Autonomous vehicle perception using a driving simulator: the capabilities of artificial LiDAR data

João Espadinha
joao.m.espadinha@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

January 2021

Abstract

In autonomous driving, object detection and semantic segmentation are critical tasks since it is required to not only detect and track objects on the road but also to have a deep understanding of the surroundings. Since annotating 3D point clouds is a high resource and time-consuming task, artificial data is seen as an upcoming resource that could solve these issues. In this work we propose a study of the capabilities and potential of artificial LiDAR data for object detection and semantic segmentation, using deep learning methods. For 3D vehicle perception challenges, training deep neural networks in big, sparse and unordered point clouds has shown to be a hard task, in part due to the lack of publicly available data. We used simulator CARLA to generate our data, and studied in depth ways of mitigating the differences between artificial and real world data. We modelled both the noise from real world point clouds, and the missed reflections (that we called point dropout) that occur in real world data collection. We also explored potential benefits of using pre-trained models on artificial data when fine tuning with all or a fraction of the available real world data. We found clear benefits when using artificial data to pre-train a network, which allowed us to use a reduced amount of real world data, and boosting the performance of our models.

Keywords: Object detection, semantic segmentation, artificial LiDAR data, deep learning.

1. Introduction

The overall goal of autonomous driving is allowing the vehicle to make the most informed decision it can, based on the circumstances it faces. So having a deep understanding of its surroundings and the behaviour of the dynamic agents is of the utmost importance. Object detection and semantic segmentation are some of the key tasks used to help tackle those problem, and the ones we will address in this work.

Significant progress has been made on 2D object detection and semantic segmentation but, for 3D vehicle perception challenges, training Deep Neural Networks (DNNs) with a more complex representation, like point clouds, has shown to be a more challenging task. As a result, there is a gap between performances for 2D and 3D perception, which could be narrowed with additional publicly available data. This lack of data can be explained by the intense process of 3D point cloud annotation, which requires time and resources. Besides that, not only the available data usually has big class imbalances, but it is also prone to human error when labeling.

As a result, artificial data can be an important asset to tackle the previous problems. Simulators usually provide highly precise annotations and, once we set up the data capturing process, annotated data is generated with less time, cost and human resources. In addition, during the data creation process we can adjust class statistics to our needs and create scenes which are hard to replicate in real world.

The main objective of this work is to study the capabilities and potential of artificial LiDAR data for object detection and semantic segmentation. We want to assess the utility of artificial data using deep learning methods, and apply the gathered knowledge to real world scenarios. The main challenges that we have to overcome in order to achieve our goal are: (1) reducing the dissimilarity between artificial and real data, and (2) discovering how we can benefit from using a pre-trained model with artificial data in comparison with training the DNN from scratch. To solve challenge (1), we created a noise model and a point dropout model to simulate real world effects, and applied them on our artificial data. To solve challenge (2), first, we evaluated two different transfer learning strategies, by breaking down the DNNs into two different sections: encoding and decoding. We decided whether we should freeze the weights from the encoding section or not when fine tuning our DNNs and, as a result, we assessed the similarity of encoded features with artificial and real data. Then we performed a set of experiments to find out how the use of a pre-trained model on artificial data could affect performance and amount of real data required to fine tune our DNNs.

2. Background

LiDAR (LIght Detection And Ranging) is an active remote sensing system which provides precise depth measurements of the surroundings, creating a 3D representation of the environment. LiDAR data points have a total of four elements: the 3D coordinates (x,y,z) and an intensity value, which represents the energy that each pulse returns with.

Currently, DNNs are the state-of-the-art methods on 3D autonomous vehicle perception challenges and have proven to be a powerfull tool when quality data is provided. The performance of this methods highly rely on the data that is used and, this way, publicly available datasets started to emerge, in order to contribute to the development of algorithms that work on the real world. Some of the available datasets with annotated LiDAR data are H3D [14], nuScenes [3] and the one used in this work, KITTI [6]. As mentioned before, the creation of these types of datasets requires a lot of time and resources, reasons why recent studies have tried to create and employ artificial data for autonomous driving purposes. This data has been generated, in some cases, in video games like *Grand Theft Auto V* and, in other cases, using simulators design for this purpose, like SYNTHIA and CARLA [4].

2.1. Object detection

Object detection has the purpose of identifying and detecting objects of a certain class. This identification can be performed, for example, in images or, in our case, in a 3D representation like a point cloud.

In early works, it was established that Convolutional Neural Networks (CNNs) were the preferred architectures when performing object detection on images. Later, works as [8] suggested a two-stage approach. In a first stage, a Region Proposal Network (RPN) is used to extract several regions of interest from a scene, which are regions where the objects may be. Then, in a second stage, these proposals are verified and classified by a second network, like a CNN. Although this approach requires more inference time, it has shown to outperform the one stage approaches, which do not use a RPN. For object detection in point clouds, works like [12] follow the same line of thought and apply directly 3D convolutions. However, this approach requires a lot of computational resources and time. To solve this issue, newer methods transform the point cloud into 2D representations, to then apply 2D convolutions. In some cases the point cloud is projected into the ground plane, in others it is projected into the image plane and, for the network used in this work, PointPillars [11], a pseudo-image is created.

2.2. Semantic Segmentation

Semantic segmentation is the process of linking each element of a certain representation to a class label. If we perform this task on an image, the elements are pixels, if we perform on a point cloud, each element is a three dimensional point.

In images, a commonly used architecture is the Fully Convolutional Network (FCN). This approach follows the architecture of a CNN, with the difference that the fully connected layer is substituted by convolutional layers. This way, an output map with smaller size than the image (due to the previous pooling layers) is obtained, which is than converted to the original image size using backwards convolution. Following studies, such as [1] build upon FCN, proposing an encoder decoder architecture. The encoder block uses a CNN while the decoder restores the lower dimension features to the original image resolution. In the end, the feature representation is mapped to the desired number of semantic classes. As before, CNN based approaches for semantic segmentation in point clouds use a 2D or a 3D approach. Projection based methods are some of the leading methods due to their smaller computational needs and good results. In this work, we use RangeNet++ [13], which uses a spherical projection approach to create the 2D representation, which is fed into an encoder-decoder network.

2.3. Related work

Artificial data for autonomous driving is becoming, recently, a subject of study. Even though there are some works that explore the creation and use of artificial images [15] using deep learning methods, both for object detection and semantic segmentation, there is a lack of work regarding artificial 3D scene generation. Some of these few works explore its capabilities for object detection and instance segmentation, and compare the performance obtained when using KITTI [6].

At [5], a slightly different approach than ours is proposed. Instead of creating fully artificially generated scenes, scans from real world scenarios are generated and augmented with synthetic obstacles. A data-driven approach was developed which extracts, from real traffic scenes, the distribution of obstacles' positions, orientations and shapes. This learned distribution is then used to augment the real world scenarios with the computer generated objects. Results showed that retaining the complexity and realism of real world scenarios brings added value in comparison with fully simulated data. However, this approach is a middle-ground between training DNNs with only real or simulated data. Although we can generate different scenes from the same backgrounds, if we want a large sample size of scenarios, human annotation is still needed.

PreSIL [9] is the most identical work to ours. In their approach, fully artificial data is created using *Grand Theft Auto V* (GTA V), and used to detect pedestrians. GTA V provides information regarding object position, dimensions, heading, type and vehicle model, and the point cloud is created using depth images, as we do in our work. There are, however, some limitations in this approach, like the fact that GTA V does not provide reflectivity information or the fact that full 360º scans are not provided. Afterwords, it was concluded that artificial data can boost performance when using a pre-trained model on this data. This is especially true when detecting objects with low class representability in real world datasets, like pedestrians. It can be explained by the fact that datasets created using simulators can be more easily adapted to the task's needs, for example including more pedestrians.

3. Methodology

The work developed consists in a study of the capabilities and potential of artificial LiDAR data for the tasks of object detection and semantic segmentation in autonomous driving scenarios, using deep learning methods.

We can divide our work pipeline in four major sections: data creation, data pre-processing, learning process and the evaluation phase, as it is shown in Figure 1. The main contributions of this work are in the pre-processing and evaluation phase.

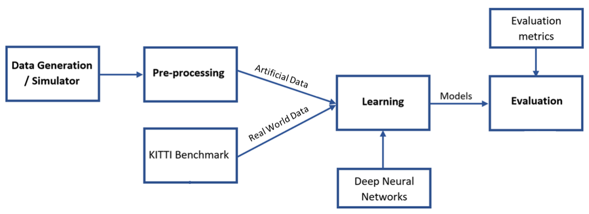

Figure 1: Pipeline of the work that was developed.

3.1. Data Generation

During the data creation process, we didn't have direct access to a LiDAR sensor in our simulator and, because of that, we needed to create one. A typical LiDAR outputs the values of 3D positions (x, y, z) of each observed point, as well as a value of the energy that each point returned to the LiDAR (E_{ret}). To create our point cloud, we used other resources available in the simulator, in this case, a depth camera, that occupied the same position as the LiDAR in our sensor setup.

To create our point could, first, we defined all the unit vectors with the directions of the simulated shots from the LiDAR. These vectors were created based on the values of the horizontal and vertical field of view and resolutions, from the specifications of the LiDAR that was being simulated (Velodyne HDL-64E). With respect to the vertical and horizontal resolutions, after comparing the projections of real point clouds from Velodyne HDL-64E and simulated point clouds with the original resolution values from that scanner, a mismatch was observed. The real point clouds were visibly sparser and, as a consequence, we changed the original hoerizontal and vertical resolution values to 0.125º and 0.485º.

After the unit vectors were created, we projected them in the image, using the depth camera parameters, and obtained the pixels coordinates. With these values, we can calculate the depth for each point that is located within the direction of the original vector. When we had a point with image coordinates located between pixels, we obtained the depth value using bilinear interpolation of the depth values from the four pixels around it. Finally, we multiplied each unit vector by the corresponding depth of the generated point, in order to get its 3D position.

Regarding the returned energy value (E_{ret}), it was calculated by the following equations:

$$E_{ret} = E_{emit} \times R_{rel} \times R_{ia} \times R_{atm} \,, \tag{1a}$$

$$R_{ia} = (1 - \cos\alpha)^{0.5} \,, \tag{1b}$$

$$R_{atm} = exp(-\sigma_{air} \times D) \,. \tag{1c}$$

where E_{emit} is the energy of the original laser pulse, R_{rel} is the reflectivity of the surface material, R_{ia} denotes the reflection rate with respect to the laser incident angle and R_{atm} is the air attenuation rate, since each laser beam is absorbed and reflected when travelling in the air. R_{rel} is obtained from a prediction of the material of a certain point, based on the class it belongs to. In R_{atm}, reprsented by (1c), σ_{air} is a constant, equal to 0.004, and D is the distance from the LiDAR center to the target.

Regarding the ground truth, we followed the predefined formats used in the 3D object detection dataset from KITTI Benchmark [7] and the semantic segmentation dataset from Semantic KITTI [2].

With respect to object detection, all the ground truth was written in two *.txt* files. The first one had the information of the intrinsic and extrinsic matrices. The first one maps points from camera coordinate system to image coordinate system and the second one maps the points from LiDAR to camera coordinate systems. The second file had information about all the objects in the scene. For each object, we stored information regarding its: (i) class, (ii) truncation level, (iii) occlusion level, (iv) angle alpha[1] (observation angle of the object), (v) location, (vi) dimensions, (vii) 2D bounding box, (viii) rotation_y[1] (angle of rotation around the vertical axis). For this task we generated objects from the following classes: “car”, “pedestrian”, “cyclist”, “truck” and “motorcycle”, from which “car” was the class to be detected.

Regarding semantic segmentation ground truth, a *.label* file was created for each point cloud, which contained the semantic class that each point belonged to. Each semantic class is represented by an integer and the simulator provides 25 different ones. However, there were two main problems that we had to tackle before being able to use the data. First, from the 28 semantic classes from KITTI and the 25 from CARLA, we observed that some classes from different instances were assigned with the same integer value. To solve this issue, we mapped all CARLA classes to the integer value from the closest KITTI class. Secondly, it was observed that, not only some classes from CARLA were more broad, but also that some classes from KITTI were not represented in CARLA. This way, we started by isolating the nineteen semantic classes used in RangeNet++ [13]. After that, the previously detected specific classes were mapped to a broader class, and the ones not represented in CARLA were excluded. In the end we ended up with the following fourteen classes: “car” ,“bicycle”, “motorcycle”, “truck”, “person”,“road”, “sidewalk”, “other-ground”, “building”, “fence”, “vegetation”, “terrain”, “pole” and “traffic-sign””.

3.1.1 Scene Generation and Selection

Since we want to study the capabilities of LiDAR artificial data for real world applications, there is a gap between domains (real and artificial) that could undermine the performance of our models. One way to attenuate this gap was explained in [16], where it was shown that if the variability in simulation is high enough, models trained in simulation should generalize to the real world with no additional training.

This variability can be accessed in terms variety of backgrounds, agents (vehicles’ models, pedestrians), as well as in terms of agents’ orientations, relative position to our car and occlusion and truncation levels. With respect to the background, we used all the eight different ones available in our simulator, which included scenarios from rural areas as well as cities and highways. Regarding the foreground, we had access to fifteen different pedestrians and twenty-seven different vehicles, which included nineteen different cars, three different bicycles, three different motorcycles and two different trucks.

Our data collection process consisted in driving around a car with our sensor suite, in each town/area, while capturing consecutive frames. The movement of our car and the other dynamic agents was simulated. During the creation of our scenes we tried to balance two major elements: fidelity in relation to the real world and diversity of agents and its attributes (orientation, occlusion levels, etc). The fact that we could not control movement of our agents meant that we frequently captured frames with a small amount of objects. To circumvent this problem, we spawned an additional amount of static objects (cars, motorcycles, etc) in our background, ending up with scenes as the one in Figure 2. These objects were spawned

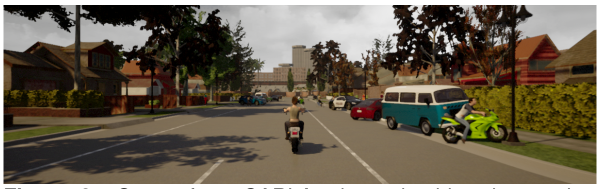
Figure 2: Scene from CARLA where, besides the moving agents, we can see several static agents that provide added variety to our datasets.

with random orientation, to further contribute to the variability of our dataset. Additionally, we spawned new objects every 100 frames and deleted the existing ones every 500 frames. This was necessary not only to diversify our scenes, but also to ensure that, in some cases, our car would not get blocked by other objects on the road for too long.

The data that we got from the simulator was organized in folders with 4000 sequential frames and each folder had frames from a single scenario. This way, after creating several data folders with all the different scenarios, a selection of the ones with the highest quality data was required. To ensure a high quantity and variability of agents from all the different classes, we chose the data folders with: (i) the highest amount and variety of agents in the scenes and (ii) the least amount of repeated frames (repeated frames appeared when the car got blocked by other objects). After this first stage, we ended up with a set of folders with sequential frames. Since we wanted to keep the same structure between the artificial and KITTI [2, 6] datasets, after this stage, we obtained our semantic segmentation data.

Unlike semantic segmentation data that is orga-

[1]For more detailed explanation, see https://towardsdatascience.comorientation-estimation-in-monocular-3d-object-detection-f850ace91411

nized in sequential frames, the KITTI object detection dataset is composed of random frames from different scenarios, which means that an additional selection process was needed. Since, in this work, we are mostly interested in detecting cars, this additional stage consisted in a selection of each frame based on the amount of cars in it. As shown in [6], to each object, a difficulty mode ("Easy", "Moderate" or "Hard") can be applied depending on its level of occlusion, truncation and bounding box height. In PointPillars, if the attributes for the difficulty mode "Hard" were not met, the ground truth of that object would not be used for training. This way, we estimated the amount of cars in a frame after applying the same filtering technique, which allowed us to better control the true statistics of this class.

3.2. Pre-processing

Our main goal in this step was to study real world effects, in order to model them and apply them on the generated artificial data. The real world effects we studied were the noise in the point clouds and the missing points due to missed reflections, which we call point dropout.

3.2.1 Noise Model

With our noise model, we aim at characterizing the statistics of the noise vector at each point of the point cloud, considering its distance to the origin of the LiDAR reference frame and the angle between its direction and the forward axis (x axis) of the same reference frame. Our model considers that the noise samples follow a normal distribution with a mean value of 0 and a standard deviation that was obtained from KITTI data (real world data).

To calculate our standard deviation, we started by collecting points from different flat surfaces, obtained from the KITTI dataset point clouds. For each one we used Least Squares to fit a plane to the corresponding points, in order to simulate the noiseless surface. After that, for each point, we computed the distance to the origin of the LiDAR reference frame, the angle with the forward axis and the perpendicular distance to the estimated plane. Finally, we made a quadratic regression of the perpendicular distances as a function of distance to origin and angle with x axis. The obtained function is, by approximation, our standard deviation, and its expression is given by:

$$\begin{aligned}\sigma = &-0.050196\,\alpha^2 - 4.582916\times 10^{-5}\,d^2 \\ &- 0.001986\,d\,\alpha + 0.097530\,\alpha \\ &+ 0.003070\,d - 0.031166\end{aligned} \tag{2}$$

where σ is our standard deviation, α is the angle between the direction of a point and the forward axis and d is the distance to the origin of the LiDAR reference frame.

In order to apply noise to each point, based on its parameters, we extracted the value of the standard deviation from (4). Then, we sampled a value from the normal distribution, and used its absolute value as the magnitude of the noise vector. Finally, we randomly selected the direction of the noise vector and added its coordinates to the selected point.

3.2.2 Point Dropout Model

Regarding our point dropout model, we had the objective of creating a model that would give us a decision of removal or non-removal for each point of the point cloud, considering the same attributes as before. Our model consists in a Bernoulli distribution that gives us the previously mentioned decision, from a value of the probability of removing a certain point, that was obtained from KITTI data.

To obtain the mentioned probability, we started by selecting several small surfaces from KITTI point clouds, from which we computed the theoretical amount of points the surface should have and the amount of points from the actual point cloud. Then, for each surface, we computed the distance between its mean point and the origin of the LiDAR reference frame, the angle between the mean point and the forward axis and the probability of a point being removed, given by:

$$p_r = \frac{np_t - np_a}{np_t} \tag{3}$$

where p_r is our probability value, np_t is the theoretical number of points and np_a is the actual number of points in a certain surface. Finally, we applied a quadratic regression to the probability values that we computed before. In the end, we obtained a function that can estimate the values of the average probability of removal for each combination of distance to the origin and angle with the x axis. From this function we obtain, by approximation, the probability of a certain point being removed, and it is represented by:

$$\begin{aligned}p = &\,0.186136\,\alpha^2 + 6.984331\times 10^{-5}\,d^2 + 1.648670 \\ &\times 10^{-4}\,d\,\alpha - 0.164589\,\alpha + 0.004652\,d - 0.173883\end{aligned} \tag{4}$$

In order to simulate the missed reflections from our simulated LiDAR shots, we started by computing our probability of removal from (4) and the parameters of each point. Finally, we used a Bernoulli distribution with the previous probability value to obtain our final decision of removal or non removal.

3.3. Learning

For our experiments we used datasets from two different domains: real and artificial. Our artificial data was created following the methodology described in this section and was only used for training. On the other hand, real world data was used for both training and validation purposes. We decided to use only real world data for validation to prevent the model from overfitting to the artificial data. For object detection, we obtained our data directly from the 3D object detection dataset, provided by KITTI Benchmark, which is divided into 7481 training instances and 7518 test instances. Since the test set did not provide ground truth annotations, we only used the 7481 training instances. We divided the annotated subset into 3712 frames used only for training and 3769 frames used for validation. For semantic segmentation, we used data provided by Semantic KITTI which consists in 22 different folders with sequential frames. Only the first eleven (0 to 10) were annotated, with 28 different semantic classes, from which we used the sequence 8, with 4071 frames, as validation data and the remaining ones, with 19 130 frames, as training data.

During a training process, the main objective is to achieve a good generalization ability. In a situation where we want to learn a certain task in a domain (source domain), and apply it to a different one (target domain), just validating in a set of data of the target domain may not be enough. This way, we explored an additional strategy to help us tackle difference between artificial and real domains: transfer learning. We wanted to use this approach to assess whether transferring the knowledge from a model trained with artificial data to a new one trained with real world data could give us an added value in our studies. The training strategy that we choose depends mainly on the similarity between the source and the target tasks and datasets. For our case, the source and target tasks are the same and the datasets only differ in domain, reasons why we decided to not initialize layers with random weights. Since we did not know for sure how similar the features from both domains are, we decided to investigate two different approaches. In the first one, we followed the standard procedure and transferred all pre-trained weights, from which we froze the encoding layers and trained the remaining ones. In the second approach, we also transferred all pre-trained weights but fine tuned the whole network. Performing experiments with both procedures allowed us to discover which approach we should and to assess the similarity between the features from real and artificial domains.

4. Experimental Setup

In this section, we will describe all the programs used and implementations that were applied in order to perform our experiments and obtain our numerical results.

4.1. Implementation Details

In order to create our artificial data, we used simulator CARLA[4] (Car Learning to Act). This is an open-source simulator[2] created to support development, training and validation of autonomous driving systems. As a result, it provides flexibility in terms of sensor suites specifications and gives some control over static and dynamic actors, environmental conditions and more.

The the sensor setup used, consists in a RGB camera and a LiDAR scanner identical to the ones used in the KITTI setup [7]. We follow the specifications of the original LiDAR model (Velodyne HDL-64E), with the exception of the vertical and horizontal resolutions, which we changed, as we explained in Subsection 3.1. Besides that, the reference frames of our sensors are defined with the same convention as in KITTI [7] and our LiDAR and camera occupy the same positions as, respectively, the LiDAR scanner and camera 0, in KITTI setup.

For the learning process, we used the implementation of PointPillars [11] from traveller59[3] (for object detection) and of RangeNet++ [13] from Photogrammetry & Robotics Bonn[4] (for semantic segmentation). In both networks, we obtained the most accurate model based on the performance it achieved on the real domain validation set. In the case of PointPillars, we measured this performance with the Average Precision (AP) for the overlap of 3D bounding boxes (BB) metric, while for semantic segmentation we used the values from the mean Intersection over Union (IoU). Furthermore, we performed an initial learning rate tuning step, following the idea introduced in [10]. As a result, for object detection we obtained a configuration of 0.0001 and 0.5 for, respectively, the initial learning rate and decay values while, for semantic segmentation, we obtained a configuration of 0.001 and 0.99 for the same hyperparameters.

4.2. Evaluation

In previous works [5, 9] using the 3D object detection dataset from KITTI, it was observed that, due to the lack of annotated data, the experimental results were obtained using the validation set. Since we faced the same issue we decided to follow this practice. For semantic segmentation, even though DNNs can be evaluated in a server using the KITTI test set, the high volume of experiments did not allow us to use that tool. Thus, we followed the same procedure as for the previous task, we evaluated and compared our results on the validation set.

[2]CARLA simulator github repository https://github.com/carla-simulator/carla

[3]Provided in https://github.com/traveller59/second.pytorch

[4]Provided in https://github.com/PRBonn/lidar-bonnetal/tree/master/train/tasks/semantic

Dataset	AP of 3D detections			AP of detections in BEV			AP of 2D detections			AOS		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Raw data	13.20	11.14	10.58	45.78	35.90	31.07	56.84	40.91	37.40	55.26	39.39	35.78
Point Drop.	14.26	11.49	11.25	40.84	33.57	30.98	58.98	45.17	**43.87**	54.85	41.46	**39.75**
Noise	**14.78**	**12.01**	**11.47**	**51.95**	**40.44**	**34.71**	**60.48**	**45.47**	42.56	**56.24**	**41.99**	39.24
Both	13.84	11.29	11.00	41.95	31.97	28.87	59.88	44.65	41.78	58.34	42.78	**39.75**

Table 1: Results after evaluating the models trained with datasets with different pre-processing implementations (Object Detection).

Dataset	car	bicycle	motorcycle	truck	person	road	sidewalk	other-ground	building	fence	vegetation	terrain	pole	traffic-sign	mIoU
Raw data	23.3	0.3	2.5	**0.5**	9.3	54.6	28	**0.7**	16.3	**2.2**	0.6	5.1	**4.7**	1.2	10.7
Point Drop	**41.3**	**0.7**	3.5	0.4	8.8	**58.1**	27.7	0.6	14.1	1.7	1.1	8.5	3.5	2.2	**12.3**
Noise	27.4	**0.7**	**3.9**	0.3	8.3	55.8	27.4	0.3	**19.7**	1.6	0.4	6.6	4.0	**2.4**	11.3
Both	37.6	0.6	3.4	0.4	**10.3**	55.4	**29.7**	0.4	15.4	2.1	**1.4**	**9.3**	4.6	1.8	12.314

Table 2: Results after evaluating the models trained with datasets with different pre-processing implementations (Semantic Segmentation).

4.3. Metrics

We used the official KITTI evaluation metrics from [2, 6] for, respectively, object detection and semantic segmentation, since they are widely used in these types of works.

Regarding object detection, we required an Intersection over Union (IoU) of 70% between the predicted and ground truth BB, for that prediction to be considered a True Positive (TP). Besides that, if multiple detections of a same object were counted, the one with higher IoU was considered a TP and axis aligned Non Maximum Suppression (NMS) was applied, with an overlap threshold of 0.5 IoU. In the end, the evaluation metrics used for this task are: (i) AP for the overlap of 3D BB, (ii) AP for the overlap of 2D BB, (iii) AP for the overlap of BB in Birds Eye View (BEV) and (iv) Average Orientation Similarity (AOS). It is also important to mention that all this metrics are stratified into three difficulty modes ("Easy", "Moderate" and "Hard"). For each difficulty mode, only ground truth with an equal or lower difficulty index is considered when training.

For the case of semantic segmentation, we used the Intersection over Union metric, also known as Jaccard Index, for each class, as well as the final mean Intersection over Union.

5. Results & discussion

In this chapter, we will describe all the experimental studies that were performed and the conclusions that we took from the obtained results. With these experiments, we have the goal of evaluating, empirically, the effect of changes in certain properties of our training sets and the effect of different learning techniques.

5.1. Study A

In this study, we want to assess the impact that our pre-processing models have in the performance of the network. For that reason, we created four datasets, all with artificial data and with either: (i) no pre-processing model applied (raw data), (ii) only noise, (iii) only point dropout or (iv) both noise and dropout. For object detection, all datasets have 4000 frames, while for semantic segmentation they have 20 000 frames. For the first task, the datasets were created using the additional frame selection process described in Section 3.1.1, and have a similar distribution of cars per frame as in the 3D object detection dataset from [6], as shown in figure 3:

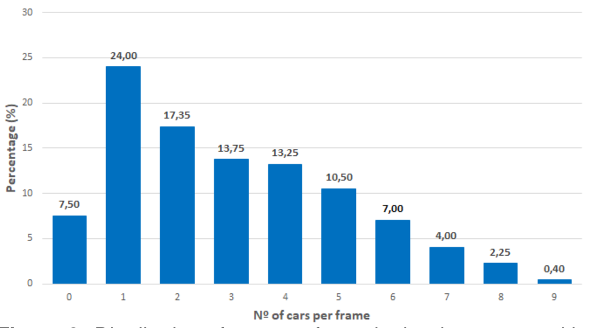

Figure 3: Distribution of cars per frame in the datasets used in Study A.

As we can see in Tables 1 and 2, we get an overall better performance, for both tasks, when we apply our pre-processing models on the point cloud. This way, we can verify that our pre-processing step is helping to close the gap between real and artificial domains, as it was intended to. For object detection, our best model is the one with noisy point clouds, while for semantic segmentation we obtain a better performance when we apply both noise and dropout. One possible reason for the small contribution of point dropout in object detection is the difference between the DNNs' architectures, more specifically in the pillar feature net section from PointPillars [11]. In this block of the network the point cloud is divided into a set of pillars, and each one can have a maximum of 100

Datas. Size	AP of 3D detections			AP of detections in BEV			AP of 2D detections			AOS		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
4 000	**14.78**	**12.01**	**11.47**	**51.95**	**40.44**	**34.71**	**60.48**	**45.47**	**42.56**	56.24	41.99	39.24
10 000	13.21	11.34	10.89	41.36	31.77	28.05	57.48	43.74	41.49	**56.34**	**42.63**	**40.22**
20 000	12.87	11.11	10.81	41.45	34.14	30.56	53.31	42.08	39.87	51.90	40.60	38.27
40 000	12.48	10.90	10.79	33.62	28.35	25.95	51.74	39.99	39.55	51.31	39.36	38.64

Table 3: Results after evaluating the models trained with datasets with different sizes (Object Detection).

Datas. Size	car	bicycle	motorcycle	truck	person	road	sidewalk	other-ground	building	fence	vegetation	terrain	pole	traffic-sign	mIoU
20 000	37.6	0.6	3.4	**0.4**	10.3	55.4	29.7	0.4	15.4	2.1	1.4	**9.3**	4.6	**1.8**	12.314
34 000	**41.0**	0.8	3.4	0.3	**11.3**	**63.0**	31.7	0.7	13.1	1.9	1.1	8.3	**5.1**	1.7	13.1
50 000	38.1	**1.5**	**4.3**	**0.4**	10.6	62.0	**32.3**	**0.9**	**16.2**	**2.2**	**2.1**	8.9	4.8	**1.8**	**13.3**

Table 4: Results after evaluating the models trained with datasets with different sizes (Semantic Segmentation).

points. When this amount of points is exceeded, the ones in excess are removed randomly. This way, not only points can be removed even without using our point dropout model, but there could also occur cases where, even after using our model, the amount of points could still exceed the maximum allowed. This way, we believe that this first block of PointPillars could influence the effectiveness of our point dropout model.

5.2. Study B

For Study B, we have the objective of inferring the impact that the increase in dataset size could have in the performance of our networks. Thus we created three additional training sets with artificial data for object detection, with 10 000, 20 000 and 40 000 frames, and two additional ones for semantic segmentation, with 34 000 and 50 000 frames. Additionally, we applied to our data the pre-processing models that performed the best in our previous study. Thus our object detection datasets had noisy point clouds, while the semantic segmentation ones had noise and point dopout applied. In this case, the object detection datasets were not fully created using the additional frame selection process. Besides the initial 4000 frames from the previous study, that were included in all these datasets, the one with 10 000 frames included 2000 more selected frames, and the remaining ones included 2000 more selected frames than the later one. The remaining data samples were taken directly from the simulator, which means that our datasets would have a higher amount of frames with a smaller amount of cars (between 0 and 4). With the objective balancing the distribution of cars per frame, all of the new selected ones had an amount of cars ranging from 5 to 9.

From Table 3 and Table 4, we can observe pretty distinct results. For object, we obtained better performance when training with only 4000 frames, while for semantic segmentation we observe an increase in performance as the dataset got bigger. A more obvious reason for the observed behaviour in the first task is the data selection process, since the smaller dataset had fully selected data based on the amount of cars per frame. This extra step allowed us to not only choose higher quality data but also to have a bigger control on the dataset statistics. Besides that, the number of agent models in CARLA is limited and, as a consequence, the variability in the object detection annotations becomes very limited, since weight, height and length are fixed values to each car model. In the semantic segmentation case,the RangeNet++ pre-processing step, in which all points are projected into a spherical image, provides added variability, since the projections of the point-wise annotations will differ significantly more as we move.

5.3. Study C

For this study, we have the goal of verifying whether we are able to get similar performance when just fine tuning the non-encoding blocks of the network in comparison with fine tuning the whole network. This way, we can also get additional information regarding the similarity between the features that are encoded with real data and artificial data. For object detection we used the model trained with 4000 frames of noisy data to initialize our network, while for semantic segmentation we used the one trained with 20 000 frames of data with noise and dropout. We used real world data from the training sets specified in Section 3.3 to fine tune either the whole networks or just, for PointPillars, the decoder and single shot detector blocks and, for semantic segmentation, the decoder and segmentation head.

From Tables 5 and 6, we can draw the same conclusion: we obtain better results when fine tuning the whole network. The fact that freezing the encoding blocks decreased performance allows us to conclude that there are differences in the encoded features when using artificial or real data. Thus we can also infer that the used simulator does not provide high enough realism for us to reuse the same

Model	AP of 3D detections			AP of detections in BEV			AP of 2D detections			AOS		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
FT 2000	**84.24**	**69.61**	**67.90**	**90.22**	**85.05**	**80.42**	**90.48**	**87.08**	**84.35**	**90.27**	**86.36**	**83.26**
Dec 2000	70.02	54.94	51.71	89.00	77.99	75.31	88.70	77.36	73.98	87.57	75.13	71.15
FT 3712	**85.34**	**75.58**	**69.63**	**90.14**	**86.51**	**81.23**	**90.50**	**88.18**	**86.53**	**90.34**	**87.57**	**85.54**
Dec 3712	71.62	55.85	53.49	89.13	78.15	75.61	88.79	77.73	74.76	87.75	75.71	72.18

Table 5: Results after fine tuning the whole network (FT) or just the decoding/detection section (Dec), using real world datasets (Object Detection).

Model	car	bicycle	motorcycle	truck	person	road	sidewalk	other-ground	building	fence	vegetation	terrain	pole	traffic-sign	mIoU
FT 10 117	**89.3**	**12.0**	**17.9**	**23.2**	**21.1**	**90.6**	**73.6**	**22.5**	**74.6**	**27.9**	**80.4**	**66.7**	**29.8**	**24.9**	**46.7**
Dec 10 117	84.4	2.7	7.1	9.3	16.8	85.6	63.7	13.8	62.3	12.4	66.1	62.4	20.3	15.5	37.3
FT 19 130	**90.2**	**15.4**	**29.3**	**25.8**	**32.8**	**92.2**	**77.4**	**34.6**	**79.1**	**43.2**	**81.8**	**69.9**	**33.0**	**25.9**	**52.22**
Dec 19 130	84.6	3.7	12.5	7.0	22.5	87.4	67.0	20.5	66.3	14.4	70.8	66.4	18.6	16.1	39.8

Table 6: Results after fine tuning the whole network (FT) or just the decoding/segmentation section (Dec), using real world datasets (Semantic Segmentation).

encoded features when training with artificial and real world data.

5.4. Study D

Finally, in this last study, we tried to find the peak performance of our networks as well as their capacity to maintain a stable performance as we decrease the size of the real world training set. These comparisons were drawn between the cases where we fine tuned the whole network, initialized with the weights from a pre-trained model on artificial data, and when we trained the network from scratch. The pre-trained models used are the same as the ones in study C, and we decreased the size of the real world dataset.

From Tables 7 and 8, we can observe two different behaviours. In the first one, regarding object detection, using a pre-trained model with artificial data allowed us to better stabilize the performance of our networks when we decreased the size of the training set. On the other hand, for semantic segmentation, we can observe a general improvement in performance when using a pre-trained model. In the end, we obtained clear contributions from artificial data, both for object detection and semantic segmentation.

6. Conclusions

With this work we intended to study the capabilities and potential of artificial LiDAR data for object detection and semantic segmentation on real world scenarios. To do that we setup a data generation process on CARLA [1] that allowed us to create data with the highest quantity and variability of agents and scenarios possible. We also created a noise and point dropout models to help reducing the gap between real and artificial domains. In this work, we performed experiments to not only assess the highest performance we can achieve when training with artificial data, but also to study the potential of using pre-trained models on artificial data to fine tune our DNNs. In the end, we observed that simulators still cannot achieve a high enough degree of realism to replicate the complexity of the real world. However, we showed how we still can benefit from using artificial data when applying the right training strategies. We can use a pre-train model on artificial data to not only boost performance, but also to allow us to use a smaller amount of real world data when training.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[3] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[4] 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*, pages 1–16, 2017.

[5] J. Fang, D. Zhou, F. Yan, Z. Tongtong, F. Zhang, Y. Ma, L. Wang, and R. Yang. Augmented lidar simulator for autonomous driv-

Model	AP of 3D detections			AP of detections in BEV			AP of 2D detections			AOS		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
FT 3712	**85.34**	**75.58**	**69.63**	90.14	**86.51**	**81.23**	**90.50**	**88.18**	**86.53**	**90.34**	**87.57**	**85.54**
FT 2000	84.24	69.61	67.90	**90.22**	85.05	80.42	90.48	87.08	84.35	90.27	86.36	83.26
FT 1000	80.22	69.06	66.87	90.14	84.00	80.11	90.21	85.72	80.73	90.09	84.94	79.68
FT 500	78.64	67.39	65.27	90.17	84.77	79.99	90.20	80.96	79.73	90.01	80.15	78.49
R 3712	**85.58**	**75.27**	**69.66**	90.03	**86.99**	**84.17**	**90.71**	**88.66**	**87.11**	**90.61**	**88.13**	**86.08**
R 2000	81.04	70.55	67.25	**90.21**	84.01	80.65	90.51	86.91	82.42	90.25	86.16	81.36
R 1000	79.34	67.57	65.43	90.01	80.43	79.74	90.26	81.78	80.17	90.05	80.95	78.82
R 500	73.12	59.33	56.42	89.06	79.34	77.76	88.66	78.03	76.67	87.58	75.96	73.94

Table 7: Results after fine tuning the whole network (FT) or training it from scratch (R), using increasingly smaller real world datasets (Object Detection).

Model	car	bicycle	motorcycle	truck	person	road	sidewalk	other-ground	building	fence	vegetation	terrain	pole	traffic-sign	mIoU
FT 19 130	**90.2**	**15.4**	**29.3**	25.8	**32.8**	**92.2**	**77.4**	**34.6**	**79.1**	**43.2**	**81.8**	**69.9**	**33.0**	**25.9**	**52.22**
FT 10 117	89.3	12.0	17.9	23.2	21.1	90.6	73.6	22.5	74.6	27.9	80.4	66.7	29.8	24.9	46.7
FT 5 305	86.3	5.3	6.0	**26.8**	16.9	87.3	67.4	8.6	69.7	19.6	74.2	61.4	26.0	22.8	41.3
FT 2 173	75.0	1.3	1.2	5.4	7.1	82.3	52.4	2.2	51.5	7.7	63.9	49.7	19.0	17.6	31.2
R 19 130	**89.3**	**16.0**	**17.9**	**32.9**	**26.6**	**92.1**	**76.9**	**34.2**	**76.6**	**39.5**	**80.6**	**68.8**	**28.3**	**24.4**	**50.03**
R 10 117	87.1	17.5	10.8	29.8	16.0	90.4	73.0	21.9	72.4	24.2	78.4	65.9	23.9	23.3	45.3
R 5 305	84.9	9.6	2.7	21.1	9.8	87.2	67.4	9.7	67.6	17.4	73.1	60.2	22.5	23.0	39.7
R 2 173	73.7	0.7	0.1	11.9	1.9	78.9	49.9	2.1	52.1	9.7	65.0	47.7	14.2	15.6	30.2

Table 8: Results after fine tuning the whole network (FT) or training it from scratch (R), using increasingly smaller real world datasets (Semantic Segmentation).

ing. *IEEE Robotics and Automation Letters*, PP:1–1, 01 2020.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[7] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[8] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

[9] B. Hurl, K. Czarnecki, and S. Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529, 2019.

[10] S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. *Width of Minima Reached by Stochastic Gradient Descent is Influenced by Learning Rate to Batch Size Ratio: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4–7, 2018, Proceedings, Part III*, pages 392–402. 10 2018.

[11] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[12] B. Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518, 2017.

[13] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[14] A. Patil, S. Malla, H. Gang, and Y.-T. Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. IEEE International Conference on Robotics and Automation (ICRA), 2019.

[15] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2232–2241, 2017.

[16] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017.