A Case Study: Deep Learning Methods for Vehicle Damage Detection

Wai Po Kevin Teng

15.02.2022

Abstract

Detecting external damages on vehicles is crucial for insurance claim, repair purpose and reselling purpose, which is tedious and time consuming. Recent advancement in the field of computer vision has made it feasible for visual inspection to be fast, scalable and robust with deep learning methods. The emergence of convolutional neural networks (CNNs) inspired a series of state-of-the-art methods in the area of object detection, such as the R-CNN family and YOLO for accurate and close to real time detection. These methods along side other deep learning techniques are ideal for vehicle damage detection. We proposed a vehicle damage detection pipeline that employed Faster R-CNN for vehicle count in an image and the possibility of implementing instance segmentation and object detection using Mask R-CNN or merely object detection using YOLO framework.

1 Introduction

In this modern society, vehicle has been one of the most important mode of transport for people to travel between places. Vehicles are however, susceptible to damages arising from man-made causes or natural disaster. To no surprise, vehicle insurance companies would often need to deal with frequent insurance claims. The claim amount depends on the degree of damage, damage type and damage position [15]. However, manually verification of claims on a large scale is both laborious and time consuming, which will hinder the claiming process. Other than vehicle insurance claim, the resell value of the vehicle also largely depends on the vehicle's external condition. Similarly, visual inspections on vehicle external condition both tedious and inefficient. This calls for a dire need of an automated system to streamline claiming process and visual inspection without compromising speed and efficiency. With the field of computer vision progressively gaining traction in tackling image or video problems, this motivates us to venture into computer vision methods in search of an efficient automation process.

In computer vision, there is a research field that particularly standout, namely deep learning. Deep learning solves the need of feature engineering through the extraction of high level features using deep neural networks [6]. Such paradigm shift led to the discovery of convolutional neural networks (CNN) proposed by LeCun et al. [11] inspired by the receptive field of visual cortex. Ever since CNN model architecture has achieved astonishing performance in image classification task from ImageNet challenge [21], CNN has thus became the *De Facto Standard* for computer vision applications. Given the wide success of CNNs, it was applied on the problem of object detection. Unlike classification task, object detection compose of object localisation and object classification. Object localisation refers to the positional description of one or more objects in the image through bounding boxes. Object classification refers to identification of the object by assigning it to it's respective label. Some state-of-the-art object detection frameworks are R-CNN [5], Fast R-CNN [4], Faster R-CNN [20] and YOLO [19] that dis exceptionally well in the PASCAL VOC 2012 challenge [2]. The R-CNN family is categorised as two-stage detector because it depends on separate module to generate region proposals. Whereas, YOLO is categorised as single-stage detector because it does not rely on secondary module.

To this extend, CNNs are without the doubt the go-to method for vehicle damages detection. In this work, the primary focus of the vehicle type is car, due to it's vast availability and the possibility of having a whole car fit into an image. The purpose of this work is to give an overview about the pipeline for car damages detection and the corresponding deep learning architectures. This work is presented without actual implementation. The structure of this report is as follow, section 2 will discuss about the background of the methods employed in this work, section 3 will discuss about the literature reviews relevant to this work, section 4 will give an overview about the experiment setup given the use case and finally section 5 will be the closing remark and future work.

2 Background

This section will define what is a backbone model and explain the evaluation metric implemented in object detection frameworks. The focus of this section will be on the explanation of object detectors.

2.1 Backbone

Backbone models are the network architectures that extract features given an input. The performance of object detectors depends on these backbone architectures for effective learning, hence the word backbone. Some state-of-the art backbone models includes, VGGNet [22], GoogleNet [23], ResNet [7] and DenseNet [9], which achieved outstanding performance in the ImageNet challenge [21]. These backbone model architectures are often fine tuned on several large data sets with the model weights widely accessible to everyone. This made some backbone models suitable as a pre-trained models for fine tuning or transfer learning task. However, backbone model architectures are not the main focus of this work as we will not provide in-depth explanations.

2.2 Evaluation Metric

The most crucial metric for object detection is mean Average Precision (mAP). To understand mAP, we need to first understand precision and recall. Precision is defined as the ratio of true positive (TP) over predicted positives, which is the sum of true positive (TP) and false positive (FP), expressed in equation 1. Recall is defined as the ratio of true positive (TP) over actual positives, which is the sum of true positive (TP) and false negative (FN), expressed in equation 2.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Therefore, in the context of object detection, precision and recall is derived from intersection over union (IoU), which is the ratio of area of overlap and the area of union given an IoU threshold. Intuitively, the IoU threshold is set at a value of 0.5, where if the IoU is more than the threshold, it is classified as TP, else FP on the flip-side. If no object was detected in the ground truth, it is classified as FN. A schematic diagram illustrating the concepts of IoU associated with bounding boxes is depicted in Figure 1.

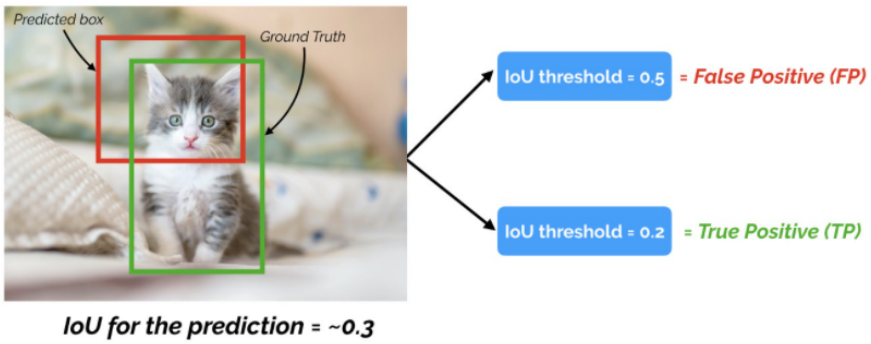
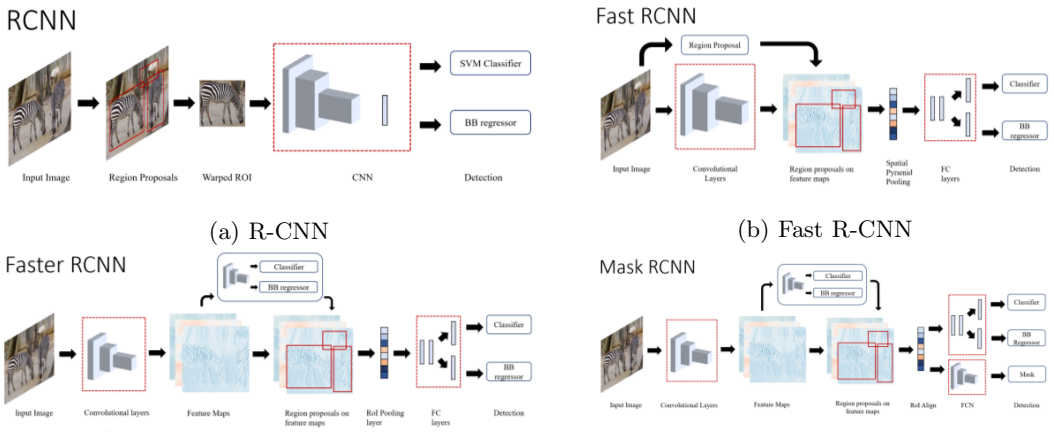


Figure 1: Illustration about IoU with predicted bounding box and ground truth on image. Image source from [27].

Average precision (AP) is defined as the area under the precision-recall curve, where mAP is the average of AP over all object classes.

2.3 Object Detectors

There are pioneering works in object detection prior to deep learning era. These frameworks consisted of Viola-Jones object detector [26] that incorporate sliding window on Haar-like fea-



(a) R-CNN

(b) Fast R-CNN

(c) Faster R-CNN)

(d) Mask R-CNN

Figure 2: Illustration of different two stage detectors architectures. Image adapted from [28].

tures, as well as Deformable Parts Model (DPM) [3] that follows the philosophy of *divide and rule*. However, this was not included in the scope of this work where we will discuss more on state-of-the-art methods using deep learning. In general, state-of-the-art object detectors can be classified into two categories, which is two stage detector and single stage detector. Two stage detector has two separate stage, where the first stage relies on region proposal module to find selective number of object proposals and attempts to classify and localize the objects in the second stage. On the other hand, single stage detectors classify and localize objects in a single module.

2.4 Two Stage Detectors

2.4.1 R-CNN

In the first stage of R-CNN [5] framework, class agnostic region proposal module was implemented with selective search [25]. This module attempts to search parts of the image that has higher probability of finding an object. These candidates serve as inputs to the backbone model where the CNN network attempts to extract salient features. The feature maps are pass on to support vector machines (SVM) to obtain class-specific confidence scores. Follows, non-maximum suppression (NMS) is applied over the scored region based on its IoU and class. A bounding box regressor is applied to predict center coordinates, width and height of the bounding box over the identified class. The building blocks of R-CNN is depicted in Figure 2 (a). R-CNN relies on pseudo-metrics, such as SVM for classification task and multi stage training process that prevents the framework to train in an end-to-end fashion. While R-CNN has large improvement in detection performance as compared to conventional object detection frameworks, the training process is slow. However, R-CNN has laid the foundation for two stage detectors and inspired an avalanche of state-of-the-art methods.

2.4.2 Fast R-CNN

Fast R-CNN [4] attempts to address the shortcoming of R-CNN that requires to train multiple pipeline separately by proposing a single end-to-end system. As shown in Figure 2 (b), feature maps are extracted from the input image. The feature maps are then pass through a region of interest (RoI) pooling layer for region proposal. The pooling layer is connected to two fully connected layers where one layer deals with the classifier and another layer is a bounding box regressor. Fast R-CNN was introduced with the aim of object detection speed enhancement without sacrificing accuracy. According to Girshick [4], Fast R-CNN is approximately 146 times the speed of R-CNN.

2.4.3 Faster R-CNN

While Fast R-CNN [4] has proven its capability to performance well in close to real time object detection, the region proposal generation serve as the bottle neck to speed up model performance. Taking this into consideration, Ren et al. [20] proposed fully connected network (FCN) [14] as trainable region proposal network (RPN). RPN introduces the concept of anchor boxes where the network takes an input image and outputs a set of window candidates. Each candidates is associated with a score which shows the likelihood of the object. RPN applied multiple bounding boxes of different aspect ratios for object localisation and object classification. A schematic diagram about Faster R-CNN architecture is depicted in Figure 2 (c). Feature maps are first obtained through a backbone model from an image input. The feature maps are use as RPN inputs where the network attempts to generate bounding boxes and their classifications. The proposed regions are then mapped back to the feature maps previously obtain from the backbone model before feeding into RoI pooling layer. The RoI pooling layer is then connected to two fully connected layer before feeding into a classifier and a regressor. Faster R-CNN is an enhancement of Fast R-CNN with RPN as its region proposal module.

2.4.4 Mask R-CNN

Often times, object detectors are able to detect the object in the images but it does not display the exact boundary of the object at a pixel value within a scene. This is exceptionally crucial when we want to estimate the coverage of car damage from the image. Mask R-CNN [8] is a special extension of Faster R-CNN [20] that introduces a framework capable of doing object detection and object instance segmentation concurrently. A schematic figure about Mask R-CNN framework is depicted in Figure 2 (d). Mask R-CNN framework is similar to the framework of Faster R-CNN, except the addition of an auxiliary output that connects a FCN [14] network for instance segmentation. Another difference was Mask R-CNN adapted RoIAlign layer, rather than RoI pooling layer after feature extraction and regional proposal. This was intended to prevent pixel-wise misalignment due to spatial quantization [28].

2.5 Single Stage Detector

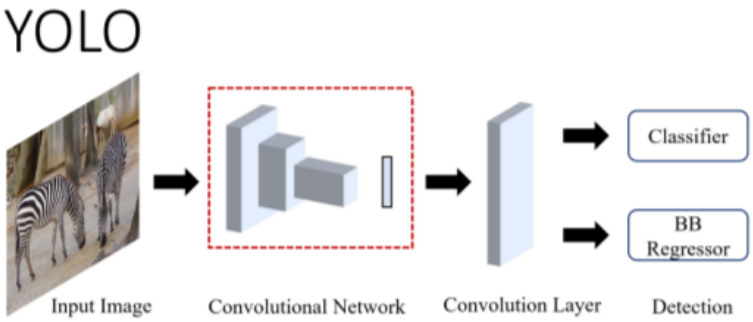


Figure 3: YOLO framework architecture. Image adapted from [28].

2.5.1 YOLO

In the previous section, two stage detectors involving solving object detection as a classification problem and a complimentary module that proposed regional candidates for the network to determine if its a foreground or a background. You Only Look Once (YOLO) [19] was introduced to readdress object detection as a regression problem by attempting to simultaneously predicts multiple bounding boxes and class probabilities for those boxes. A schematic diagram depicting the architecture of YOLO is shown in Figure 3. In YOLO framework, the input image is divided into a grid size of $S \times S$. The cell where the object's center located is responsible for detecting it. Each grid cells predicts multiple bounding boxes and confidence scores of the boxes. The confidence scores indicates how confident is the box containing an object and how accurate the

box predicts. In the context of YOLO, the confidence score is defined as the multiplication of probability of the object in the box and IoU, shown in equation 3.

$$Confidence = Pr(Object) * IoU \tag{3}$$

According to Redmon et al. [19], YOLO excel in real time object detection without sacrificing speed. However, the drawbacks of YOLO is the localisation for small or clustered object as well as its limitation to number of objects per cell.

2.6 Comparative Results

Model	Year	Backbone	Size	$AP_{0.5}$	FPS
R-CNN	2014	AlexNet	224	58.50 %	0.02
Fast R-CNN	2015	VGG-16	Variable	65.70 %	0.43
Faster R-CNN	2016	VGG-16	600	**67.00** %	5
YOLO	2015	GoogleNet	448	57.90 %	**45**

Table 1: Comparative results between R-CNN, Fast R-CNN, Faster R-CNN and YOLO for PASCAL VOC 2012 [2] data set. $AP_{0.5}$ is the average precision (AP) with IoU > 0.5. Table adapted from [28].

Table 1 shows a comparative results of two stage detectors and single stage detector introduced in this work on PASCAL VOC 2012 [2] data set. The models are compared on average precision (AP) and processed frames per second (FPS) at inference time. As observed from Table 1, two stage detectors (R-CNN, Fast R-CNN and Faster R-CNN) has better $AP_{0.5}$ performance as compared to single stage detectors (YOLO), with Faster R-CNN having the best performance. However, on FPS performance, YOLO is a magnitude faster to all two stage detectors. This indicates that YOLO is much more suitable for real-time object detection with fast inference time but compromise on AP. On the other hand, Faster R-CNN which is an refinement of the R-CNN family, gives better prediction for object detection but is not suitable for real-time object detection as compared to YOLO.

3 Related work

Vehicle evaluation is crucial to gauge the condition of the vehicle for various purpose, such as insurance claims, repair purpose and reselling purpose. At a glance, the condition of the vehicle depends on the severity and numbers of external damages. Therefore, visual inspection to identify damage areas of the vehicle depends heavily on the performance of the object detection algorithm. Patil et al. [16] presented work on deep learning based car damage classification by comparing multiple state-of-the-art backbone models as feature extractors as well as linear support vector machine (SVM) and softmax function for network output as classifiers. Due to the lack of car damage images, Patil et al. [16] employed convolutional auto encoders (CAE) for salient feature extractor pre-trained on Stanford cars data set to boost the performance of their classifiers. In the work by Li et al. [12], they proposed a deep learning framework on anti-fraud system for car insurance claim to prevent claim leakage. The system is conducted in two stages. Li et al. [12] first employed YOLO framework for car damage detection. The second step involved damage re-identification, which employed pre-trained VGG16 [22] network for local features of the damage region and then merge with global features such as colour histograms before comparing features in the post-claim history database. The input image act as a query and retrieve a gallery of top nearest neighbour images from the database through cosine similarity. If the traits of the query image matched the top nearest neighbour in the database, there could be potential fraud. In the work by Malik et al. [15], they explored different deep learning backbone models for car damage classification and detection for insurance claim. Malik et al. [15] employed YOLOv3 [18] framework to detect the damaged region and a CNN trained model to classify the type of damage. Kyu et al. [10] introduced four models for four stages of granularity in the car damage assessment pipeline. All four models are VGG inspired models, where the first model was used to classify if there are any car in the image or not; the second model attempts to classify damaged

and undamaged car; the third model attempts to categorised the severity of the damaged car; whereas the fourth model attempts to classify the location of the car damage area.

4 Experiment Setup

The problem statement of this work is to build an algorithm to identify damaged areas of vehicles such that the vehicle can be valued and sold accordingly. This section aims to provide an overview about the building blocks of car damaged area detection.

4.1 Data Set Preparation

This section explains how the data set will be formulated and the possible preprocessing steps for model training.

4.1.1 Data Set Description

Due to the scarcity of publicly available data set for car damage classification, work by [16], [12], [15] and [10] has created their own data set by collecting images via web scraping through keywords, such as 'scratch', 'dent', etc. Also, mentioned in the work by [15], some of the challenges of vehicle images include the reflective metallic bodies that will contribute to false positive during damage detection as well as the quality of photographs taken in an uncontrolled environment. Before begin to collect our data set, we need to formulate the types of damage that we anticipate. For example, the damage types granularity used in the work by [15] are bumper dent, scratches, door dent, glass shatter, head-lamp broken, tail-lamp broken and smashed, which are depicted in Figure 4. Furthermore, in the work by [12], in order to mimic customer behaviours, [12] employed image evidence captured from mobile devices.

Figure 4: Image gallery of different damage-types, starting from (left column to right column) bumper dent, scratches, door dent, glass shatter, head-lamp broken, tail-lamp broken and smashed. Image adapted from [15].

4.1.2 Data Preprocessing

Since the data set are self collected, we need to manually filtered and categorised the images into the type of damages we defined. Next, we need to resize all the data set images into the same size, at the same time retaining the highest possible resolution without loss of information. Data set with image of the same size would enable ease of data loading and training process. Follows, we will need to normalise the image such that the pixel intensity is between the range of 0 to 1. This will help to speed up the model convergence during training phase. Furthermore, the image data set has to be annotated with bounding boxes for its corresponding labels. An additional pixel wise image annotation of the ground truth will be required if the task objective involves instance segmentation.

4.1.3 Data Augmentation

It has been emphasize in the previous work ([16], [12], [15], [10]) that the data set are small and deep learning methods required abundant data set for effective learning. To overcome the problem of model overfitting, data augmentation is employed to artificially increase the data set size. Some common data augmentation techniques involve rotation, zooming, shearing, horizontal flip, etc. However, data augmentations should be handled with care and take natural behaviour into consideration. Vertical flipping and 90 degree clock-wise rotation of car images does not fulfill the requirements of the actual car state we witness in our everyday life.

4.2 Transfer Learning

Transfer learning is an inexpensive way to tackle overfitting on small data sets by leveraging the optimised pre-trained model on similar task. Instead of training the backbone model from scratch, we can make use of the backbone models that are pre-trained on large data set, such as ImageNet data set [21], assuming that the transfer knowledge from the pre-trained model will share some similar features with the target task. Fortunately, almost all state-of-the-art backbone models are trained on large data sets with the model weights being widely available. We can speed up the convergence of our training model by adopting trained weights of the model on ImageNet data set and fine tuning the top layer of our backbone model, such that we can learn high level features from our data set.

Model	Parameters	Accuracy
AlexNet	60M	89.89 %
VGG19	144M	94.9 %
Inception V3	5M	74.18 %
ResNet50	25.6M	90.26 %
MobileNets V1.0	4.2M	70.8 %

Table 2: Car damage classification test accuracy of different pre-trained (on ImageNet) models. Table adapted from [15].

For the purpose of reference, Table 2 shows the car damage classification test accuracy of different pre-trained models with data augmentation in the work by [15]. Although, VGG19 has the highest accuracy value at 94.9 %, the number of parameters is the highest among the models. This would hinder the model training computation speed. ResNet50 has the best trade-off by exhibiting moderate number of parameters without compromising accuracy. ResNet is also fast at converging and prevents vanishing of gradient during training thanks to residual connections [7].

4.3 Car Damaged Area Detector

A simple car damaged area detection pipeline flow chart is depicted in Figure 5. In total, it is possible to train three models for the detection pipeline. First, Faster R-CNN [20] is employed to count the number of cars inside the picture. The images may contained single car, multiple car or no car at all. Ideally, for it is only possible to inspect only one car from one image the rest would be irrelevant information. This pipeline acts as a screening process for rejecting irrelevant images. Therefore, we need a system to detect the number of cars inside the image and reject these images when the number of car is not equal to one. Faster R-CNN [20] framework was chosen for the task because of drawbacks of YOLO [19] not being able to detect clustered objects.

On the next step, we have the option of choosing whether we need to include instance segmentation to determine the exact boundary of the damage areas. If instance segmentation is required, then the go-to framework is Mask R-CNN [8] for car damaged area detection and segmentation task. Else, YOLO [19] framework is employed in the final stage for detecting car damaged area due to it's speed without sacrificing accuracy. We recommend ImageNet pre-trained ResNet50 as the backbone model for both Faster R-CNN and Mask R-CNN. Whereas, the backbone model for YOLO is recommended for Darknet53 [17] due to it's excellent performance shown in the work by [15].

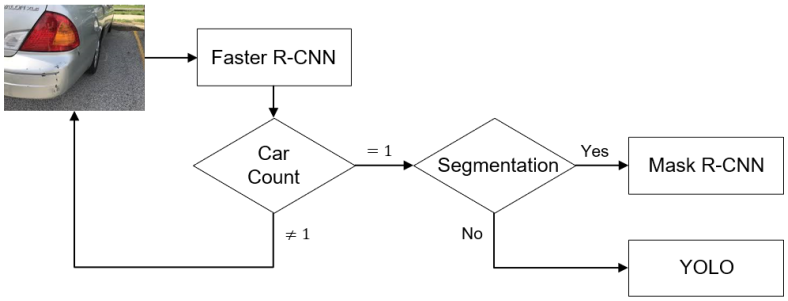


Figure 5: Flow chart on car damaged area detection pipeline. Car image adapted from [12].

5 Conclusion

This work presented a brief overview about the object detection architectures, such as R-CNN family ([5], [4], [20]) and YOLO [19] in section 2. This work also presents the building blocks of car damaged area detector in section 4. In a nutshell, R-CNN family are slower as compared to YOLO due to the existence of region proposal module. However, YOLO does not perform well in localising small and clustered objects. This work lacks the introduction on other contemporary state-of-the-art object detection frameworks, such as the YOLO family ([18], [1]), as well as the trending transformer inspired object detection framework ([24], [13]). Also, this work lacks the ablation studies of actual implementation which it will be included as future work.

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *CoRR* abs/2004.10934 (2020).

[2] M. Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: vol. 88. 2. June 2010, pp. 303–338.

[3] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. "A discriminatively trained, multiscale, deformable part model". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587597.

[4] Ross B. Girshick. "Fast R-CNN". In: vol. abs/1504.08083. 2015.

[5] Ross B. Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: vol. abs/1311.2524. 2013.

[6] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[7] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: vol. abs/1512.03385. 2015.

[8] Kaiming He et al. "Mask R-CNN". In: vol. abs/1703.06870. 2017.

[9] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: vol. abs/1608.06993. 2016.

[10] Phyu Mar Kyu and Kuntpong Woraratpanya. "Car Damage Detection and Classification". In: *Proceedings of the 11th International Conference on Advances in Information Technology*. ACM, July 2020.

[11] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: vol. 1. 1989, pp. 541–551.

[12] Pei Li, Bingyu Shen, and Weishan Dong. "An Anti-fraud System for Car Insurance Claim Based on Visual Evidence". In: vol. abs/1804.11207. 2018.

[13] Ze Liu et al. "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows". In: *CoRR* abs/2103.14030 (2021).

[14] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: vol. abs/1411.4038. 2014.

[15] Hashmat Shadab Malik et al. *Deep Learning Based Car Damage Classification and Detection*. EasyChair Preprint no. 3008. EasyChair, 2020.

[16] Kalpesh Patil et al. "Deep Learning Based Car Damage Classification". In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Dec. 2017.

[17] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. http://pjreddie.com/darknet/. 2013–2016.

[18] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018).

[19] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: vol. abs/1506.02640. 2015.

[20] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: vol. abs/1506.01497. 2015.

[21] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: 2014.

[22] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[23] Christian Szegedy et al. "Going Deeper with Convolutions". In: vol. abs/1409.4842. 2014.

[24] Mingxing Tan, Ruoming Pang, and Quoc V. Le. "EfficientDet: Scalable and Efficient Object Detection". In: *CoRR* abs/1911.09070 (2019).

[25] J. R. R. Uijlings et al. “Selective Search for Object Recognition”. In: vol. 104. 2. Springer Science and Business Media LLC, Apr. 2013, pp. 154–171.

[26] Paul Viola and Michael Jones. “Rapid object detection using a boosted cascade of simple features”. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* Vol. 1. 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[27] Shivy Yohanandan. *”mAP (mean Average Precision) might confuse you!”*. https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2, [accessed 2022-02-14].

[28] Syed Sahil Abbas Zaidi et al. “A Survey of Modern Deep Learning based Object Detection Models”. In: vol. abs/2104.11892. 2021.