applied sciences

MDPI

Article

Automated Car Damage Assessment Using Computer Vision: Insurance Company Use Case

Sergio A. Pérez-Zarate [1,2,*], Daniel Corzo-García [1], Jose L. Pro-Martín [1,2], Juan A. Álvarez-García [3], Miguel A. Martínez-del-Amor [2,4] and David Fernández-Cabrera [1]

[1] Valora Peritaciones, 41011 Sevilla, Spain; danielcorzog@gmail.com (D.C.-G.); jpro@us.es (J.L.P.-M.); tecnicos@valoraperitaciones.com (D.F.-C.)
[2] Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, 41012 Sevilla, Spain; mdelamor@us.es
[3] Department of Languages and Computer Systems, Universidad de Sevilla, 41012 Sevilla, Spain; jaalvarez@us.es
[4] SCORE Lab, I3US, Universidad de Sevilla, 41012 Sevilla, Spain
* Correspondence: sperez3@us.es

Abstract: Automated car damage detection using computer vision techniques has been studied using several datasets, but real cases for insurance companies are usually dependent on private methods and datasets. Furthermore, there are no metrics or standardized processes that describe the situation in which the company analyzes the customer's images, the models used for the inference, and the results. We perform extensive experiments to show that our proposal, an ensemble of 10 deep learning detectors based on YOLOv5, improves the state-of-the-art not only in terms of typical metrics but also in terms of inference speed, allowing scalability to thousands of instances per minute. A comparison with YOLOv8 is carried out, showing the differences between both ensembles. Furthermore, a dataset called TartesiaDS, labeled under the supervision of professional appraisers from insurance companies, is available to the community for evaluation of future proposals.

Keywords: object detection; car damage detection; car damage assessment; ensemble; supervised learning; deep learning

Citation: Pérez-Zarate, S.A.; Corzo-García, D.; Pro-Martín, J.L.; Álvarez-García, J.A.; Martínez-del-Amor, M.A.; Fernández-Cabrera, D. Automated Car Damage Assessment Using Computer Vision: Insurance Company Use Case. *Appl. Sci.* **2024**, *14*, 9560. https://doi.org/10.3390/app14209560

Academic Editors: Vincent Gan, Liangliang Jiang and Ke Yan

Received: 26 September 2024
Revised: 16 October 2024
Accepted: 17 October 2024
Published: 19 October 2024

1. Introduction

The automotive insurance industry must adhere to strict regulations that sometimes cause delays in procuring insurance for their customers. In addition, they are often faced with a number of challenges, such as providing an accurate assessment of a vehicle's damage and reducing claim handling costs. The damage assessment process typically involves manual inspection of numerous images, along with physical inspection by an automotive appraiser, which is time-consuming and costly for the company. The lack of efficiency in claims management usually results in losses for companies, often referred to as claims leakage.

In the U.S., it is estimated that the insurance industry loses approximately USD 18 billion annually due to claims leakage. Insurance companies need to adapt the use of technology to speed up processes and generate accurate estimates. One alternative is the use of artificial intelligence, which is being used in various industries to improve processes and operations. The main artificial intelligence solution to claims leakage is based on assessing car damage (https://www.insurancethoughtleadership.com/claims/how-stop-claims-leakage, accessed on 20 December 2022); based on the pictures that the customer sends to the company at the time of activating or purchasing an insurance policy.

Different types of object detectors [1–3] have been implemented and show that deep learning is capable of accurately detecting and classifying vehicle damage. Since deep

learning models require a sufficient amount of data to achieve acceptable performance, it is possible to turn to public datasets such as Wang et al.'s work [4], where 4000 high-resolution car damage images with more than 9000 cases of six damage categories are provided.

However, in order to perform vehicle evaluation and recognize damage as an expert would, it is necessary to have a dataset similar to that obtained by an appraiser, usually between six and eight images of the exterior of the vehicle, without occlusions and at an adequate distance. For this reason, in this work, we have focused on creating a dataset under the supervision of appraisers, which has 7896 images of vehicles with the presence of damage intended for the training process. Additionally, we contribute a public dataset of 108 images, again under the supervision of insurance appraisers, to allow for comparisons with future models. To the best of our knowledge, there is no public dataset of this type. Another important feature about both datasets is that the photo acquisition was guided by a web app-based process that ensures very uniform shooting angles and distances to the picture target throughout all the images.

We should also take into account the requirements of insurance companies for a car damage detector:

1. The ultimate goal of the industry is to make the damage assessment process require as little supervision as possible, so very high precision values (i.e., low false positive rate) are preferred.
2. In terms of efficiency, we need bounded execution times, as well as a system that is easily scalable and parallelizable and able to respond to dozens of concurrent requests.

Based on our dataset, we propose an inference voting system based on single-stage YOLOv5 and YOLOv8-type detectors, applying ensemble learning and confidence threshold optimization. Our proposal succeeds in identifying vehicle damage types while maintaining the minimum number of false positives. Furthermore, we make a comparison with existing damage detection models and demonstrate a significant improvement in both performance metrics (Area under the ROC Curve (AUC), precision, and recall) and inference times, making our model suitable for scalable applications in insurance companies and reducing claims leakage.

In summary, the main contributions of this study are the following:

1. We propose an ensemble of YOLOv5 detectors and another of YOLOv8 detectors combined with confidence threshold optimization and new bounding box generation for the detection of car damage for insurance companies.
2. We compare our model with existing models and surpass the state-of-the-art in damage detection in both performance and inference time while keeping the number of false positives low.
3. We make publicly available the first test dataset labeled by insurance company vehicle appraisers called TartesiaDS available at https://github.com/tartesia/TartesiaDS (accessed on 13 October 2023).

The rest of this paper is organized as follows. Section 2 presents related work in object detection, car damage detection, and relevant datasets. Section 3 describes the proposal and methodology for ensemble learning and bounding box combination, including the confidence threshold optimization. Section 4 describes the experimental set-up, datasets used, and experiments carried out. Section 5 shows and discusses the results for each experiment, comparing our proposal with the state-of-the-art. Finally, Section 6 concludes this study and describes the future steps to be analyzed.

2. Related Works

This section reviews the state-of-the-art in object detection, vehicle damage detection, and existing vehicle damage datasets.

2.1. Models for Object Detection

Object detection has been evolving at an impressive pace, with two groups of detectors becoming known: two-stage detectors and single-stage detectors. Two-stage detectors are usually characterized by high accuracy, but they have major limitations in terms of speed. Fast RCNN [5], faster R-CNN [6] and cascade RCNN [7] are some of the most popular two-stage models used in the task of object detection. However, new models have emerged that surpassed the state-of-the-art, such as InternImage [8], which explores the use of deformable convolutions (DCN) [9] to address large-scale model training and M3I pre-training [10], which is a one-stage pre-training paradigm, integrating pre-training methods such as supervised pre-training, weakly supervised pre-training, and self-supervised pre-training. This is leveraged to train the InternImage model.

On the other hand, single-stage detectors have been presented as an alternative to two-stage detectors, since they can perform inference in a single step and can be executed in real-time. Some single-stage models that stand out are the SSD [11], where two techniques are introduced: the multi-reference technique and the multi-resolution technique. The multi-reference technique consists of defining a set of references in each location of the image to subsequently predict the detections based on these references. The multi-resolution technique allows for the detection objects of different scales in different layers of the network. Next, RetinaNet [12] is proposed with the goal of reducing the accuracy gap between single-stage detectors and two-stage detectors. The authors of RetinaNet found that the main problem was in the extreme imbalance present between the foreground and background classes at the time of training. To solve the imbalance problem, they created a new function called focal loss, which is based on a transformation of the standard cross-entropy loss that allows the detector to focus on misclassified examples during training.

Finally, to address real-time object detection, the YOLO family models are proposed [13–16], including Ultralytics (https://github.com/ultralytics/ultralytics, accessed on 16 October 2024) YOLOv8. Those models have allowed researchers to address problems such as human behavior analysis [17], autonomous driving [18], traffic assessment [19], multiple target tracking [20], medical diagnostic analysis [21], and detection of road defects by means of a light model applied in real time [22], characterized by high speed and accuracy.

In our approach, we will use an ensemble of 10 YOLOv5 models to achieve a high-precision model [23]. YOLOv8 gives us lower performance compared to YOLOv5, so we chose YOLOv5 as our base detector. Our intuition about the better performance of YOLOv5 with respect to YOLOv8 is that the former is better adapted to real scenarios. This has been shown in the literature [24,25], where YOLOv5 outperforms YOLOv8 in terms of recall in particular cases and additionally generates higher reliability by having a low standard deviation. However, we extend the use of our ensemble to YOLOv8 models for the purpose of comparing the performance of more recent models.

2.2. Damage Detection

To move from the domain of object detection to vehicle damage detection, it is only necessary to consider damage types as object classes. In [26], the authors chose the most common types of damage such as bumper dents, door dents, glass breakage, headlight breakage, scratches, and crushing. They experimented with various techniques to train CNNs, such as direct training and transfer learning. In [2], the authors opted for the approach of dividing damage assessment into three subtasks: identification, localization, and severity level. By using two CNN models, VGG16 and VGG19, they managed to achieve an accuracy of 95.22% in identification, 76.48% in localization, and 58.48% in damage severity level.

In [27], the authors divide their implementation into two main parts. The first one consists of verifying if the vehicle in an image is damaged, then two additional models are used to obtain the location and severity level of the damage. The second part consists of using mask RCNN [28] to mask the damaged region. The authors compared various

models such as VGG16, VGG19, Resnet50, and InceptionV3, and VGG16 provided the best results.

In [29], the authors opted for a one-stage implementation to mark the damage box and identify its type. The model they used was YOLOS [30], characterized by the use of transformers. On the other hand, in a more recent work [31], the authors opted for the implementation of a YOLOv5-type detector trained on a proprietary dataset and succeeded in adapting it to the problem of damage detection. These previous approaches are similar to the one proposed in this work, as we use our own dataset, but such datasets usually lack complexity, such as the presence of small and irregularly shaped frictions. Therefore, these works allow us to identify the ability of one-stage models to address the problem of vehicle damage detection, but the need to implement a method to minimize false positives is evident.

On the other hand, in [32], an RCNN-type object detector [33] is used to identify the possible location of the damage and complement it with a RCNN mask-type detector to define the exact area of the damage. Similarly, in [33], a two-step workflow is applied: the combination of the segmentation of the parts of a vehicle, together with the segmentation of the damage. Two mask RCNN detectors were used to perform this workflow, and it should be noted that the use of segmentation models often requires high costs for a company to create a dataset focused on segmentation. Previous works [34,35] address damage quantification by implementing a rule-based system based on the detection of the damage level and structured information such as detected parts, price associated with a part, repair costs, etc. This approach complements object detection and takes a step towards damage quantification. The quantification of damage has an important limitation, as the costs and methods of repair work in different countries may differ significantly. The quantification of damages is therefore outside the scope of this research.

Finally, DCN+ [4], an improved version of DCN including multiscale learning and focal loss techniques, marks the current state-of-the-art in damage detection.

In the field of vehicle damage detection, the state-of-the-art uses one- or two-stage models but not ensembles, which have been used in other fields with considerable success [36,37]. Although in [38,39], the implementation of ensemble learning for object detection is proposed and guidelines are established, for the detection of damage to vehicles, it is necessary to implement additional steps such as searching for the best thresholds for each model and simplifying the non-maximum suppression algorithm by merging boxes. Based on the work mentioned above, the use of single-stage models has the risk of generating a false positive rate not allowed by insurance companies. For this reason, we propose the use of an ensemble of detectors from the YOLO family that can be run in parallel without increasing the detection time while reducing the number of false positives, which is essential for insurance companies. To determine the optimal number of detectors and their importance in the inference of boxes, a Particle Swarm Optimization (PSO) algorithm is employed. We used the outcome of this algorithm to combine the results using a new bounding box fusion method.

3. Methodology

To address the task of vehicle damage assessment, we propose the implementation of a system based on ensemble learning and YOLOv5, as visualized in Figure 1, allowing us to keep the number of false positives to a minimum and guaranteeing low inference times.

Our system is divided into three parts:

- Selection of object detectors;
- Optimization of confidence thresholds for each detector;
- Combination of predictions.

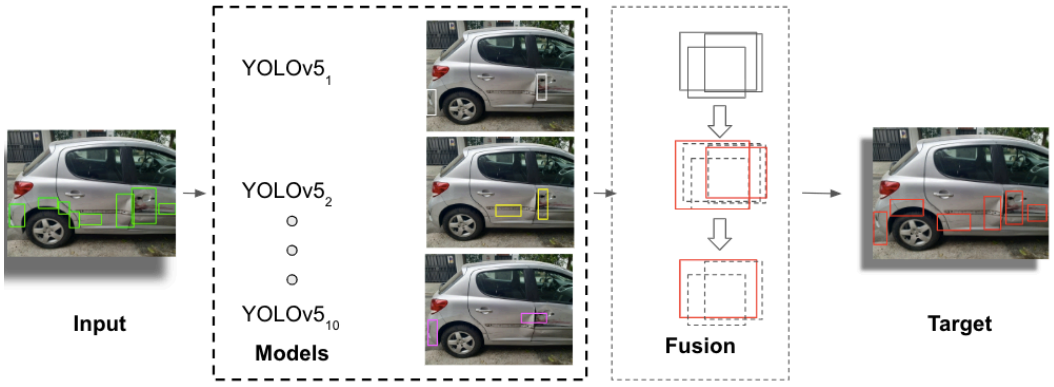


Figure 1. General scheme of our proposal for efficient damage detection.

3.1. Selected Models

In [4], the DCN+ model is used to locate and mask vehicle damage, but this model has inference times of 12 s in a production environment, which is too slow to be supported in our setting. The DETR-type models [40,41] have recently stood out for their high performance in detecting objects; however, they have high inference times and are currently not suitable for a production environment.

For this reason, we approach damage verification using a fast and robust detector. In this case, we have chosen YOLOv5 as our base detector, as it has inference times from 45 to 766 ms on CPU (https://github.com/ultralytics/yolov5, accessed on 25 September 2024), and its performance is usually close to its variants, such as all of the latest YOLO versions, from v6 to v8 on general character datasets. On our vehicle damage datasets, YOLOv8 gives superior performance in terms of AUC and mAP compared to YOLOv5; however, the false positive rate is higher with respect to YOLOv5. For this reason, we focused on training YOLOv5, which has been trained on the COCO dataset [42] as a base detector.

The YOLOv5 framework allows for fine-tuning of the model in different depths such as YOLOv5n (1.9 million parameters), YOLOv5s (7.2 million parameters), YOLOv5m (21.2 million parameters), YOLOv5l (46.5 million parameters), and YOLOv5x (46.7 million parameters). Although the differences between the previous models include the depth of the network and its number of parameters, we have found that we can take advantage of the joint inference of the aforementioned models because the behavior of each model varies enough with respect to the other models to locate new damage. However, they usually maintain similar predictions, allowing them to reinforce the presence of possible damage. The variability in the predictions can be seen in Figure 2. For this reason, we have chosen to train YOLOv5 from version n to x, with input sizes of (640 × 480) and (1280 × 960).

For each YOLO-type model, we define the hyperparameters as indicated in Table 1, highlighting the most important ones.

Table 1. YOLOv5 hyperparameters (lr stands for learning rate).

Hyperparameters	Values
Initial lr (lr0)	0.01
Final lr (lrf)	0.01
Weight decay	0.0005
Warmup bias lr	0.1
Mosaic	1
Flip left–right	0.5

We start with the basic data augmentation settings, such as the use of horizontal flips, changing the image saturation, scale changes, and the application of the mosaic method, which increases the diversity of the training set and improves the robustness of the model.

We were able to identify that the YOLOv5x, YOLOv5l, and YOLOv5s models achieved a balance between the number of correct predictions and false positives by adjusting their confidence values. This allowed us to generate more balanced interactions when interacting with other models.

To obtain an optimal number of models and avoid increasing inference times, we used the PSO algorithm [43] to perform a search and limit the number of models to be used. We defined the PSO inertia values at c1 = 2, c2 = 2, and w = 0.9 based on the work presented in [44].As a first step, we performed several rounds of training, varying the input image resolution and the size of the YOLOv5 model dimensions. Then, we performed several tests using PSO and all the trained models, varying the minimum number of predictions to discard false positives. Starting from 12 models and varying the PSO parameters, we managed to reduce the number of models to 10, providing a balance between precision and recall.

(**a**) Ground truth (**b**) YOLOv5l (**c**) YOLOv5m (**d**) YOLOv5n

Figure 2. YOLOv 5 generates variability in predictions according to the depth with which a model has been trained. Thus, by combining the predictions, it is possible to identify new areas of possible damage and to reinforce predictions that are repeated across several models.

3.2. Confidence Threshold Optimization

To restrict the number of predictions provided by each model, it is necessary to define thresholds based on the confidence score. In general, the confidence score used to generate predictions is usually set to a value higher than 90% to ensure a low false positive rate. However, due to the high complexity of damage types in our dataset, an individual model cannot generate a sufficient number of correct predictions with high confidence score thresholds. This fact forces us to use predictions generated with low thresholds, which drastically increases the number of false positives. Therefore, we define a threshold for each model to restrict the number of false positives. In order to find the confidence score threshold values that fit each model, we used PSO and set the confidence score threshold limits between 10% and 99%.

3.3. Bounding Box Generation

We defined a prediction box merging process with the objective of unifying the predictions generated by each model. The general box-merging algorithm is detailed in Algorithm 1.

The first step consists of filtering out the predictions (list P) that do not exceed the confidence score established for the model that generated it. Subsequently, the predictions are ordered based on their confidence score. This is based on the premise that predictions with higher confidence scores tend to have a higher probability of being a correct prediction.

After obtaining the list of predictions ordered by their confidence score values, Q, we proceeded to iterate it and call the current one as the candidate prediction. Then, we obtained the predictions that intersect with the candidate prediction based on the value of Intersection over Union (IoU), which we call *IouThreshold* and define as 0.5.

Algorithm 1 Box fusion pseudocode

Inputs:
$P = [p_1 ... p_n]$ ▷ n prediction boxes
$S = [s_1 ... s_m]$ ▷ m score thresholds, one per model
$IouThreshold$ ▷ IoU threshold
$NumVotes$ ▷ Minimum number of votes
Output: F ▷ Final prediction boxes
$P' \leftarrow []$
for $i \leftarrow 1 \dots n$ **do** ▷ Filter predictions by scores
    $j \leftarrow model(p_i)$ ▷ Get model from prediction
    if $score(p_i) \geq s_j$ **then** ▷ Score is greater than model's threshold
        $P' \leftarrow concat(P', p_i)$ ▷ Add box for next step
    end if
end for
$Q \leftarrow sort(P', score)$ ▷ Sort predictions by scores
while $size(Q) \neq 0$ **do** ▷ Merge boxes
    $votes \leftarrow 0$
    $candidate \leftarrow Q[1]$ ▷ Get first element of Q
    $B \leftarrow [candidate]$ ▷ For contributing boxes
    $Q' \leftarrow []$ ▷ For remaining boxes
    for $i \leftarrow 2 \dots size(Q)$ **do** ▷ Get intersections
        if $IOU(candidate, Q[i]) > IouThreshold$ **then**
            $B \leftarrow concat(B, Q[i])$
            $votes \leftarrow votes + 1$ ▷ Accumulate votes
        else
            $Q' \leftarrow concat(Q', Q[i])$ ▷ Keep no intersects
        end if
    end for
    $Q \leftarrow Q'$ ▷ Intersections were deleted
    if $votes \geq NumVotes$ **then**
        $boxMaxArea \leftarrow getBoxWithMaxArea(B)$
        $F \leftarrow concat(F, boxMaxArea)$ ▷ Add box to F
    end if
end while
return F

Each prediction intersecting with the candidate prediction (also known as the cluster) will contribute with a vote. If the number of votes surpasses the defined variable $NumVotes$, then we will proceed to compute the best prediction box that binds all of them together. This is $boxMaxArea$, and it has the largest area including all the boxes. Finally, we add it to our final list of predictions F and remove all predictions that intercepted with the candidate prediction by updating the list Q. To identify the optimal minimum number of votes, we performed several iterations with the PSO algorithm, allowing us to set $NumVotes$ to 9, achieving a balance between correct predictions and false positives.

To address the scenario where different types of damage overlap, we define each prediction as a general damage and classify them using a previously trained classifier to identify the type of damage.

Figure 3 exemplifies the results of applying prediction fusion to different scenarios. We use as an example four vehicles with different damages. In column (a), we define the labeled damages in our dataset; in column (b), we visualize the damages generated by all the models, which are grouped into clusters according to their IoU value with respect to the bounding boxes of other damages; and in column (c), we show the resulting clusters after performing the box fusion process and applying the voting system.

Focusing on the first vehicle in Figure 3, we observe that five possible clusters can be generated. In column (b), the clusters with the highest probability of being chosen as

valid predictions are $c2$, $c3$, and $c4$. This is based on the number of matching predictions. However, in order to decrease the number of false positives, the minimum number of matching predictions in the cluster to be accepted as a valid prediction is taken into account.

In the second vehicle, clusters $c1$ and $c2$ stand out for having the highest number of intercepted predictions, but cluster $c3$, although very close to $c2$, is not considered a candidate for merging due to the IoU value between it and $c3$. This allows us to generate predictions that adapt to small damages and also to avoid generating predictions with large sizes.

For the third vehicle, four clusters stand out, of which clusters $c3$ and $c5$ have more candidates or votes. Cluster $c3$ has a higher probability of being considered as a prediction compared to the other clusters, since they have few predictions that intercept with each other. Finally, cluster $c3$ is the only one that remains as a valid prediction, and the other clusters are eliminated, since most of them are predictions generated by one or two models.

For the last vehicle, when the fusion algorithm defines the clusters, one of the prediction boxes seems to be assigned to cluster $c1$. Nonetheless, this prediction box is associated with cluster $c2$ because it shares a better IoU. In this way, we achieve variability in the generation of predictions, which allows us to cover multiple types of damage.

(**a**) Labeled damages (**b**) All predict damages (**c**) Final predicted damages

Figure 3. Examples of bounding box generation. In column (**a**), we visualize the actual damage for each image, while in column (**b**), we show the predictions generated by all models and the selected clusters (C1, C2). Column (**c**) shows the final results after applying the box fusion using a voting number of 6 for this example.

4. Materials and Methods

In order to determine whether the proposal is adequate, the following experiments were considered: (1) comparison of isolated YOLOv5 models with ensemble of multiple YOLOv5, and (2) comparison of the proposal with the state-of-the-art method DCN+ [4].

The models were trained on Ubuntu System 20.04 with Python 3.7.0, PyTorch 1.12.1, and CUDA 11.6 on an NVIDIA RTX 2080 GPU (installed in a server based at Grupo Valora, Sevilla, Spain). The inference process was carried out using the same system but with an Intel Core i5 processor, given that it is cheaper in a production environment than GPUs. We used the protocol currently used by insurance companies. First, the photographs were analyzed by a computer vision system. If no damage was found, the system moved on to contracting or renewing the insurance. However, if the system detected any damage, it moved on to an appraiser. In this way, we evaluated the detection of damage without distinguishing the category of damage.

For each experiment, the following metrics were considered:

- Precision: Defined as $\frac{TP}{TP+FP}$, precision allows us to identify which models tend to generate false positives, since this value will be affected by them. In the production environment, the presence of false positives implies increases in operational costs. For this reason, false positives should be kept to a minimum.
- False positive rate: Defined as $\frac{FP}{FP+TN}$, the false positive rate allows us to identify which models tend to generate false positives, which are negative results falsely labeled as positive. This is a very important metric for insurance companies, given that when a positive result appears, an appraiser will check it.
- Recall: Defined as $\frac{TP}{TP+FN}$, recall allows us to identify models that can detect the greatest number of damaged areas.
- AUC: This metric allows us to summarize the overall performance of a model from the precision vs. recall curve.
- Inference time: An important aspect is to measure the inference speed of the model, as it allows us to see the scalability of the model if it is to be deployed in a production environment, as well as the computational cost. This is measured in milliseconds.

Below, we define the datasets that have been used to both train and evaluate the voting system.

4.1. Datasets

We have opted for the creation of a new vehicle damage dataset where we increase the number of images available for the training process and add the supervision of experts in the field. Our dataset was created from scratch by experts in the field, making it a unique dataset focused on the verification of vehicle damage following the protocol of taking photos required by insurance companies.

The dataset used in this work is divided into three parts:

- Set 1: Dataset intended for training, fine-tuning, and model selection.
- Set 2: Dataset intended for model evaluation in a production environment.
- Set 3: Public dataset intended for model comparison for future work within the research community.

4.1.1. Dataset 1

This dataset consists of 7896 images of vehicles with damage and 2464 images of vehicles without damage. The damage has been divided into eight types: scratches, deformations, breakage—crack, breakage—loss of material, paint defect—peeling, paint defect—lacquering, missing part, and special damage.

The main types of damage are defined as follows:

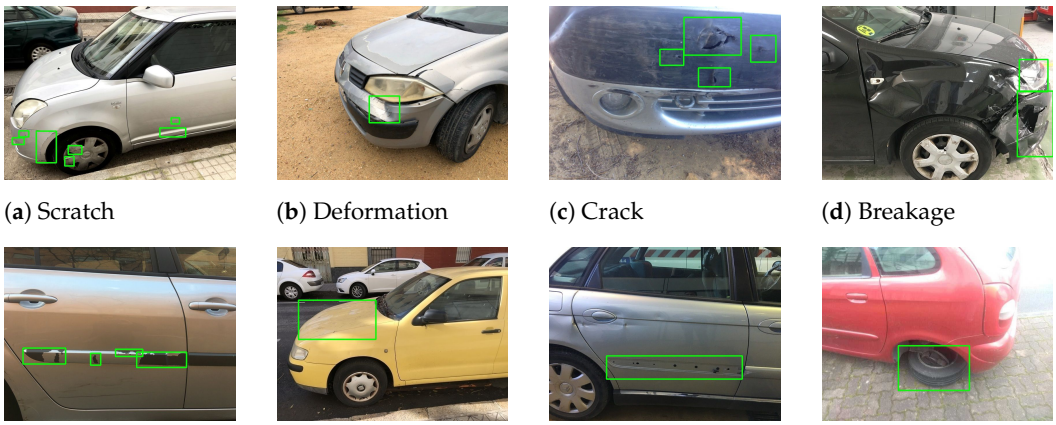
- Scratch: scratch without deformation produced on some elements of the vehicle's exterior bodywork. This category includes damage to the exterior paintwork, usually by detachment, mainly suffered by plastic bodywork elements.
- Deformation: Dent in any of the parts of the exterior bodywork of a vehicle.
- Breakage: Cracks, indentations, and detachment/lack of material from any of the bodywork parts. They usually appear on plastic parts, although they can be found in

the form of a hole in metal parts. It takes the form of peeling of the lacquer or veiling of the paint.

- Paint damage: Damage not resulting from an accident but from the attack of atmospheric or chemical agents, or even from defects in previous repairs. It takes the form of peeling of the lacquer or veiling of the paintwork.
- Special damage: This includes all damage that has not been considered above and indicates the existence of damage, but the type of damage is unknown. This type of damage is also affected by occlusions and usually requires further examination.

To ensure the consistency of the annotated data, a labeling protocol is specified, which includes non-linear damage. We address this by using small overlapping rectangles to cover the entire damage. Parallel scratches are labeled individually, even if they belong to the same area of damage.

The types of damage that have been labeled can be visualized in the Figure 4, where only damage of the corresponding type is displayed.



(**a**) Scratch (**b**) Deformation (**c**) Crack (**d**) Breakage

(**e**) Paint defect (**f**) Lacquer (**g**) Absence (**h**) Special

Figure 4. Types of damage. Each type is framed by a bounding box in each image.

We limited the damage to be detected from eight categories to the two most common: scratches and deformations. This is because they tend to be more recurrent in damage assessment than the other types.

4.1.2. Dataset 2

This dataset was specially created to carry out the evaluation process in real scenarios. A protocol for photo capture was followed, which consisted of taking six photos of each car, representing all possible views including the front view, rear view, front right side view, rear right side view, front left side view, front left side view, and rear left side view.

This dataset has been divided into two subsets:

Detection Set

This set consists of 955 images, manually labeled by expert appraisers, where damage types are classified into scratches and deformations. This dataset allowed us to evaluate the detection capability of the models.

False Positive Test Set

This dataset consists of 17,328 images manually reviewed by the staff of an insurance company in which all vehicles were identified as undamaged. This dataset allowed us to evaluate the robustness of a model by calculating the ratio of false positives.

4.1.3. Dataset 3

We provide a public dataset, called TartesiaDS, in order to allow other researchers to evaluate their models using a dataset under an appraisal standard.

This dataset consists of 108 images in total, divided as follows:

- 24 images with no damage present.
- 84 images with damage present.

In the images with damage, we found 226 areas of damage labeled as scratches, 93 areas of damage labeled as deformations, and 18 areas of damage of other types such as paint flaws, missing parts, and breakage.

4.2. Experiments

We fine-tune YOLOv5 models with (s, n, m, l, x) depths in combination with 640 and 1280 resolutions on dataset 1. Additionally, we train a YOLOv5x model starting from a different seed of YOLOv5x to start from different weights, which we name as YOLOv5_x2. Through the optimization of confidence scores and the process of merging boxes, as explained in Section 3, we obtain our final version of the ensemble.

4.2.1. YOLOv5 Model Comparison

We trained each YOLOv5 model defined above. In Table 2, we visualize the inference times, the size of the model, and the AUC value of each model on dataset 2.

Table 2. YOLOv5 models comparison for dataset 2.

Model	CPU Time (ms)	AUC
YOLOv5n_640	60	0.25
YOLOv5m_640	240	0.27
YOLOv5l_640	450	0.27
YOLOv5x_640	750	0.28
YOLOv5x_2_640	750	0.28
YOLOv5n_1280	190	0.27
YOLOv5s_1280	440	0.27
YOLOv5m_1280	930	0.26
YOLOv5l_1280	1700	0.29
YOLOv5x_1280	2820	0.26

4.2.2. Isolated YOLOv5 vs. YOLOv5 Ensemble

To validate our set of models, we use the precision, recall, and AUC metrics to compare an individual YOLOv5 model to a YOLOv5 ensemble. For the individual YOLOv5 model, we set the confidence score to 10% in order to obtain the maximum possible recall value, and we consider the worst-case scenario for precision. The confidence score values for the YOLOv5 ensemble are obtained using the PSO algorithm, allowing us to set the values of YOLOv5m_1280, YOLOv5n_1280, and YOLOv5s_1280 to 11.13%, 23.75%, and 55.01%, respectively. For the other models, it remains at 10%.

Additionally, in order to identify the behavior of an isolated YOLOv5 model at different confidence scores, we define a set of confidence scores as 40%, 50%, and 68%. We performed this test in order to identify problems when using a single model on our dataset.

4.2.3. YOLOv5 Ensemble vs. Other Models

We trained the DCN+ model on dataset 1 and evaluated it using our ensemble model. For this, we set the confidence score of DCN+ to 64% for the purpose of matching or beating the ensemble in terms of AUC. We carried out the evaluation on dataset 2 to identify the false positive rate. This is because the false positive rate cannot be higher than 10% to be accepted in a real environment. Then, we set the confidence score of DCN+ 76% in order to match or exceed the false positive rate with respect to the ensemble and to be able to compare the AUC metric.

In addition, we compared the YOLOv5 ensemble and the DCN+ model with an ensemble of seven faster R-CNN models to identify the advantages of using one-stage versus two-stage models, and we extended our ensemble methodology to YOLOv8-type models. We implemented the ensemble faster R-CNN using tensorflow models using the same voting and optimization system, defining the number of votes as seven. The models used were faster R-CNN with Resnet101 as the backbone, and we applied data augmentation such as clipping and brightness and saturation adjustments, together with changes in hyperparameters such as modifying the threshold non-maximum suppression IoU and the IoU threshold in the second stage.

4.3. Cost Analysis

4.3.1. Test Cost

To form the set of models, we train each model individually, where the largest model with a 1280-pixel resolution tends to generate a training time in dataset 1 (Section 4.1.1) of 24 h on the GPU specified in Section 4. Given that we are using an ensemble of several models, the total training time is close to 5 days. Once all the models were trained, we carried out threshold optimization by running PSO 10 times, involving different combinations of vote numbers and penalty values. We defined a range of number of votes from 5 to 9 and the penalty values as 1.5 and 2.5.

4.3.2. Inference Time Analysis

To verify the feasibility of using a model in a real scenario, we calculate the inference times for each model on a CPU using dataset 2 (Section 4.1.2), as this is usually the means available for a real scenario, allowing companies to avoid increasing costs. In the case of DCN+, the inference time is 12 s, which is higher than the YOLOv5 inference time of 2.82 s and the YOLOv8 inference time of 1.1 s. However, the time may be a high inference time if it is to be assumed for each model, so we use parallelization to assume a maximum time of 12 (DCN+), 2.82 (YOLOv5), or 1.1 (YOLOv8) seconds.

4.3.3. Evaluation of the Public Dataset

In Table 3, we list several datasets that have been created to carry out vehicle damage assessment, most of which are of a private nature. However, recently public datasets have been published, such as [4], which covers six types of damage (dents, scratches, cracks, glass breakage, punctures, and glass breakage) includes around 4000 images, collected using sources such as flickr5 and Shutterstock6, focusing on obtaining high-quality and diverse images. In [45], a dataset with 14,054 images was created, where 9 views of a vehicle are annotated as back, back left, back right, left, right, front, front left, front right, and zoomed in. Additionally, the authors added 3818 examples of three types of damage: scratches, cracks, and dents. However, as the scratch category damage tends to be more frequent, there is an imbalance in the dataset, which is solved by the authors through the application of data augmentation techniques, applying rotation, saturation, contrast, and brightness transformations.

In [26], a dataset is created for vehicle damage classification, dividing damage into seven types of damage: bumper dents, door dents, broken glass, broken headlights, broken taillights, broken tail lights, scratches, and breaks. These images were collected from the web and manually annotated.

In [46], a dataset was created using images from an insurance claims database, where five types of damage are defined: scratches, major dents, minor dents, and cracks. In [47], a public dataset is provided using images extracted from Google Images using Selenium. The images are divided into three sets: vehicle identification, view identification, and damage level identification.

Table 3. Comparison of vehicle damage datasets.

Dataset	Size	Categories	Task	Appraiser
Patel et al. [48]	326	N/A	D	N
Balci et al. [49]	533	2	C	N
Waqas et al. [50]	600	3	C	N
Deijn [51]	1007	4	C	N
Dwivedietal. [52]	1077	7	C, D	N
Kyu [2]	1150	N/A	C	N
Neo [47]	1150	8	C	N
Shirode [27]	1439	4	D, C	N
Widjojo [53]	2100	N/A	S, C	N
Singhetal. [46]	2822	5	S	N
Patil [26]	3092	7	D, C	N
Luca et al. [45]	3818	3	D	N
CarDD [4]	4000	6	C, D, S, SOD	N
Ours private dataset	10,360	8	C, D	Y
Ours public dataset	108	8	C, D	Y

C: damage classification, D: damage detection, S: damage segmentation, SOD: salient object detection, N/A: information not provided by the authors, N/Y: whether the dataset was created in the context of an appraisal.

Although some of the datasets named above are public and have a considerable number of images, we consider that it is necessary to adapt vehicle damage detection to a real appraisal. Therefore, we created a dataset under the supervision of experts in the field and focused on carrying out an image acquisition protocol based on the requirements of the insurance companies, which consists of taking six photos of each car representing all possible views: front view, rear view, front right side view, rear right side view, front left side view, front left side view, and rear left side view.

In our datasets, damage segmentation labeling was discarded due to a lack of agreement between experts on the boundaries of the damage. Particularly, in the case of scratches and deformations, these boundaries may depend on subjective perceptions, making the labeling inconsistent. Furthermore, from the point of view of insurance companies, a rectangular box is sufficient for damage detection and also speeds up the labeling process by the experts.

Finally, in order to be able to compare our work with future work, we performed tests on dataset 3 (Section 4.1.3) with all the models used in previous sections.

5. Results and Discussion

5.1. Isolated YOLOv5 vs. YOLOv5 Ensemble

Once we obtained the results of Experiment 1 shown in Table 2, we focused on using the precision value to compare all the trained models and the proposed set. Although the recall and AUC values allowed us to determine the performance of a model, the precision value in an industrial setting is more important because low precision generates costs to companies in terms of time and effort when checking for false positives. In Figure 5, we

can see the precision of all the models used together with the ensemble of models in order to highlight the precision value obtained by the ensemble.

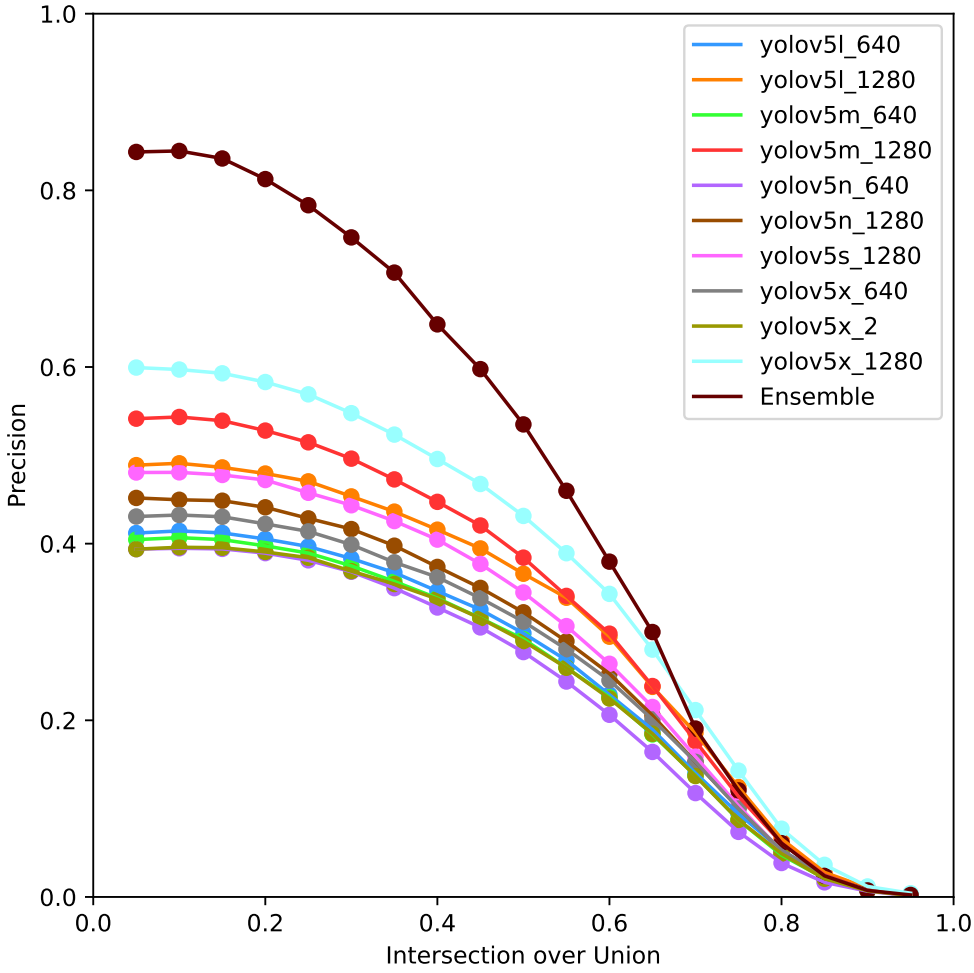


Figure 5. Precision achieved by the models individually vs. the ensemble.

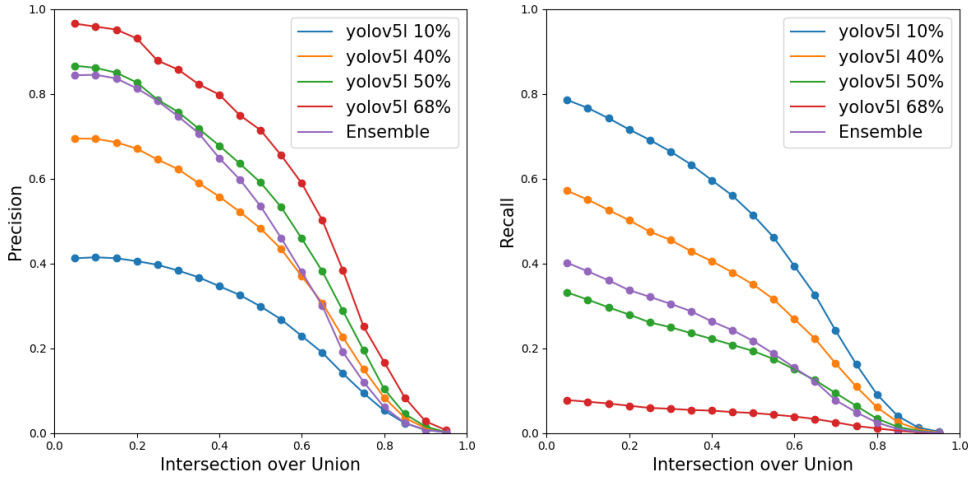
We performed an additional evaluation utilizing YOLOv5l_1280 (YOLOv5l), since it obtained one of the best AUC values among all models, and it was the proposed ensemble to identify the false positive rate. For this, we evaluated these models on our false positive set, and we show the results in Table 4 including the AUC vs. IoU value, which is the average of the AUC values over the set IoU range. We can observe that although the YOLOv5l model has a higher AUC compared to the ensemble, the false positive rate of YOLOv5l is very high. For this reason, this model is not feasible for the damage verification process in a real scenario. In contrast to YOLOv5l, the proposed ensemble manages to keep the false positive rate at 9%.

In addition, we evaluate YOLOv5l using different confidence score thresholds against the ensemble model. To gain detailed insight into the behavior of each model, each metric is derived based on a range of IoU values at the time we performed the evaluation. We visualize the results in Figure 6, where we can see that the ensemble implementation achieves intermediate results across all the metrics, thus maintaining a balance between precision and recall.

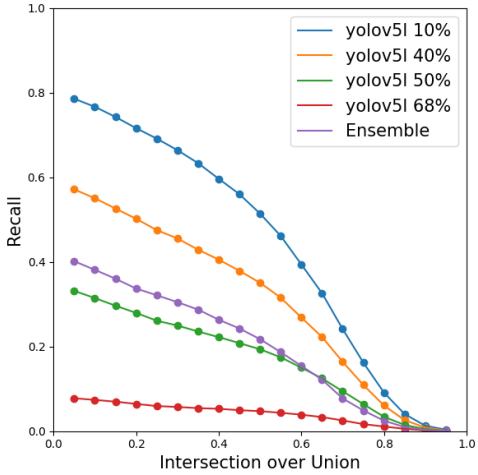
In order to extend the use of our methodology, we applied the ensemble to the YOLOv8 models and carried out the evaluation presented in Figure 7 on dataset 2. This evaluation allows us to present an alternative to the use of YOLOv5 in case a lower accuracy is required.

Table 4. Metrics summary of the models for dataset 2. YOLOV5l 10% graphics can be seen in Figure 6. DCN+ 64% and DCN+ 76% graphics can be seen in Figure 8. mAP is calculated with the IoU threshold 50. The standard deviation is not included, as the models are trained with a fixed partition on dataset 1 and tested on the entire dataset 2. Inference time is the maximum of each model.

Models	FP Rate	AUC vs. IoU	mAP_{50}	Inference Time (CPU)
YOLOv5l 10%	69%	0.2779	7.20%	1700 ms
DCN+ 64%	30%	0.1541	4.33%	12,000 ms
DCN+ 76%	8%	0.067	2.17%	12,000 ms
Ensemble faster R-CNN	3.3%	0.0784	1.36%	3540 ms
Ensemble YOLOv8 (ours)	16%	0.1849	4.14%	1110 ms
Ensemble YOLOv5 (ours)	9%	0.1507	3.77%	2820 ms

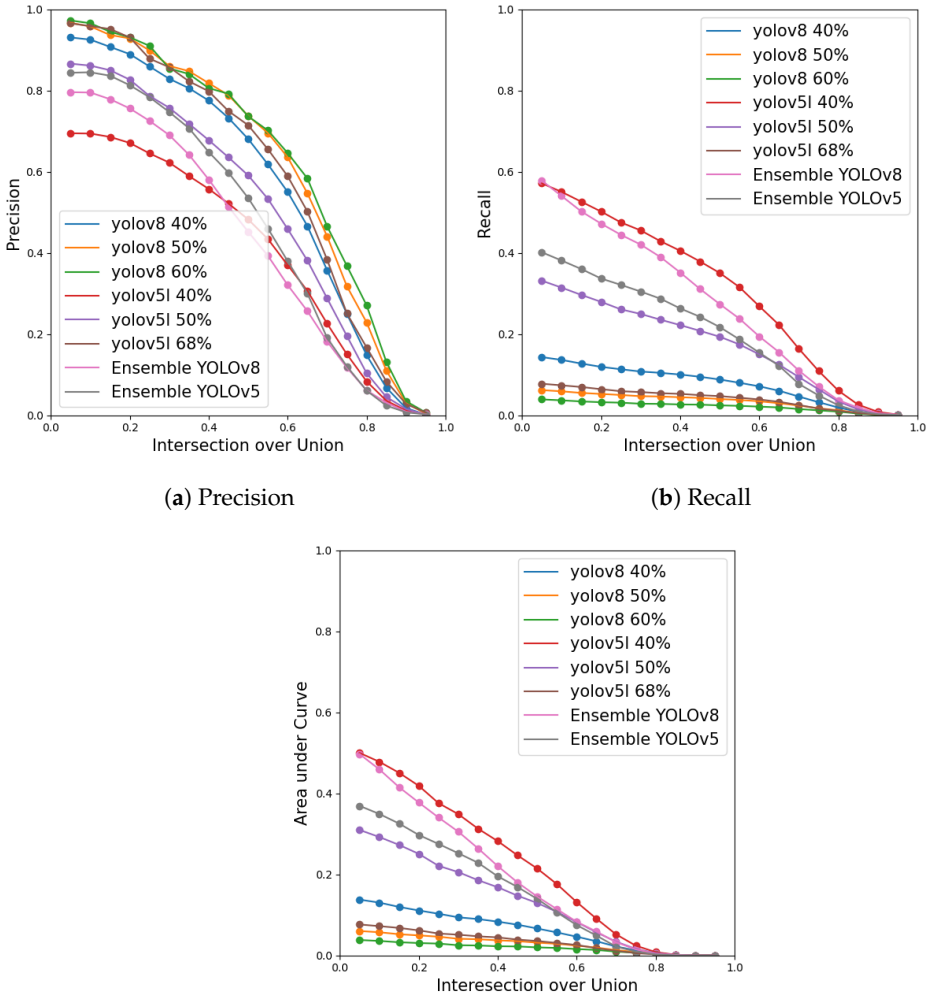


(**a**) Precision (**b**) Recall



(**c**) AUC

Figure 6. YOLOv5l thresholds vs. ensemble.



(**a**) Precision (**b**) Recall

(**c**) AUC

Figure 7. YOLOv 5l ensemble vs. YOLOv8 ensemble vs. YOLOv5 and YOLOv8 individually.

5.2. YOLOv5 Ensemble vs. Others Models

To validate our proposed model against the state-of-the-art, we evaluated two YOLO ensembles (YOLOv5 and YOLOv8) against the faster R-CNN ensemble and the DCN+ model in the same way as for the individual YOLOv5l model and the ensemble. We set the confidence values of DCN+ to 64% and 76%, as explained in Section 4.2.3, and we compared it with the ensembles in dataset 2, as shown in Figure 8.

We add the mAP_{50} metric where the classes of each prediction are taken into account. This metric allows us to compare the results with future work, but in our case, we have not used it individually, as both the AUC vs. IoU and the mAP value must be complemented with the false positive rate.

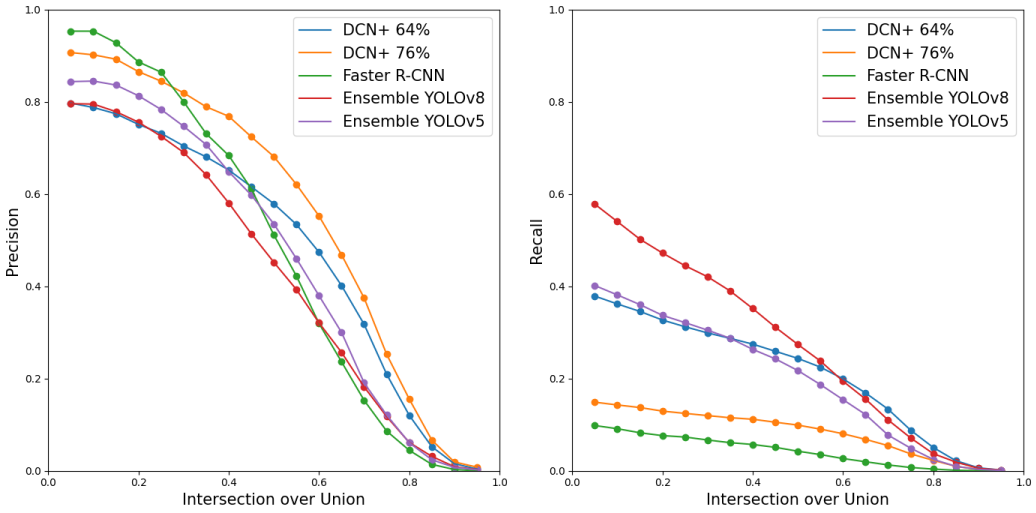
When evaluating DCN+ with the confidence score at 64% on the false positive set, according to Table 4, we obtain a false positive ratio of 30%, while the ensemble maintains a ratio of 9%. In the same way, using the confidence score at 76%, the false positive rate is slightly better than the ensemble (we set this threshold to achieve this), but the recall and AUC metrics are worse than the ensemble.

At first glance, DCN+ could be a candidate to replace one of the YOLOv5 models in the ensemble. However, due to its false positive rate and its CPU inference time, which more than four times slower than the ensemble, it is not feasible to use it in the ensemble for our damage detection problem.

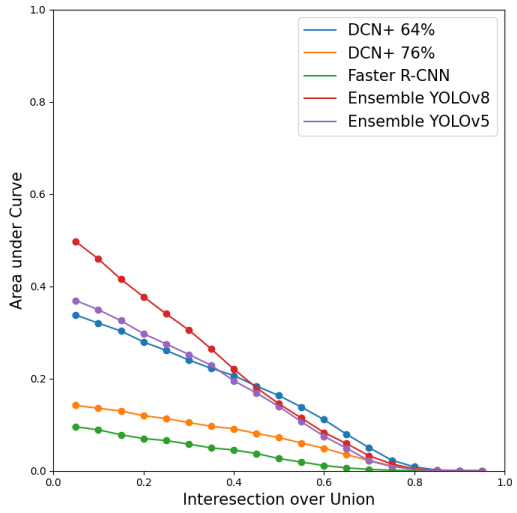
On the other hand, the YOLOv8 ensemble can also be considered as a possible candidate. We even consider this ensemble as an alternative to the YOLOv5 ensemble since,

due to its inference time and high mAP50 value, it can be useful in a scenario where it is allowed a false positive rate greater than 10%.

On the faster R-CNN side, it achieves a good performance in the precision metric, but in the other metrics, its performance falls below the other models.



(**a**) Precision (**b**) Recall



(**c**) AUC

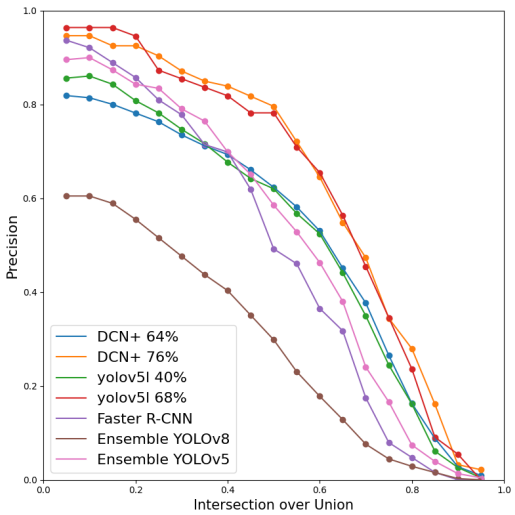
Figure 8. Ensemble vs. others models. This figure is complemented by the data in Table 4. The more balanced DCN+ (64%) has a high false positive rate (30%), and the DCN+ inference time is more than 4 times slower than our ensemble. The faster R-CNN ensemble achieves a good result in the precision metric but is outperformed by the other models in the recall and AUC metrics. The YOLOv8 ensemble excels in the recall and AUC metrics, followed by the YOLOv5 ensemble, which performs better in terms of precision.

5.3. Evaluation Using the Public Dataset

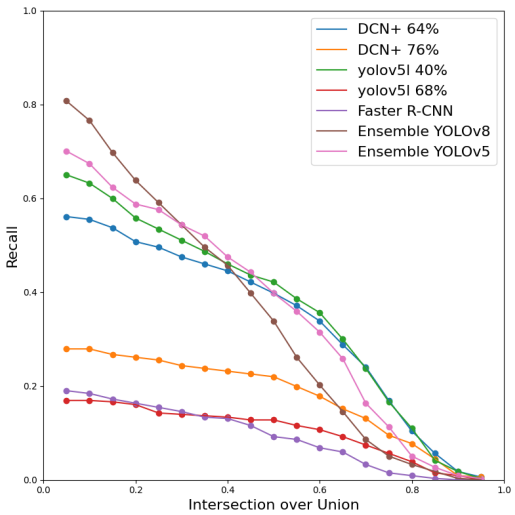
Finally, to verify the result of our ensemble model trained on dataset 1, we tested it on dataset 3 or TartesiaDS. The precision, recall, and AUC metrics are shown in Figure 9. As we can see, when compared to the other models, it still obtains balanced results, highlighting the precision metric. It should be considered that this last evaluation is performed to show the results on the public dataset, since the robustness of the ensemble proposal and the speed of inference time have been tested with the previous datasets, maintaining a low

CPU inference value, 2820 ms, which allows for scaling of the system in a production environment.

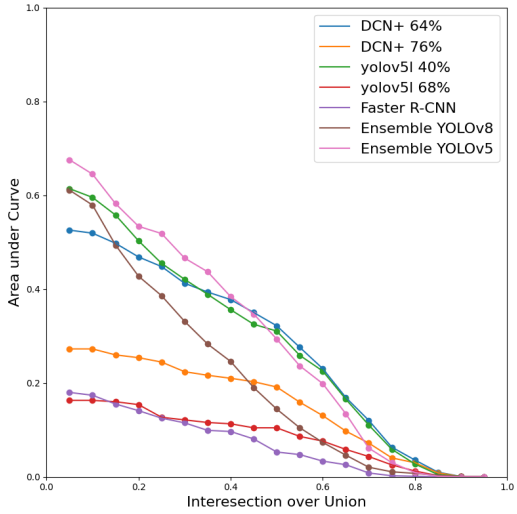
Finally, through tests on the public dataset, we can verify that the ensembles allow us to obtain the best precision and that the YOLOv8-type ensembles can be an alternative for even a scenario where greater recovery is required, risking precision.



(**a**) Precision



(**b**) Recall



(**c**) AUC

Figure 9. Evaluation with our public dataset TartesiaDS.

6. Conclusions and Future Work

In this paper, we have presented our implementation of a voting system and a box fusion process using ensemble learning to perform damage detection in cars. This is done in the context of insurance companies, where it is required to have a low false positive rate and a good efficiency for scalability.

Through several tests, we managed to highlight the advantages of using ensemble learning based on YOLOv5 detectors, together with optimization and box fusion processes, allowing us to implement a fast and robust system and minimize the number of false positives.

We also published a careful selection of 108 images containing vehicles with and without damage, named TartesiaDS. This can serve as a testing dataset for the research community to validate the evolution of the state-of-the-art for professional vehicle damage

appraisers. Our ensemble model achieves a very good balance between false positive rate; a metric that is very important for insurance companies, given that appraisers must verify every positive instance, precision, recall, AUC, and inference time in CPU, which is the better option to scale the system in the cloud at the best price.

Our solution, compared to the DCN+ model, achieve better metrics, and the inference speed is four times faster, improving the state-of-the-art.

We also provide an alternative for specific cases where greater ensemble-based detection capacity is required using more up-to-date models such as YOLOv8.

For future work, we will focus on vehicle damage detection based on videos. For this, we will use spatio-temporal context using transformer-based models such as [54–56], supported by a new video-based dataset under expert supervision. Relying on works such as [57,58], we will be able to face the challenges of camera movements, changes in environmental conditions such as rain, lights, changes in the appearance of objects, and above all, achieve the tracking of an object that is often restricted by the instability of a detector. In addition, we will consider implementing utilities to facilitate the construction of a new dataset by generating masks using only bounding boxes and generating damage trajectories using tracking algorithms.

Author Contributions: Conceptualization, J.A.Á.-G., M.A.M.-d.-A., J.L.P.-M. and S.A.P.-Z.; Methodology, J.A.Á.-G., M.A.M.-d.-A., J.L.P.-M. and S.A.P.-Z.; Formal analysis, S.A.P.-Z. and D.C.-G.; Investigation, S.A.P.-Z.; Data curation, D.F.-C.; Writing—original draft, S.A.P.-Z.; Writing—review and editing, J.A.Á.-G., M.A.M.-d.-A., S.A.P.-Z. and J.L.P.-M.; Supervision, M.A.M.-d.-A., J.A.Á.-G. and J.L.P.-M.; Project administration, J.A.Á.-G.; Funding acquisition, J.L.P.-M. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the Valora Group with its own funds, with partial support from the HORUS project (grant no. PID2021-126359OB-I00) funded by MCIN/AEI/10.13039/5011000 11033 and by the European Union NextGenerationEU/PRTR. This research was also supported by grants from NVIDIA and used NVIDIA A100 donated to Miguel A. Martinez-del-Amor.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study can be requested from the corresponding author through the repository. The data are not available to the public for privacy reasons.

Acknowledgments: We are grateful for the hard work of the Valora Group's team of experts in developing the dataset. We also thank the HORUS project (grant no. PID2021 126359OB-I00) funded by MCIN/AEI/ 10.13039/501100011033 and the support of Miguel A. Martinez-del-Amor for facilitating the use of the NVIDIA A100 resource.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. van Ruitenbeek, R.; Bhulai, S. Convolutional Neural Networks for vehicle damage detection. *Mach. Learn. Appl.* **2022**, *9*, 100332. [CrossRef]
2. Kyu, P.M.; Woraratpanya, K. Car damage detection and classification. In Proceedings of the 11th International Conference on Advances in Information Technology, Bangkok, Thailand, 1–3 July 2020; pp. 1–6.
3. Parhizkar, M.; Amirfakhrian, M. Recognizing the Damaged Surface Parts of Cars in the Real Scene Using a Deep Learning Framework. *Math. Probl. Eng.* **2022**, *2022*, 5004129. [CrossRef]
4. Wang, X.; Li, W.; Wu, Z. CarDD: A New Dataset for Vision-Based Car Damage Detection. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 7202–7214. [CrossRef]
5. Girshick, R. Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
6. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2016**, arXiv:1506.01497. [CrossRef]
7. Cai, Z.; Vasconcelos, N. Cascade R-CNN: Delving into High Quality Object Detection. *arXiv* **2017**, arXiv:1712.00726.

8. Wang, W.; Dai, J.; Chen, Z.; Huang, Z.; Li, Z.; Zhu, X.; Hu, X.; Lu, T.; Lu, L.; Li, H.; et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 14408–14419.
9. Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; Wei, Y. Deformable convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 764–773.
10. Su, W.; Zhu, X.; Tao, C.; Lu, L.; Li, B.; Huang, G.; Qiao, Y.; Wang, X.; Zhou, J.; Dai, J. Towards all-in-one pre-training via maximizing multi-modal mutual information. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 15888–15899.
11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14; Springer: Berlin/Heidelberg, Germany, 2016; pp. 21–37.
12. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
13. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
14. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
15. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976.
16. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.
17. Yajing, L.; Zhongjian, D. Abnormal Behavior Detection in Crowd Scene Using YOLO and Conv-AE. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1720–1725.
18. Liu, Y.; Hong, W. Target detection based on DB-YOLO in road environment. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1496–1501.
19. Azimjonov, J.; Özmen, A. A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways. *Adv. Eng. Inform.* **2021**, *50*, 101393. [CrossRef]
20. Lv, X.; Lian, X.; Tan, L.; Song, Y.; Wang, C. HPMC: A multi-target tracking algorithm for the IoT. *Intell. Autom. Soft Comput.* **2021**, *28*, 513–526. [CrossRef]
21. Rivero-Palacio, M.; Alfonso-Morales, W.; Caicedo-Bravo, E. Mobile application for anemia detection through ocular conjunctiva images. In Proceedings of the 2021 IEEE Colombian Conference on Applications of Computational Intelligence (ColCACI), Virtual, 26–28 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
22. Roy, A.M.; Bhaduri, J. DenseSPH-YOLOv5: An automated damage detection model based on DenseNet and Swin-Transformer prediction head-enabled YOLOv5 with attention mechanism. *Adv. Eng. Inform.* **2023**, *56*, 102007. [CrossRef]
23. Xu, J.; Wang, W.; Wang, H.; Guo, J. Multi-model ensemble with rich spatial information for object detection. *Pattern Recognit.* **2020**, *99*, 107098. [CrossRef]
24. Casas, E.; Ramos, L.; Bendek, E.; Rivas-Echeverria, F. YOLOv5 vs. YOLOv8: Performance Benchmarking in Wildfire and Smoke Detection Scenarios. *J. Image Graph.* **2024**, *12*, 127–136. [CrossRef]
25. Gašparović, B.; Mauša, G.; Rukavina, J.; Lerga, J. Evaluating Yolov5, Yolov6, Yolov7, and Yolov8 in underwater environment: Is there real improvement? In Proceedings of the 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech), Split and Bol, Croatia, 20–23 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–4.
26. Patil, K.; Kulkarni, M.; Sriraman, A.; Karande, S. Deep learning based car damage classification. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 50–54.
27. Shirode, A.; Rathod, T.; Wanjari, P.; Halbe, A. Car damage detection and assessment using CNN. In Proceedings of the 2022 IEEE Delhi Section Conference (DELCON), New Delhi, India, 11–13 February 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–5.
28. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
29. Mallikarjuna, B.; Arun Kumar, K.L. Vehicle Damage Detection and Classification Using Image Processing. *Int. J. Adv. Res. Sci. Commun. Technol.* **2022**, 568–574. [CrossRef]
30. Fang, Y.; Liao, B.; Wang, X.; Fang, J.; Qi, J.; Wu, R.; Niu, J.; Liu, W. You only look at one sequence: Rethinking transformer in vision through object detection. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26183–26197.
31. Hoang, V.D.; Huynh, N.T.; Tran, N.; Le, K.; Le, T.M.C.; Selamat, A.; Nguyen, H.D. Powering AI-driven car damage identification based on VeHIDE dataset. *J. Inf. Telecommun.* **2024**, 1–19. [CrossRef]
32. Reddy, D.A.; Shambharkar, S.; Jyothsna, K.; Kumar, V.M.; Bhoyar, C.N.; Somkunwar, R.K. Automatic Vehicle Damage Detection Classification framework using Fast and Mask Deep learning. In Proceedings of the 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 8 September 2022; pp. 1–6. [CrossRef]

33. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587.
34. Elbhrawy, A.S.; Belal, M.A.; Hassanein, M.S. CES: Cost Estimation System for Enhancing the Processing of Car Insurance Claims. *J. Comput. Commun.* **2024**, *3*, 55–69. [CrossRef]
35. Kalshetty, J.N.; Hrithik Devaiah, B.; Rakshith, K.; Koshy, K.; Advait, N. Analysis of Car Damage for Personal Auto Claim Using CNN. In *Sentimental Analysis and Deep Learning: Proceedings of ICSADL 2021*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 319–329.
36. Bahhar, C.; Ksibi, A.; Ayadi, M.; Jamjoom, M.M.; Ullah, Z.; Soufiene, B.O.; Sakli, H. Wildfire and Smoke Detection Using Staged YOLO Model and Ensemble CNN. *Electronics* **2023**, *12*, 228. [CrossRef]
37. Doshi, K.; Yilmaz, Y. Road damage detection using deep ensemble learning. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 5540–5544.
38. Casado-García, Á.; Heras, J. Ensemble methods for object detection. In Proceedings of the ECAI 2020, Santiago de Compostela, Spain, 29 August–8 September 2020; IOS Press: Amsterdam, The Netherlands, 2020; pp. 2688–2695.
39. Lee, J.; Lee, S.K.; Yang, S.I. An ensemble method of CN models for object detection. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 17–19 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 898–901.
40. Zhang, H.; Li, F.; Liu, S.; Zhang, L.; Su, H.; Zhu, J.; Ni, L.M.; Shum, H.Y. DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection. *arXiv* **2022**, arXiv:2203.03605.
41. Liu, S.; Li, F.; Zhang, H.; Yang, X.; Qi, X.; Su, H.; Zhu, J.; Zhang, L. DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR. *arXiv* **2022**, arXiv:2201.12329.
42. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; Proceedings, Part V 13; Springer: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
43. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 29 November–1 December 1995; Volume 4, pp. 1942–1948. [CrossRef]
44. Pedersen, M.E.H. Good parameters for particle swarm optimization. *Hvass Lab. Cph. Den. Tech. Rep. HL1001* **2010**, 1551–3203.
45. Maiano, L.; Montuschi, A.; Caserio, M.; Ferri, E.; Kieffer, F.; Germanò, C.; Baiocco, L.; Ricciardi Celsi, L.; Amerini, I.; Anagnostopoulos, A. A deep-learning–based antifraud system for car-insurance claims. *Expert Syst. Appl.* **2023**, *231*, 120644. [CrossRef]
46. Singh, R.; Ayyar, M.P.; Pavan, T.V.S.; Gosain, S.; Shah, R.R. Automating car insurance claims using deep learning techniques. In Proceedings of the 2019 IEEE fifth international conference on multimedia big data (BigMM), Singapore, 11–13 September 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 199–207.
47. Neo, T. Car Damage Detective. 2023. Available online: https://github.com/neokt/car-damage-detective (accessed on 13 February 2023).
48. Patel, N.; Shinde, S.; Poly, F. Automated damage detection in operational vehicles using mask R-CNN. In Proceedings of the Advanced Computing Technologies and Applications: Proceedings of 2nd International Conference on Advanced Computing Technologies and Applications—ICACTA 2020, Mumbai, Maharashtra, 28–29 February 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 563–571.
49. Balci, B.; Artan, Y.; Alkan, B.; Elihos, A. Front-View Vehicle Damage Detection using Roadway Surveillance Camera Images. In Proceedings of the VEHITS, Crete, Greece, 3–5 May 2019; pp. 193–198.
50. Waqas, U.; Akram, N.; Kim, S.; Lee, D.; Jeon, J. Vehicle damage classification and fraudulent image detection including moiré effect using deep learning. In Proceedings of the 2020 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), London, ON, Canada, 26–29 April 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–5.
51. De Deijn, J. Automatic car damage recognition using convolutional neural networks. In Proceedings of the 2018 Internship Report MSc Business Analytics, Amsterdam, The Netherlands, 29 March 2018.
52. Dwivedi, M.; Malik, H.S.; Omkar, S.; Monis, E.B.; Khanna, B.; Samal, S.R.; Tiwari, A.; Rathi, A. Deep learning-based car damage classification and detection. In *Advances in Artificial Intelligence and Data Engineering: Select Proceedings of AIDE 2019*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 207–221.
53. Widjojo, D.; Setyati, E.; Kristian, Y. Integrated Deep Learning System for Car Damage Detection and Classification Using Deep Transfer Learning. In Proceedings of the 2022 IEEE 8th Information Technology International Seminar (ITIS), Surabaya, Indonesia, 19–21 October 2022; pp. 21–26. [CrossRef]
54. Yan, B.; Jiang, Y.; Sun, P.; Wang, D.; Yuan, Z.; Luo, P.; Lu, H. Towards grand unification of object tracking. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–24 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 733–751.
55. Cheng, B.; Choudhuri, A.; Misra, I.; Kirillov, A.; Girdhar, R.; Schwing, A.G. Mask2former for video instance segmentation. *arXiv* **2021**, arXiv:2112.10764.
56. Meinhardt, T.; Kirillov, A.; Leal-Taixe, L.; Feichtenhofer, C. Trackformer: Multi-object tracking with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 8844–8854.

57. Zeng, F.; Dong, B.; Zhang, Y.; Wang, T.; Zhang, X.; Wei, Y. MOTR: End-to-End Multiple-Object Tracking with Transformer. *arXiv* **2022**, arXiv:2105.03247.
58. Yan, F.; Luo, W.; Zhong, Y.; Gan, Y.; Ma, L. Bridging the Gap Between End-to-end and Non-End-to-end Multi-Object Tracking. *arXiv* **2023**, arXiv:2305.12724.