Machine Learning with Applications 9 (2022) 100332

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Convolutional Neural Networks for vehicle damage detection

R.E. van Ruitenbeek [*], S. Bhulai

De Boelelaan 1081a, 1081 HV Amsterdam, Netherlands

ARTICLE INFO

Keywords:
Computer vision
Image recognition
Object detection
Deep learning
Vehicle damage detection

ABSTRACT

Vehicle damages are increasingly becoming a liability for shared mobility services. The large number of handovers between drivers demands for an accurate and fast inspection system, which locates small damages and classifies these into the correct damage category. To address this, a damage detection model is developed to locate vehicle damages and classify these into twelve categories. Multiple deep learning algorithms are used, and the effect of different transfer learning and training strategies is evaluated, to optimize the detection performance. The final model, trained on more than 10,000 damage images, is able to accurately detect small damages under various conditions such as water and dirt. A performance evaluation with domain experts shows, that the model achieves comparable performance. In addition, the model is evaluated in a specially designed light street, indicating that strong reflections complicate the detection performance.

1. Introduction

With vehicle sharing initiatives at the rise, the relevance of insurance management increases. The growth of commercial car sharing, peer-to-peer sharing, and home delivery results in a higher number of drivers per vehicle. With this trend, the complexity and liability for insurance companies and car owners increases drastically. Therefore, a thorough inspection at each handover is preferred. To avoid delays in the process and maintain viable car sharing, in terms of cost and ease of usage, a more automated and efficient inspection is required.

Automatic inspection has frequently been studied in the literature. A variety of studies focuses on structural health monitoring and inspection for less accessible areas. Most studies in structural health monitoring use ultrasonic waves in combination with a learning algorithm (Brincker et al., 1995; Lee et al., 2006; Liew & Veidt, 2009; Shardakov et al., 2017). The benefits of ultrasonic waves are especially obtained for less accessible areas such as gas and oil pipelines (Lee et al., 2006) and oil platforms (Brincker et al., 1995). Alternatively, fiber strain sensors are applied to monitor structures continuously (Matveenko et al., 2019; Zhang et al., 2020). Other research uses pattern recognition for images to monitor the condition of structures. Such as cloud modeling and the classification of cubemap images using Convolutional Neural Networks (CNNs) (Isailović et al., 2020), using edge detection filters and Kalman filtering (Cha, Chen et al., 2017), or by applying CNNs directly on the image (Cha, Choi et al., 2017; Shihavuddin et al., 2019).

Outside structural health monitoring, extensive research is found for damage inspection. Bodnarova et al. (2002) use 2-D Gabor filters, in combination with a maximizing Fisher function to detect textile flaws. Huang et al. (2020) apply kNN, SVM, Logistic Regression, Random Forest and a CNN for damage detection on steel wire ropes, showing that CNNs largely outperform other machine learning methods. Alternatively, Zhao et al. (2020) used bilateral filtering, in combination with a feature gradient histogram to estimate the size and location of damage in fishing nets.

Other ways to detect anomalies are by comparing an object of interest with an undamaged representation. Jayawardena (2013) used a 3D Computer-Aided Design (CAD) of undamaged cars and implemented a 3D pose estimation algorithm to align the damaged vehicle with an undamaged CAD. Matching two visualization, using a sparse representation in combination with a classifier, has been widely applied for face recognition (Ahmed et al., 2015; Guo et al., 2012) and object similarity matching (Chen et al., 2015; Hsu et al., 2018).

One of the major challenges for object inspection, in contrast to structures, is the movement of the object. Making the application of fiber strain sensors less applicable. Another challenge in damage inspection is the robustness against different light conditions, where Cha, Choi et al. (2017) show a strong benefit of CNNs to extract patterns over edge detection and Kalman filtering (Cha, Chen et al., 2017; Cha, Choi et al., 2017). Hsu et al. (2018) explain that varying outdoor illumination results in an in-precise representation of edge features, requiring an appropriate feature-based presentation rather than an edge-based presentation. Illumination changes and reflective surfaces such as vehicle bodies complicate the feature extraction which increases the false positive rate (Jayawardena, 2013).

* Corresponding author.
E-mail addresses: revanruitenbeek@gmail.com (R.E. van Ruitenbeek), s.bhulai@vu.nl (S. Bhulai).
URLs: https://orcid.org/0000-0003-4774-1511 (R.E. van Ruitenbeek), http://www.math.vu.nl/~sbhulai (S. Bhulai).

https://doi.org/10.1016/j.mlwa.2022.100332
Received 16 December 2021; Received in revised form 19 April 2022; Accepted 13 May 2022
Available online 20 May 2022

CNNs have largely proven its applicability to automatically extract the most discriminating features (Shihavuddin et al., 2019; Simonyan & Zisserman, 2014). The use of handcrafted features such as Scale Invariant Feature Transformation (SIFT) (Lowe, 1999) and Histogram of Gradients (HOG) (Dalal & Triggs, 2005) are largely outperformed by CNNs, due to its robustness against scale, rotation, and illumination (Liu et al., 2020). Automatic feature extraction received a tremendous growth by the development of AlexNet (Krizhevsky et al., 2017), resulting in development of real-time object detection models such as Region-based Convolutional Neural Network (R-CNN) (Ren et al., 2017), Single Shot multi-box Detector (SSD) (Liu et al., 2016), and You Only Look Once (YOLO) (Redmon & Farhadi, 2018). Shihavuddin et al. (2019) show that CNNs, in combination with Pyramid and Patching Augmentation (PPA), are able to extract small damages on wind turbine blades from high-resolution drone images.

So far, limited research has been conducted in the field of automatic detection for vehicle damages. Although, Jayawardena (2013) used an alignment approach of CAD models to detect anomalies, most research focuses on CNNs for 2D images (De Deijn, 2018; Liu et al., 2018; Patil et al., 2017). De Deijn (2018) and Patil et al. (2017) used approximately 1000 samples of damaged vehicles to distinguish damaged vehicles from undamaged vehicles using a CNN. De Deijn (2018) classified the damages into four damage categories and four damage locations and shows that solely classification suffers from large intra-class interference. Li et al. (2018) went beyond damage classification and added damage localization, using YOLO (Redmon & Farhadi, 2018) object detection. Although Li et al. (2018) used damage localization, they made no distinction between damage categories. Previous research shows that classification of damage on vehicles is non-trivial due to different shapes of the vehicle and the damage (Liu et al., 2018; Patil et al., 2017).

In this paper, we extend previous research in three ways. Firstly, we use a significantly larger dataset by extending images available on the internet, with data obtained from Pon.[1] We apply the damage classification and localization on twelve damage categories, to limit the intra-class interference faced by De Deijn (2018) and Patil et al. (2017) and extend the research of Li et al. (2018). Secondly, we evaluate different object detection models with various backbones using transfer learning and quantify the impact of different fine-tuning techniques. Thirdly, we evaluate the model against domain experts and asses the performance in a production environment under the strong light conditions of a light street.

We use a manual annotation technique to construct the ground-truth labels for a total of 5000 images and observe that localizing and classifying damage categories is challenging due to the inter-class similarity, as explained by Patil et al. (2017). Additionally, the high reflective vehicle surface, as well as dirt on the vehicle, complicate this process. Contrary to classical object detection tasks, the notion of a clear boundary for damages is absent, as different annotation granularities result in different ground truth situations.

We observe that deep learning is able to detect vehicle damages. More precisely, this research shows that deep learning is able to achieve performance comparable to humans. Furthermore, we observe that a large performance difference exists between different object detection models with different backbones. Various fine-tuning experiments indicate, that a carefully selected set of trainable layers results in a major detection improvement. Lastly, we show that more detailed object categories result in less inter-class interference for damage detection.

2. Related work

To the best of our knowledge, only Li et al. (2018), Patil et al. (2017), and De Deijn (2018) used a deep learning approach to identify vehicle damages from 2D images. That is, Patil et al. (2017) evaluated the ability of CNNs to classify vehicle damages. They used three different approaches to classify the damage into seven damage categories[2] and one undamaged category, using a total of 1200 images with vehicle damages and 1271 images without damage. Training from scratch, results in a classification accuracy of 72.46 percent, whereas the use of Convolutional AutoEncoders (CAE) improves this slightly to an accuracy of 73.43 percent.

The third applied method of Patil et al. (2017) made use of transfer learning, increasing the classification accuracy to 88.24 percent. They argue that pre-trained models, trained on a broad object range, outperform pre-trained models trained specifically for vehicle classification. Giving rise to the idea that vehicle damages detection might require different features, in contrast to recognizing vehicles itself (Patil et al., 2017). Lastly, they show that a linear ensemble method can improve the overall accuracy to a final 89.5 percent. They point out that vehicles with small damages often result in false negatives, as the majority of the vehicle is undamaged. Moreover, they indicate that the classification task is non-trivial due to the large inter-class similarity of damages.

De Deijn (2018) conducted research comparable to Patil et al. (2017). A cascade of three CNNs has been used to conduct the damage classification. The first classifier recognizes the presence of a vehicle, followed by classifying if damage is present, where the last model identifies the class, location, and size of the damage. For their research, 29,000 images without a vehicle, 16,185 images with undamaged vehicles, and 1007 images with damaged vehicles have been used. To increase the dataset size, they solely used horizontal flipping as image augmentation to avoid overfitting (De Deijn, 2018).

De Deijn (2018) explains that ambiguity in labeling complicates the classification process for location and size, being in line with the statement of Patil et al. (2017). Moreover, De Deijn (2018) argues that the classification of damage size suffers from large interference between the categories small and medium, which might be biased by the manual labeling process. In addition, vehicles with multiple damage classes are assigned to the dominant damage class in the image, which could increase class interference. This might result in the large misclassification for co-occurring dents and scratches in the research of De Deijn (2018).

Li et al. (2018) went beyond image classification and extended damage classification with damage localization. They used this technique to develop a system, which is able to identify similar damages to cope with insurance fraud. 1790 images from the internet are extended with 98 vehicles captured from parking lots. Despite the different damage types within their dataset, they focused on detecting damage itself and did not apply multi-class detection. They used the one-stage object detector: YOLO, to detect the damage and point out that localizing vehicle damage is a challenging task. This as, unlike normal object detection, each damage can be of different shape.

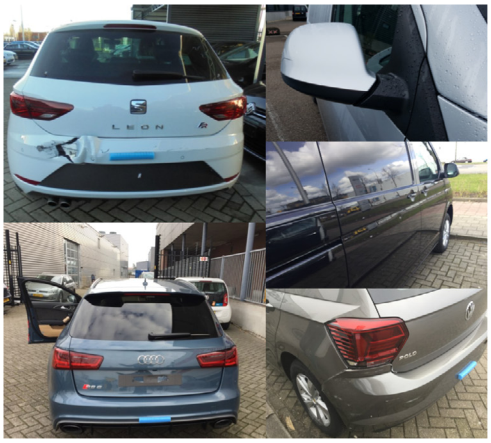
Even though Li et al. (2018) used slightly more images compared to De Deijn (2018) and Patil et al. (2017), they still applied transfer learning to increase the training dataset. To handle the various light conditions, they applied Local Response Normalisation (LRN) layers to reduce the false positives from light reflections. This approach increases the precision from 32.75 percent to 37.96 percent and the recall from 57.58 percent to 81.75 percent. Despite this, their results show that the accurate estimation of a bounding box can be difficult. This falls in line with the statement of Patil et al. (2017), that localizing damage is non-trivial due to the different shapes of the damages.

3. Data

To the best of our knowledge, there is no publicly available dataset for vehicle damage detection. Therefore, we constructed our own dataset and annotated each image manually. Pon provided the first

[1] Company website: Pon.com.

[2] Categories: bumper dent, door dent, glass shatter, head-lamp broken, tail-lamp broken, scratch, smash.


(a) *Damage Dossiers.*

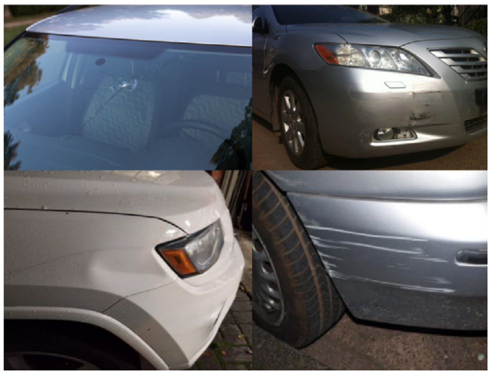

(b) *Damage Web.*

Fig. 1. Dataset excerpts.

dataset, further referenced to as *Damage Dossiers*. To increase the image diversity, we extended this dataset by collecting images from the internet (*Damage Web*) and installed a demo environment with cameras in a light street to mimic real deployment of the model (*Light Street*).

3.1. Damage dossiers

This dataset consists of 2499 *Damage Dossiers*, where each dossier contains multiple[3] images of the damaged vehicle. A diverse range of viewpoints and zoom levels is present, as well as images of vehicles without any damage. Using preprocessing of Section 3.4 we reduced the full dataset from 19,907 images to 3513, of which an excerpt is presented in Fig. 1(a).

The majority of images have been captured using a fixed set of mobile phones, resulting in a low diversity of image dimensions: $\{(w,h) : w \in [640,4800]; h \in [680,4100]\}$. Fig. 2(a) shows the dimension diversity and Fig. 2(b) the object class frequency from the 7901 manual annotated damages. The damage dimensions with respect to the image ranges from 1 percent to 100 percent of the image in both the width and height. Both the range and distribution of the damage sizes varies strongly per damage type. Rust, Tire crack, and Bump covers never more than 50 percent of the image. Whereas, the other classes range up-till or close to 100 percent. Bending and Glass damage is normally distributed, Hail and Cover damage are Beta distributed and all other classes are Log-normal distributed.

3.2. Damage web

To enrich the *Damage Dossiers*, we constructed a dataset with images from the internet. In addition, the external data is used to compare the performance of a model trained on internal data with a model trained on external data. The external data contains images of higher resolution and vehicles are, on average, less clean, making the model potentially more robust.

By use of web-scraping, approximately 2500 images have been extracted from Google image search. Preprocessing, according to Section 3.4, yields a dataset of 1338 images of which Fig. 1(b) provides an excerpt. In contrast to the *Damage Dossiers*, stronger fluctuations are present in the image dimensions: $\{(w,h) : w \in [150,9216]; h \in [130,4800]\}$. Fig. 3(a) visualizes the dimension diversity, whereas Fig. 3(b) presents the object class frequency. The final dataset consists of 3797 damage objects. The damage dimensions with respect to the image size are in line with the findings from Section 3.1.

[3] Average: 8, min: 0, and max: 89.

3.3. Light street

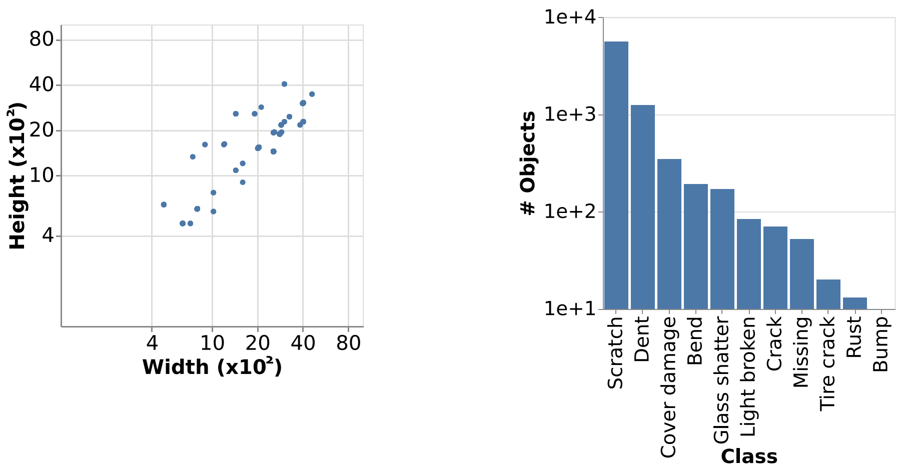
To evaluate the model in a practical setup, we installed cameras in a specially designed light street. This light street is in use for regular damage inspections, making it ideal for practical evaluation of the model. As proof of concept, we installed four cameras to capture the vehicle from multiple sides. Although, not all areas of the vehicle are captured, it enables an initial performance evaluation.

Based on preliminary research on the *Damage Dossiers* dataset, the side, front, rear, rim, and roof are most frequently damaged. To avoid a strong angle between the camera and the rim, we install a camera on both sides as close as possible to the vehicle. The front and rear cameras are placed further away, as no closer mounting option is available. The installed setup is shown in Fig. 4(a), for which the side-cameras are installed at a height of 265 cm to capture the side and roof for each passenger car. The front and rear cameras are placed at a lower height of 200 cm, as the roof is covered by the side cameras and lower placement improves the coverage of the bumper. Since we were limited by the 4 cameras, we are not able to capture the side sill, the bottom of the front bumper, and the bottom of the rear bumper. Furthermore, the doors restrict the cameras from being placed directly in front of the vehicle and behind the vehicle. As such, there exists a small blind-spot at the right front bumper and right rear bumper.

To remove the need for manual interference, a video stream is used instead of pictures. This creates an autonomous system and increases the number of captured frames, giving the flexibility to select the frames where no employees are blocking the view. The Frames Per Second (FPS) has been reduced to the lowest supported configuration of 2 FPS, to minimize the generated data flow. The cameras provide a resolution of 2560 × 1920 at 5 MegaPixel, well above average resolution of the *Damage Dossiers* and the *Damage Web* dataset. Fig. 4(b) shows an excerpt from all four cameras, according to the setup of Fig. 4(a).

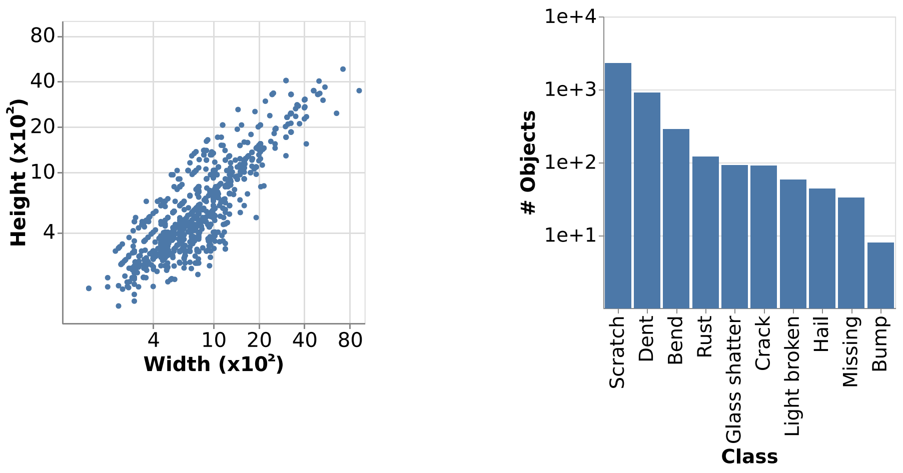
3.4. Preprocessing

We removed duplicate images by comparing the average image hash, inspired by the approach of Zhang et al. (2013). Contrary to Zhang et al. (2013), only exact duplicates have been removed to keep images of the same damage from different angles. Therefore, an average hash distance of zero is used. The duplicate removal is followed by a manual cleaning process, where images are removed when no vehicle is present or when the damage is not visible. Consecutively, we split the data in a train and validation set, according to Section 3.4.1. Lastly, we manually annotate the data with polygons to avoid the exclusion of specific deep learning models.



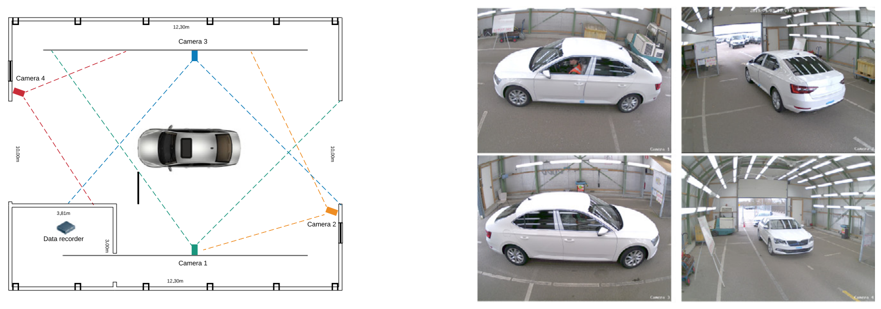
(a) Image dimensions. (b) Object class frequency.

Fig. 2. Statistics of the *Damage Dossiers* dataset.



(a) Image dimensions. (b) Object class frequency.

Fig. 3. Statistics of the *Damage Web* dataset.



(a) Camera setup. (b) Camera excerpt.

Fig. 4. Light Street setup.

3.4.1. *Train and validation split*

We set the train fraction to 80 percent, slightly lower than the 90 percent proposed by Flach (2012). Preliminary results with a 10 percent validation set were significantly depending on the prior selected validation set. Increasing the validation set to 20 percent, removed the dependency between the performance and the prior selected validation set. The requirement for this larger validation set, compared with the recommendation of Flach (2012), might be due to the small dataset size, in combination with the relatively high diversity in capture angles, image sizes, damage types, and vehicle models.

Each image of the *Damage Web* dataset is assigned to the train set with probability 80 percent. Applying the same random image split to the *Damage Dossiers* would leak information from the train set into the validation set, as each dossier can contain multiple images.

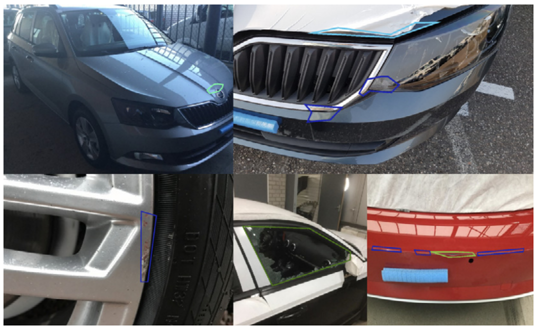


Fig. 5. Excerpts from the annotations.

Therefore, the *Damage Dossiers* are split randomly on a dossier level, with probability 80 percent of ending up in the train set. An assigned dossier to either the train or the validation set assigns all images of the corresponding dossier to the train or validation set.

3.4.2. Annotation process

Each dataset is manually annotated in a detailed manner. We make use of a manual annotation process, since there is currently no computer vision model available that can detect and classify vehicle damages in large detail. Furthermore, we aim to capture the knowledge from domain experts in the annotations during this manual annotation process. The damage labeling *Bump*, *Dent*, *Hail*, and *Scratch* is inspired by the internal damage evaluation system of Pon. According to domain experts from Pon, *Cover Damage* frequently turned out to be vehicle damage. Therefore, *Cover Damage* should alarm the employees on potential vehicle damage. Overlapping damage types, such as a *Scratch* contained within a *Dent*, frequently occur. As both De Deijn (2018) and Patil et al. (2017) show that classification suffers from large class interference, we assign a bounding box to each damage. With this, a *Scratch* can be annotated within a *Dent*, potentially improving the learning process. Fig. 5 provides an excerpt of the annotated images. The following object classes are used:

- Bend
- Bump
- Cover Damage
- Crack
- Dent
- Glass Shatter
- Hail
- Light Broken
- Missing
- Rust
- Scratch
- Tire Crack

Bounding box dimensions differ within and across classes. Especially *Scratch*, *Dent*, and *Cover Damage* have a diverse range of bounding box sizes. *Crack*, *Tire Crack*, *Bump*, and *Rust* are relatively small. *Hail* damage is mostly large, as it is frequently affecting the complete vehicle. Therefore, object size and class are expected to correlate.

4. Methodology

We conduct six consecutive experiments to train and optimize the damage detection and to evaluate its performance. The experiments are listed below and further described in the following sections:

1. Optimize Hyperparameter.
2. Compare Transfer Learning and Fine-Tuning Methods.
3. Compare Models and Backbones.
4. Estimate Robustness and Train Data Influence.
5. Compare Performance between Model and Domain Experts.
6. Evaluate Performance in a Specially Designed Light Street.

All experiments make use of the Adam optimizer, in combination with an adaptive learning rate schedule, which reduces the learning rate by a factor of 0.5 if the evaluation loss did not improve for three consecutive epochs. To limit overfitting, we set the L1 regularization to $\alpha = 1e^{-4}$, following the approach of Liu et al. (2016). The threshold for the Jaccard overlap in YOLO v3 and all SSD models is set to 0.5 and the Non-Maximum Suppression (NMS) threshold to 0.5, as proposed by Liu et al. (2016) and Redmon and Farhadi (2018). Lastly, we set the maximum number of detections to 100 to restrict the computation time of the NMS algorithm. To ensure

4.1. Optimize hyperparameter

This experiment evaluates the impact of different parameters on the performance of YOLO v3 for the *Damage Web* dataset. We focus on hyperparameter optimization in particular, as previous research in damage detection did not evaluate this effect to a large extent. Since we use YOLO v3 for the initial parameter tuning, we make use of the default loss function of YOLO v3 which is developed by Redmon and Farhadi (2018).

Augmentation

We optimize cropping and padding using grid-search, recommended by Liu et al. (2016) for SSD models, in the domain $\{(\alpha_{crop}, \alpha_{pad}) : \alpha_{crop} \in \{0.1, 0.3, 0.5, 0.7\}$ and $\alpha_{pad} \in \{1.1, 1.3, 1.5, 1.7\}\}$. In addition, we implement Horizontal Flipping and evaluate the effect of: Brightness Adjustment, Gaussian blur, and Rotation. This approach might improve the robustness of the model for light fluctuation, quality loss of low-quality cameras, and different camera angles. Furthermore, most scratches are horizontally orientated. Therefore, rotation might improve the detection of vertical and diagonal oriented scratches.

Input Normalization

The model performance strongly depends on the used normalization technique. We define $x_{lijk} \in [0, 255]$ as the original pixel intensity in row i, column j, and channel k for image l. Here, we define $k \in RGB$, where $RGB = \{red, green, blue\}$. Define $\tilde{x}_{lijk}$, as the normalized pixel intensity, in a similar way. Then the mean pixel intensity of image l and channel k is given by Eq. (1). Lastly, the channel means for a dataset with N instances is defined by Eq. (2). Using these definitions, we compare the following normalization techniques:

- **Dataset Mean**: Scale each image channel (RGB) to have zero mean across the entire dataset: $\tilde{x}_{lijk} = \frac{x_{lijk}}{x_{...k}}$ $\forall l, i, j, k$.
- **Image Mean**: Scale each image channel (RGB) to have zero image mean: $\tilde{x}_{lijk} = \frac{x_{l..k}}{x_{...k}}$, $\forall l, i, j, k$.
- **[0, 1] Scaling**: Scale each pixel to the range $[0, 1]$: $\tilde{x}_{lijk} = \frac{x_{lijk}}{255}$, $\forall l, i, j, k$.

$$x_{l..k} = \frac{1}{H_{img} W_{img}} \sum_{i=1}^{H_{img}} \sum_{j=1}^{W_{img}} x_{lijk}. \tag{1}$$

$$x_{...k} = \frac{1}{N} \sum_{l=1}^{N} x_{l..k}. \tag{2}$$

Image Resize

We compare two image resize methods: resizing while preserving the aspect ratio and resizing while ignoring the aspect ratio. When preserving, the image is resized to have the largest dimension fit the target size and is placed randomly in a $n \times n$ target canvas. We use random placement on the canvas to make the model more robust. When ignoring the aspect ratio, the image gets resized and stretched to fit the target canvas. We use the inter cubic resize method, since Atwood (2017) points out that inter cubic preserves more information compared with Inter Linear and Nearest Neighbor for downsizing images. We prefer this method, since, Section 3 shows that most images are larger than the input size of the network.

Batch Size and Learning Rate

Previous research on damage detection did not evaluate on the optimal batch size and learning rate. Therefore, we optimize these variables using a 2D grid-search to incorporate the dependence between the learning rate and batch size. We search in the domain: $\{(LR, BS) : LR \in [1^{-3}, 5^{-3}, 1^{-4}, 5^{-4}, 1^{-5}];\ BS \in [16, 32, 64]\}$.

4.2. Compare transfer learning and fine-tuning methods

We transfer weights from task X into our model and use fine-tuning to optimize the weights for our specific task Y. We prefer this method over the use of a Support Vector Machine (SVM) on top of a feature extraction layer, as Pu et al. (2019) explain that the latter method requires the original task X to be close to the new task Y.

Patil et al. (2017) explain that transfer learning for damage detection works better when the initial model is trained on a broad set of objects. Therefore, we use pre-trained models for PASCAL VOC 2012 and COCO 2014, with twenty and eighty object classes, respectively. We compare both datasets as Pont-Tuset and Gool (2015) show that COCO 2014 contains smaller objects, making the performance potentially better for small damages. To avoid overfitting, we initially train only the extra layers, confidence layers, and location layers. The backbone, as well as any normalization and pyramid layers, are initially frozen. Based on the first results, we evaluate the impact of unfreezing more layers on the mAP, training loss, and validation loss.

Anchor Boxes

Liu et al. (2016) provide a dataset-independent calculation for the construction of the default anchors for SSD. Therefore, the proposed default anchors from the initial authors of SSD (Liu et al., 2016), FSSD (Li & Zhou, 2017), and RFB-SSD (Liu et al., 2018) will be used. Redmon and Farhadi (2017) propose to use K-means clustering, to construct 9 default anchors for each dataset. Therefore, we normalize each bounding box, according to Eq. (3), to compensate for the differences in image size. The normalized boxes are represented by $\widetilde{H}_{bb} \in [0, 1]$ and $\widetilde{W}_{bb} \in [0, 1]$, whereas the original bounding box is of size $H_{bb} \times W_{bb}$ and the image of size: $H_{image} \times W_{image}$. Consequently, we apply K-means clustering for each dataset with $k = 9$, as proposed by Redmon and Farhadi (2017).

$$\left(\widetilde{W}_{bb}; \widetilde{H}_{bb}\right) = \left(\frac{W_{bb}}{W_{image}}; \frac{H_{bb}}{H_{image}}\right). \tag{3}$$

4.3. Compare models and backbones

Although it is explained by Liu et al. (2016) and Redmon et al. (2015) that one-stage models largely outperform two-stage models, we incorporate R-CNN in the comparison to serve as a benchmark for the two-stage models. More specifically, we make use of Faster R-CNN, proposed by Ren et al. (2017), which is an optimization of Fast R-CNN (Girshick, 2015) in terms of speed.

To meet real-time inference, we evaluate multiple single-shot object detection models. We include two major variants, You Only Look Once (YOLO) proposed by Redmon et al. (2015) and Single Shot multi-box Detector (SSD) proposed by Liu et al. (2016). We incorporate several improvements and extensions, as Liu et al. (2016) and Redmon et al. (2015) point out that YOLO and SSD lack detection performance for small objects. For YOLO, we solely use the latest variant (YOLO v3, Redmon and Farhadi (2018)), which has been iteratively improved from YOLO and YOLO 9000 (Redmon & Farhadi, 2017) to detect small objects using multi-scale detection and up-sampling (Redmon & Farhadi, 2018).

To overcome the lack of detection performance for small objects with the SSD model, we incorporate several model extensions. We use Feature Fusion Single Shot multi-box Detector (FSSD) and Receptive Field Block Single Shot multi-box Detector (RFB-SSD), proposed by Li and Zhou (2017) and Liu et al. (2018), respectively. Both models aim to provide more contextual information to the classifier, being especially beneficial for small objects. We exclude Deconvolutional Single Shot Detector (DSSD) from the comparison, which uses a deconvolutional module to transfer contextual information with a similar goal (Fu et al., 2017). We exclude the DSSD configuration, as Liu et al. (2018) explain that the performance improvement of DSSD is largely due to the ResNet-101 backbone. Furthermore, Li and Zhou (2017) describe that FSSD reduces the incorrect detection of multi-parts of an object, making it more beneficial over DSSD.

We compare all one stage models across four backbones: Darknet-53 (Redmon & Farhadi, 2018), ResNet-50 (He et al., 2016), VGG-16 (Simonyan & Zisserman, 2014), and MobileNet v2 (Sandler et al., 2018). The first three are selected due to proven performance. MobileNet v2 is added to give a fair estimation of the performance when implemented on a mobile device (Sandler et al., 2018). For R-CNN, we solely use ResNet-50 as R-CNN serves as a benchmark and is expected to perform worse, compared with the one-stage methods. Furthermore, the low FPS of R-CNN makes it less applicable in practice.

Since we use different models, we train each model using the proposed loss function from the paper where the model was originally proposed. We therefore report the loss values of a model with respect to its own loss function. However, when comparing different models, we solely compare using the mAP. When different training strategies are proposed for the same model.

4.4. Estimate robustness and train data influence

With this experiment, we identify how additional data affect the overall model performance. We construct a cross-performance between training a model on dataset y and evaluating the model on dataset x, where $(x, y) \in \{\text{Damage Web}, \text{Damage Dossiers}, \text{Damage Web} + \text{Damage Dossiers}\}$[2]. Using this unique approach, we are able to identify the robustness of the model by analyzing how a model trained on one dataset performs on another dataset. Furthermore, the effect of adding an external dataset (*Damage Web*) to an internal dataset (*Damage Dossiers*) can be described.

4.5. Compare performance between model and domain experts

We compare the performance of the model with domain experts to evaluate the practical relevance. To make the comparison, we used 100 images from the *Damage Dossiers* dataset, instead of the *Damage Web* dataset, as the first dataset has validated ground truth labels from the repair process. Two domain experts receive the 100 images and are allowed to debate about each image annotation as long as needed, to mimic insurance claim procedures. Each annotation consists of a bounding box and a class name. The model receives the same images, and the results are compared in an aggregated way. To ignore the different ways of annotating damage, we count each damage class only once in each image and assign *True* if the model/experts predicts the damage class correctly. This corresponds to having a bounding box overlap of at least 30 percent with the ground truth. We assign *False* otherwise.

4.6. Evaluate performance in a specially designed light street

To evaluate the applicability of the model in a practical setting, we designed the detection pipeline that is visualized in Fig. 6. The model performance is evaluated in parallel with a manual inspection of 3 min, conducted by two employees. To reduce the number of frames in this time period, we sub-sample the video-stream by a factor 10. For each frame, object detection is executed for vehicles and persons simultaneously. All frames without a vehicle are excluded, and detected persons are blurred for privacy matters. We took images from 50 vehicles with detected damage. Furthermore, we added 50 vehicles without detected damage in the supply chain. To ensure this, we sampled 50 undamaged vehicles from the vehicles which have been transported to a dealer or

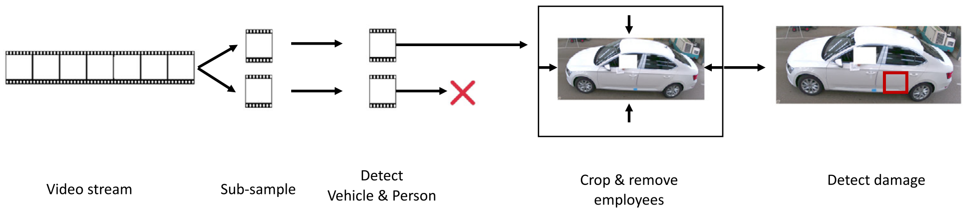


Fig. 6. Preprocessing steps on the Light Street video stream.

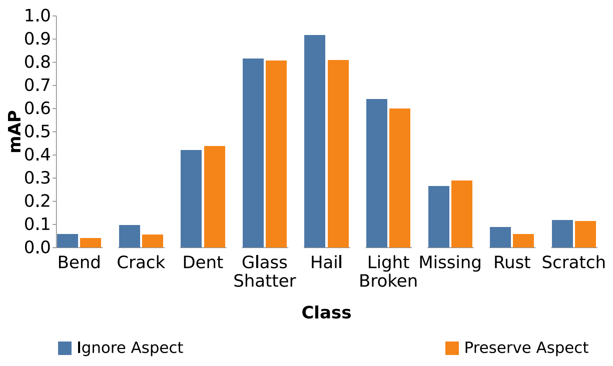


Fig. 7. mAP comparison for resize without perseverance of the aspect ratio (blue) and with perseverance of the aspect ratio (orange). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Effect of α_{crop} and α_{pad} on the mAP for YOLO v3 with Darknet-53, learning rate $1e{-}3$, batch size 16, 30 epochs, 0–1 scaling, and ignoring the aspect ratio.

Pad	Crop			
	0.1	0.3	0.5	0.7
1.1	0.170	0.183	0.177	0.153
1.3	0.181	0.198	0.182	0.164
1.5	0.216	0.243	0.220	0.197
1.7	0.239	**0.251**	0.244	0.239

customer. By doing this, we are almost certain that all used negative examples are indeed negative.

To avoid the blocking of the vehicle by the employees, any frame where the employee is between the camera and vehicle is excluded. For this, we define the bounding box of object i as $(x_1^i, x_2^i, y_1^i, y_2^i)$ and its corresponding class as c_i. Using this, we exclude the frames if Condition (4) holds for c_i = Person and c_j = Vehicle. Using this, frames will not be excluded if the driver is recognized. Lastly, the image is cropped to the size of the vehicle, and the image is passed through the damage detection algorithm.

$$\min(y_1^i, y_2^i) < \min(y_1^j, y_2^j) \quad \text{and} \quad (x_1^j, x_2^j) < (x_1^i, x_2^i) < (x_1^j, x_2^j). \tag{4}$$

5. Results

In this section, we describe the results for the six experiments from Section 4.

5.1. Optimize hyperparameter

Table 1 shows the results of the grid-search, where $\alpha_{crop} = 0.3$ outperforms independently of α_{pad} and $\alpha_{pad} = 1.7$ outperforms independently of α_{crop}. The total grid-search yields $(\alpha_{crop} = 0.3, \alpha_{pad} = 1.7)$. Using these settings for the optimization of image normalization, results in normalization by dataset mean (0.286) to outperform 0–1 scaling (0.251) and image mean normalization (0.234).

Table 2
Batch size and learning rate optimization. Using $\alpha_{crop} = 0.3$ and $\alpha_{pad} = 1.7$, 50 epochs, dataset mean scaling, and ignoring the aspect ratio. Results reported in terms of the mAP.

BS	LR				
	$1e^{-3}$	$5e^{-3}$	$1e^{-4}$	$5e^{-4}$	$1e^{-5}$
16	0.286	0.289	0.291	0.287	0.286
32	0.303	0.313	**0.333**	0.292	0.288
64	0.234	0.216	0.207	0.196	0.179

Table 3
Effect of augmentation on scratch detection, using a subset of images which contains at least one scratch. Using hyperparameters: $\alpha_{crop} = 0.3$, $\alpha_{pad} = 1.7$, horizontal flipping($p = 0.5$), resize while ignoring the aspect ratio, $LR = 1e^{-4}$, and $BS = 32$.

Evaluation	1	2	3	4	5	6	7
Dimension	416	680	416	416	416	416	680
Rotation			10	15		30	15
Gaussian Blur					✓	✓	✓
Brightness					✓	✓	✓
mAP	0.067	0.076	0.132	0.152	0.142	**0.161**	0.115
Total loss	4.04	**3.62**	4.11	4.08	4.16	3.92	3.77
Confidence loss	2.33	**2.24**	2.27	2.32	2.32	2.19	2.16
Location loss	1.71	**1.38**	1.84	1.76	1.84	1.73	1.61

No significant improvement has been found when preserving the aspect ratio over ignoring the aspect ratio. On a class level, preserving the aspect ratio increases the mAP for the class *Missing* while ignoring the aspect ratio seems to improve the mAP on the class *Hail* and the class *Scratch*. Fig. 7 shows a detailed class comparison. We ignore the aspect ratio in the further extension of this research as the majority of damages is of class *Scratch*. Optimization of the BS and LR, results in $(BS, LR) = (32, 1e^{-4})$. Table 2 shows that a batch size of 32 outperforms all evaluated batch sizes, where an increased batch size of 64 demands a higher learning rate.

Fig. 7 indicates large performance diversity between the damage classes. *Bend*, *Crack*, *Rust*, and *Scratch* are lacking behind in performance. Although the performance on these four classes is relatively low, the number of objects of class *Scratch* is large. Therefore, all images containing at least one scratch were isolated, and a model was trained on this dataset in isolation.

Table 3 shows the performance on the Scratch dataset for seven different evaluations, where evaluation 1 serves as a benchmark. For a larger image size (evaluation 2), the model tends to locate the objects more precisely. However, the mAP does not increase to a large extent. As the mAP takes objects into account with an IoU of at least 0.5, it gives rise to the idea that an increased image size improves box locations, which already had an IoU of 0.5. Evaluation 3–6 show that the mAP benefits from Rotation, Gaussian Blur, and Brightness adjustment. The best mAP is achieved with evaluation 6.

5.2. Transfer learning and fine-tuning

We start by describing the effect of the anchor size, in combination with the pre-trained weights and close by comparing different fine-tuning approaches.

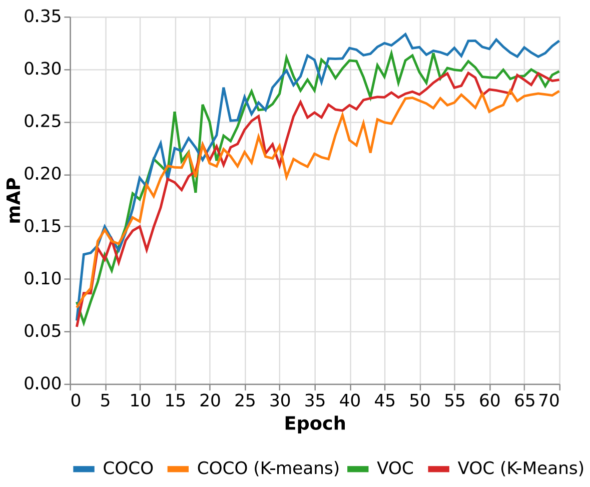


Fig. 8. Influence of pre-trained weights and default anchors on the mAP, using YOLO v3 with Darknet-53.

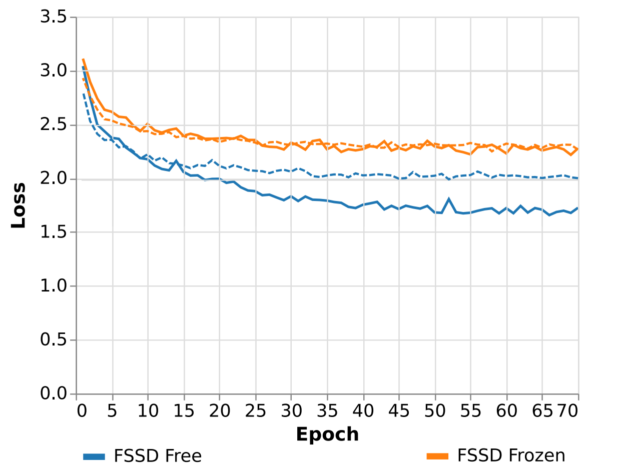


Fig. 9. Confidence loss with *free* (freezing backbone) and *frozen* (extra, confidence and localization trainable).

Anchor Size and Pre-trained Weights

Fig. 8 illustrates the mAP performance for pre-trained weights: COCO and PASCAL VOC 2012, using different anchor sizes: COCO, PASCAL VOC 2012, and K-means clustering as suggested in the paper of Redmon and Farhadi (2018). Surprisingly, the anchors constructed with K-means clustering result in a higher loss and lower mAP, compared with the anchors used during training of the pre-trained weights. This is in contradiction with the results from Redmon and Farhadi (2018). In contrast to the COCO dataset, our dataset has smaller objects. Using K-means clustering, the resulting default anchors are smaller compared to coco. As a result, object classes with relatively large objects (Hail, Glass shatter, and missing) perform worse with our K-means anchors. This might explain the decreased mAP using the K-means anchors.

Fine-Tuning

Table 4 shows the mAP after 70 epochs for two different levels of frozen layers. The category *frozen* freezes all layers and solely trains the extra layers, the location layers, and the confidence layers. The category *free* freezes the base network; all other layers.[4],[5],[6] are trainable. FSSD benefits largely from the additional trainable weights, increasing the mAP by 58.57 percent, where the loss reduction is mainly due to the location loss as shown in Fig. 9 The training loss (solid line) and validation loss (dashed line) shows a high model bias for FSSD frozen, as the training and validation loss are relatively close. Unfreezing the extra layers increases the flexibility of the model, resulting in a lower loss value for both training and evaluation.

As this preliminary research shows promising results for FSSD, we construct a more detailed comparison for the effect of the number of trainable layers on the loss and the mAP. Table 5 summarizes the results of different evaluations, where we consequently increase the number of trainable layers. The validation loss decreases, and the mAP increases until evaluation 4, showing large overfitting when unfreezing the backbone in evaluation 5. The best performance is obtained when the base network is frozen during the initial 50 epochs and set trainable during the last 20 epochs. Using this, the base network is used during the fine-tuning process while the learning rate is already reduced.

[4] FSSD: normalization, extra, transformation, pyramid, location, and confidence.

[5] RFB-SSD: normalization, extras, location, and confidence.

[6] SSD: normalization, extras, location, and confidence.

Table 4
Influence on the mAP when unfreezing additional layers. All previously optimized parameters are used.

Model	SSD	RFB-SSD	FSSD
Frozen	0.273	0.268	0.211
Free	**0.286**	**0.281**	**0.334**
Improvement (%)	4.76	4.85	**58.57**

Table 5
Effect of the number of trainable layers on the loss and mAP. Trained on 70 epochs.

Evaluation	1	2	3	4	5	6
Base					✓	>50
Normalization				✓	✓	✓
Transformation			✓	✓	✓	✓
Pyramids			✓	✓	✓	✓
Extras		✓	✓	✓	✓	✓
Location	✓	✓	✓	✓	✓	✓
Confidence	✓	✓	✓	✓	✓	✓
mAP	0.139	0.211	0.252	0.331	0.221	**0.341**
Training loss	4.644	3.818	3.627	3.427	**2.073**	2.807
Validation loss	4.764	4.281	4.122	3.988	4.315	**3.883**

Table 6
Cross comparison, on the mAP, between object detection algorithms and various backbones trained with frozen base.

Model	FSSD	RFB-SSD	SSD	YOLO v3	R-CNN
Darknet-53	**0.330**	0.275	**0.271**	**0.333**	–
ResNet-50	0.254	0.272	0.257	0.297	**0.241**
VGG-16	0.278	**0.298**	0.234	0.257	–
MobileNet v2	0.273	0.207	0.223	0.253	–

5.3. *Object detection algorithm and backbone*

Table 6 summarizes the results for different object detection algorithms with various backbones. The best performing configurations are YOLO v3 with Darknet-53 and FSSD with Darknet-53. Inspired by the results of Table 5, we add the base network to the trainable layers and train for an additional 20 epochs. The performance of YOLO v3 improves from 0.333 to 0.413, whereas FSSD improves from 0.330 to 0.341. No significant improvement has been found for SSD and RFB-SSD. A breakdown per object class, visualized in Fig. 10, shows that especially the lower performing classes benefit from unfreezing the base network.

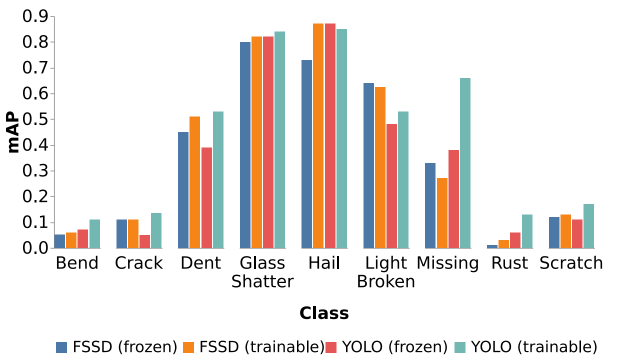


Fig. 10. mAP comparison per class for FSSD and YOLO v3 before unfreezing the base network and after unfreezing the base network.

Table 7
mAP performance for the best model, trained on the dataset of axis y and evaluated on the dataset of axis x. Excluded *Hail*, *Bump*, *Cover Damage*, and *Tire Crack*.

	Damage Web	Damage Dossiers	All Data
Damage Web	**0.413**	0.159	0.280
Damage Dossiers	0.170	0.332	0.247
All Data	0.293	**0.373**	**0.333**

5.4. Damage Dossiers vs. Damage Web

The two best performing models on the *Damage Web* dataset are used to set a benchmark on the *Damage Dossiers*. YOLO v3 with Darknet-53 backbone is used, as well as FSSD with Darknet-53. We first optimize the LR and BS in a similar way as done for the *Damage Web* dataset. The best mAP (0.311) is achieved with batch size 32 and learning rate $5e^{-3}$, which slightly differs from the *Damage Web* dataset. All other parameters are equivalent to the optimized parameters from the *Damage Web* dataset.

The *Damage Dossiers* dataset is more sensitive to overfitting as the model directly started to overfit after unfreezing the backbone Therefore, we resumed training from epoch 50 and added heavier data augmentation with a padding of 2.0 and cropping of 0.1. This improved the validation loss of the FSSD model from 3.021 to 2.781 (confidence) and from 2.391 to 1.942 (localization). The same methodology was followed for YOLO v3. In contrast with the results of the *Damage Web* dataset, FSSD outperforms YOLO v3 on all classes. Fig. 11(a) shows excerpts from the detection result on the *Damage Dossiers* dataset. It indicates the applicability of the model to locate the damage and classify the damage into the correct category. The robustness of the model against water, is displayed in Fig. 12.

5.5. Training data influence

Table 7 shows the used training data in the rows and the evaluation data in the columns. To construct an unbiased comparison, we compare all models on the same classes. Therefore, we remove the class *Hail* from the *Damage Web* dataset and removed the classes *Bump*, *Cover Damage*, and *Tire Crack* from the *Damage Dossiers*. Increased performance on the *Damage Dossiers* is achieved when the *Damage Web* data is added to the training instances. Contrary, the performance on the *Damage Web* data decreased when the *Damage Dossiers* are added to the training instances.

5.6. Compare performance between model and domain experts

The model used approximately 1 min on a CPU to process the 100 images, where the domain experts used approximately 2 h and 15 min. The confusion matrix in Table 8 shows that both the experts and FSSD Darknet-53 made a relatively low number of errors between the damage classes. The last column presents the false positives, whereas the false negatives are shown in the last row. Compared with the experts, the model is especially better in detecting the class *Bend*, as well as the class *Cover Damage*. Contrary, the experts seem to be better at detecting the classes *Dent* and *Scratch*. Fig. 11(b) compares the performance of the experts (left) with the performance of FSSD with Darknet-53 (right). The first row shows that the experts assigned the label *Scratch*, instead of *Scratch* and *Dent*. The second row shows that the model did not detect the scratch at the rim and incorrectly detected a scratch on the tire. The examples indicate that the model seems localizes the damages in more detail.

5.7. Evaluate performance in a specially designed light street

Table 9 shows the confusion matrix, constructed in a similar way as done in Section 5.6. The model shows a relatively low precision with a moderate recall. It has a high number of false positives, ending up in the last column of the confusion matrix. The confusion between the different damage classes is relatively low. With this, the model seems to have trouble identifying the difference between background and damage. When damage is present, the model seems to accurately predict the correct damage class.

We noticed that the environmental variables in the Light Street are strongly influencing the detection performance. Fig. 13 (bottom row) illustrates that the strong light influence, in combination with the low resolution, is creating multiple false positives. Fig. 13 (middle row) shows a vehicle at the Light Street before removal of the cover (left) and after removal of the cover (right). The damage below the cover is not detected (left), but the right image shows that the model accurately detected the partly removed cover and the *Dent*. The top row of Fig. 13 shows that the model accurately detects the *Dent* for both vehicles.

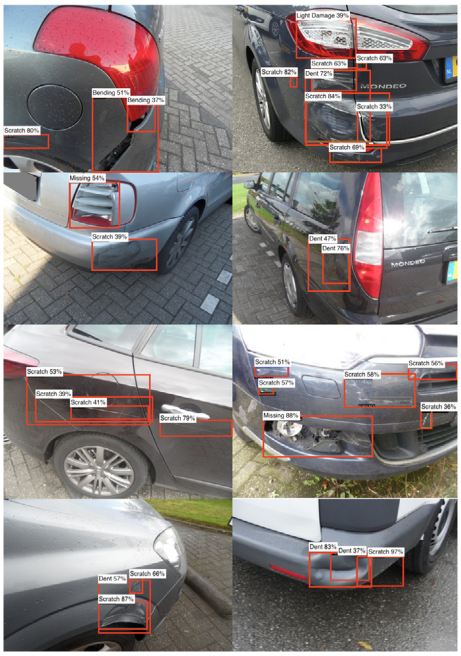
6. Conclusion

We showed that deep learning is able to accurately detect and classify vehicle damages. Our approach of detection and classification showed that the damage detection results in a relatively low interference between classes. FSSD with Darknet-53 and YOLO v3 with Darknet-53 yields the best mAP on, respectively, the *Damage Dossiers* and *Damage Web* dataset. With this, damage detection seems to benefit from models that focus on small objects and contextual information. We showed that, in our case, external data for training enhances the detection performance on the internal dataset, but not the other way around. We quantified the effect of different strategies for freezing and unfreezing network layers during training, achieving more than 50 percent improvement on the mAP.

Our model achieved performance comparable with domain experts and constructs bounding boxes in more detail. Domain experts make slightly fewer false negatives for *Dents*, where the model outperforms in the classes *Bend* and *Cover Damage*. Additionally the model is less accurate in identifying the background (no damage) from damage within the light street, but is still able to distinguish the different damage classes.

Table 8
Confusion matrix with the prediction on the rows and the ground truth on the columns with confidence threshold 0.25.

	Domain experts									FSSD Darknet-53								
	Bend	Cover Damage	Crack	Dent	Glass Shatter	Light Broken	Missing	Scratch	*No Damage*	Bend	Cover Damage	Crack	Dent	Glass Shatter	Light Broken	Missing	Scratch	*No Damage*
Bend	3									8								
Cover Damage		5									8							
Crack			2									2						
Dent				35				1	5				27				3	3
Glass Shatter					8									7				
Light Broken						2									3			
Missing							2									2		
Scratch								47	4				3				42	5
No Damage	6	4	1	2	2	1		3	–	1	1	1	7	3			6	–


(a) *Damage Dossiers.*

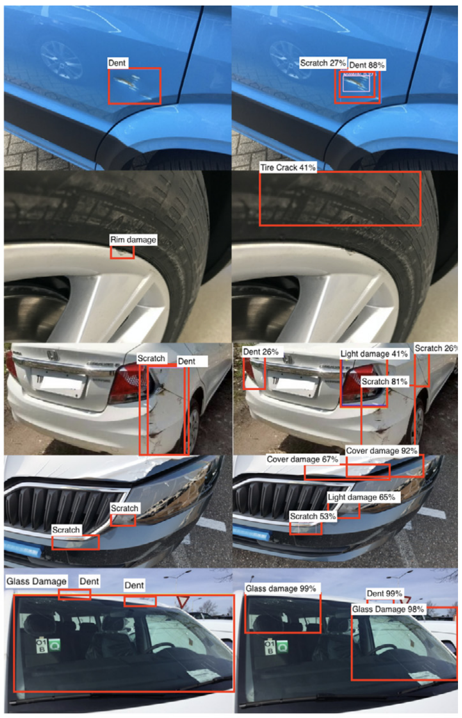

(b) Domain experts (left) and FSSD Darknet-53 with confidence threshold 0.25 (right).

Fig. 11. Excerpts from the detection results.

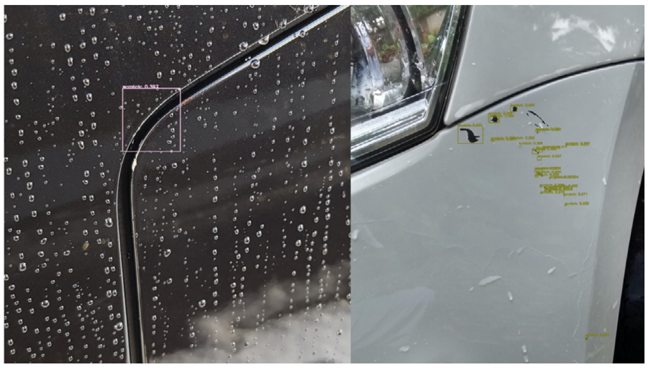


Fig. 12. Damage detection on surfaces with water.

7. Discussion

As a result of the small dataset, robustness against different light conditions, camera angles, and zoom levels are not optimal. The *Damage Dossiers* contain more reflections, compared with images from the web, which might explain the lower mAP. Evaluating the model in the light street, shows that the model got largely influenced by strong light reflections, resulting in many *false positives*. Therefore, we suggest exploring the possibilities of reflection removal, such as polarizing filters, before applying CNNs. Additionally, the localization of each damage is non-trivial, making the reported performance largely influenced by the annotators' ground truth. We recommend further research to implement cross-validation between annotators and study the effect of different annotation granularities. mAP comparison between classes is rather ambiguous, since the non-triviality in class labeling

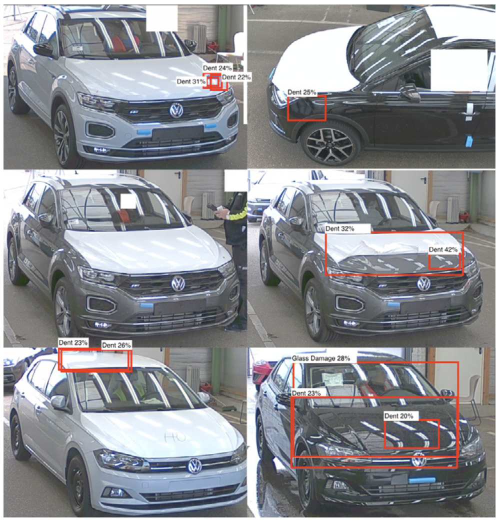


Fig. 13. Excerpts from the light street detection results.

Table 9
Confusion matrix with the prediction (rows) and the ground truth (columns) with confidence >0.25. Images without predicted or ground truth damage excluded.

	Bend	Cover Damage	Crack	Dent	Glass Shatter	Light Broken	Missing	Scratch	*No Damage*
Bend	2								1
Cover Damage		9							23
Crack									
Dent				14				1	43
Glass Shatter					1				18
Light Broken									3
Missing									
Scratch								2	8
No Damage		4		6		1		8	–

is stronger present for class *Scratch* than for class *Glass Shatter*. To limit the validation time, we used a relatively small dataset with few domain experts. As a result, the sparse confusion matrix made it hard to compare the performance in detail which can be improved by adding more domain experts.

CRediT authorship contribution statement

R.E. van Ruitenbeek: Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review and editing. **S. Bhulai:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

We would like to thank Pon for providing data and computer power, required to conduct this research.

References

Ahmed, E., Jones, M., & Marks, T. K. (2015). An improved deep learning architecture for person re-identification. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition, Vol. 7* (pp. 3908–3916). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2015.7299016.

Atwood, J. (2017). Better image resizing. URL: https://blog.codinghorror.com/better-image-resizing/ (accessed 2020-02-15).

Bodnarova, A., Bennamoun, M., & Latham, S. (2002). Optimal gabor filters for textile flaw detection. *Pattern Recognition*, *35*(12), 2973–2991. http://dx.doi.org/10.1016/S0031-3203(02)00017-1.

Brincker, R., Kirkegaard, P. H., Andersen, P., & Martinez, M. E. (1995). Damage detection in an offshore structure.

Cha, Y.-J., Chen, J., & Büyüköztürk, O. (2017). Output-only computer vision based damage detection using phase-based optical flow and unscented Kalman filters. *Engineering Structures*, *132*, 300–313. http://dx.doi.org/10.1016/j.engstruct.2016.11.038, URL: https://linkinghub.elsevier.com/retrieve/pii/S014102961631313X.

Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. *Computer-Aided Civil and Infrastructure Engineering*, *32*(5), 361–378. http://dx.doi.org/10.1111/mice.12263.

Chen, L. C., Hsieh, J. W., Yan, Y., & Chen, D. Y. (2015). Vehicle make and model recognition using sparse representation and symmetrical SURFs. *Pattern Recognition*, *48*(6), 1979–1998. http://dx.doi.org/10.1016/j.patcog.2014.12.018.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR 2005: Vol. I, Proceedings - 2005 IEEE computer society conference on computer vision and pattern recognition* (pp. 886–893). http://dx.doi.org/10.1109/CVPR.2005.177.

De Deijn, J. (2018). *Automatic car damage recognition using convolutional neural networks* (MSc thesis), (p. 56). Vrije Universiteit Amsterdam, URL: https://science.vu.nl/en/Images/stageverslag-deijn_tcm296-882561.pdf.

Flach, P. (2012). *Machine learning: the art and science of algorithms that make sense of data* (p. 415). Cambridge: Cambridge University Press, http://dx.doi.org/10.1017/CBO9780511973000, URL: http://ebooks.cambridge.org/ref/id/CBO9780511973000.

Fu, C.-Y., Liu, W., Ranga, A., Tyagi, A., & Berg, A. C. (2017). DSSD : Deconvolutional single shot detector. arXiv:1701.06659. URL: https://arxiv.org/abs/1701.06659.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440–1448). Santiago, Chile: IEEE, http://dx.doi.org/10.1109/ICCV.2015.169, URL: https://ieeexplore.ieee.org/document/7410526.

Guo, H., Wang, R., Choi, J., & Davis, L. S. (2012). Face verification using sparse representations. In *IEEE computer society conference on computer vision and pattern recognition workshops* (pp. 37–44). http://dx.doi.org/10.1109/CVPRW.2012.6239213.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition, Vol. 2016-Decem* (pp. 770–778). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2016.90, URL: https://ieeexplore.ieee.org/document/7780459.

Hsu, S. C., Chang, I. C., & Huang, C. L. (2018). Vehicle verification between two nonoverlapped views using sparse representation. *Pattern Recognition*, *81*, 131–146. http://dx.doi.org/10.1016/j.patcog.2018.02.031.

Huang, X., Liu, Z., Zhang, X., Kang, J., Zhang, M., & Guo, Y. (2020). Surface damage detection for steel wire ropes using deep learning and computer vision techniques. *Measurement*, Article 107843. http://dx.doi.org/10.1016/j.measurement.2020.107843, URL: https://linkinghub.elsevier.com/retrieve/pii/S026322412030381X.

Isailović, D., Stojanovic, V., Trapp, M., Richter, R., Hajdin, R., & Döllner, J. (2020). Bridge damage: Detection, IFC-based semantic enrichment and visualization. *Automation in Construction*, *112*, Article 103088. http://dx.doi.org/10.1016/j.autcon.2020.103088.

Jayawardena, S. (2013). *Image based automatic vehicle damage detection* (Ph.D. thesis), http://dx.doi.org/10.25911/5D74E7FFC3917, URL: http://hdl.handle.net/1885/11072.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. http://dx.doi.org/10.1145/3065386.

Lee, D. H., Park, K. S., Jo, Y. D., & Choi, S. C. (2006). Long range inspection of city gas pipeline using ultrasonic guided waves. In *12th Asia-Pacific conference on NDT (d)* (pp. 1–5). URL: https://www.researchgate.net/publication/242149816_Long_Range_Inspection_of_City_Gas_Pipeline_Using_Ultrasonic_Guided_waves.

Li, P., Shen, B., & Dong, W. (2018). An anti-fraud system for car insurance claim based on visual evidence. arXiv:1804.11207. URL: http://arxiv.org/abs/1804.11207.

Li, Z., & Zhou, F. (2017). FSSD: Feature fusion single shot multibox detector. arXiv:1712.00960 URL: http://arxiv.org/abs/1712.00960.

Liew, C. K., & Veidt, M. (2009). Pattern recognition of guided waves for damage evaluation in bars. *Pattern Recognition Letters*, *30*(3), 321–330. http://dx.doi.org/10.1016/j.patrec.2008.10.001.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot MultiBox detector. In *European conference on computer vision, Vol. 9905* (pp. 21–37). Springer, http://dx.doi.org/10.1007/978-3-319-46448-0, URL: https://link.springer.com/chapter/10.1007%2F978-3-319-46448-0_2.

Liu, S., Huang, D., & Wang, Y. (2018). Receptive field block net for accurate and fast object detection. In *European conference on computer vision, Vol. 11215* (pp. 404–419). Springer, http://dx.doi.org/10.1007/978-3-030-01252-6.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, *128*(2), 261–318. http://dx.doi.org/10.1007/s11263-019-01247-4.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the IEEE international conference on computer vision, Vol. 2* (pp. 1150–1157). IEEE, http://dx.doi.org/10.1109/iccv.1999.790410.

Matveenko, V. P., Kosheleva, N. A., Serovaev, G. S., & Voronkov, A. A. (2019). Registration of local damage based on the use of fiber-optic strain sensors and numerical simulation results. *17*, In *Procedia structural integrity* (pp. 363–370). Elsevier B.V., http://dx.doi.org/10.1016/j.prostr.2019.08.048.

Patil, K., Kulkarni, M., Sriraman, A., & Karande, S. (2017). Deep learning based car damage classification. In *ICMLA 2017*: *Vol. 2017-Decem*, *Proceedings - 16th IEEE international conference on machine learning and applications* (pp. 50–54). Institute of Electrical and Electronics Engineers Inc., http://dx.doi.org/10.1109/ICMLA.2017.0-179.

Pont-Tuset, J., & Gool, L. V. (2015). Boosting object proposals: From pascal to COCO. In *Proceedings of the IEEE international conference on computer vision, Vol. 2015 Inter* (pp. 1546–1554). http://dx.doi.org/10.1109/ICCV.2015.181, URL: https://ieeexplore.ieee.org/document/7410538?denied=.

Pu, Y., Apel, D. B., Szmigiel, A., & Chen, J. (2019). Image recognition of coal and coal gangue using a convolutional neural network and transfer learning. *Energies*, *12*(9), 1735. http://dx.doi.org/10.3390/en12091735, URL: https://www.mdpi.com/1996-1073/12/9/1735.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition, Vol. 2016-Decem* (pp. 779–788). IEEE Computer Society, URL: http://arxiv.org/abs/1506.02640.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *CVPR 2017*: *Vol. 2017-January*, *Proceedings - 30th IEEE conference on computer vision and pattern recognition* (pp. 6517–6525). Institute of Electrical and Electronics Engineers Inc., http://dx.doi.org/10.1109/CVPR.2017.690.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv. URL: http://arxiv.org/abs/1804.02767.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. http://dx.doi.org/10.1109/TPAMI.2016.2577031, URL: https://ieeexplore.ieee.org/document/7485869.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition* (pp. 4510–4520). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2018.00474.

Shardakov, I., Bykov, A., Shestakov, A., & Glot, I. (2017). Crack control in concrete using shock wave techniques. In *Procedia structural integrity, Vol. 5* (pp. 210–216). Elsevier B.V., http://dx.doi.org/10.1016/j.prostr.2017.07.114.

Shihavuddin, A. S., Chen, X., Fedorov, V., Christensen, A. N., Riis, N. A. B., Branner, K., Dahl, A. B., & Paulsen, R. R. (2019). Wind turbine surface damage detection by deep learning aided drone inspection analysis. *Energies*, *12*(4), http://dx.doi.org/10.3390/en12040676.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. CoRR, 1409.1556. URL: http://arxiv.org/abs/1409.1556.

Zhang, Q., Fu, H., & Qiu, G. (2013). Tree partition voting min-hash for partial duplicate image discovery. In *Proceedings - IEEE international conference on multimedia and expo* (pp. 1–6). San Hise: IEEE Computer Society, http://dx.doi.org/10.1109/ICME.2013.6607460, URL: https://ieeexplore.ieee.org/document/6607460.

Zhang, L., Wu, G., & Cheng, X. (2020). A rapid output-only damage detection method for highway bridges under a moving vehicle using long-gauge strain sensing and the fractal dimension. *Measurement: Journal of the International Measurement Confederation*, *158*, Article 107711. http://dx.doi.org/10.1016/j.measurement.2020.107711.

Zhao, Y. P., Niu, L. J., Du, H., & Bi, C. W. (2020). An adaptive method of damage detection for fishing nets based on image processing technology. *Aquacultural Engineering*, *90*, Article 102071. http://dx.doi.org/10.1016/j.aquaeng.2020.102071.