented images" below).

In order to get rid of images like this in our training set like this we decided to do an elimination process. We basically calculated the percentages of black pixels in each image. And if that percentage is above to the threshold value we set, then we removed those images from the training set. Training our model with better segmented images boosted our validation accuracy results. See the last model for further discussions.

CNN Model Architecture

The architecture comprises two main components: a feature extraction network and a classification network.

Feature Extraction

- The feature extractor consists of three convolutional blocks, each followed by Batch Normalization and ReLU activation for stability and non-linearity.
- The blocks use:
  - Strided convolutions to reduce spatial dimensions.

  - MaxPooling layers for further downscaling and feature emphasis.
  - An Adaptive Average Pooling layer ensures the output feature map size remains fixed (4x4), regardless of input dimensions, enhancing flexibility in handling varying image sizes.

Classification

- The flattened feature maps are passed through a fully connected layer with 64 neurons, incorporating a Dropout layer (rate: 0.5) to mitigate overfitting.
- A final fully connected layer outputs predictions for two classes: damaged or undamaged.

Datasets

The dataset used in this project was provided by a car-sharing company and consisted of real-world images. It consisted of 10000 total images evenly split between 5000 depicting damaged cars and 5000 depicting undamaged cars. The raw nature of the dataset, captured in real-world scenarios, resulted in low image quality. Many images featured unusual angles and complex backgrounds, often including other vehicles, which sometimes caused our segmentation model to focus on the wrong car. Additionally, some images of damaged cars were extreme close-ups, emphasizing the damage itself. These real-world challenges significantly increased the difficulty of accurately classifying cars as damaged or undamaged. All of the data can be found at: `https://drive.google.com/drive/folders/1znDzs-fC9ysmS673rcSSWKB2hgRzb7Qq?usp=drive_link`

Figure 4: Car in the background example from dataset

Figure 5: Close-up image examples from the dataset.

Evaluation Results

Baseline Model

The baseline model performed surprisingly well. The early stopping mechanism was triggered at epoch 22, achieving the following results:

- **Training Accuracy:** 94.0%
- **Validation Accuracy:** 85.1%
- **Training Loss:** 0.1440
- **Validation Loss:** 0.4609
- **Damaged Accuracy:** 93.43%
- **No Damage Accuracy:** 76.06%

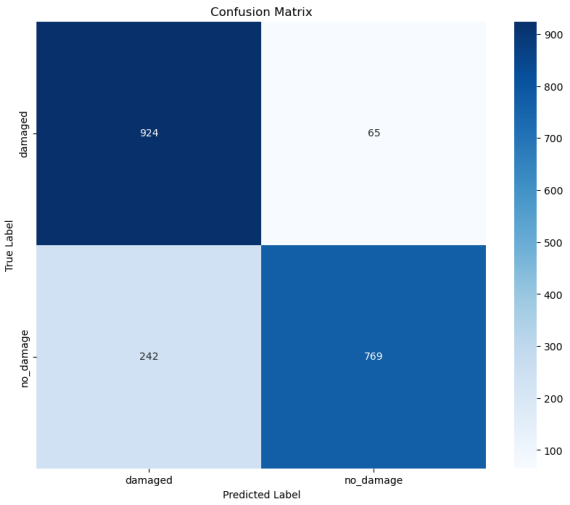


Figure 6: Baseline model confusion matrix.

Baseline Model Prediction Visualizations: GradCam was used to visualize the areas influencing the CNN's decisions. For cases where the model correctly predicts damage, the visualizations show that the network focuses on the damaged areas, demonstrating meaningful reasoning. Conversely, in instances where the model incorrectly predicts no damage, the attention often shifts to background details instead of the car itself. Lastly, for images misclassified as damaged, the decision is typically influenced by regions around the headlights which is an interesting pattern.

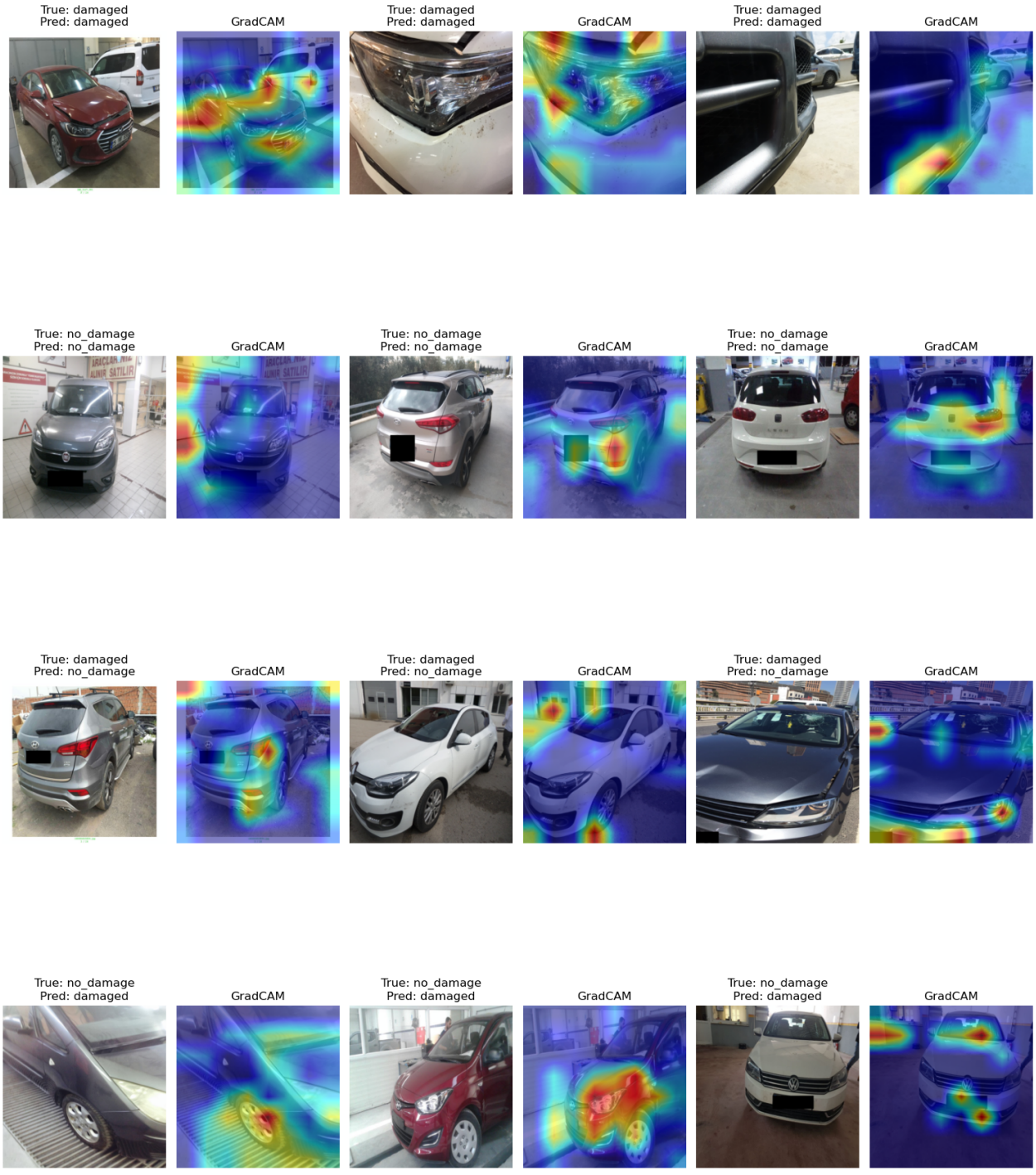


Figure 7: Baseline model predictions.

Model Trained on Segmented Images

The model trained on segmented images performed worse than the baseline model. This decline in performance was caused by the reduction in data quality in the segmented images. In some cases, the SAM model segmented background cars or other irrelevant features instead of the primary vehicle, leading to suboptimal training data. The early stopping mechanism got triggered at epoch 27 and this model achieved the following results:

- **Training Accuracy:** 84.9%
- **Validation Accuracy:** 78.8%
- **Training Loss:** 0.32 %
- **Validation Loss:** 0.52 %
- **Damaged Accuracy:** 78.3%
- **No Damage Accuracy:** 79.3%

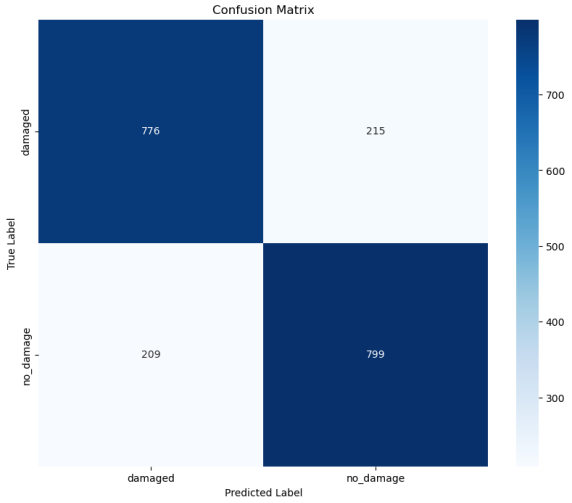


Figure 8: Segmented Model confusion matrix.

Visualizing Predictions for Model Trained on Segmented Data: GradCam was used to visualize the areas influencing the CNN's decisions. When the model correctly predicts damage, the network correctly focuses on the damaged regions, demonstrating meaningful reasoning. However, for images correctly classified as "no damage," there are instances where the network mistakenly focuses on areas that should have been segmented out. In cases of incorrect predictions—both for "damage" and "no damage"—the network

fails to concentrate on specific areas, instead focusing on broader regions. This suggests an inability to reason effectively in these scenarios.

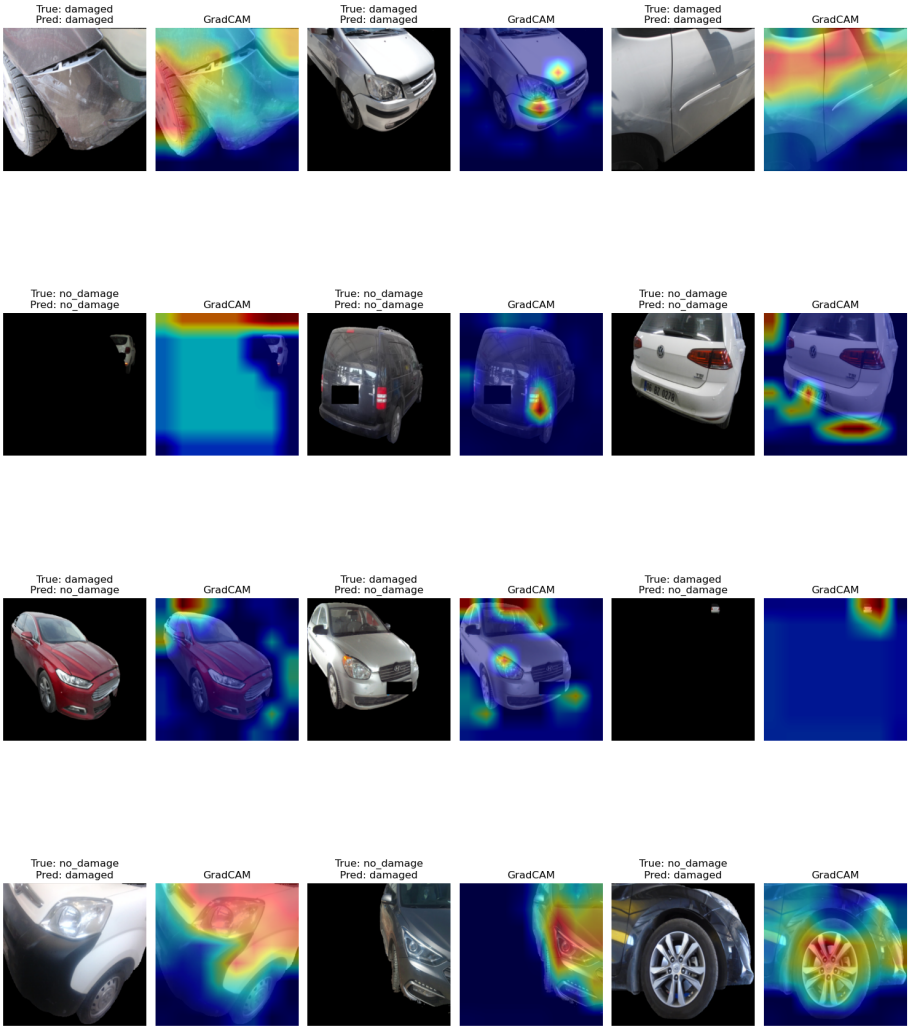


Figure 9: Segmented Model predictions.

Baseline Model with Reduced Dataset

This model was trained on the unprocessed (only license plate masking) images that are used in the segmented and reduced model. Despite being trained on a smaller subset of the original dataset, this model outperformed the one trained on segmented images. This highlights the extent to which the initial segmentation process failed to meet expectations,

ultimately degrading data quality and causing poorer performance than a baseline model trained with less data. The model reached its optimal performance at epoch 24, achieving the following results:

- **Training Accuracy:** 95.1%
- **Validation Accuracy:** 86.8%
- **Training Loss:** 0.12 %
- **Validation Loss:** 0.37 %
- **Damaged Accuracy:** 86.72%
- **No Damage Accuracy:** 86.82%

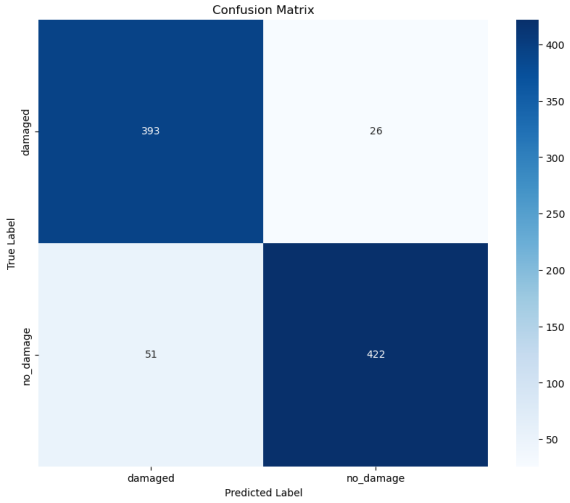


Figure 10: Segmented and Filtered model confusion matrix.

Final Model with Reduced and Segmented Dataset

This model outperformed the one trained on low-quality segmented images. By filtering out images with more than 80% black pixels, we minimized the inclusion of poorly segmented samples during training, which help to significantly increase the model's performance. The early stopping mechanism was triggered at epoch 29, and the model achieved the following results:

- **Training Accuracy:** 96.2%
- **Validation Accuracy:** 91.4%

- **Training Loss:** 0.08 %
- **Validation Loss:** 0.26 %
- **Damaged Accuracy:** 93.79%
- **No Damage Accuracy:** 89.22%

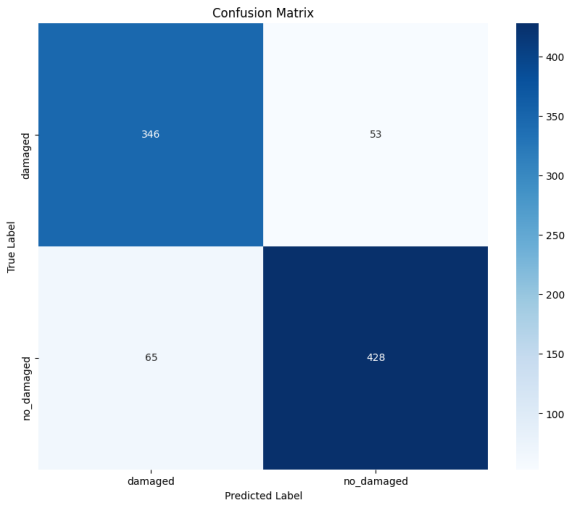


Figure 11: Segmented and Filtered model confusion matrix.

Visualizing Predictions for the Model Trained on Segmented Data: GradCam visualizations show that, in most cases of correctly classified as damaged, the model correctly identifies the areas where damage is present with few occurrences of focusing on black pixilated areas. However, for instances correctly predicted as "no damage," the model often focuses on black-pixeled regions. Interestingly, in cases misclassified as "damaged," the model tends to focus on the car's tires, indicating a recurring pattern in its decision-making process.

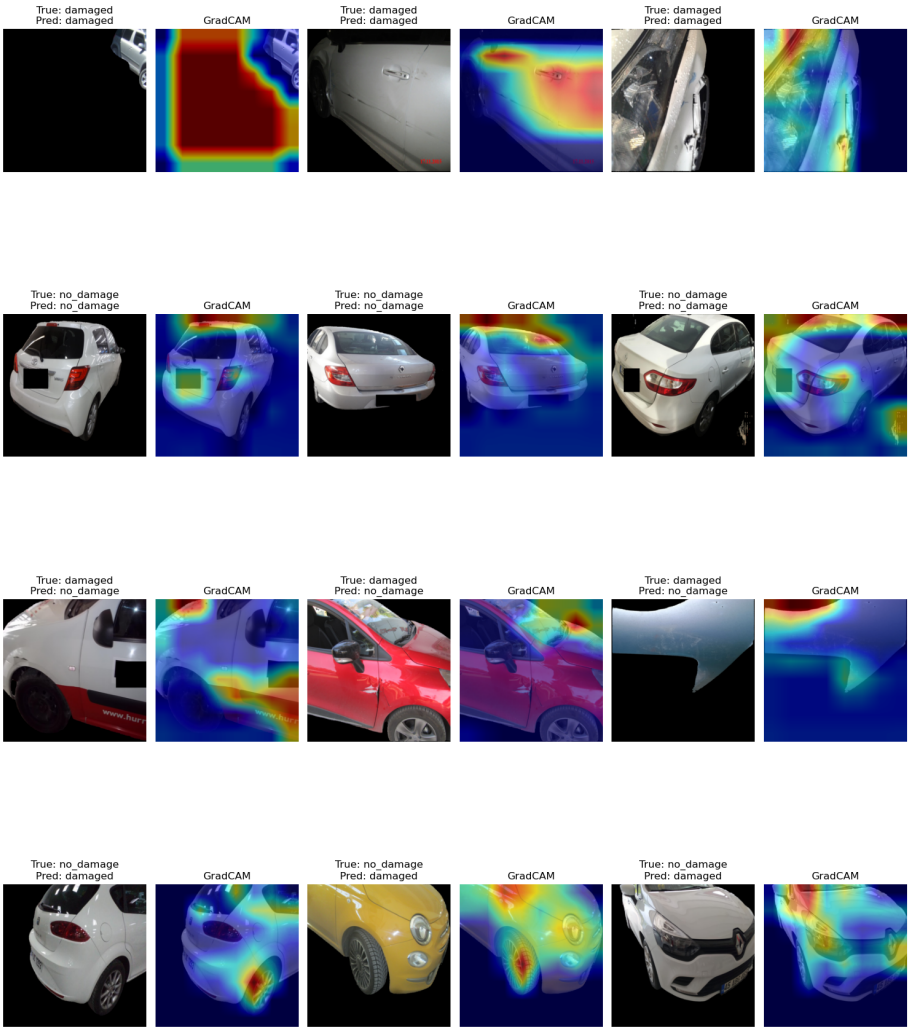


Figure 12: Segmented Model predictions.

Conclusion

This project successfully implemented advanced deep learning techniques to address the challenge of detecting vehicle damage in car-sharing scenarios. Our methodology integrated state-of-the-art models, including YOLOv4 and Segment Anything (SAM), for image segmentation and vehicle detection. While the baseline model performed robustly, achieving a validation accuracy of 85.1%, subsequent models with refined datasets highlighted the critical importance of data quality in model performance.

Through iterative refinement, the final model, trained on a reduced and segmented dataset, achieved the highest validation accuracy of 91.4%, demonstrating its capability to effectively classify damaged and undamaged vehicles. GradCam visualizations provided insights into the model's decision-making process, showcasing its ability to focus on relevant features when making predictions.

The future work for this model might involve creating a better algorithm to more accurately segment all car images, ensuring no training data is lost. With more data and better-segmented results, it is possible to achieve even higher accuracy and more reliable predictions.

References

[1] `https://www.kaggle.com/datasets/andrewmvd/car-plate-detection`

[2] `https://github.com/roboflow/supervision.git`

[3] Alexey Bochkovskiy et al *YOLOv4: Optimal Speed and Accuracy of Object Detection.* 2020.

[4] Alexander Kirillov et al *Segment Anything.* 2023. `https://github.com/facebookresearch/segment-anything.git`