Including transfer learning and synthetic data in a training process of a 2D object detector for autonomous driving

Miguel Antunes, Luis M. Bergasa, Javier Araluce, Rodrigo Gutiérrez, J. Felipe Arango, and Manuel Ocaña

RobeSafe research group, Electronics Department, University of Alcalá, Spain
{miguel.antunes, luism.bergasa, javier.araluce, rodrigo.gutierrez, juanfelipe.arango, manuel.ocanna}@uah.es

[1]This work has been funded from Scholarship for Introduction to Research activity by University of Alcalá

Abstract. Nowadays the use of deep learning (DL) based systems is widely extended in several areas such as facial recognition, voice and audio processing or perception systems. The training process that must be performed for proper functionality requires a large amount of data with the required characteristics needed for the task to be executed. The process of obtaining new adequate training data is complex and tedious, therefore multiple techniques such as data augmentation or transfer learning have been developed in order to have a greater amount of knowledge in the network without the need to search for new data sources. The aim of this paper is to study the effect of the inclusion of knowledge from transfer learning in a 2D image detector trained with real world and synthetic data from multiple sources. The detector that is going to be trained will be focused on autonomous driving tasks, therefore we decide to use KITTI as the real world data source and our AD PerDevkit (based on CARLA) and Virtual-KITTI as the synthetic sources.

Keywords: Deep Learning, transfer learning, 2D object detection, simulation, KITTI, CARLA.

1 INTRODUCTION

In recent years, Deep Learning (DL) based systems have become a fundamental tool for computer vision tasks, especially thanks to their remarkable performance improvement and the evolution of the hardware on which they are executed. The training from scratch of DL based detectors requires a very large amount of data rich in diverse information, therefore many techniques exist to accelerate and improve this process. The number of training data can be increased by using techniques such as traditional data augmentation, which applies geometric transformations such as rotation or compression to the images, or even using synthetic data from some source to complement the real data. Another common approach is the use of transfer learning, in which weights trained on an external

database are used to provide initial knowledge to the system to later specialize in the task to be performed. In this paper we study the effect of using and mixing several of these methods on the training process of a 2D object detector focused on autonomous driving tasks (Figure 1). The images must come from specialized and open-source databases, for this reason the real data used for training, test and validation comes from two of the most used datasets for this type of tasks: KITTI [1] and Waymo [2]. On the other hand, one of the most widely used tools to test the performance of driving architectures in simulation is CARLA [3], which, using Unreal Engine and ROS, provides realistic looking environments with accurate information about the vehicle's sensors and the rest of the elements of the simulation. Taking advantage of the data provided by CARLA, we have developed our own ground truth generation tool: AD PerDevkit, which will be used to obtain part of the synthetic data. The rest of the synthetic images are obtained from the Virtual Kitti 2 dataset [4][5] which applies a recreation of the Kitti data in a virtual environment based on Unity [6] from which more synthetic sequences are generated with different conditions such as weather or time. The main difference between the two methods is that CARLA is a simulator, which means that a multitude of tests can be performed on its maps and a virtually unlimited amount of data can be generated. VKitti, on the other hand, tries to bring the knowledge provided by a real database to a virtual domain to provide a better generalization of object characteristics while maintaining similarity with the original dataset.

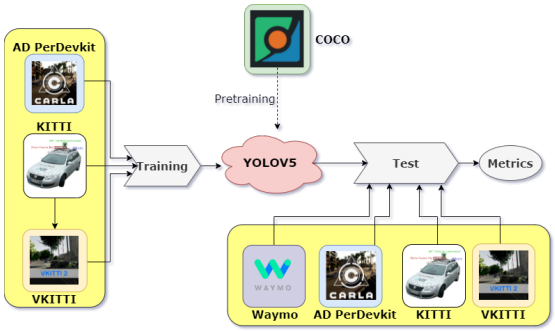

Fig. 1: Training, test and validation images mix for 2D object detection.

2 RELATED WORK

2.1 Data augmentation

Data augmentation is one of the most common techniques in DL based system training. In [7] several traditional methods such as rotation, compression, filtering or color changes are presented along with other state of the art DL based methods such as Generative Adversarial Network (GAN) or Adversarial Training. These methods can contribute to the network with generalized knowledge that helps to improve the results and avoid overfitting with training data. In this

paper we will combine synthetic data from various sources with various classic data augmentation methods such as blur, transformation to grayscale, image compression or image contrast modification.

2.2 Artificial data from virtual environments

The CARLA autonomous driving simulator [3] is an open source tool that aims to provide a realistic environment in which to test the performance of complete autonomous driving architectures. In the case of perception tasks the simulator provides data from various sensors such as LiDAR, radar or camera with easily configurable parameters that mimic the real hardware with which you want to work afterwards. In the case of vehicle camera images CARLA provides different formats such as RGB or Depth. The strength of using a simulator as a data source is the possibility of obtaining new images in a practically unlimited way from user defined scenarios with parameters such as weather, time, environment and even the route and speed of the car. There are several tools that help us to obtain the data we need from the simulator. One of them is AIODrive[8] which providesa dataset with multiple environments with labeled objects based on direct information from the simulator. The main problem with this dataset is that it labels objects out of range as it does not take into account the visibility of the objects relative to the car. The other tool examined is our AD PerDevkit dataset [9] which extracts information in a similar way to AIODrive but removes objects that are not of interest for training tasks or are out of range, improving weak points of the former dataset. The data in our tool is in a similar format to KITTI with some extra information such as the frame identifier to facilitate the association with the raw images. The main problem for camera-based detection from simulation is the visual difference between objects in the synthetic world and in the real world, although with the advances in virtual environments in recent years it will become more and more realistic. Another way to obtain synthetic data is presented by Virtual-KITTI [4][5], in which the images provided have been generated by a semi-automatic procedure that aims to create virtual environments called *'proxy'* that are clones of a real-world dataset. In this case, *'proxy'* worlds are generated in Unity [6] from several KITTI scenes and transformations are applied to obtain those same environments with different weather or time of day conditions. Unlike the CARLA approach, this method does not aim to generate a huge amount of data, but instead uses real images to generate its clones to help the generalization process of the characteristics from the original dataset objects.

2.3 Bidimensional state-of-the-art object detectors

Currently there are multipe DL based 2D detectors with state-of-the-art performances. Some of the more popular are multi-stage systems such as fast-RCNN [10] and faster-RCNN [11] or one-stage detectors such as Yolo [12][13][14] and EfficientDet [15]. Table 1 shows differences between the latest and heaviest models in Average Precision (AP) on COCO and their inference time.

Model	COCO AP 0.5	Latency (ms)
Yolov5L	67.3	2.7
Yolov5X	68.9	4.8
EfficientDet D6	71.5	92.8
EfficientDet D7	74.3	153
Faster RCNN	42.7	200

Table 1: Performance of different state-of-art 2D detectors [11][14][15]

Faster RCNN obtains much worse results than the rest of the networks, so it is discarded. EfficientDet obtains slightly better results than Yolov5, however, the difference in execution time is very high. Due to the focus on autonomous driving tasks, low inference times are required, so Yolov5 is chosen as detector.

3 CAMERA DATA

3.1 COCO and Waymo datasets

COCO [16] is one of the largest datasets for general object detection, with 200k images labeled representing 80 object classes. The authors of Yolov5 provide weights with a pre-training of 300 epochs on this dataset which will be used in the training of the paper, particularizing the knowledge only for the class *'Car'*. It should be noted that in order to train with the different datasets, a transformation of the ground truth annotations to the COCO data format has been performed. Waymo [2] is used to evaluate the trained detectors on a real dataset different from the ones used for training. This dataset specializes in autonomous driving tasks, offering data segments recorded by multiple real-world environments from a vehicle with information from multiple sensors compressed into multiple files. Camera information was extracted from the files and transformed into COCO format for evaluation.

3.2 Kitti and Virtual-Kitti datasets

KITTI dataset [1] is one of the most widely used autonomous driving datasets since its release in 2013. Like Waymo, KITTI provides information from different sensors on a vehicle moving through real environments. In this paper KITTI will be used in practically all experiments as the source of real camera data, using the 4781 images it provides, of which 80% have been used for training, 10% for validation and the rest for testing. The KITTI labeling format provides object information such as class, camera visibility, position, rotation, 3D dimensions and the 2D bonding box. As with the other datasets, only the 2D bounding box and class information is used, which is transformed into COCO format.

Virtual-Kitti [4][5] (VKitti) uses a semi-automatic technique to generate virtual environments on Unity's engine based on camera information provided by KITTI. These *'proxy'* environments are modified to reflect different weather conditions or time of day, allowing the trained DL model to better generalize

information about the original dataset, which in this case is KITTI. The first version of the dataset was released in 2015 and provided visual information and the ground truth of the objects, while the second one has been released in 2020 and improves the graphic section as well as providing additional stereo camera and ground truth information such as class segmentation or depth.

3.3 AD PerDevkit dataset

The synthetic data from the CARLA simulator is obtained through our AD PerDevkit dataset [9]. The synthetic data coming from the CARLA simulator are obtained through our tool. Using the ROS bridge the information from the sensors is exported, both the raw data (images, LiDAR pointcloud and Radar pointcloud) and the labeled objects in a csv with the required information such as each object class or 2D bounding box.

The dataset is composed of 11 challenging scenes recorded in 5 towns included in default CARLA under different weather conditions, each containing between 1400 and 3200 frames with full set of annotations. Only about 10000 daytime synthetic images will be used to maintain the similarity with KITTI, which only has daytime images.

(a) (b)

Fig. 2: a) Comparison between original KITTI, VKITTIv1 and VKITTIv2 and KITTI. b) CARLA 2D ground truth example.

4 2D OBJECT DETECTION ARCHITECTURE

The detector that is going to be trained is Yolov5 [14] due to its good results and the flexibility it offers thanks to its multiple models. The base architecture is the one shown in Figure 3, with an image input size of 640x640 and using layers such as CSP Bottleneck with 3 convolutions (C3) or Spatial Pyramid Pooling (SPPF). This architecture scales following the idea presented in EfficientDet [15] to offer various performance and execution times, shown in the table 2.

From the data shown in table 2 it was decided to use the Yolov5L model as it offers similar results to Yolov5X with half the parameters in the network, and the Yolov5S model as it is the lightest model without taking into account Yolov5N which does not offer sufficient results for the task to be performed.

Model	mAP^{val} 0.5	Speed V100 (ms)	Params (M)
Yolov5n	46.0	6.3	1.9
Yolov5s	56.0	6.4	7.2
Yolov5m	63.9	8.2	21.2
Yolov5l	67.2	10.1	46.5
Yolov5x	68.9	12.1	86.7

Table 2: Yolov5 models comparison. Validation on COCO val2017.

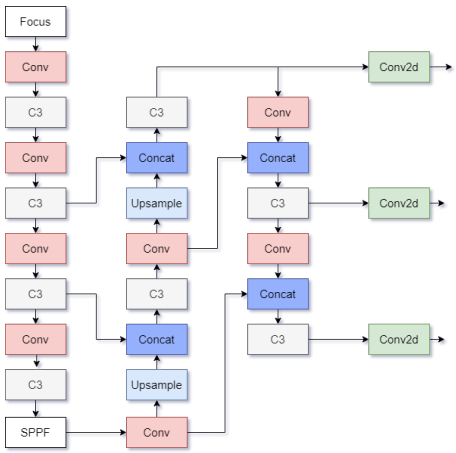

Fig. 3: Yolov5 detector layers

5 EXPERIMENTAL RESULTS

Multiple experiments have been conducted to evaluate the effect of using synthetic data with other techniques such as transfer learning or classical data augmentation to improve the network performance. All training processes are performed on a GTX 1080Ti with 25 epochs and a batch size of 32 for Yolov5L and 64 for Yolov5S models, applying the geometric data augmentation techniques presented in 2.1 and calculating the precision, recall and mAP metrics. The tables presented below highlight in one color the best results for each case, with black being the best using the 100% of Kitti training data, orange using the 50% and blue using the 25%.

5.1 Training without pretrained weights

The first experiment was to train the detectors from scratch, in other words, without any prior knowledge of COCO. The training was performed on separate datasets and various combinations with the multiple synthetic sources.

The table 3 shows the results of training on individual databases without prior knowledge. As expected, the best results are obtained by evaluating on

the same dataset used for training. With no prior knowledge, the network only has the information of the characteristics of one database, which can lead to overfitting on the training data, deteriorating the metrics on the rest of the datasets.

Model	Train	Test	P	R	mAP
Yolov5S	Kitti	Kitti	0.887	0.807	0.908
		Waymo	0.520	0.294	0.327
		Carla	0.298	0.303	0.172
	Carla	Kitti	0.152	0.058	0.043
		Waymo	0.016	0.029	0.013
		Carla	0.640	0.890	0.767
	VKitti	Kitti	0.671	0.380	0.425
		Waymo	0.029	0.053	0.016
		VKitti	0.973	0.921	0.980
Yolov5L	Kitti	Kitti	0.924	0.869	**0.943**
		Waymo	0.618	0.350	0.409
		Carla	0.276	0.198	0.154
	Carla	Kitti	0.078	0.077	0.038
		Waymo	0.023	0.023	0.013
		Carla	0.621	0.893	0.780
	VKitti	Kitti	0.681	0.416	0.462
		Waymo	0.048	0.047	0.017
		VKitti	0.982	0.941	0.984

Table 3: Training results on one database from scratch

Tables 4 and 5 show the results of training by combining a percentage of KITTI data with each of the synthetic datasets.

Using KITTI and CARLA (table 4) the detector can generalize the knowledge better than if it is trained only on one database, obtaining the same or better results on these datasets even than in the case of training and evaluating the network with only one of them (table 3). For the Yolov5L model, KITTI improves from a mAP of 0.943 to 0.949, thus improving performance on the real data base, which is the desired effect.

In the case of using VKitti and KITTI (table 5) both datasets have more similar features since VKitti is built from KITTI, this is reflected in the results of evaluating on KITTI, which improve even more than training with CARLA, reaching a mAP of 0.961 if we use Yolov5L and train with both complete datasets.

It is worth noting that Waymo also benefits slightly from training with mixed datasets, but its results still leave room for improvement.

5.2 Transfer learning experiments with pretrained weights

Once the training tests had been performed without the knowledge from COCO, the training processes were repeated starting from the pre-trained weights.

Model	Train	Test	P	R	mAP
Yolov5S	Kitti 100% + Carla	Kitti	0.884	0.853	0.925
		Waymo	0.533	0.315	0.352
		Carla	0.669	0.879	0.776
	Kitti 50% + Carla	Kitti	0.925	0.873	0.901
		Waymo	0.755	0.519	0.605
		Carla	0.699	0.796	0.811
	Kitti 25% + Carla	Kitti	0.831	0.755	0.841
		Waymo	0.476	0.268	0.292
		Carla	0.675	0.804	0.724
Yolov5L	Kitti 100% + Carla	Kitti	0.933	0.874	**0.949**
		Waymo	0.598	0.350	0.356
		Carla	0.687	0.767	0.772
	Kitti 50% + Carla	Kitti	0.910	0.841	**0.930**
		Waymo	0.521	0.277	0.314
		Carla	0.657	0.775	0.753
	Kitti 25% + Carla	Kitti	0.873	0.794	**0.889**
		Waymo	0.506	0.263	0.303
		Carla	0.652	0.769	0.723

Table 4: Training results on CARLA and KITTI from scratch

Model	Train	Test	P	R	mAP
Yolov5S	Kitti 100% + Vkitti	Kitti	0.913	0.850	0.936
		Waymo	0.578	0.348	0.395
		Vkitti	0.968	0.926	0.981
	Kitti 50% + Vkitti	Kitti	0.889	0.803	0.901
		Waymo	0.615	0.347	0.407
		Vkitti	0.968	0.900	0.970
	Kitti 25% + Vkitti	Kitti	0.864	0.758	0.862
		Waymo	0.580	0.353	0.404
		Vkitti	0.958	0.919	0.979
Yolov5L	Kitti 100% + Vkitti	Kitti	0.943	0.900	**0.961**
		Waymo	0.652	0.374	0.433
		Vkitti	0.987	0.931	0.984
	Kitti 50% + Vkitti	Kitti	0.918	0.878	**0.940**
		Waymo	0.644	0.360	0.420
		Vkitti	0.975	0.939	0.982
	Kitti 25% + Vkitti	Kitti	0.887	0.820	**0.908**
		Waymo	0.637	0.325	0.385
		Vkitti	0.978	0.945	0.986

Table 5: Training results on VKitti and KITTI from scratch

To evaluate the initial level of knowledge provided by the pre-trained weights, an initial evaluation was performed on the real-world datasets used to evaluate the results of the other experiments: KITTI and Waymo.

The table 6 shows that the pre-trained weights lead to better performance results in Waymo compared to those on KITTI. This may be due to the fact that

Model	Test	P	R	mAP
Yolov5S	Kitti	0.753	0.531	0.590
	Waymo	0.815	0.650	0.763
Yolov5L	Kitti	0.743	0.567	0.609
	Waymo	0.778	0.692	0.784

Table 6: Pretrained weights evaluated on KITTI and Waymo

Waymo images are more similar to those used for pre-training in COCO than the ones in KITTI. Despite this, it is observed that COCO provides a high degree of initial knowledge for both datasets, since before any training is performed, mAP values of 0.78 for Waymo and 0.6 for KITTI are already obtained. The results of training on a database starting from the COCO weights are represented in the table 7, in which each model has been trained on the synthetic datasets and on KITTI and various percentages of its data to test the metrics when having limited real-world data. The evaluation is performed on Waymo and KITTI and a synthetic database depending on the training case. Comparing with the results of training without COCO weights (table 3) the improvement is pretty evident. Even training with exactly the same data leads to results improvement, for example Yolov5L with KITTI reaches a mAP of 0.943 which rises to 0.969 with pre-training.

For the case of training KITTI in combination with CARLA (table 8) there is a slight deterioration of the results with respect to training from scratch. The reason for this is that in the case of training from scratch (table 4) there is some overffiting since it is not able to generalize the information correctly, increasing the metrics on the training data but worsening in other datasets (Waymo).

Training with a combination of KITTI and VKitti (table 9) achieves surprising results due to the generalization of information induced by the synthetic dataset. First of all, it manages to improve the results compared to not using transfer learning (table 5) especially in the cases with lower KITTI data proportion. For example, for Yolov5L with only 25 percent of KITTI increases the mAP from 0.908 to 0.931 without affecting Waymo as much (decreases only to 0.679 instead of the 0.385 it had without transfer learning). On the other hand, this combination of data also achieves better results than training with transfer learning but only on the equivalent single database (table 7). The difference in results increases again as the KITTI data decreases, resulting in an 5% higher mAP when mixed with the 25% of KITTI. From this data it can be deduced that adding VKitti to KITTI training will always lead to improved results.

It should be noted that some tests have also been performed in which synthetic data was used for training and then fine tuning was performed with KITTI data, but they do not provide any improvement in results, so they have not been incorporated. For example training on CARLA and performing fine tune on all KITTI training images results on exactly the same mAP as training only on KITI.

Model	Train	Test	P	R	mAP
Yolov5S	Kitti 100%	Kitti	0.903	0.823	0.911
		Waymo	0.763	0.559	0.647
		Carla	0.504	0.541	0.419
	Kitti 50%	Kitti	0.862	0.668	0.807
		Waymo	0.793	0.568	0.674
	Kitti 25%	Kitti	0.783	0.654	0.751
		Waymo	0.789	0.586	0.695
	Carla	Kitti	0.638	0.427	0.459
		Waymo	0.741	0.531	0.617
		Carla	0.691	0.763	0.762
	VKitti	Kitti	0.788	0.586	0.632
		Waymo	0.820	0.594	0.709
		VKitti	0.942	0.913	0.970
Yolov5L	Kitti 100%	Kitti	0.953	0.910	**0.969**
		Waymo	0.776	0.582	0.674
		Carla	0.409	0.559	0.439
	Kitti 50%	Kitti	0.921	0.867	**0.942**
		Waymo	0.786	0.568	0.693
	Kitti 25%	Kitti	0.869	0.781	**0.883**
		Waymo	0.798	0.622	0.716
	Carla	Kitti	0.651	0.466	0.532
		Waymo	0.614	0.509	0.552
		Carla	0.710	0.842	0.793
	VKitti	Kitti	0.797	0.622	0.656
		Waymo	0.838	0.663	0.769
		VKitti	0.966	0.935	0.983

Table 7: Training result on one database with transfer learning

Model	Train	Test	P	R	mAP
Yolov5S	Kitti 100%+ Carla	Kitti	0.886	0.828	0.912
		Waymo	0.767	0.581	0.670
		Carla	0.660	0.734	0.728
	Kitti 50%+ Carla	Kitti	0.760	0.810	0.806
		Waymo	0.784	0.578	0.674
		Carla	0.685	0.749	0.742
	Kitti 25%+ Carla	Kitti	0.767	0.764	0.775
		Waymo	0.794	0.570	0.672
		Carla	0.717	0.730	0.742
Yolov5L	Kitti 100%+ Carla	Kitti	0.950	0.911	**0.969**
		Waymo	0.800	0.579	0.676
		Carla	0.745	0.760	0.829
	Kitti 50%+ Carla	Kitti	0.802	0.871	**0.889**
		Waymo	0.792	0.600	0.694
		Carla	0.610	0.901	0.800
	Kitti 25%+ Carla	Kitti	0.817	0.808	**0.846**
		Waymo	0.799	0.570	0.673
		Carla	0.702	0.731	0.776

Table 8: Training result on CARLA and KITTI with transfer learning

Model	Train	Test	P	R	mAP
Yolov5S	Kitti 100%+ Vkitti	Kitti	0.896	0.813	0.904
		Waymo	0.783	0.568	0.664
		Vkitti	0.945	0.910	0.969
	Kitti 50%+ Vkitti	Kitti	0.864	0.772	0.868
		Waymo	0.782	0.567	0.663
		Vkitti	0.955	0.906	0.970
	Kitti 25%+ Vkitti	Kitti	0.815	0.732	0.816
		Waymo	0.804	0.570	0.675
		Vkitti	0.945	0.909	0.970
Yolov5L	Kitti 100%+ Vkitti	Kitti	0.949	0.915	**0.967**
		Waymo	0.770	0.597	0.683
		Vkitti	0.973	0.929	0.983
	Kitti 50%+ Vkitti	Kitti	0.938	0.882	**0.955**
		Waymo	0.779	0.597	0.682
		Vkitti	0.971	0.925	0.982
	Kitti 25%+ Vkitti	Kitti	0.895	0.878	**0.931**
		Waymo	0.781	0.586	0.679
		Vkitti	0.979	0.930	0.984

Table 9: Training result on VKitti and KITTI with transfer learning

6 DISCUSSION AND CONCLUSIONS

The experiments conducted have made it possible to quantify the impact of various techniques on the results of a training process. First, multiple trainings from scratch have been performed with different databases (table 3) and mixing real and synthetic data (tables 4 and 5). Subsequently, these trainings have been repeated but applying the transfer learning technique, that is, starting from pre-trained weights in COCO (tables 7, 8 and 9). It has been observed how applying a transfer learning technique helps to avoid overfitting problems that appear in training from scratch, obtaining even higher performances. This is why in DL network training processes it is recommended to use this technique whenever possible due to the benefits it provides. On the other hand, the results have also allowed us to verify which approach to obtain synthetic data between CARLA and VKitti provides the most improvement. Training with only one of these two datasets does not provide any improvement over real data (KITTI or Waymo), so it is advisable to mix them with real-world data to improve the results. In the case of performing this combination, it is obtained that in the case of training from scratch, both data sources provide better results than using only the real database. When using pre-trained weights, CARLA does not achieve better results than training with a single real database, but VKitti does achieve an improvement, especially as the percentage of KITTI is reduced. The use of transfer learning for DL training is recommended as long as the original database has similarities to those to be used for the particular case. For example, COCO [16] can be used for general camera detection. The proxy worlds approach used by VKitti [4] provides a more noticeable improvement than using data from

a simulator such as CARLA. However, obtaining data by the former method is more expensive and more limited than using CARLA, which is able of generating unlimited labeled data in a simple way.

References

1. A. Geiger, P. Lenz, and R. Urtasun, "Are we sready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
2. P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
3. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
4. A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4340–4349.
5. Y. Cabon, N. Murray, and M. Humenberger, "Virtual kitti 2," 2020.
6. "Unity." [Online]. Available: https://unity.com/
7. C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, Jul 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0
8. X. Weng, Y. Man, D. Cheng, J. Park, M. O'toole, and K. Kitani, "All-in-one drive: A large-scale comprehensive perception dataset with high-density long-range point clouds," 12 2020.
9. J. de la Peña, L. M. Bergasa, M. Antunes, F. Arango, C. Gómez-Huélamo, and E. López-Guillén, "Ad perdevkit: An autonomous driving perception development kit using CARLA simulator and ROS - Accepted ITSC 2022."
10. R. Girshick, "Fast r-cnn," 2015.
11. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.
12. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.
13. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.
14. G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, NanoCode012, TaoXie, Y. Kwon, K. Michael, L. Changyu, J. Fang, A. V, Laughing, tkianai, yxNONG, P. Skalski, A. Hogan, J. Nadar, imyhxy, L. Mammana, AlexWang1900, C. Fati, D. Montes, J. Hajek, L. Diaconu, M. T. Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106. (2021) ultralytics/yolov5: v6.0 - yolov5 models, roboflow integration, tensorflow export, opencv dnn support. [Online]. Available: https://doi.org/10.5281/zenodo.5563715
15. M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.
16. T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.