Automated On-Vehicle Road Defect Data Collection and Detection

Zachary Todd(✉) and Heyang Li

Department of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand 8041
zachary.todd@pg.canterbury.ac.nz, thomas.li@canterbury.ac.nz

Abstract. This paper proposes a pipeline for the automated on-vehicle data collection, filtering, and classification of road surface defects. The proposed pipeline provides a flexible framework that allows for the integration of a variety of systems. The pipelines flexibly allow for various sensors such as camera, 3D camera and lidar; computational resources such as on-vehicle edge computing or cloud computing; data transfer such as 5G or on-site upload; and data storage. The pipeline was tested using an edge computer on board a contracted road sweeping vehicle with an image taken every 10 s with image processing and evaluation occurring between. Post installation, the pipeline required no input from the driver of the sweeper vehicle besides turning on the road sweeper. The data was transferred via WiFi as the road sweeper was pulling up at the end of its shift. During operation around 21k road, defects were identified with over 90% of these images containing road defects.

Keywords: Deep learning · Data collection · Edge computing

1 Introduction

There have been strong improvements in the camera sensing technology, and the area of autonomous vehicles technology [12,18], with improved driver assistance, and a better understanding of the road environments [2,8,10,11,13,14].

However, there is still a significant gap in using those technologies for automatic road defects data collection and detection. Much of the research and development in this area has been using data collected manually, with manual data transfer and filtration steps that are not suitable for the large-scale automation needed to detect the whole road network.

Performing these tasks post-data collection allows for greater flexibility that is only limited by the computational resources available and the limitations of the data itself. Whereas, performing these tasks on-vehicle available decreases the available power resources which in turn decreases the computational potential. However, there are several advantages, with real-time data collection allowing for the filtering of data, as well as periodic or real-time reporting. In this context can in inform when a section of the road has been significantly damaged and needs to be fixed.

H. Aziz et al. (Eds.): AI 2022, LNAI 13728, pp. 3–14, 2022.
https://doi.org/10.1007/978-3-031-22695-3_1

The paper aims to utilise the potential advantages gained from performing data collection and detection on a collection vehicle while creating a flexible pipeline that allows for the proposed approach to fit a wide variety of needs and applications. The pipeline, in short consists of several modules, these being; image capture, location tracking, image prepossessing, image evaluation, data transfer, and data storage.

Initial validation of the approach is demonstrated by evaluating the image evaluation module and testing its ability to perform detection. This is then followed up using by running a case study utilising this model. The case study consists of a real-world application, in which a road sweeper completes its normal tasks while having the proposed application pipeline installed on the collection vehicle. The desired outcome of the case study is that despite running data collection and detection on-vehicle that the model is able to detect road defects and do so in a reliable manner.

2 Method

This section covers two pipelines used in both training applications. In addition, this section also covers the dataset used to train the image evaluation models and the apparatus the overall apparatus used in the application pipeline (Fig. 1).

2.1 Pipeline

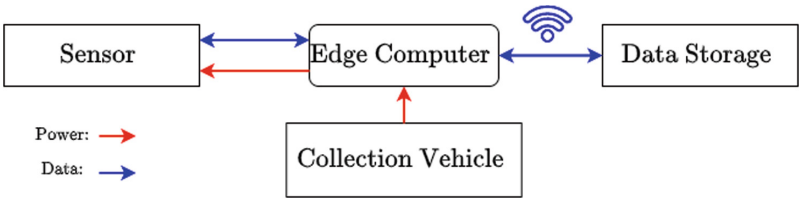

Fig. 1. Overview of the proposed pipeline.

Our approach consists of two pipelines. These are the training and application pipelines. The training pipeline is responsible for training and evaluating the model being used with the application pipeline. The application pipeline is the pipeline used in practices and the case study, with it being is made up of several modules, these being; image capture, location tracking, image prepossessing, image evaluation, data transfer, and data storage.

The image capture module's responsibility is to capture images and to send the captured images to the edge computer for preprocessing. For this, four implementations methods were considered:

(1) Wireless camera such as GoPro and communicate image collected over Bluetooth to edge computer.
(2) Smartphone powered via USB from the vehicle.
(3) Wired camera connected to the vehicle battery and communicating image using power over ethernet (PoE) to edge computer.
(4) Wired camera connected and powered using USB to edge computer.

With implementation (4) being used in the application pipeline. Implementation (1) is the most flexible on its face though it requires the battery of the wireless device to be charged. In addition to this problem, a number of reliability issues with the automatic pairing of a variety of wireless camera systems with the edge computer after they had been turned off, thereby requiring a person to interact with the system to ensure that the camera was charged and paired with the edge computer. Implementation (2) was the cheapest of the four implementations. However, like (1) there are several reliability issues such as image processing slowing down and the unreliable image capture rate, with the likely cause of these issues being due to wear and tear of the smartphone and overheating after extensive use. Implementation (3) though is very similar to implementation (4) because the increased power demand of the PoE made the powering of the system less reliable. Implementation (4) mitigates the reliability issues of the other implementations with there being no overheating, communication issues or major power fluctuations.

For location tracking, a GPS receiver was connected to the edge computer to provide the GPS position of the collection vehicle. This position information is provided with the captured images as metadata so that detected problems can be located.

For computation, a Intel NUC was selected, other options included low-power small GPU systems such as Nvidia Jetson Nanoor Nvidia Jetson Xavier, or the aforementioned smartphone. With the NUC over a GPU system because of cost and availability.

For image prepossessing, the image was cropped to remove the sky and other unrelieved features within the image and resized to the specification of the image evaluation model.

After prepossessing, the image evaluation model evaluates the images, with the responses sent to the data transfer and storage module. The evaluation model is dependent on the type of sensor used to capture the image, the computational resources available and the type of evaluation being performed. As performing the evaluation on-vehicle limits the potentially available power; the available computational resources were also limited.

The data transfer and storage module is responsible for transferring the image and metadata from the edge computer on the vehicle to where the data is stored. There are several methods to transfer the data from the edge computer. The following implementations were considered:

(1) 5G transfer direct from the edge computer to cloud storage.
(2) On board storage and manual retrieval and upload of the data.
(3) On-site WiFi connection with that data being transferred when the collection vehicle returns.

Implementation (1) allows for all of the collected images to be transferred from the edge computer during collection and allows for the possibility of real-time reporting, with the main limiting factor of this implementation being the cost of using 5G infrastructure. Implementation (2) is the simplest from a technology perspective. However, this would result in a lag in detection and reporting time, as well as requiring someone to interact with the edge computer regularly. Implementation 3 provides a compromise between the aforementioned implementations, allowing for a report per shift of the collection vehicle and requiring no manual interaction with the edge computer. However, implementation (3) does limit the number of images that can be transferred due to the bandwidth limitation of transferring data over WiFi while the collection vehicle is leaving and arriving from its station. To mitigate this, only the images (with associated metadata) in which detections occurred are transferred, thereby significantly decreasing the amount of data being transferred (Fig. 2).

2.2 Dataset

The dataset consists of 120k images collected around the South Island, New Zealand, specifically Canterbury, Nelson, Otago and Tasman regions. The dataset consists of all road segments of State-highways 65, 69, 73, 75, 85, and 87 and part of State-highways 1, 6, 7, 8 and 72, with the dataset encompassing over 3000 Km. The annotated dataset is a subset of this, including 19k images taken from these collections. The dataset was labelled with two classes, these being potholes and other defects. During annotations, the images were labelled with an encapsulating polygon. The images were labelled in two passes, with the first labelling the image in a batch of 500 images and then a reviewer to confirm their labels. On average each image took 20s to label with this being just over 30s including the second pass. Around 8% of the labelled images contained other defects and less than 1% contained potholes.

2.3 Apparatus

The training pipeline uses a Nvidia RTX2080 Ti GPU to train the models running with python 3.8, TensorFlow 2.0 and CUDA 10.1.

In the application pipeline, the collection used a road sweeper vehicle that was contracted to sweep roads around several suburbs in Christchurch, New Zealand. To capture the images a Logitech USB web camera was used. For computing, an Intel NUC with an I7 processor was used, with the NUC set to start upon receiving power. The NUC receive power using the road sweeper's auxiliary power outlet. To provide location information for the images, GlobalSat BU-353-S4 USB GPS receiver was used. Running on the NUC was python 3.8

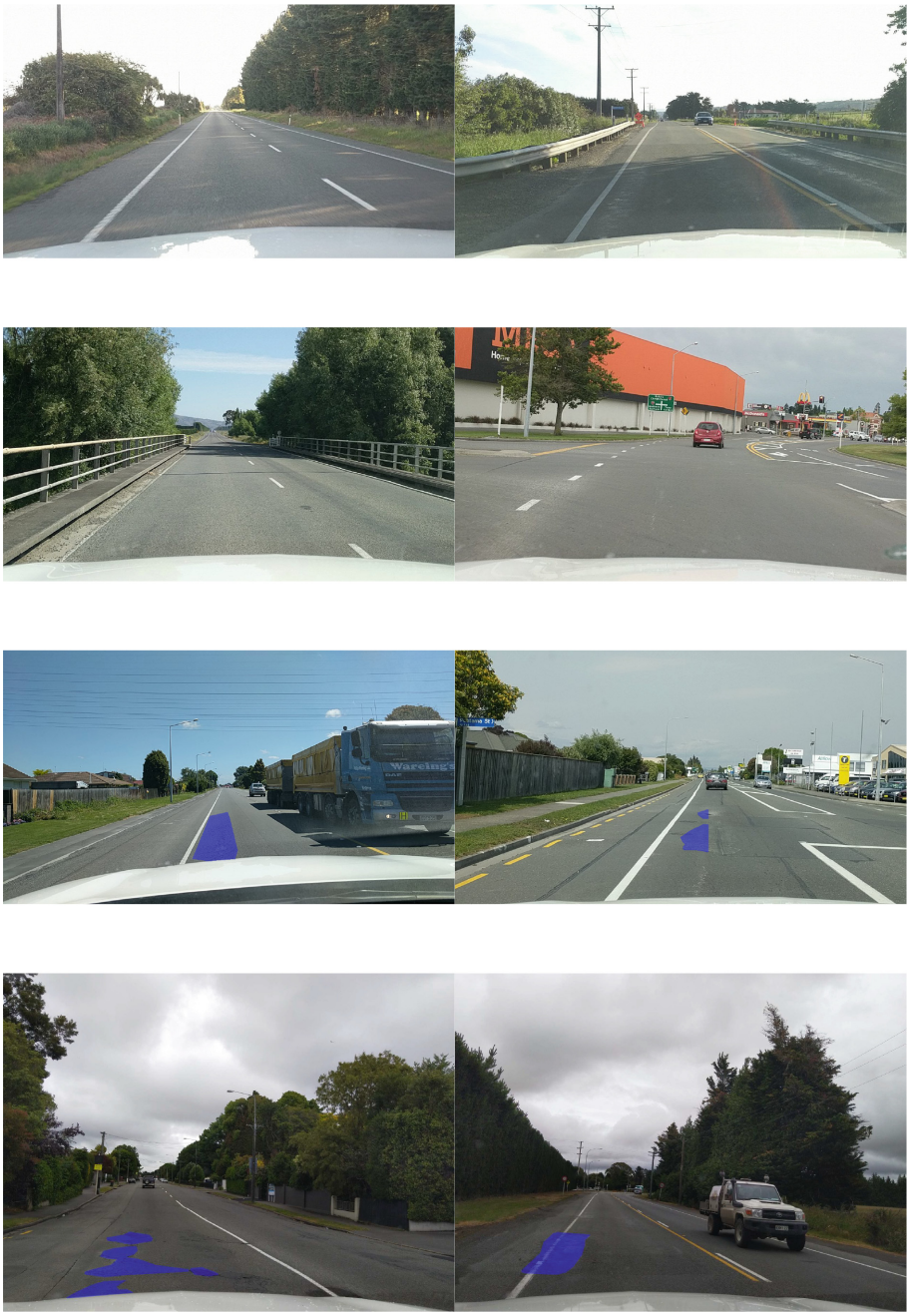

Fig. 2. Dataset examples. The top row consists of example images without defects and the second row contains annotated images with defects.

with TensorFlow 2.0 and Intel Optimization for TensorFlow to run the image evaluation model. For data transfer, the data is transferred over WiFi at the road sweeper station as the road sweeper is departs and arrives at the station.

The main limiting factor of the apparatus is the computing platform. The NUC with an I7 processor [1] was chosen as a compromise between power consumption and computation with lower computation processors such as Pentium and I3 providing similar power requirements though with less computation resources. A potentially better computation platform to use would a low power consumption small GPU systems such as Nvidia Jetson Nano or Nvidia Jetson Xavier. However, due to global supply chain issues, this was not possible at the time.

2.4 Model Training

Two types of models were considered for the image evaluation model; these being image classification and instance segmentation models. Image classification models classify an image into a number of set classes, with each image having an assorted class. Instance segmentation models provide an encapsulating polygon or mask of all of the instances of the relevant objects within an image, as well as classifying these objects. Each input of instance segmenting results in either a set of polygons-class or image-class pairs to communicate what parts of the image belong to each instance object and their assorted classes.

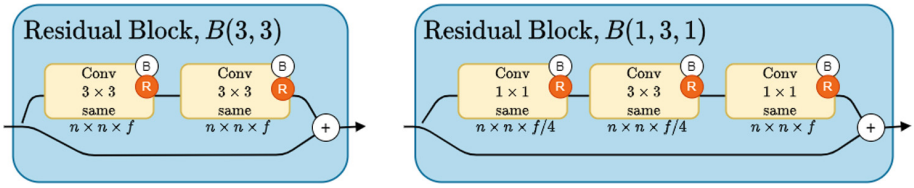

Fig. 3. ResNet Block, With the size of the output volume of each conv layer displayed below each conv layer.

The ResNet [7] convolutional neural network (CNN) was selected as the classification model. ResNet main feature is its residual block (Fig. 3), with the block consisting of two parallel branches. The first is a branch is a series of convolutional (conv) layers either two 3×3 conv layers (B(3, 3)) or 1×1, 3×3 and 1×1 conv layers (B(1, 3, 1)) and the second branch make no changes and passes the input volume froward to be added to the first branch. ResNet starts with a 7 conv layer followed by a maxpool and then followed by fours stages of residue blocks. The number of residual blocks per stage is different for each ResNet variant. After each conv layer there is batch normalisation [9] followed by ReLU activation [17]. After the fourth stage, there are two fully connected layers to finish the network.

Commonly used ResNet variants:

- ResNet18: block: B(3, 3), blocks per stage [2, 2, 2, 2]
- ResNet34: block: B(3, 3), blocks per stage [3, 4, 6, 3]
- ResNet50: block: B(1, 3, 1), blocks per stage [3, 4, 6, 3]
- ResNet101: block: B(1, 3, 1), blocks per stage [3, 4, 23, 3]
- ResNet152: block: B(1, 3, 1), blocks per stage [3, 8, 36, 3]

For the classification experiment, ResNet50 was selected as it is the largest of the ResNet variant while meeting the computation limitation of the edge computer.

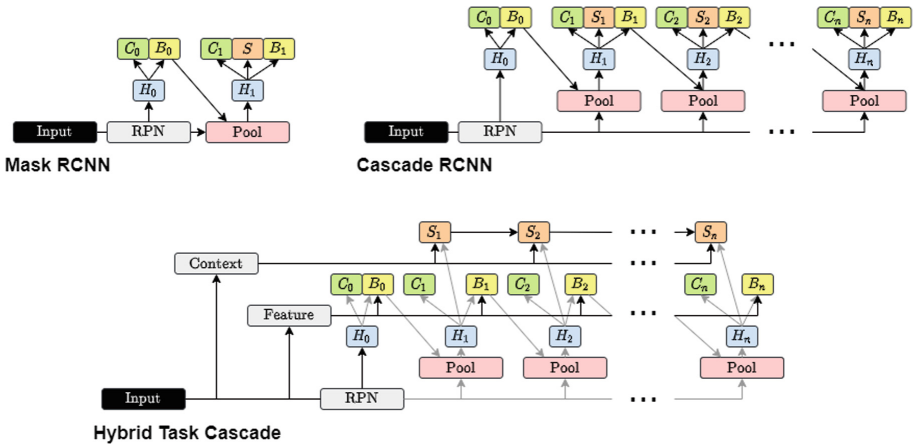

Fig. 4. Instance segmentation method structure.

For instance segmentation, three CNN models were considered; Mask RCNN, Cascade RCNN, and Hybrid Task Cascade (HTC) [3,4,6]. All three methods work similarly by using a backbone network such as ResNet to generate region proposals and the difference in the overall structure is shown in Fig. 4. Mask RCNN then uses a pooling layer to warp the variable size region proposals into a predefined size shape. Finally, these pooled regions are then connected to a head layer that predicts the class, bounding boxes and mask.

Cascade RCNN has two improvements over Mask RCNN these being cascade bounding box (bbox) regression and cascade detection. The cascade bbox regression improves quality by providing a series of regressions with each regressor having a stricter intersection over union (IoU) acceptance threshold. The cascade detection resampling of the positive examples of previous regressors with a higher IoU, as a regressor with an IoU threshold of μ, will produce bboxes with IoU score higher than μ.

HTC interleaves mask and bbox detectors with each bounding box and masks being connected and each aggressor feeding to the next regressor in the cascade. HTC also used a separate spatial context branch for masks instead of using the region proposal network.

For the instance segmentation experiment, HTC with ResNet50 was selected. This is because HTC has better results on large general datasets [15] than Cascade RCNN and Mask RCNN while requiring amount a similar amount of computational resources.

3 Experiments and Results

This section covers the experiments used to evaluate the selected models on the dataset, as well as discusses the case study used to evaluate the application pipeline.

3.1 Model Evaluation

For evaluation, the annotated dataset is split into two subsets with 80% being used to train and 20% being used to test the models. Both the classification and segmentation models use the AutoAugment [5] method to augment the annotate dataset and AdamW optimiser [16] to optimise the gradient descent, and were trained for 300 epochs.

The classification result as shown in Table 1 shows accuracy across all classes of 91.4%. With none of the Pothole images being classified as 'No Defect' and

Table 1. Classification confusion matrix

	No defect	Pothole	Other defect	Total
No Defect	3202	14	297	3513
Pothole	0	32	8	40
Other Defect	15	2	338	355
Total	3217	48	643	3908

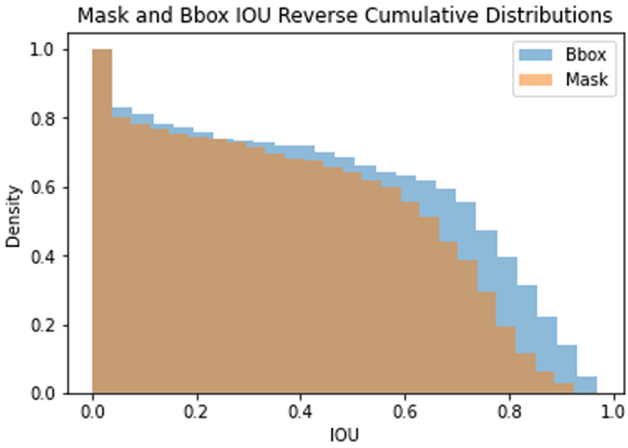

Fig. 5. Mask and Bounding box results

Fig. 6. Detected defect examples, with pothole detection examples in the first two rows and other defects detection examples in the last two rows.

only 15 of the 'Other defect' class being classed as 'No Defect'. These results demonstrated that the model is able to distinguish between images that do and do not contain defects, as well as distinguishing between defects that are and are not potholes (Fig. 5).

The instance segmentation results show the HTC model on the test dataset has a mask and bbox AP@50 (average precision using 0.5 IoU as the acceptance threshold) of 0.69 and 0.71 respectively, and a mask and bbox AP@75 of 0.58 and 0.63 respectively, with there being a sharp drop-off in acceptance with an IoU of 0.6. These results demonstrate that the detection and segmentation are viable in most cases with almost 70% of the images having a greater than 50% mask and bbox IoU.

3.2 Case Study

The computer, camera, and GPS receiver were placed inside and connected to the road sweeper vehicle. The road sweeper ran as normal for a month collecting data and running the application pipeline. The model used to determine if a defect was present was ResNet50. Although the segmentation model performs the classification method has a faster inference time as it does not have to run the associated detection and segmentation components of HTC.

During operation, the only reliability issues experienced were interference between the GPS receiver and a GPS receiver built into the collection vehicle. This caused several of the detected defect images to not have assorted metadata. Although this unreliability can easily be remedied in the case where a collection vehicle has GPS then the image captured time code and the time code of the collection vehicle GPS can be synced to get the location of the captured image. During the month of operation over 21k images were identified with 92.4% containing defects (Fig. 6).

4 Conclusion

The application pipeline demonstrates the viability of automating the detection of road defects in a flexible, accurate and reliable manner. The flexibility is demonstrated in the relatively low complexity of the modules allowing for the accommodation of a variety of sensors, computation platforms and data transformation methods. The accuracy is demonstrated in the results on the training dataset, as well as the presence of road defects within the case study. However, the result that over 90% of the reported images have defects does not guarantee that the result of the dataset has been able to transfer to the context of the case study as there is not a full picture of the images that were determined to not contain a defect. It does provide some confidence that this mapping between the two similar contexts has occurred. Finally, the reliability of the application pipeline is demonstrated by the reliability of the case study, with only minor reliability issues such as the interference in the GPS receiver.

A natural progression from these outcomes would be to increase both the resolution of the problem from the perspective of both the types of road defects being detected, as well as the complexity of reporting being performed by the edge computer. Increasing the class resolution could achieve by training an evaluation model on more diverse classes, for example, adding road defect classes such as cracking, surface defects and surface distress. Increasing the complexity could be achieved by running an instance segmentation model to provide the location and the size of defects within an image, as well as using an onboard GPU to speed up the computation.

References

1. Comparison charts for intel® core™ desktop processor family. https://www.intel.com/content/www/us/en/support/articles/000005505/processors.html
2. Agrawal, R., Chhadva, Y., Addagarla, S., Chaudhari, S.: Road surface classification and subsequent pothole detection using deep learning. In: 2021 2nd International Conference for Emerging Technology (INCET), pp. 1–6. IEEE (2021)
3. Cai, Z., Vasconcelos, N.: Cascade r-cnn: high quality object detection and instance segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **43**, 1483–1498 (2019)
4. Chen, K., et al.: Hybrid task cascade for instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4974–4983 (2019)
5. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: Autoaugment: learning augmentation policies from data. arXiv preprint arXiv:1805.09501 (2018)
6. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask r-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV). IEEE (2017). https://doi.org/10.1109/iccv.2017.322
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2016)
8. Hu, Y., Furukawa, T.: Degenerate near-planar 3d reconstruction from two overlapped images for road defects detection. Sensors **20**(6), 1640 (2020)
9. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
10. Jung, J., Bae, S.H.: Real-time road lane detection in urban areas using lidar data. Electronics **7**(11), 276 (2018)
11. Kim, J., Kim, J., Jang, G.J., Lee, M.: Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection. Neural Netw. **87**, 109–121 (2017)
12. Levinson, J., et al.: Towards fully autonomous driving: systems and algorithms. In: 2011 IEEE Intelligent Vehicles Symposium (IV), pp. 163–168. IEEE (2011)
13. Li, H.T., Todd, Z., Bielski, N.: Equirectangular image data detection, segmentation and classification of varying sized traffic signs: a comparison of deep learning methods (2022)
14. Li, H.T., Todd, Z., Bielski, N., Carroll, F.: 3d lidar point-cloud projection operator and transfer machine learning for effective road surface features detection and segmentation. Visual Comput. **38**(5), 1759–1774 (2022)
15. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48

16. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
17. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML, pp. 807–814 (2010). https://icml.cc/Conferences/2010/papers/432.pdf
18. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: common practices and emerging technologies. IEEE Access **8**, 58443–58469 (2020)