Complex & Intelligent Systems (2022) 8:3613–3625
https://doi.org/10.1007/s40747-021-00397-8

ORIGINAL ARTICLE

Evaluation of deep learning algorithms for semantic segmentation of car parts

Kitsuchart Pasupa[1] · Phongsathorn Kittiworapanya[1] · Napasin Hongngern[1] · Kuntpong Woraratpanya[1]

Received: 24 January 2021 / Accepted: 10 May 2021 / Published online: 22 May 2021

Abstract

Evaluation of car damages from an accident is one of the most important processes in the car insurance business. Currently, it still needs a manual examination of every basic part. It is expected that a smart device will be able to do this evaluation more efficiently in the future. In this study, we evaluated and compared five deep learning algorithms for semantic segmentation of car parts. The baseline reference algorithm was Mask R-CNN, and the other algorithms were HTC, CBNet, PANet, and GCNet. Runs of instance segmentation were conducted with those five algorithms. HTC with ResNet-50 was the best algorithm for instance segmentation on various kinds of cars such as sedans, trucks, and SUVs. It achieved a mean average precision at 55.2 on our original data set, that assigned different labels to the left and right sides and 59.1 when a single label was assigned to both sides. In addition, the models from every algorithm were tested for robustness, by running them on images of parts, in a real environment with various weather conditions, including snow, frost, fog and various lighting conditions. GCNet was the most robust; it achieved a mean performance under corruption, mPC = 35.2, and a relative degradation of performance on corrupted data, compared to clean data (rPC), of 64.4%, when left and right sides were assigned different labels, and mPC = 38.1 and rPC = 69.6% when left- and right-side parts were considered the same part. The findings from this study may directly benefit developers of automated car damage evaluation system in their quest for the best design.

Keywords Semantic segmentation · Object detection · Car parts segmentation · Deep learning

Introduction

Recently, the insurance business has grown rapidly because more people have started to insure their life and property seriously to control the risks of extensive repair costs for a damaged car or property, after an accident. Car insurance is a major insurance business; it is mandatory for cars that have not been fully paid off yet. A crucial process in the operation of a car insurance company is the intricate car damage evaluation process, that requires evaluators to have comprehensive experience and skills in handling car damage. The evaluators will base their task on evidence, e.g., video recorded from car's camera, photos taken from mobile phones showing the damages as pieces of evidence of the accident and log data from IoT devices—for example telematics [1,2]. They must also present their damage evaluation to several parties and estimate the repair cost. This process not only takes a long time, but is also prone to human errors, fatigue or bias. Insurance companies desire to make this process more accurate, without needing to hire many highly paid damage evaluators.

New technology has made computers more powerful: machine learning enables a computer to learn from big data and provide clues for decision makers; computer vision enables a computer to recognize objects in an image or a video clip, which is directly applicable to the business. Edge computing enables front-end devices, e.g., smartphones, to analyze images in real time. This applies to the insurance business too. This technique pushes the heavy computation tasks, e.g., artificial intelligence, computer vision and complex algorithms, from centralized computing to the edge of the network—a front-end device. The front-end device will

✉ Kitsuchart Pasupa
kitsuchart@it.kmitl.ac.th

Phongsathorn Kittiworapanya
60070055@kmitl.ac.th

Napasin Hongngern
59070084@kmitl.ac.th

Kuntpong Woraratpanya
kuntpong@it.kmitl.ac.th

[1] Faculty of Information Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok,10520, Thailand

benefit from privacy, reliability and lower network latency [3–5]. Evaluators can use a smartphone to capture complete views of a car and analyze the captured image or video, in real-time, to evaluate damages and estimate the repair cost instantly. Any insurance company requires photos of damages to an insured car or property as pieces of evidence. Therefore, we brought in the new computer technologies to automate some steps of damage evaluation from photos of the damaged car—(1) identification of car parts; (2) identification of damaged parts; (3) damage evaluation for each part; and (4) repair cost estimation. These steps are illustrated in the schematic diagram in Fig. 1.

Here, we used image segmentation to automatically identify car parts. An image segmentation technique is similar to object detection; it detects where, in an image, an object is located, but adds recognition of the context of the object. An essential difference between the two techniques is that image segmentation works at the pixel level, whereas object detection works at the level of bounding boxes around the object. Image segmentation can be either semantic segmentation, where identical objects in the image are considered to be the same object, or instance segmentation, where identical objects are recognized as different instances. In particular, we used instance segmentation, since we wanted to differentiate different instances of the same object. For example, some car parts come in a left and right pair; instance segmentation enabled us to differentiate between the two members of this pair. A literature review showed that papers on car part segmentation are still limited, and no standards or criteria for this process have been established. Therefore, we tested a set of state-of-the-art deep-learning algorithms on a self-developed car part data set, containing images annotated with descriptions of the object in them. Our contributions are:

1. Development of an extensive car part data set—annotated images of car parts from multiple viewpoints—some were taken from the Internet and some were taken by our team.
2. Comparative evaluation in terms of mean average precision between Mask R-CNN (baseline technique) with ResNet Backbone and four state-of-the-art instance segmentation algorithms—the top four algorithms reported by paperswithcode.com [6].
3. Robustness testing in terms of mPC and rPC of models from four state-of-the-art instance segmentation algorithms and the baseline model against real weather elements and lighting conditions in the photos.

The rest of this paper is arranged as follows: the second section briefly describes related works; the third section briefly describes the tested algorithms; the fourth section discusses the experimental setup and the data sets; the fifth section reports and discusses results, and the final section concludes.

Related works

Edge computing has emerged to push the computation capability closer to end-devices. It can improve response times and reduce required network bandwidth. With a combination of front-end devices, edge nodes and cloud computing, many applications that use machine learning and computer vision techniques have been successfully deployed. Many researchers developed their algorithms to fully operate on front-end devices to enhance system efficiency. Velichko et al. [7] proposed a lightweight neural network algorithm called "LogNNet", that used filters based on logistic mapping for image recognition task. It can be employed in low-memory devices. Howard et al. [8] and Sandler et al. [9] developed MobileNets and MobileNetV2, which are efficient lightweight Convolution Neural Network (CNN) models, designed to work on mobile device. Tuli et al. [10] developed an object detection framework, EdgeLens, that integrated IoT, fog and cloud computing.

Applications of instance segmentation have included detection of individual humans in an image based on their posture. In addition, Zhang et al. [11] presented an instance segmentation method for human detection based on a human pose skeleton. It enabled recognition of the context of a posture even though, in the image, there was another human nearby or an overlap with another human. This capability differentiated it from other instance segmentation algorithms, e.g., Mask R-CNN [12]. Other instance segmentation applications include identification of biological objects in an image. In one instance, Yi et al. [13] presented an instance segmentation method for biological objects, that worked on heat map images.

Currently, several new instance segmentation algorithms have been proposed. For instance, CenterMask [14] did not use a bounding box but used a spatial attention-guided mask. It differed from algorithms that use a fully connected layer, e.g., Mask R-CNN. In addition, it used a fully convolutional one-stage object detector (FCOS) [15] rather than Faster R-CNN [16] in the object detection task, resulting in a higher detection accuracy of both still images and video frames. In another example, Wang et al. [17] developed an instance segmentation algorithm, "SOLO", a one-step algorithm, that did not use bounding box in object detection but, instead, divided an image into a number of squares and detected the interested object in each square. It used a semantic category branch technique to determine semantic category as well as an instance mask branch to determine instance category. SOLO was improved into SOLOv2 [18]. Mask learning was developed based on dynamic convolution. No weights

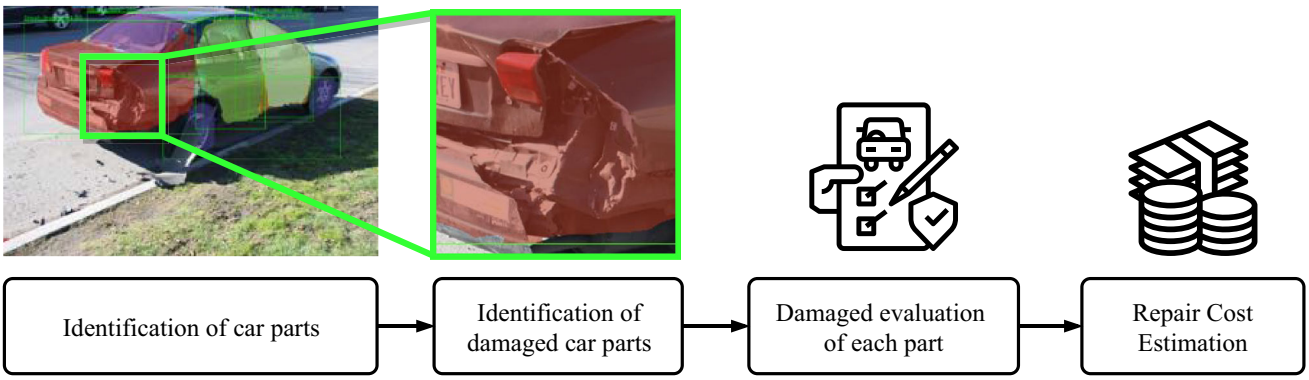

Fig. 1 Car damage evaluation steps

or parameters in the model were set as a fixed value, so that the feature map could be adjusted to various kinds of input. The model had two types of mask branch: a Mask Kernel Branch for learning the convolution kernel and a Mask Feature Branch for learning convoluted features. Non-Maximum Suppression Matrix was used to reduce processing time, which was shorter than any other tested algorithms.

Recently, one-stage instance segmentation methods, that do not have different branches for performing different functions, have gained more attention from researchers than two-stage methods, e.g., PolarMask [19], RDSNet [20] and YOLACT++ [21]. A two-stage method performs object detection first, then constructs a mask branch to predict each mask in a bounding box. Example of these methods are Mask R-CNN, PANet [22] and Mask Scoring R-CNN [23]. Chen et al. [24] proposed a BlendMask with an improved FCOS Object Detector. They added a blender module to an attention map. The blender module included both high- and low-resolution masks in every bounding box mask, enabling the model to predict the mask more accurately and rapidly than Mask R-CNN or other two-stage algorithms.

In a review of studies on car part segmentation, Lu et al. [25] presented a semantic segmentation method for car parts, based on landmark assignment and boundaries of each part. They used a graphical model to find relationships between car parts, then a segmentation by a weighted aggregation method (SWA) [26] to pair two nearby landmarks, then a Segment Appearance Consistency (SAC) technique, to connect segments of nearby landmarks, in every level of a hierarchical segmentation and to determine whether the same segment was represented in every hierarchical level. The outcome was a group of pixels that could classify various car parts. Nevertheless, in SAC and hierarchical segmentation for every hierarchical level, the meanings of a car part of different levels differed. In other words, an SAC, after only one round of SWA, was not able to segment all car parts in an image. Singh et al. [27] built a system to detect different car parts and localize their damages. However, the algorithms used in their system—Mask R-CNN, PANet and an ensemble model, that was based on Mask R-CNN and PANet—did not perform well. The MAP was lower than 0.5 across all algorithms. Dhieb et al. [28] used Inception-ResNetV2 to classify damage severity level, localize and detect part damage. Patil et al. [29] and Dwivedi et al. [30] used various CNN models to classify the car part damage, but these works only focused on a small set of car parts.

A website, paperswithcode.com, ranked all instance segmentation methods and determined the state-of-the-art ones [6]. They were benchmarked on various data sets, e.g., PASCAL VOC [31] and Common Objects in Context (COCO) Challenge [32]. Since we needed the best model for instance segmentation of car parts, we evaluated several algorithms on a large COCO Test-dev Task data set with a large number of categories, using Mask R-CNN with ResNet as baseline. The evaluated methods were the top four, as ranked by paperwithcode.com on 30/09/2019, that also used ResNet as Backbone: HTC [33], CBNet [34], PANet [22] and GCNet [35]. These algorithms are briefly described in the next section.

Methodology

The top-ranked algorithms from paperwithcode.com, on 30/09/2019, are briefly described here.

Mask region-based convolutional neural network (Mask R-CNN)

Instance segmentation Mask R-CNN algorithm [12] was a development of Faster R-CNN [16]. Faster R-CNN was only able to detect, where a target object was in an image and recognize it, but Mask R-CNN was also able to perform instance segmentation. Mask R-CNN had two main parts: (1) a backbone that extracted features from an image with Residual Neural Network (ResNet), a CNN 50–101 layers deep [36], in combination with Feature Pyramid Network (FPN) [37]

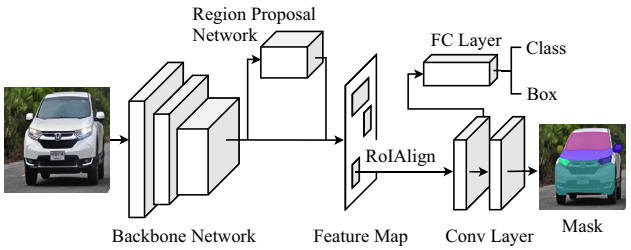

Fig. 2 Mask R-CNN

and (2) a head that constructed a bounding box around a Region of Interest (ROI) and predicted the type of object in the box. The additional step of Mask R-CNN over Faster R-CNN constructed a mask for each ROI. In Mask R-CNN, after the backbone extracted features from an image, these features were input into a Region Proposal Network (RPN), which constructed anchor boxes of various sizes, that contained an object of interest and passed them to an ROI Extractor, that extracted the features from each ROI. Each ROI Map was forwarded to fully connected layers, consisting of two parallel components: the original components of Faster R-CNN for predicting bounding boxes and objects of interest (classification) and an additional component for predicting a mask in a bounding box. The flowchart of Mask R-CNN is illustrated in Fig. 2. Mask R-CNN was ranked number five by paperswithcode.com.

Global context network (GCNet)

GCNet [35] had a similar structure to Mask R-CNN, as can be seen in Fig. 2. However, the ResNet-FPN backbone was augmented with a global context block (Fig. 3). The Non-local Network (NLNet), a part of the block, solved the long-range dependency issue of deep neural networks [38]. NLNet worked in combination with a Squeeze-Excitation Network (SENet) to find the relationships between channels of each feature [39]. GCNet was as effective as NLNet, but computed faster, because it used fewer convolution and operation layers than NLNet. It was ranked number four by paperwithcode.com.

Path aggregation network (PANet)

PANet was developed by Liu et al. [22]. It had a similar structure to Mask R-CNN, as shown in Fig. 4, but the RPN and ROI Extractor were replaced by bottom-up path augmentation and adaptive feature pooling components. The bottom-up path augmentation component took an input from the previous stage and processed it together with an output of each FPN layer to generate feature maps. Those maps were used to better mix high and low-level features. Then, the adaptive feature pooling component processed the feature maps from

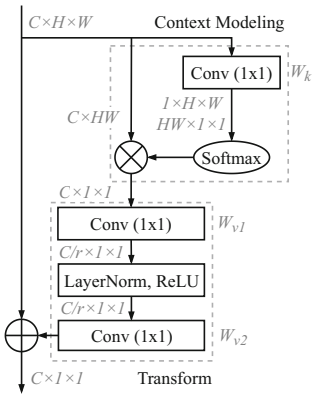

Fig. 3 Global context (GC) block. The feature map has size, $C \times H \times W$—channel number C, height H and width W. $\otimes$ denotes matrix multiplication and $\oplus$ denotes broadcast element-wise addition. r is the bottleneck ratio and C/r denotes the hidden representation dimension of the bottleneck

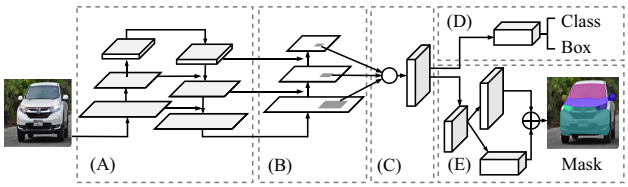

Fig. 4 PANet **A** FPN backbone, **B** bottom-up path augmentation, **C** adaptive feature pooling, **D** Box branch, **E** fully-connected fusion

every layer, concatenated all of its output, then sent them to the head component, consisting of many fully connected layers, to detect objects, construct masks and bounding boxes and classify detected objects. Because of those processes, PANet was highly accurate. It was able to take advantage of all levels of feature maps, from low to high level features in each feature map. PANet was ranked number three by paperswithcode.com.

Cascade mask R-CNN with composite backbone network (CBNet)

This method combined Cascade Mask R-CNN [40] and Composite Backbone Network [34]. First, CBNet improved the feature extraction step, using a number of connected backbones called Assistant Backbones. Each connected backbone extracted some features and sent a feature map to the next backbone, which also extracted some features and sent a new feature map to the next-to-next backbone and so on. The last backbone was called a 'Lead Backbone'. It generated the final feature map, that was consecutively concatenated with features extracted from all previous backbones in the connection. Because of this repeated extraction step, low-level and high-level features were extracted into a more effective mix than a mix that Mask R-CNN generated. Second, Cascade Mask R-CNN, whose head was modified from that of

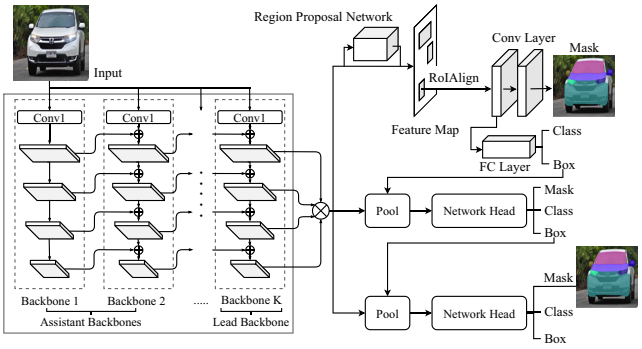

Fig. 5 Cascade mask R-CNN with CBNet. The composite backbone—a combination of assistant backbones and lead backbone—helped improve prediction accuracy

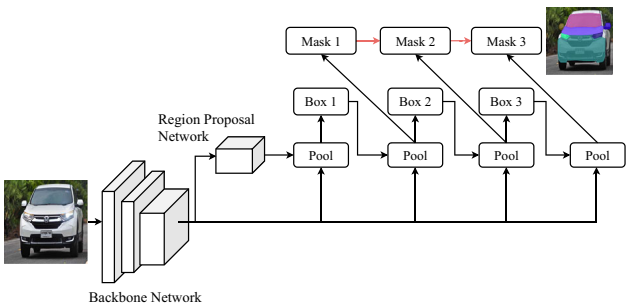

Fig. 6 Hybrid task cascade for instance segmentation

Mask R-CNN, improved prediction accuracy. The bounding box head in a previous branch was forwarded to the ROI Extractor of the next branch to improve prediction accuracy, as illustrated in Fig. 5. This method was ranked number two by paperswithcode.com.

Hybrid task cascade for instance segmentation (HTC)

This algorithm was developed by Chen et al. [33] improving the efficiency of instance segmentation task. In this algorithm, the bounding box head, mask head and ROI extractor were interleaved in a cascade, illustrated in Fig. 6, and so bounding box prediction and mask prediction tasks proceeded in parallel instead of independently. A multi-stage mask branch technique was introduced. It took into account the mask from a previous branch in the generation of a mask in the current branch to improve information flow. Lastly, a semantic mask branch was connected to the head of every mask to enable the model to understand the context of the information in every mask better. All of these features improved the information flow in every task. This method was the top in the paperswithcode.com ranking.

Experimental framework

Data set

The data set contained 500 images of sedans, pickups and sports utility vehicles (SUVs) collected from the Internet and taken from public parking spaces. Images of these vehicles were taken in multiple views—front, back and angled views. The car identification number was blurred to hide individual vehicle details. Each image was annotated by the 18 listed instance masks and bounding boxes: `back_bumper`, `back_glass`, `back_left_door`, `back_left_light`, `back_right_door`, `back_right_light`, `front_bumper`, `front_glass`, `front_left_door`, `front_left_light`, `front_right_door`, `front_right_light`, `hood`, `left_mirror`, `right_mirror`, `tailgate`, `trunk` (of trucks and SUVs), and `wheel` (wheel and tire). The number of instances per category is shown in Fig. 7 and examples of the images in the data set are in Fig. 8. The DSMLR Car Part data set contains images and annotation in COCO Challenge format and is available for download at https://github.com/dsmlr/Car-Parts-Segmentation.

Experimental procedures and settings

We evaluated five algorithms: Mask R-CNN [12], HTC [33], CBNet [34], PANet [22] and GCNet [35], that used ResNet-50 and ResNet-101 as backbones, in terms of correctness and robustness on the car part data set. The algorithms were implemented with an MMDetection toolbox [41]. The experimental steps are described next. First, we resized all input images to 1024 × 1024 pixels, while maintaining the aspect ratio by zero-padding. Next, we randomly partitioned the car part data set into a training set (80% of the entire data set) and a test data set (20%). Then, since it was necessary to determine the best number of epochs for training the model for every evaluated algorithm, we ran a five-fold cross-validation by training for 200 epochs on each fold. The best number of epochs for each algorithm was the number that provided the lowest average five-fold validation loss. Validation loss was computed from 5 types of loss: (1) loss in classification task, (2) loss in bounding box task, (3) loss in segmentation task (Loss mask), (4) RPN loss in classification task and (5) RPN loss in bounding box task. Validation losses (4) and (5) were calculated by a Cross Entropy loss algorithm, embedded in the RPN. Next, we used a Stochastic Gradient Descent (SGD) to finding optimal parameters, setting the learning rate at 0.02 and weight decay at 0.0001. The optimal models were trained with the training set for the optimal number of epochs. The experiment was run five times with different random splits.

Furthermore, we evaluated the robustness of algorithm for semantic segmentation and object detection tasks on

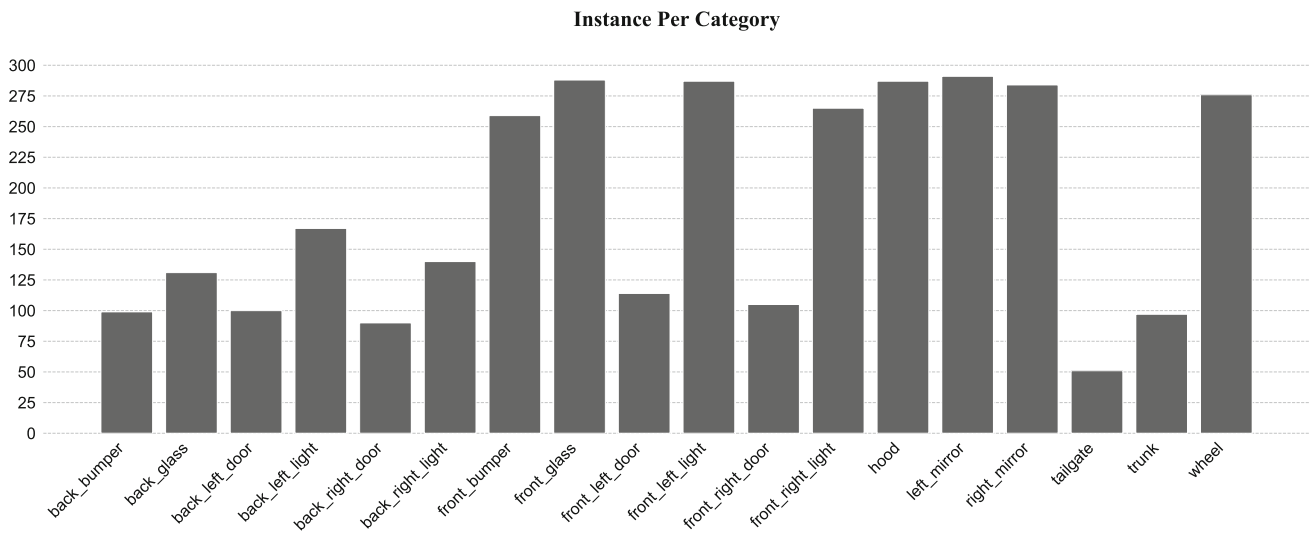

Fig. 7 Number of annotated instances per category for the DSMLR Car Part data set

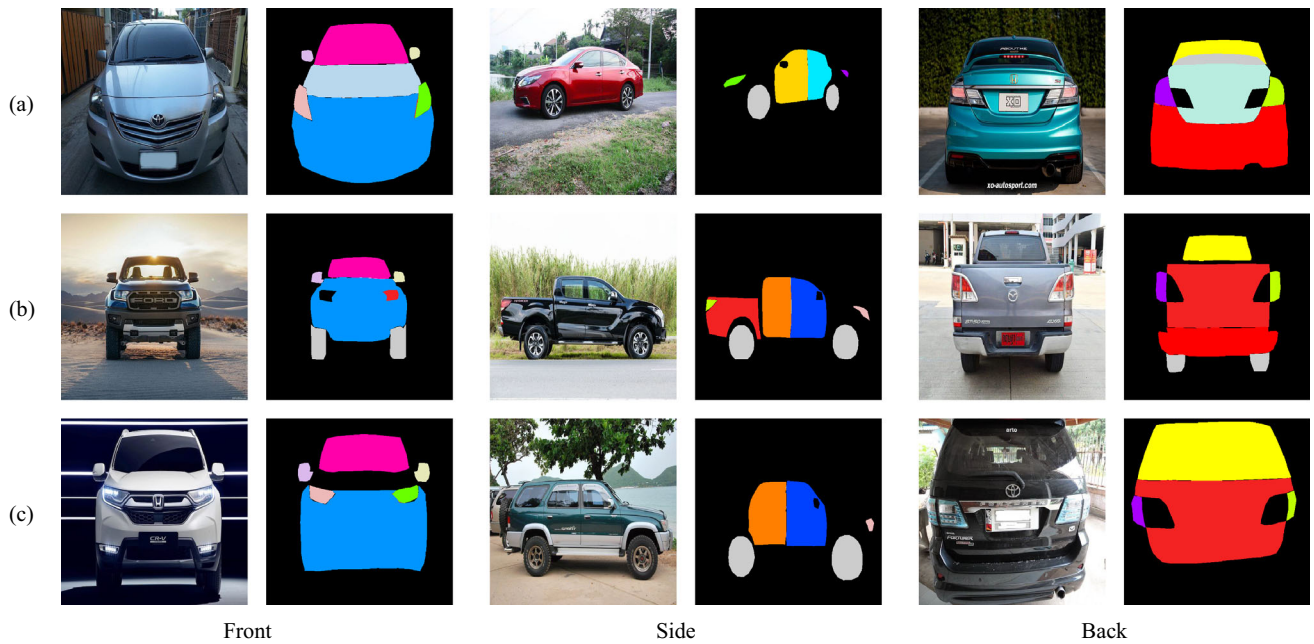

Fig. 8 Samples of pair images and instance mask from the DSMLR Car Part data set: **a** sedan, **b** pickup and **c** sports utility vehicle (SUV)

corrupted data, simulating four real weather conditions and lighting, i.e., snow, frost, fog and ambient light at five levels of severity. The corrupted examples were generated by methods described by Hendrycks and Dietterich [42] (visualized in Fig. 10).

Performance evaluation

Correctness

Each algorithm was evaluated for average precision (AP), based on the COCO Challenge, an established evaluation method for object detection tasks. AP was calculated from the Intersection over Union (IoU) of each interested object. IoU was calculated by

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}. \quad (1)$$

A model was considered to successfully detect an object, if the IoU was equal to or higher than a threshold that we assigned. The AP_{50} and AP_{75} means that the IoU are greater than or equal to the threshold at 0.50 and 0.75, respectively. Then, the mean average precision (mAP), based on COCO Challenge, is the average over IoUs between the threshold at

0.50 and 0.95, computed as:

$$\text{mAP} = \frac{\sum_{i=0}^{9} \text{AP}_{50+5\cdot i}}{10}. \tag{2}$$

Since car parts take different sizes, we also evaluated the AP across scale of the car part, i.e.,APS for small parts with an area lower that 32^2 pixels, AP_M for medium parts, with area between 32^2 and 96^2 pixels, and AP_L for large parts, with area greater than 96^2 pixels. It is noted that AP on the COCO Challenge was reported in percent.

Robustness

Robustness was measured using two metrics—mean performance under corruption (mPC) and relative performance under corruption (rPC) metrics [43].

mPC is calculated:

$$\text{mPC} = \frac{1}{N_c}\sum_{c=1}^{N_c}\frac{1}{N_s}\sum_{s=1}^{N_s} P_{c,s}, \tag{3}$$

where $N_c = 4$ indicates the number of corruptions and $N_s = 5$ the number of severity levels (as set in this work), and $P_{c,s}$ is the performance measure evaluated on test data, that was corrupted with corruption type, c, under severity level, s. Although several metrics could be used for P, in this work, P levels were calculated using mAP. A higher mPC indicates a more robust algorithm.

rPC measured the relative degradation of performance on corrupted data compare to original data. It was calculated by

$$\text{rPC} = \frac{\text{mPC}}{P_\text{original}}, \tag{4}$$

where P_original is the performance of algorithm on the original data, that is mAP of the original data, rPC $\in [0, 1]$. rPC = 1 represents 'perfect' robustness, while 0 represents negligible robustness.

Experimental results and discussion

In this section, several comparisons were made and discussed:

1. We compared overall algorithm performance based on two tasks—object detection and semantic segmentation tasks.
2. We discussed robustness in potential real weather elements and lighting conditions.
3. We further discussed performance and robustness, when left- and right-side parts were grouped under one label.

Overall performance of object detection and semantic segmentation tasks

The performances of all the algorithms are illustrated in Table 1, that includes mAP and AP with different thresholds. It can be seen that HTC with ResNet-101 encoder achieved the best mAP at 54.3 in object detection. In addition, it worked best on small and medium car parts, resulting in AP_S and AP_M at 35.6 and 52.0, respectively. This was followed by HTC with ResNet-50 encoder at 54.1 of mAP. The stricter metric, AP at IoU $\in$ (0.75, 0.95], came in second at 62.4, while HTC with ResNet-50 was the best contender at 63.6. Further, HTC with ResNet-50 performed best on large car parts, resulting in AP_L at 61.1. Surprisingly, Mask R-CNN with ResNet-50—the baseline—scored highest on AP_{50} at 77.0, but it did not perform well on the stricter metric. This was because Mask R-CNN tried to detect, classify and segment the car parts with low-level features, whereas other algorithms used global features or high-level features for segmentation tasks. On the other hand, Mask R-CNN, with the ResNet-101 encoder, achieved the highest mAP at 55.4 in the semantic segmentation task, as well as in the strictest metric, AP_{75}, at 65.2, which is in-line with HTC with ResNet-50 encoder. Here, HTC, with the ResNet-50 encoder, secured the second best mAP at 55.2, with a small difference in mAP from Mask R-CNN with ResNet-101 encoder. It also worked best with the large car parts—AP_L at 63.6. In addition, PANet performed best on small car parts, yielding AP_S at 38.5.

In terms of performance related to the size of the car part, the models performed better on large parts followed by medium and small parts. The average AP_L, AP_M and AP_S across all the models in the object detection task were 55.3, 46.9, and 32.1, respectively, and, in semantic segmentation, the scores were 59.2, 48.7, and 33.2, respectively. Larger parts led to better performance. Figure 9 shows a sample of object detection and semantic segmentation by the models with ResNet-50 and ResNet-101 encoders.

To determine which combination of model and encoder achieved the best overall performance, we used Kendall's coefficient of concordance (W) to measure agreement between evaluation metrics. We rank the 10 candidate models (5 models with 2 encoders each) on 12 performance metrics (2 tasks with 6 metrics each). Then, we reported sum of the ranks of each candidate model as shown in Table 1 that leads to the ranking of the candidate models. Next, we calculate W by

$$W = \frac{12\left(\sum_{i=1}^{k} R_i^2\right) - 3k^2n(n+1)^2}{k^2n(n^2-1) - k\sum_{j=1}^{k}(T_j)}, \tag{5}$$

where n is the number of candidate models, R is the sum of ranks for the i-th candidate, k is the number of the perfor-

Table 1 Overall model performance on object detection and semantic segmentation tasks

Model	Encoder	Object detection						Semantic segmentation						Sum rank	Rank
		mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L		
Mask R-CNN	ResNet-50	50.4	**77.0**	56.8	30.9	47.7	55.2	54.0	77.5	62.5	34.3	51.0	59.4	64.5	6
	ResNet-101	50.8	75.4	59.2	35.4	49.1	54.9	**55.4**	77.0	**65.2**	33.0	**53.0**	60.6	45.5	3
GCNet	ResNet-50	50.9	76.8	57.7	32.3	45.6	56.7	54.6	**78.2**	63.9	34.9	48.3	61.7	52.5	4
	ResNet-101	48.5	63.8	45.1	27.3	35.7	45.6	43.0	64.9	49.6	34.0	40.7	49.2	115.0	10
PANet	ResNet-50	48.8	76.5	56.4	32.9	48.6	51.8	54.0	77.3	63.5	**38.5**	51.4	58.7	63.5	5
	ResNet-101	49.6	73.9	59.8	31.0	46.1	54.2	54.3	75.1	64.8	30.4	49.5	58.9	76.0	8
CBNet	ResNet-50	51.9	71.6	60.8	28.6	48.3	57.9	53.0	72.2	63.0	28.6	48.3	61.7	74.5	7
	ResNet-101	49.5	67.5	58.3	32.2	45.6	55.5	49.5	67.1	57.8	29.6	44.7	56.1	96.5	9
HTC	ResNet-50	54.1	75.7	**63.6**	34.4	50.4	**61.1**	55.2	76.1	**65.2**	36.1	50.2	**63.6**	**29.0**	**1**
	ResNet-101	**54.3**	74.8	62.4	**35.6**	**52.0**	60.0	54.5	75.2	63.0	32.6	50.2	52.0	43.0	2
Average		49.9	73.3	58.0	32.1	46.9	55.3	52.8	74.1	61.9	33.2	48.7	59.2	–	

Bold—best

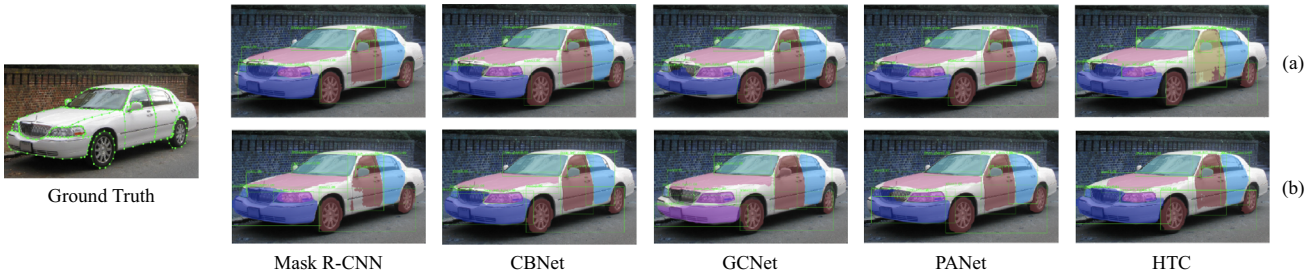

Fig. 9 Sample of object detection and semantic segmentation results: **a** ResNet-50 Encoder and **b** ResNet-101 Encoder

mance metrics, and T is a correction factor, based on tied ranks (see [44] for more details). Here, $n = 10$ and $k = 12$. Thus, W=0.5079 that is transformed to a χ^2 value of W for significance testing against a null hypothesis of no agreement,

$$\chi^2 = k(n-1)W. \tag{6}$$

Thus, $X^2 = 54.8350$ leads to $p < 0.01$ for 9 degrees of freedom. the led to $p < 0.01$. Thus, we rejected the null hypothesis. Therefore, we confirmed that HTC with ResNet-50 and HTC with ResNet-101 are the first and the second rank, respectively.

Robustness

In this section, the models used in the previous subsection were further evaluated. They were tested on the modified test data, including the set of real weather elements and lighting conditions, with different severity levels as shown in Fig. 10. We illustrate the overall robustness test results, showing results for different types of noise for object detection and semantic segmentation tasks in Table 2. GCNet with the ResNet-50 encoder was the best contender; it achieved the highest robustness, based on rPC, in object detection at 64.8% and semantic segmentation at 64.4%. It yielded the best mPC in all weather conditions for both object detection and semantic segmentation tasks, except brightness changes in object detection task. It was clear that the worst was CBNet, with the ResNet-50 encoder, as it retained only 48.1% and 47.3% of the performance in object detection and semantic segmentation tasks, respectively. HTC, with ResNet-101, in the object detection task, achieved the highest mAP, with the normal condition image, but although it only retained 53.2% of the performance, when the images were corrupted, its mPC was still ranked second at mPC= 28.9, after GCNet with ResNet-50. Moreover, HTC with ResNet-101 obtained the highest mPC with the brightness changes at 42.4. This also applied to the semantic segmentation task, HTC, with ResNet-101, ranked second in overall performance, based on mPC = 29.3, similar to Mask R-CNN with ResNet-101. We also found that the factors, that degraded performance for all algorithms, were snow and frost conditions, because they degraded the performance to less than 50% of the performance without corruption in both tasks. However, the algorithms tolerated changes in brightness and fog conditions well: they were still able to keep performance at 79.3%

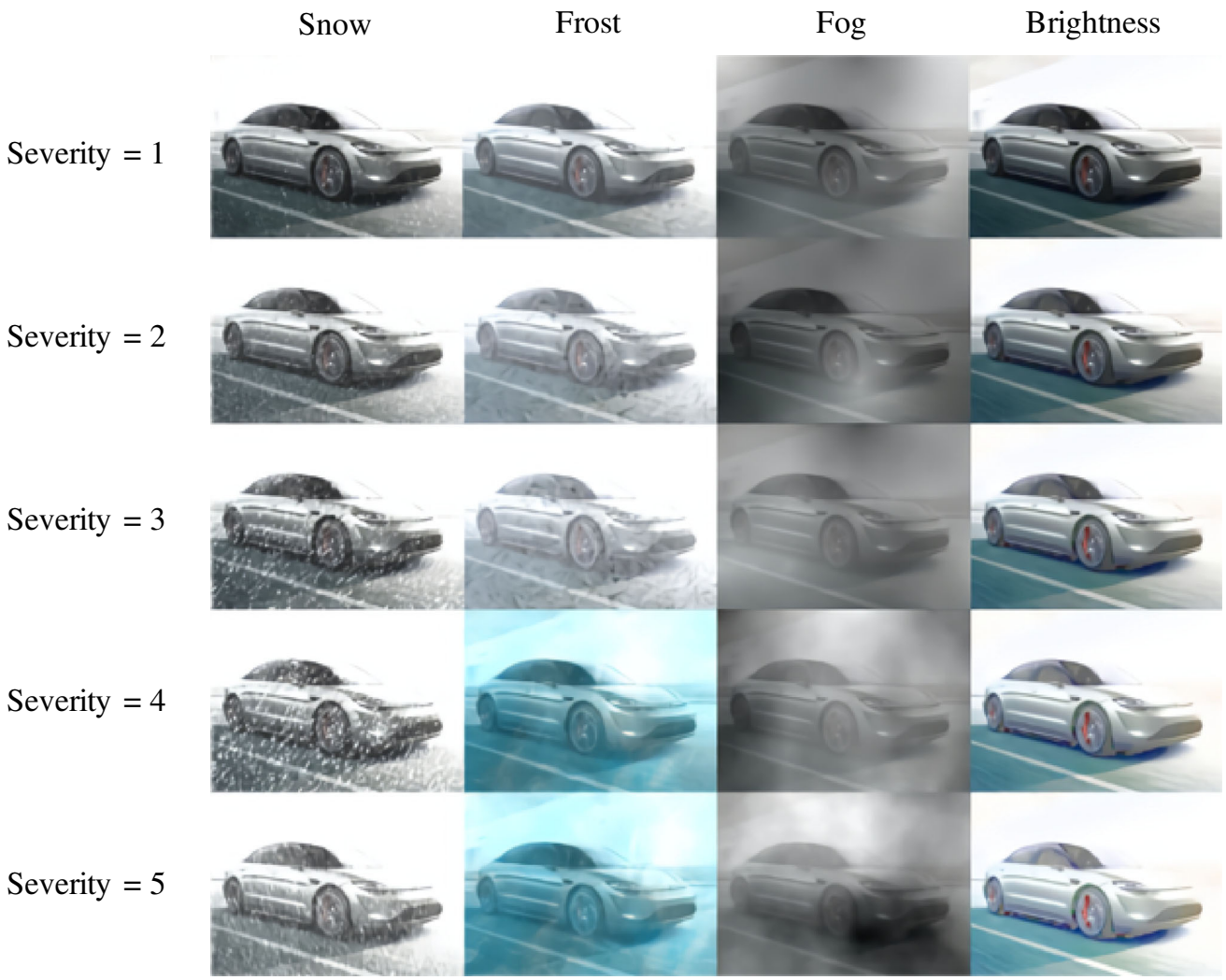

Fig. 10 Images in real environments and varied lighting conditions

(light changes) and 63.0% (fog) in the object detection, and 78.4% and 63.0% in semantic segmentation.

Merging left- and right-side car part as one label

After evaluating overall performance and robustness, we ran an error analysis to seek a way to improve the task. We found that the algorithms were usually confused with left- or right-side parts, e.g., predicting `left_mirror` as a `right_mirror` or vice versa. Therefore, we created a new set of data, that assigned a single label to left and right sides of a part. Then we fine-tuned each pre-trained model from the original labels at 100 epochs—other settings remained the same.

Table 3 shows the overall performance on both object detection and semantic segmentation, with left- and right-side part labels merged. All performances were higher than when left- and right-side parts were considered separately (Table 1): mAP increased by 5.76% for object detection and 5.27% for semantic segmentation for all models. The table shows that HTC, with the ResNet-101 encoder, yielded the highest mAP = 59.4, followed by HTC, with the ResNet-50 encoder, with mAP = 59.1 in object detection. HTC, with ResNet-101, performed best on large car parts—the highest value of AP_L at 65.4—while HTC, with ResNet-50, encoder achieved the best performance on small and medium car parts, resulting in AP_S = 34.5 and AP_M = 53.5. In addition, HTC, with ResNet-50, was the best contender with the most strict metric AP_{75} = 68.6. Although Mask R-CNN, with ResNet-50, received the highest AP_{50} score, it was still worse than HTC, with ResNet-50 or ResNet-101, using the strictest metric. In semantic segmentation, HTC, with ResNet-101, also ranked first with mAP = 60.1, followed by Mask R-CNN, with ResNet-50 or ResNet-101. Apparently, Mask R-CNN performed well in semantic segmentation, resulting in the highest performance on AP_{50} = 81.9, AP_{75} = 71.3, and

Table 2 Performance of each method for object detection and semantic segmentation, including a robustness test with challenging real environments

Model	Encoder	P	Overall		Snow		Frost		Fog		Brightness	
			mPC	rPC [%]	mPC	rPC [%]	mPC	rPC [%]	mPC	rPC [%]	mPC	rPC [%]
Object detection												
Mask R-CNN	ResNet-50	50.4	27.4	54.3	19.1	37.6	21.5	42.6	31.8	63.0	41.2	81.7
	ResNet-101	50.8	28.3	55.7	20.7	40.7	19.9	39.1	30.7	60.4	42.1	**82.8**
GCNet	ResNet-50	50.9	**33.0**	**64.8**	**24.4**	**63.4**	**26.8**	**52.6**	**38.8**	76.2	41.8	82.1
	ResNet-101	38.5	24.3	63.1	17.3	35.5	17.0	44.1	30.9	**80.2**	31.0	80.5
PANet	ResNet-50	48.8	26.6	54.5	18.7	37.7	21.0	43.0	26.9	55.1	38.2	78.2
	ResNet-101	49.6	26.9	54.2	20.5	39.5	23.1	46.5	28.8	58.0	40.8	82.2
CBNet	ResNet-50	51.9	25.0	48.1	16.9	34.1	15.3	29.4	29.3	56.4	38.3	73.7
	ResNet-101	49.5	25.7	51.9	18.0	33.3	18.7	37.7	26.4	53.3	37.9	76.5
HTC	ResNet-50	54.1	28.1	51.9	19.8	36.5	18.5	34.1	34.8	64.3	41.9	77.4
	ResNet-101	**54.3**	28.9	53.2	18.7	34.4	19.3	35.5	34.4	63.3	**42.4**	78.0
Average		48.9	27.4	55.2	19.4	39.3	20.1	40.5	31.3	63.0	39.6	79.3
Semantic segmentation												
Mask R-CNN	ResNet-50	54.0	28.5	52.7	20.0	37.0	21.2	39.2	33.5	62.0	43.2	80.0
	ResNet-101	**55.4**	29.3	52.8	21.0	37.9	20.0	36.1	33.0	59.5	**44.2**	79.7
GCNet	ResNet-50	54.6	**35.2**	**64.4**	**26.4**	**48.4**	**28.4**	**52.0**	**41.7**	76.3	**44.2**	80.9
	ResNet-101	43.0	25.9	60.2	17.5	40.7	17.6	40.9	33.7	**78.3**	33.7	78.3
PANet	ResNet-50	54.0	28.7	53.1	19.9	36.9	21.9	40.5	30.3	56.1	41.7	77.2
	ResNet-101	54.3	29.2	53.7	22.1	40.7	23.1	42.5	31.4	57.8	44.1	**81.2**
CBNet	ResNet-50	53.0	25.1	47.3	17.1	32.3	15.3	28.8	29.4	55.4	38.7	73.0
	ResNet-101	49.5	26.0	52.5	18.4	37.2	18.7	37.7	27.5	55.5	38.3	77.3
HTC	ResNet-50	55.2	28.9	52.3	20.3	36.8	18.5	33.5	35.7	64.6	43.0	77.8
	ResNet-101	54.5	29.3	53.7	19.5	35.8	19.3	35.4	34.9	64.0	42.6	78.1
Average		52.8	28.6	54.3	20.2	38.4	20.4	38.7	33.1	63.0	41.4	78.4

Bold—best

AP_M = 55.2. Again, we used Kendall's coefficient of concordance (W) to evaluate agreements between algorithms. The overall performance rank changed: Mask R-CNN, with ResNet-50, was now the first ranked, followed by HTC, with ResNet-50 and ResNet-101. The rankings in the table are significant at $p < 0.01$ for 11 degrees of freedom ($W = 0.4905$ and $\chi^2 = 52.9691$).

We also evaluated algorithms robustness in the merged sides of a car part scenario on both tasks as shown in Table 4. The overall picture was very much the same as considering left side and right side separately. GCNet was still the most robust algorithm, while the worst was CBNet. Moreover, snow and frost were still the top most challenging conditions to corrupt the data, that impacted the algorithms.

Conclusion

Computer network technology and end-devices are becoming more powerful. Also, the car insurance business is rapidly growing. Thus, an automated system for damage evaluation is necessary. In this work, we describe an automatic car part identification system based on images by deep learning techniques. We compared the performance of several state-of-the-art deep learning algorithms on a part segmentation task, using a car part data set, created for this work, that is now publicly available. Our experiments showed that HTC was the best model, followed by Mask R-CNN and GCNet, in both object detection and semantic segmentation tasks in normal weather conditions. Also, we evaluated algorithm robustness in real environmental and lighting conditions, simulating conditions that would occur in the field, when we take a photo using a smartphone. GCNet was the most robust model, because it achieved the best performance in overall pictures and in real conditions, except in varying brightness. Currently, edge computing has become more practical and able to overcome limitations of end-devices. Therefore, edge computing enabled the models to operate in the end-device, leading to a solution for real-time image analysis.

Table 3 Overall performances of the selected models on object detection and semantic segmentation tasks with merged side of the car part scenario

Model	Encoder	Object Detection						Semantic segmentation						Sum Rank	Rank
		mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L		
Mask R-CNN	ResNet-50	56.5	**81.0**	67.7	33.3	52.2	60.1	59.4	**81.9**	70.4	35.0	**55.2**	64.7	**36**	**1**
	ResNet-101	55.0	79.0	64.6	33.3	51.3	59.2	59.3	80.4	**71.3**	32.6	54.9	63.8	53.5	4
GCNet	ResNet-50	55.1	79.8	65.1	32.0	50.9	58.7	58.9	81.4	70.7	**51.6**	53.9	63.1	55.5	5
	ResNet-101	51.1	75.7	60.9	30.6	49.2	56.4	54.7	76.8	64.7	42.8	52.7	60.6	102.5	10
PANet	ResNet-50	53.5	78.9	61.8	31.4	50.1	58.4	57.9	80.6	68.3	30.7	50.6	66.2	83.5	7
	ResNet-101	55.7	78.9	65.5	31.7	51.1	62.1	59.0	80.2	69.1	31.2	52.3	**68.1**	62	6
CBNet	ResNet-50	56.9	74.6	64.4	32.5	49.2	64.7	56.9	75.8	65.9	30.5	49.2	64.7	86	8
	ResNet-101	54.1	71.8	62.2	31.1	52.5	58.7	54.8	73.3	64.2	28.6	52.1	61.7	101	9
HTC	ResNet-50	59.1	78.4	**68.6**	**34.5**	**53.5**	64.9	59.2	80.3	69.6	34.5	53.7	65.7	38	2
	ResNet-101	**59.4**	78.1	68.0	33.9	52.5	**65.4**	**60.1**	80.2	68.6	32.7	52.8	66.1	42	3
Average		55.6	77.6	64.9	32.4	51.3	60.9	58.0	79.1	68.3	35.0	52.7	64.5	–	

Bold—best

Table 4 Performance of each method for object detection and semantic segmentation on merged part sides, including a robustness test with different real environments

Model	Encoder	P	Overall		Snow		Frost		Fog		Brightness	
			mPC	rPC [%]	mPC	rPC [%]	mPC	rPC [%]	mPC	rPC [%]	mPC	rPC [%]
Object detection												
Mask R-CNN	ResNet-50	56.5	29.8	52.7	19.6	34.6	20.4	36.1	33.9	60.0	45.5	80.3
	ResNet-101	55.0	28.8	52.3	19.9	36.1	14.5	26.3	32.6	59.2	46.5	**84.5**
GCNet	ResNet-50	55.1	**39.3**	71.3	**30.5**	55.3	**33.8**	61.3	**46.1**	83.6	39.3	71.3
	ResNet-101	51.1	36.5	**71.4**	30.4	59.4	31.5	**61.6**	42.9	**83.9**	37.2	72.7
PANet	ResNet-50	52.5	28.4	53.0	21.5	40.1	16.3	30.4	29.6	55.3	43.0	80.3
	ResNet-101	55.7	31.2	56.0	23.5	**42.1**	21.1	37.8	31.0	55.6	46.7	83.8
CBNet	ResNet-50	56.9	28.3	49.7	20.0	35.1	13.8	24.2	31.4	55.1	43.7	76.8
	ResNet-101	54.1	27.6	51.0	21.7	40.1	17.4	32.1	28.0	51.7	43.9	81.1
HTC	ResNet-50	59.1	32.2	54.4	22.3	37.7	20.7	35.0	37.9	64.1	46.7	79.0
	ResNet-101	**59.4**	31.9	53.7	21.0	35.3	20.3	34.1	38.7	65.1	**47.1**	79.2
Average		55.6	30.4	56.6	23.0	41.6	21.0	37.9	35.2	63.4	44.0	78.9
Semantic segmentation												
Mask R-CNN	ResNet-50	59.4	30.7	51.6	20.3	34.1	20.3	34.1	35.6	59.9	46.7	78.6
	ResNet-101	59.3	29.5	49.7	20.4	34.4	14.7	24.7	34.0	57.3	48.1	81.1
GCNet	ResNet-50	58.9	**40.9**	69.4	**31.5**	53.4	**33.5**	56.8	**48.2**	81.8	40.8	69.2
	ResNet-101	54.7	38.1	**69.6**	31.3	**57.2**	32.3	**59.0**	45.3	**82.8**	38.3	70.0
PANet	ResNet-50	57.9	30.6	52.8	23.1	39.8	17.1	29.5	32.1	55.4	46.5	80.3
	ResNet-101	59.0	32.5	55.0	23.3	39.4	20.7	35.0	33.4	56.6	**49.4**	**83.7**
CBNet	ResNet-50	56.9	28.2	49.5	19.9	34.9	14.1	24.7	31.2	54.8	43.2	75.9
	ResNet-101	54.8	27.8	50.7	21.5	39.2	17.5	31.9	28.3	51.6	44.4	81.0
HTC	ResNet-50	59.2	32.2	54.3	22.2	37.5	20.5	34.6	38.4	64.8	47.2	79.7
	ResNet-101	**60.1**	32.2	53.5	21.1	35.1	20.7	34.4	38.3	63.7	47.5	79.0
Average		58.0	32.3	55.6	23.5	40.5	21.1	36.5	36.5	62.9	45.2	77.9

Bold—best

In future work, we will focus on developing a lighter weight model for semantic segmentation to ease the load on the end-device, without sacrificing its accuracy and robustness. We also aim to extend the work to detect, localize and estimate the severity of damage on different parts.

Acknowledgements This work was supported by King Mongkut's Institute of Technology Ladkrabang under grant agreement number 2564-02-06-002.

Author contributions KP conceived the original idea of the method, validation, and revised the final manuscript. NH carried out the software implementation. KP and PK performed formal analysis, investigation, and writing the original draft. KP and KW conceived the initial concept of the study, review and editing the draft. All authors read and approved the final manuscript.

Funding The funder had no role in the study design, data collection and analysis, decision to publish or preparation of the manuscript.

Availability of data and material The data set generated and analyzed in this study is available in the GitHub repository, https://github.com/dsmlr/Car-Parts-Segmentation.

Declarations

Conflicts of interest We declare that we have no competing interests.

References

1. Handel P, Skog I, Wahlstrom J, Bonawiede F, Welch R, Ohlsson J et al (2014) Insurance telematics: opportunities and challenges with the smartphone solution. IEEE Intell Transp Syst Mag 6(4):57–70
2. Husnjak S, Peraković D, Forenbacher I, Mumdziev M (2015) Telematics system in usage based motor insurance. Procedia Eng 100:816–825
3. Zhou Z, Chen X, Li E, Zeng L, Luo K, Zhang J (2019) Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc IEEE 107(8):1738–1762
4. Ni J, Zhang K, Lin X, Shen X (2018) Securing Fog computing for internet of things applications: challenges and solutions. IEEE Commun Surv Tutor 20(1):601–628
5. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: vision and challenges. IEEE Internet Things J 3(5):637–646
6. Papers with Code–COCO test-dev Benchmark (Instance Segmentation) (2020) https://paperswithcode.com/sota/instance-segmentation-on-coco. Accessed 1 Nov 2020
7. Velichko A (2020) Neural network for low-memory IoT devices and MNIST image recognition using kernels based on logistic map. Electronics 9(9):1432
8. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T et al (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861v1
9. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) MobileNetV2: inverted residuals and linear bottlenecks. arXiv:1801.04381v4
10. Tuli S, Basumatary N, Buyya R (2019) EdgeLens: deep learning based object detection in integrated IoT, Fog and cloud computing environments. In: Proceedings of the international conference on information systems and computer networks (ISCON), p 496–502
11. Zhang S, Li R, Dong X, Rosin P, Cai Z, Han X et al (2019) Pose2Seg: detection free human instance segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), p 889–898
12. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask R-CNN. In: Proceedings of the IEEE international conference on computer vision (ICCV), p 2980–2988
13. Yi J, Tang H, Wu P, Liu B, Hoeppner DJ, Metaxas DN et al (2019) Object-guided instance segmentation for biological images. arXiv:1911.09199v1
14. Lee Y, Park J (2019) CenterMask: real-time anchor-free instance segmentation. arXiv:1911.06667v6
15. Tian Z, Shen C, Chen H, He T (2019) FCOS: fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV), p 9626–9635
16. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
17. Wang X, Kong T, Shen C, Jiang Y, Li L (2019) SOLO: segmenting objects by locations. arXiv:1912.04488v3
18. Wang X, Zhang R, Kong T, Li L, Shen C (2020) SOLOv2: dynamic and fast instance segmentation. ArXiv:2003.10152v3
19. Xie E, Sun P, Song X, Wang W, Liang D, Shen C et al (2019) PolarMask: single shot instance segmentation with polar representation. arXiv:1909.13226v4
20. Wang S, Gong Y, Xing J, Huang L, Huang C, Hu W (2019) RDSNet: a new deep architecture for reciprocal object detection and instance segmentation. arXiV:1912.05070v1
21. Bolya D, Zhou C, Xiao F, Lee YJ (2019) Better Real-time Instance Segmentation. arxiv:1912.06218
22. Liu S, Qi L, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), p 8759–8768
23. Huang Z, Huang L, Gong Y, Huang C, Wang X (2019) Mask scoring R-CNN. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), p 6402–6411
24. Chen H, Sun K, Tian Z, Shen C, Huang Y, Yan Y (2020) BlendMask: top-down meets bottom-up for instance segmentation. arXiv:2001.00309v3
25. Lu W, Lian X, Yuille A (2014) Parsing semantic parts of cars using graphical models and segment appearance consistency. arXiv:1406.2375v2
26. Liu Y, Zou L, Li J, Yan J, Shi W, Deng D (2016) Segmentation by weighted aggregation and perceptual hash for pedestrian detection. J Vis Commun Image Represent 36:80–89
27. Singh R, Ayyar MP, Sri Pavan TV, Gosain S, Shah RR (2019) Automating car insurance claims using deep learning techniques. In: Proceedings of the IEEE international conference on multimedia big data (BigMM), p 199–207
28. Dhieb N, Ghazzai H, Besbes H, Massoud Y (2019) A very deep transfer learning model for vehicle damage detection and

localization. In: Proceedings of the international conference on microelectronics (ICM), p 158–161
29. Patil K, Kulkarni M, Sriraman A, Karande S (2017) Deep learning based car damage classification. In: Proceedings of the IEEE international conference on machine learning and applications (ICMLA), p 50–54
30. Dwivedi M, Malik HS, Omkar SN, Monis EB, Khanna B, Samal SR et al (2020) Deep learning-based car damage classification and detection. In: Advances in artificial intelligence and data engineering. Springer, p 207–221
31. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2010) The PASCAL visual object classes (VOC) challenge. Int J Comput Vis 88(2):303–338
32. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D et al (2014) Microsoft COCO: common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Proceedings of the European conference on computer vision (ECCV). Springer International Publishing, Cham, p 740–755
33. Chen K, Pang J, Wang J, Xiong Y, Li X, Sun S et al (2019) Hybrid task cascade for instance segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), p 4969–4978
34. Liu Y, Wang Y, Wang S, Liang T, Zhao Q, Tang Z et al (2019) CBNet: a novel composite backbone network architecture for object detection. arXiv:1909.03625v1
35. Cao Y, Xu J, Lin S, Wei F, Hu H (2019) GCNet: non-local networks meet squeeze-excitation networks and beyond. In: Proceedings of the IEEE/CVF international conference on computer vision workshop (ICCVW), p 1971–1980
36. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), p 770–778
37. Lin T, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), p 936–944
38. Wang X, Girshick R, Gupta A, He K (2018) Non-local neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR), p 7794–7803
39. Hu J, Shen L, Albanie S, Sun G, Wu E (2020) Squeeze-and-excitation networks. IEEE Trans Pattern Anal Mach Intell 42(8):2011–2023
40. Cai Z, Vasconcelos N (2019) Cascade R-CNN: high quality object detection and instance segmentation. IEEE Trans Pattern Anal Mach Intell 43(5):1483–1498
41. Chen K, Wang J, Pang J, Cao Y, Xiong Y, Li X et al (2019) MMDetection: Open MMLab detection toolbox and benchmark. arXiv:190607155
42. Hendrycks D, Dietterich T (2019) Benchmarking neural network robustness to common corruptions and perturbations. arXiv:1903.12261v1
43. Michaelis C, Mitzkus B, Geirhos R, Rusak E, Bringmann O, Ecker AS et al (2019) Benchmarking robustness in object detection: autonomous driving when winter is coming. arXiv:190707484
44. Siegel S, Castellan NJ (1988) Nonparametric statistics for the behavioral sciences, 2nd edn. McGraw-Hill, New York