Estimation of Speed and Distance of Surrounding Vehicles from a Single Camera

Mirko Zaffaroni[1,3(✉)], Marco Grangetto[1], and Alessandro Farasin[2,3]

[1] Computer Science Department, University of Torino, Turin, Italy
{mirko.zaffaroni,marco.grangetto}@unito.it
[2] Department of Control and Computer Engineering, Politecnico di Torino, Turin, Italy
alessandro.farasin@polito.it
[3] Fondazione LINKS, Microsoft Innovation Center, Turin, Italy
{mirko.zaffaroni,alessandro.farasin}@linksfoundation.com

Abstract. Deep Learning requires huge amount of data with related labels, that are necessary for proper training. Thanks to modern videogames, which aim at photorealism, it is possible to easily obtain synthetic dataset by extracting information directly from the game engine. The intent is to use data extracted from a videogame to obtain a representation of various scenarios and train a deep neural network to infer the information required for a specific task. In this work we focus on computer vision aids for automotive applications and we target to estimate the distance and speed of the surrounding vehicles by using a single dashboard camera. We propose two network models for distance and speed estimation, respectively. We show that training them by using synthetic images generated by a game engine is a viable solution that turns out to be very effective in real settings.

Keywords: Automotive · Deep Learning · Computer vision · Synthetic dataset

1 Introduction

The availability of large amount of indexed and labeled images is key to the successful design of many complex vision tasks leveraging on powerful Deep Learning (DL) techniques based on Convolutional Neural Networks (CNN). The creation of a large dataset able to correctly represent the target scenario and allowing the trained neural network to generalize in real applications remain a critical design step. Resorting to human visual inspection and manual labeling does not represent a viable solution in many scenarios. In fact, manual labeling does not scale very well to large datasets, except for very simple and repetitive tasks that do not require particular expertise where one can resort to crowdsourcing [3]. Moreover, doubts may arise on the quality of the collected information and potential unexpected bias. Finally for some tasks manual labeling

E. Ricci et al. (Eds.): ICIAP 2019, LNCS 11751, pp. 388–398, 2019.
https://doi.org/10.1007/978-3-030-30642-7_35

is simply not possible as is the case in the automotive scenario targeted in this work, where physical quantities such as distance and speed must be estimated from images. One option in the automotive field is to use special vehicles with ad-hoc, and usually expensive, settings and sensors capable of gathering the information required for training. The set-up and maintenance costs of such real experiments may represent a significant barrier.

In this context the use of computer graphics (CG) simulation is emerging as a powerful source of visual information. CG allows to obtain large sized dataset in a short time and with the usage of cheap resources [11]. In addition, modern video games are getting closer and closer to photorealism, thus promising to bridge the gap between visual simulation and reality that is likely to be the key to training computer vision systems that are effective in real life. Moreover, simulation makes experimental and environmental settings more flexible: i.e. in the automotive field, datasets with heterogeneous driving scenarios can be generated and subjected to different weather and lighting conditions. Higher heterogeneity can significantly improve the trained model in terms of robustness and generalization. As an example by using a simple 3D rendering technique such as Ray-casting in a virtual environment one can get a simulation of a LIDAR scanner [14,15] easily obtaining information on the distance of the elements within the image. Furthermore, it is possible to get data that are normally difficult to obtain, such as measurements of the speed of all the surrounding vehicles speed, that would require a complex setup on the real field.

Clearly, to customize and generate a dataset for visual training one either needs to design a complex CG simulation environment or exploit existing high quality game engines. The second option is viable if one has access to the source code to easily extract information on the entities and the various elements that make up the gaming environment (bounding box, size, distance from the observer, type of entity, etc.). Nowadays, there are few open source simulators that can be used to extract synthetic datasets. In the automotive environment TORCS [1] can be used; however this tool allows the representation of only a few scenarios with limited photorealism. On the other hand, commercial videogames car run very realistic CG and are equipped with intelligent agents to simulate entity actions, e.g. a pedestrian walking. For this reason the research community has recently got interest in Grand Theft Auto V (GTAV) [4,9,12,13], a popular open world videogame that, thanks to the libraries developed by third parties, allows one to extract data from the gaming environment.

In a similar fashion to the work done in [13], in this work we propose two CNN architectures to estimate distance and speed of the surrounding vehicles from a single camera with windshield view (see Fig. 1). Training has been achieved with GTAV simulations and performance validated in real life. The main contributions of this paper lie in:

- a DL model the uses a pre-trained deep CNN to extract semantic features from vehicles images and uses them to predict distance of the surrounding vehicles from a single camera with windshield view;
- a model which uses optical-flow information to predict the speed of surrounding vehicles from pixel motion between pairs of video frames.

– a training framework based on videogame simulation for the creation of training and testing datasets in the automotive field;
– we show that training with synthetic but photorealistic images represents a viable alternative to more expensive experimental data collection, with promising results in the estimation of distance and speed of the vehicles on the road, observed with a single camera.

Dataset, code and pre-trained model are publicly available and can be found at: https://github.com/mirkozaff/DeepGTAPrediction.

2 Methodology

In this section we will introduce the framework used to collect data, preprocess them and the models used to accomplish the vision task.

2.1 Data Collection

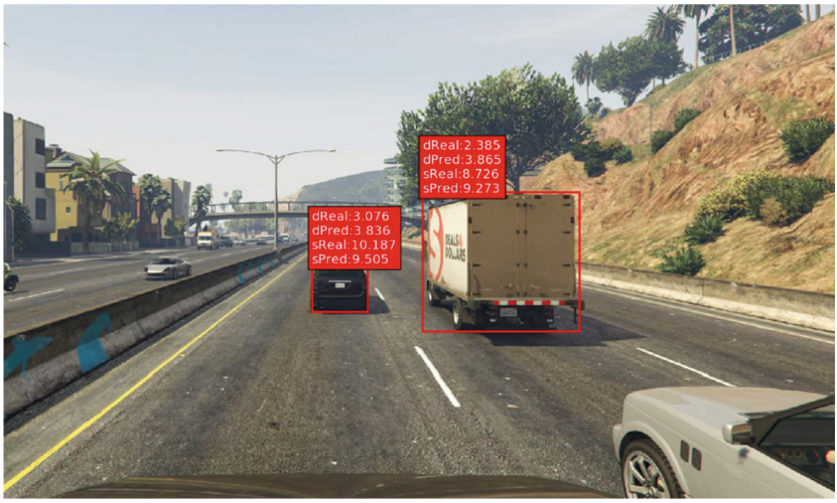

Fig. 1. Example of a photorealistic frames extracted from the game environment.

Data have been collected from GTAV thanks to Script Hook V library (SHL), which allows to easily access GTAV native function and extract information about the entities (vehicles) from the game environment. Images and corresponding information have been generated by configuring an in-game agent that drives a vehicle and letting it wander the streets; during the simulation one can collect the required information by queering the game engine through SHL calls. For our goal we built a dataset by collecting, for every vehicle in the range of 30 m from the player, the following items:

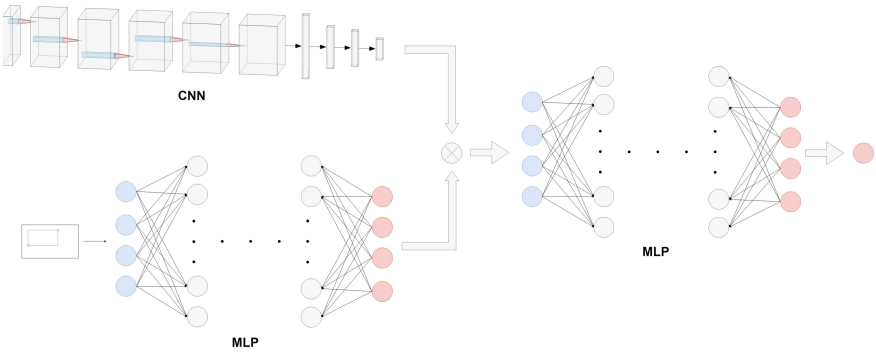

Fig. 2. Prototype architecture of our proposed distance model. A CNN is used to extract semantic features, while the MLP branch is used to learn a spatial representation of the coordinates. Then all these information are merged and decoded into output values trough a last MLP.

- *Frame:* 1920×1080 image captured at 30 Hz, gathered by setting the in-game camera on the dashboard.
- *Entity ID:* identifier of the vehicle to track it in multiple frames.
- *Entity speed, distance:* speed and distance of the vehicle.
- *Entity bounding box:* pair of coordinates that define the *bounding box* B_1 of the vehicle in the captured frame; this is computed by projecting the 3D bounding box obtained from the game engine into 2D screen coordinates.

We noted that the bounding boxes extracted by SHL are often inaccurate and present a drift caused by the delay in the response to each SHL query. To get precise bounding boxes we use pretrained Mask R-CNN [7] model to detect each vehicle in the dumped frame (the same model will be used in the testing phase on real-life images). For each detected vehicle Mask R-CNN output a bounding box B_2. To univocally map B_2 onto previously computed B_1 (and corresponding speed and distance data) we set a threshold on the intersection over union $IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2}$. Only the entities showing $IoU \geq 0.7$ are included in the dataset. The selected threshold has also the effect to filter out some vehicles that cannot be reliably detected due to poor visual conditions.

2.2 Models

Distance. The first model we present is designed to estimate distance from surrounding vehicles using a single camera with windshield view. The input is a single image from which vehicles bounding boxes are detected, e.g. by using Mask R-CNN. As shown in Fig. 2 the architecture is composed by two branches:

- *First branch:* pretrained ResNet50 [8] used to extract semantic features of the target vehicle from the corresponding frame. To this end the classification layers in ResNet50 are removed. The input is a frame crop based on the

bounding box B_2. In particular, each vehicle is extracted from the frame by cropping and resizing it at 224×224, that is the input resolution expected by ResNet50.
– *Second branch:* Multi Layer Perceptron (MLP), that is used to encode the coordinate of the bounding box B_2 into a higher multi-dimensional space. We selected the ELU activation function [5] for each layer, in place of the classic $ReLU$; we noted that in our scenario ELU is very effective to avoid the dead-neuron problem [16].

These two branches are then concatenated and processed through a final MLP responsible of predicting distance from the fused information produced by image pixels and bounding box coordinates. Semantic features provided by the first branch are important because vehicles appearing at the same scale in the image may represent different classes of object; clearly, we cannot base the distance estimation on the sole geometric information, i.e. bounding box dimension and position analyzed by the second branch. In other words, as in real life, we must take into account that cars are smaller than trucks when guessing the corresponding distance.

Speed. The model proposed to estimate the speed of surrounding vehicles is designed with a similar approach using two branches: (i) semantic based on images, (ii) geometrical based on bounding boxes. When speed is regarded one clearly has to consider at least two consecutive images to gather object displacement over time. One option would be to directly process frames. In this paper we propose to use as input the *Optical Flow* (OF) estimated from current (and previous) frame under analysis. OF is a dense vector field that represents the displacement of every pixel, e.g. computed using the Farneback method [6]. Moreover, we use the two bounding boxes of the same vehicle tracked in two consecutive frames (tracking is simply obtained in our GTAV dataset using entity IDs, while it will require additional processing in real setting). The structure of the model for speed estimation is as follows:

– *First branch:* PilotNet [2], a CNN proposed to learn salient points of the road for autonomous driving, is used to process the input OF. The OF vector fields is represented as an image with two bands representing vector magnitude and direction, respectively. The obtained OF image is cropped according to B_2 and resized to 200×66 (the resolution expected by PilotNet). We improved the original PilotNet model by adding batch normalization to the convolutional layer in order to speed up convergence and by using ELU activation function and *Dropout* on the last fully-connected layers.
– *Second branch:* same MLP structure used in previous model to encode in a higher multidimensional space the coordinates of bounding boxes; differently from the distance estimation model, now the input is represented by two bounding boxes associated to the same vehicle tracked in two successive frames.

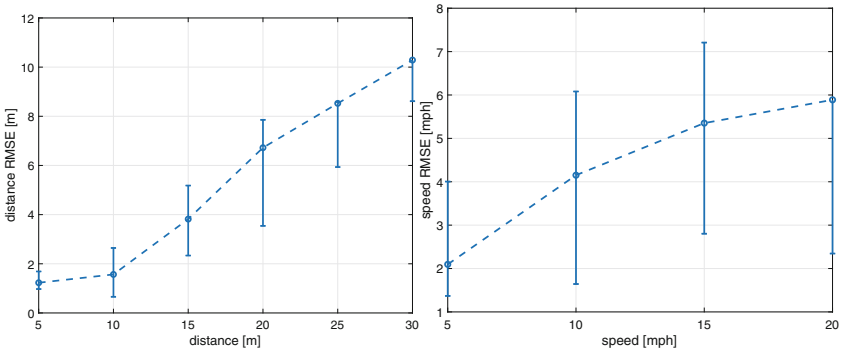

Fig. 3. RMSE on distance (a) and speed (b) estimate.

Finally, the features extracted on the two branches are concatenated and processed by final MLP as already shown in Fig. 2 to estimate speed. Clearly, we adopted a heuristic similar to the one proposed for distance: the lower branch extracts motion features based on bounding box geometrical information and displacement; the Pilonet branch encodes richer features that depend on the OF of all the pixels corresponding to a vehicle and potentially extract also semantic characteristics.

3 Network Training

Using the process presented in Sect. 2.1 it is possible to generate datasets comprising as many vehicles, labeled with distance and speed, as desired. In this work, we employ a training and validation sets with 250,000 and 2,500 samples, respectively to train the distance model. As far as the speed model is regarded, we extract from previous dataset all vehicles visible in two consecutive frames generating a set of 180,000 OF images for training and 1,500 for validation.

The proposed models have been trained using *Mean Squared Error* (MSE) as loss function and *Adam* optimizer with the following parameters: $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$. In order to avoid over-fitting the training has been stopped as soon as the loss computed on the validation set ceases to decrease; in our experiments this usually happened after about 15 training epochs.

Training of all MLP sub-networks has been done using *Dropout* with parameter $p = 0.4$. In the distance model ResNet50 weights pretrained on *ImageNet* have been kept fixed, while optimizing only the other MLP sub-networks. In the speed model all the network has been trained since no pretrained PilotNet useful in our context was already available.

Training has been run on a PC with Intel(R) Core(TM) i9-7940X CPU, 128 GB RAM and NVIDIA GeForce GTX 1080 Ti (x4). Testing was performed both on the same machine and on a lighter one with Intel(R) Core(TM) i5-6400,

8 GB RAM and NVIDIA GeForce GTX 1050 Ti. This latter has been selected as representative of the hardware that one expects to have on board a vehicle as opposed to the previous higher-end server.

4 Experimental Results

In this section we describe the experimental results obtained in different simulated and real settings.

4.1 Testing on Synthetic Dataset

As a first step, the estimation accuracy of the trained models has been evaluated on synthetic datasets of size 2,500 and 1,800 for distance and speed, respectively. These testing sets have been generated using the GTAV simulation described in Sect. 2.1. It is worth pointing out that training and testing sets have been generated with different random simulations to make them independent.

The proposed models are able to predict distance with a Root MSE (RMSE) of about **2.46** [m] and speed with RMSE of about **2.75** [mph]. In Fig. 1 we provide an example of the obtained visual results. The image shows a car and a truck with labels representing ground truth and predicted distance and speed. For the car the model predicts a distance of 3.8 m versus a real value of 3 m and 9.5 mph speed versus 10.2 mph.

In Fig. 3 we analyze in more details the estimation accuracy. In particular, Fig. 3(a) shows the RMSE on distance as a function of the actual distance range; to this end we compute RMSE (the circle marker) and standard deviation of the estimation error (vertical bars) by binning the collected results in increasing distance ranges of 5 m in the interval $(0, 30)$ m (the top error bar indicates an overestimate, whereas the bottom segment represents an underestimate). It can be noted that, as one may expects, the RMSE increases for larger distances. Overall the distance estimates are quite accurate and unbiased (almost symmetric error bars) within a range of 15 m: as an example the RMSE in the range $(0, 5)$ m is 1.23 m and in the range $(5, 10)$ m is 1.57 m. For farther vehicles the predictions are less accurate and the model tends to underestimate the distance. This can be explained by the fact that at distances greater than 15 m vehicles are represented in the image by fewer pixels limiting the information extraction capabilities of the convolutional layers.

In Fig. 3b we show similar RMSE analysis on the speed estimate as a function of the speed up to 20 mph, that is the maximum value that can be simulated in GTAV. It can be noted that speed RMSE increases as a function of speed. The obtained results shows that the proposed network can guess the speed of the surrounding vehicles at reasonable level by using a single camera view. As an example, in the speed range $(0, 5)$ mph we get RMSE equal to 2.10 mph, and in the range $(5, 10)$ mph we get RMSE equal to 4.15 mph. We expect to be able to improve such results by increasing the number of temporal frames analyzed by the model and using better OF representations.

4.2 Testing on Real Dataset

As already mentioned in Sect. 1 one of the goal of this work is to understand if CG simulation can be used to effectively train DL models that can be employed in real settings. To answer this question we need vehicles videos with annotated data. To this end we used the video sequence provided in [10] and corresponding distance estimates as an example of real dataset. In Fig. 4 we compare the RMSE on distance prediction obtained on the real and synthetic datasets subdivided in 2 m ranges (please note that images from [10] are limited to a 6 m range). It can be noted that the proposed model is quite robust and generalizes well in real life scenario, even if the actual environment can be significantly different with respect to GTAV simualtion. Indeed it can be noted that, in the experimented distance range, the RMSE of the real dataset increases by less than 0.5 m with respect to the synthetic testing set. Overall the test RMSE was **1.21** on synthetic data and **1.40** on real data. We would like to perform similar experiment with speed prediction but unfortunately, to the best of our knowledge, there is no publicly available dataset that can be employed to this end. Indeed, the setup of a real road experimentation is quite complex.

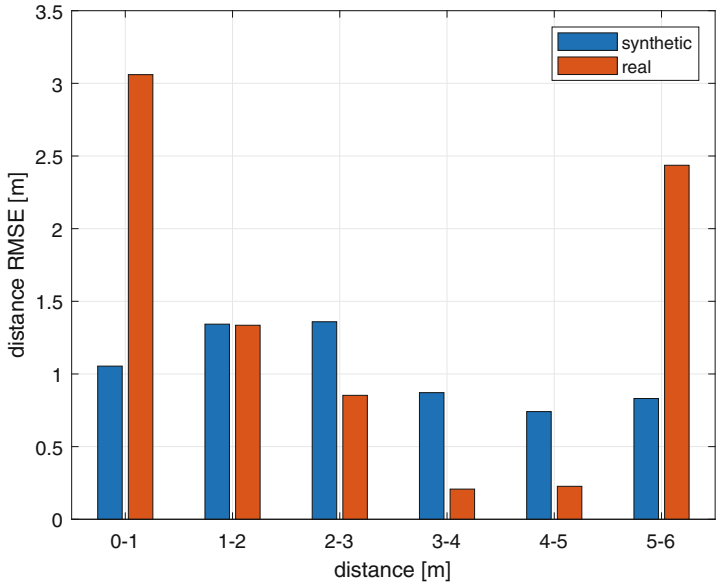

Fig. 4. RMSE on distance estimate (synthetic and real images).

Fig. 5. Example of distance estimate in real environment.

4.3 Testing on the Road

Finally, the model has been tested on a video recorded on the streets around our city using a Go Pro Hero 6 placed on the car dashboard. This last experiment was accomplished in real road environments (both urban and highway) to check the meaningfulness of the obtained predictions. In this case we do not have ground truth data. By analyzing the operations of the proposed system in real live we noted that the predicted distances are plausible and coherent, i.e. vehicles appearing at the same distance are assigned the same value, and approaching vehicles exhibit decreasing distance. As in previous experiment also in this case the model performance is not significantly impaired by the road environment that is very much different with respect the GTAV scenario.

4.4 Computational Cost

In this section we analyze the computational cost of the proposed solutions by measuring the execution time of different algorithmic steps of the two hardware architectures described in Sect. 3; these are meant to be representative of a workstation performing remote computation and lower-end hardware compatible with in vehicle system. In Table 1 we show the average time taken by the calculation of the bounding box, speed and distance estimate for an image with 5 vehicles (on average). It can be noted that to get acceptable delays (compatible with real time requirements of advanced driver-assistance systems) it is necessary to use powerful workstation. As expected the speed estimate represent the slowest module.

Table 1. Execution time of different algorithmic steps.

Work station	
290 ms	Bounding box
15 ms	Distance
45 ms	Speed
120 ms	Latency
500 ms	Total
On board PC	
2 s	Bounding box
280 ms	Distance
880 ms	Speed
3 s	Total

5 Conclusions

In this paper, we proposed two models to accomplish two different tasks: speed and distance prediction using a single camera looking at the road from the driver perspective. Since for such tasks it is either technically difficult or quite expensive to get real video sequences for training CNNs, in this paper we proposed to use simulated data generated by means of a popular game engine. Such an approach allowed us to collect photorealistic driving scenes, where all the visible vehicles can be labeled with distance and speed information. We designed two DL models built around similar ideas: one branch extracts features from the input images, a second one maps vehicles' bounding boxes (dimension and position) to higher dimensional space, and a last MLP network infers distance or speed from all the extracted features. The estimation accuracy has been evaluated on both synthetic and real data showing that the simulated images can be used to effectively train the proposed models. For future work we plan to improve the models by substituting the bounding box detection network with a lighter version and using a DL approach to estimate OF for speed prediction. Moreover, we plan to enrich the input available to the network by including parameters that can be logged on board a car such as throttle, brake and steering data to mention a few.

Acknowledgement. The research leading to these results has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 713788 ("optiTruck" project).

References

1. Torcs (2007). http://torcs.sourceforge.net/
2. Bojarski, M., et al.: Explaining how a deep neural network trained with end-to-end learning steers a car (2017). http://arxiv.org/abs/1704.07911

3. Buhrmester, M., Kwang, T., Gosling, S.D.: Amazon's mechanical turk: a new source of inexpensive, yet high-quality, data? Perspect. Psychol. Sci. **6**(1), 3–5 (2011)
4. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: DeepDriving: learning affordance for direct perception in autonomous driving. In: ICCV 2015 (2015)
5. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus) (2015). http://arxiv.org/abs/1511.07289
6. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In: Bigun, J., Gustavsson, T. (eds.) SCIA 2003. LNCS, vol. 2749, pp. 363–370. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45103-X_50
7. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN (2017). http://arxiv.org/abs/1703.06870
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). http://arxiv.org/abs/1512.03385
9. Johnson-Roberson, M., Barto, C., Mehta, R., Sridhar, S.N., Vasudevan, R.: Driving in the matrix: can virtual worlds replace human-generated annotations for real world tasks? (2016). http://arxiv.org/abs/1610.01983
10. Karagiannis, V.: Distance estimation between vehicles based on fixed dimensions licence plates. Ph.D. thesis, University Of Patras (2017)
11. 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
12. Martinez, M., Sitawarin, C., Finch, K., Meincke, L., Yablonski, A., Kornhauser, A.L.: Beyond grand theft auto V for training, testing and enhancing deep learning in self driving cars (2017). http://arxiv.org/abs/1712.01397
13. Palazzi, A., Borghi, G., Abati, D., Calderara, S., Cucchiara, R.: Learning to map vehicles into bird's eye view. In: Battiato, S., Gallo, G., Schettini, R., Stanco, F. (eds.) ICIAP 2017. LNCS, vol. 10484, pp. 233–243. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68560-1_21
14. Wu, B., Wan, A., Yue, X., Keutzer, K.: Squeezeseg: convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D lidar point cloud (2017). http://arxiv.org/abs/1710.07368
15. Yue, X., Wu, B., Seshia, S.A., Keutzer, K., Sangiovanni-Vincentelli, A.L.: A lidar point cloud generator: from a virtual world to autonomous driving. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval
16. Zaheer, R., Shaziya, H.: GPU-based empirical evaluation of activation functions in convolutional neural networks. In: 2018 2nd International Conference on Inventive Systems and Control (ICISC)