Cross-sensor deep domain adaptation for LiDAR detection and segmentation

Rist, Christoph; Enzweiler, Markus; Gavrila, Dariu

DOI
10.1109/IVS.2019.8814047

Publication date
2019

Document Version
Accepted author manuscript

Published in
Proceedings IEEE Symposium Intelligent Vehicles (IV 2019)

Cross-Sensor Deep Domain Adaptation for LiDAR Detection and Segmentation

Christoph B. Rist[a,b,1], Markus Enzweiler[a,2] and Dariu M. Gavrila[b,3]

***Abstract*— A considerable amount of annotated training data is necessary to achieve state-of-the-art performance in perception tasks using point clouds. Unlike RGB-images, LiDAR point clouds captured with different sensors or varied mounting positions exhibit a significant shift in their input data distribution. This can impede transfer of trained feature extractors between datasets as it degrades performance vastly.**

We analyze the transferability of point cloud features between two different LiDAR sensor set-ups (32 and 64 vertical scanning planes with different geometry). We propose a supervised training methodology to learn transferable features in a pre-training step on LiDAR datasets that are heterogeneous in their data and label domains. In extensive experiments on object detection and semantic segmentation in a multi-task setup we analyze the performance of our network architecture under the impact of a change in the input data domain. We show that our pre-training approach effectively increases performance for both target tasks at once without having an actual multi-task dataset available for pre-training.

I. Introduction

Environment perception using LiDAR sensors is a key enabler for intelligent vehicles and autonomous driving. Object detection and semantic segmentation on point clouds are key problems within this domain. Both tasks enrich a simple scanning of the surrounding with high-level information. See Figure 1 for an example scan. Being able to solve these two related tasks by using the same learned features saves computation time for usage in a real-time scenario.

Machine learning methods for perception typically require a sizable amount of training data to achieve state-of-the-art performance. In a supervised set-up the typical assumption is that the test data is drawn from the same distribution as the training data. A large shift in the input data domain degrades performance at test time and therefore jeopardizes the possibility to make use of annotations on already existing datasets which have a different distribution in their data samples. The acquisition of suitable training data becomes complex if a particular set of labels – 3D objects and semantic segmentation in this work – is required for supervised learning in a particular multi-task setup.

The data distribution in a automotive LiDAR dataset does not only depend on the character of the recorded scenes but also heavily on the sensor type itself and its mounting position. Thus the desired data domain for a specific vehicle setup lacks labeled data as it is challenging to transfer learned features from a different dataset to the current setup.

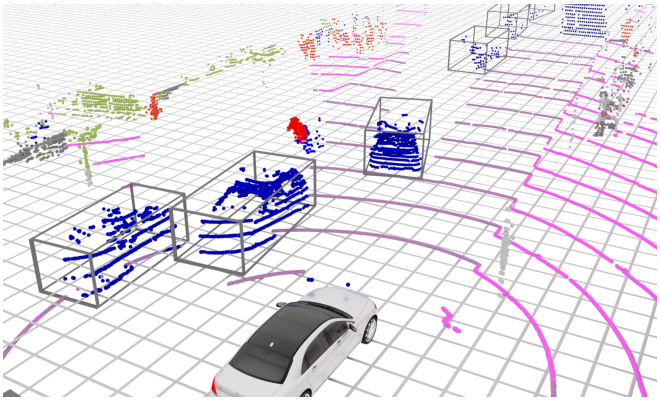


Fig. 1: **Result of our method:** Detecting objects (bounding boxes) and annotating measurements semantically (colors indicate different object classes) in full 3D space are key problems for autonomous driving and can be tackled on LiDAR point clouds.

In this paper, we are concerned with a feature extractor for LiDAR scans for the tasks of 3D object detection and semantic segmentation. In an experimental study we evaluate how pre-training on two different support datasets impacts the final performance on our multi-task problem.

In the visual domain it is common to reduce the issue of litte training data by using pre-trained feature extractors and fine-tuning them on the desired input domain and target task. This approach is intuitive as there is a canonical and obvious input representation for images in form of dense data structures for convolutional networks. This has led to the well-established strategy of exploiting large public datasets, e.g. the ImageNet challenge [1], to help solve a broad spectrum of perception problems in a different domain.

The same methodology applied to point cloud processing yields some drawbacks: diverse approaches to process sparse point cloud data with CNNs and the rather large spread of LiDAR data distributions has not yet lead to a set of distinguished feature extractors to be re-used for a range of new tasks. The dependency of a feature extractor on a certain underlying neural net architecture where there is no definitive standard for a point cloud input representation hinders establishing an equivalent to GoogleNet or ResNet in the LiDAR domain.

As a first step towards that goal, we propose a convolutional architecture based on a voxelized input representation

a) Daimler AG, Stuttgart, Germany
b) Intelligent Vehicles & Cognitive Robotics Group, Technical University Delft, The Netherlands
[1]christoph_bernd.rist@daimler.com
[2]markus.enzweiler@daimler.com
[3]d.m.gavrila@tudelft.nl

and show that it is a suitable choice for our problem definition. This work presents an architecture and training methodology for object detection and semantic segmentation on point clouds and evaluates the influence of an alternated training on different tasks and different sensors. Using the definitions from [2] and [3] we realize a heterogeneous domain adaptation in a supervised manner, i.e. labels for the target domain are available.

II. Previous Work

a) Deep learning based LiDAR environment perception: Since the advent of deep neural networks learned features on LiDAR-captured point clouds have become the state of the art to solve a variety of perception tasks relevant to automated driving: object detection (3D), semantic segmentation, odometry estimation, scene flow, and object motion.

A variety of data representations for ordered and unordered point clouds has emerged recently to be used as input for their respective deep learning architectures: Sensor view images [4], bird view images [5], voxel grids [6], [7], mixed 2.5D bird view approaches [8], [9], [10], permutohedral lattice convolutions [11], global features on an unordered point cloud [12], [13], nearest neighbor based convolutions [14], representing point clouds as graphs and performing convolutions on edges [15], and graph neural networks [16] (see Table I). For object detection a split-head architecture approach for region proposals and spatial regression has proven effective and is popular [4], [5], [6], [7], [17]. The problem to train the two output heads is sometimes also referred to as multi-task problem [5].

b) Multi-task learning: Many vision-based tasks are related and learning multiple tasks at once can help to increase the performance of the individual tasks given the right training parameters [18]. Solving multiple tasks in one inference step based on the same image features greatly reduces the required computation time compared to the usage of separate per-task networks [19]. Training multiple tasks at once on a multi-task dataset helps to successfully train a particular task for which only a comparatively small amount of annotated data is available [20].

Multi-task perception problems on point clouds have not been evaluated extensively, yet. [7] shows that the problems of scene flow, object detection, and object motion can be solved with a single set of point cloud features from consecutive point clouds using a voxel grid. Part segmentation and classification can be solved at once on artificially generated point clouds based on 3D object models [12].

c) Domain adaptation using deep networks: Domain adaptation settings are categorized by the availability of labels in the target domain, by domain divergence and task divergence. In the heterogeneous domain adaptation setting, feature spaces between source and target domain are nonequivalent and usually differ in their dimensionality, e.g. due to changed image resolution or changed number of returned LiDAR points from a different sensor. A homogeneous domain adaptation exists when only the data distributions between source and target domain differ [2].

Method	Representation	3D Perception Task
VoxelNet [6]	VFE-Encoder, + Voxel Grid	Object Detection
PointFlowNet [7]	VFE-Encoder, + Voxel Grid	Object Det./Motion + Scene Flow
PointPillars [10]	PointNet features, + Top view	Object Detection
3D FCN [17]	Hand-crafted features + Voxel Grid	Object Detection
Veh. Det. 3D [4]	Sensor-view image	Object Detection
LiLaNet [21]	Sensor-view image	Semantic segm.
PointNet [12] PointNet++ [13]	Unordered set + global description vector	Part segmentation, classification
MV3D [5]	Sensor-view + Top view	Object Detection
Complex-YOLO [9] Fast and Furious [8]	Top view (2.5D)	Object Detection /Tracking

TABLE I: Overview of LiDAR input representations of deep learning approaches for automotive-related perception tasks

Deep network architectures can exercise a domain adaptation effect via the backpropagation mechanism [2]. This is exploited when fine-tuning feature extractors for a target domain. Since the adoption of deep networks in a broad spectrum of visual perception tasks on images it is common to use weights for an architecture trained on different datasets as an initialization [22], [23], e.g. for a ResNet feature extractor for semantic segmentation [24]. [25] point out that pre-training on a high diversity dataset facilitates increased generalization capabilities of a model after fine-tuning.

Batch normalization [26] primarily enables the training of ever deeper neural networks by normalizing the distribution of activations between layers while training. In generative models it has become the default way to prevent mode collapse [27]. While improving performance and convergence it has also been shown to implicitly function as architecture-based domain adaptation technique [28].

There are numerous studies on general domain adaptation and image perception in particular [3]. Deep domain adaptation techniques have so far mainly been reviewed on visual tasks [2]. In the visual domain fine-tuning works well in practice as convolutional neural networks learn similar hierarchical visual features on a wide spectrum of datasets regardless of the actual task. These features are representative for different tasks and different datasets [22].

For LiDAR data and point cloud processing in general, this research area is largely unexplored. We consider this paper as a first step in this direction. Our main contributions are:

- We present an end-to-end trainable model for the joint prediction of 3D object detection and semantic segmentation of unordered LiDAR point clouds.
- To the best of our knowledge we are first to evaluate domain adaptation and feature re-usability on point clouds.
- We propose a training scheme for heterogeneous datasets that can effectively learn features that transfer well to another data domain.

III. Proposed Approach

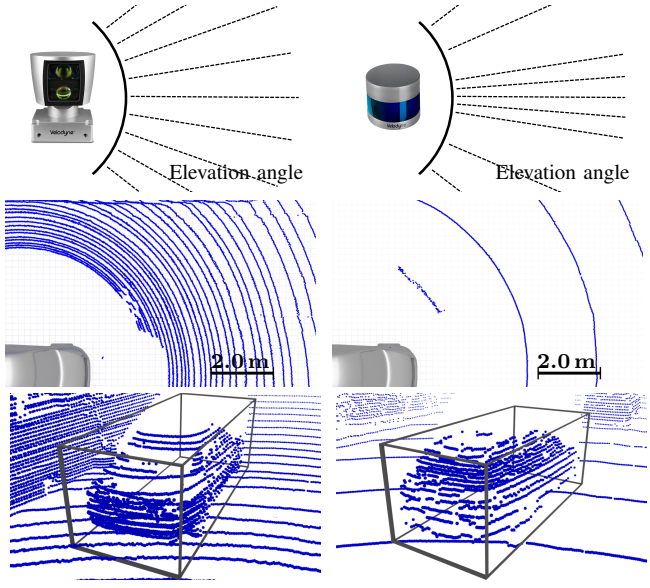
We design a CNN-based architecture with multiple task-specific output heads to solve object detection and semantic segmentation of point clouds with a single feature extractor. In our approach, neither task does depend on the time dimension and can be computed from a single LiDAR scan.

A. Deep domain adaptation on LiDAR scans

Rotating LiDAR scanners generate point clouds in a way which is fundamentally different from RGB-image sensors. The environment is sampled in a continuous motion around the sensor producing a high-frequency stream of individual columns of 3D points. It is common to collect the points of a full 360 degree rotation into a single data sample, referred to as *frame* or *scan*. This approach effectively neglects the complex time dimension and allows to treat the generated point cloud as self-contained samples from a feature space $\{x_1, x_2, \ldots, x_n\} \in \mathcal{X}$ instead of individually measured points.

Analogous to images of different resolutions the dimensionality of the point cloud input space $\mathcal{X}$ differs between sensors with a different number of layers. When considering the accumulation of points over time into a point cloud the horizontal resolution affects the dimensionality of the data space as well. Such nonequivalent feature spaces $\mathcal{X}^{\mathrm{A}} \neq \mathcal{X}^{\mathrm{B}}$ between different sensors A and B and their configuration constitute a heterogeneous domain adaptation setting. If the mounting position of a sensor is changed the distribution of data samples changes while the input dimensionality stays fixed.

The qualitative differences between the two sensors Velodyne-HDL64 and Velodyne-VLP32 [29] used in our datasets are displayed in Figure 2. The horizontal resolution of both sensors is almost identical while the vertical resolution differs substantially at the top and bottom of the vertical field of view.



(a) Velodyne-HDL64 (b) Velodyne-VLP32

Fig. 2: **Diverse LiDAR scan characteristics:** The physical properties of LiDAR scanners tie the characteristics of resulting scans to mounting position and the scanner type itself. The doubled number of layers of the Velodyne-HDL64 model and their uniform distribution over the elevation angle compared to the VLP32 sensor (top) create a different, but distinct characteristic of the recorded point clouds. In an outdoor vehicle-mounted setting this shift in the input data domain becomes most visible at the ground surface (middle row) while close objects at sensor height exhibit less change in their sampled density (bottom).

B. CNN Architecture

Voxel Feature Extractor: Convolutional networks assume neighbourhood relations. Therefore reproducing the metric space around the sensor in a deep learning architecture that convolves over these exact dimensions brings a translation invariance that we consider useful for 3D perception tasks. Hence our point cloud feature extractor is based on a metric voxel grid in 3D space. One essential challenge to overcome in voxelized architectures is the non-uniform sampling density in sparse point clouds when sampling the environment with a single sensor from a single position. Therefore we use the feature encoder layers originally proposed by [6] and employed by [7] as they have demonstrated state-of-the-art results on large real-world outdoor-scene point clouds. This feature encoder creates a per-voxel feature resulting in a dense data representation of fixed size.

An important benefit of this feature extractor is that it prevents the problems associated with a change in the input data dimensionality in heterogeneous domain adaptation. By using the metric space for a spatial arrangement of features we gain a feature space that is linked to the sensor in use only by the per-voxel feature extraction. When able to cope with different sampling densities because of different distances, this also holds the possibility to learn features independent of the vertical layer distribution or mounting position.

These observations let us presume that this feature extractor is a particularly feasible architecture choice out of the variety of LiDAR architectures to learn features that are as independent of the sensor as possible.

Per-point input features: [6] proposes to supply two sets of coordinates for each point into the VFE-Layers: The absolute sensor coordinates x_s, y_s, z_s and a set of coordinates x_r, y_r, z_r relative to the mean position of all points within a given voxel. Additionally, the measured reflectivity is provided as an input feature. We only supply the set of per-voxel relative coordinates x_r, y_r, z_r and the reflectivity value. The absence of absolute point coordinates in a sensor coordinate system forces the subsequent feature convolutions to only learn features relying on the relative positions of points and voxels among each other. We think of this measure as a way to prevent the network from basing decisions about objects or semantic categories on an absolute position dependent on the mounting position of the LiDAR sensor.

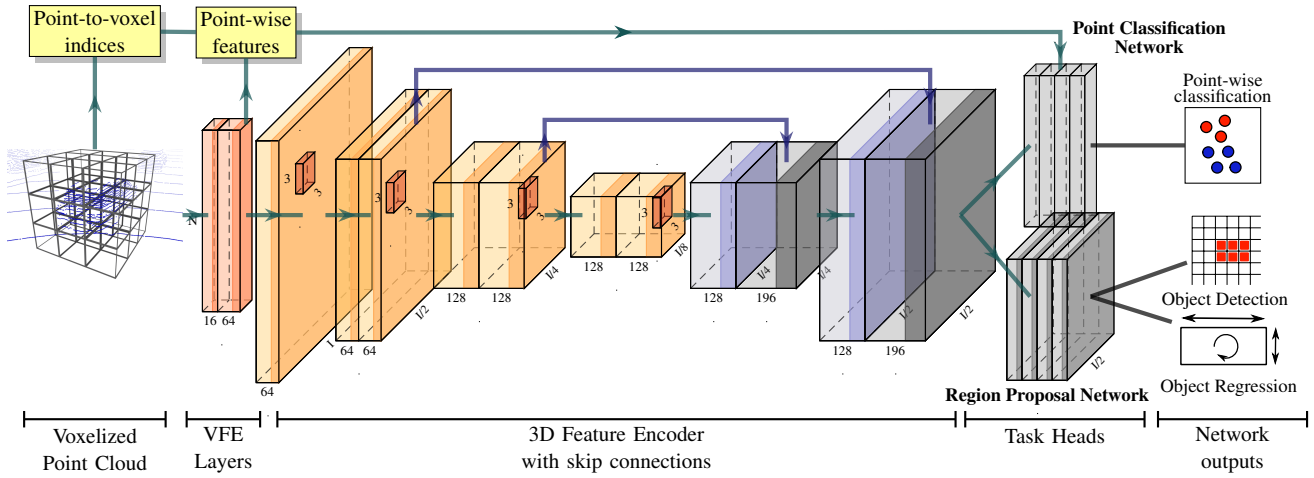


Fig. 3: **LiDAR-Multi-Task architecture**: Our CNN-architecture makes use of VFE-Layers [6] on the voxelized point cloud to generate a fixed size representation. A convolutional feature encoder as network trunk follows, branching into slim output heads for the desired tasks. The encoder samples the 3D grid resolution down and up again and connects identical grid resolutions with skip connections. The grid feature height (z dimension) is omitted for visual clarity.

For example, it cannot base a classification for the class *Road* on a absolute z_s coordinate of a point at about $-2.0\,\mathrm{m}$ relative to the sensor.

Core Feature Encoder: The core feature layers of our CNN are made up of 3D convolution layers with skip connections. We make use of batch normalization after every convolution and ReLU activations. An overview of the complete architecture is given in Figure 3.

Point-wise classification output and region proposal network: The output of the core feature encoder consists of features that are spatially arranged in 3D at half the resolution of the input voxel grid. To classify the individual LiDAR measurements we concatenate each of the input points with the feature vector of the voxel it resides in. Analogous to the VFE-Layers we apply a series of fully-connected layers on each point individually to obtain the semantic classification. We retain the regression targets of [6] but shrink the region proposal network to a depth of four convolutional layers. Thus keeping the depth of the individual network branches preferably small to minimize the amount of per-task learnable network capacity.

Loss functions: The network features three task-specific output heads: Object region proposals (cross entropy loss), object regression (L2 loss) and point-wise semantic classification (cross entropy loss). We find that employing uncertainty weighting [18] to weight all three output heads for the final loss function gives slightly better results than a constant factor weighting. Note that our loss function operates on the individual point level for semantic segmentation as well as on the bird view voxel grid for object classification and regression. Following [6], we define two detection losses: The region proposal loss

$$\mathcal{L}_{\mathrm{RPN}} = \frac{1}{N_+}\sum_{N_+}\mathcal{L}_+ + \frac{1}{N_-}\sum_{N_-}\mathcal{L}_- \qquad (1)$$

is a class-weighted sigmoid cross-entropy loss $\mathcal{L}_{+,-}$ over the bird view RPN matrix with N_+ positive and N_- negative examples. The object regression loss $\mathcal{L}_{Reg} = L1_{smooth}\,(r, r^*)$ is the smooth L1 distance between the predicted regression vectors r^* and the ground truth target vectors r for every positive object location. The segmentation loss $\mathcal{L}_{Segm}$ is the mean cross-entropy classification loss over all annotated points within the voxel grid extent. Integrating the learnable uncertainty weights $\sigma_{\mathrm{RPN,Reg,Segm}}$ results in the overall training loss

$$\mathcal{L} = \frac{1}{\sigma^2_{\mathrm{RPN}}}\mathcal{L}_{\mathrm{RPN}} + \frac{1}{\sigma^2_{\mathrm{Reg}}}\mathcal{L}_{\mathrm{Reg}} + \frac{1}{\sigma^2_{\mathrm{Segm}}}\mathcal{L}_{\mathrm{Segm}} + \log \sigma^2_{\mathrm{RPN}}\sigma^2_{\mathrm{Reg}}\sigma^2_{\mathrm{Segm}}. \qquad (2)$$

C. Training methodology

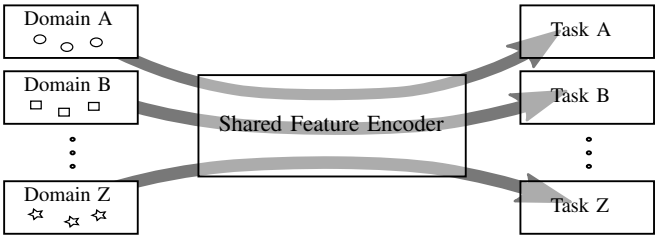


Fig. 4: **Training methodology**: An alternating training scheme between multiple tasks A, B, ... each on its respective input data with backpropagation through a shared feature encoder. This way there is no need for a single dataset to include the annotations for all output tasks.

In the case of having only a limited amount of training data, it can be useful to train a deep neural network on an existing dataset of a related domain first. An increase in the available amount of data can help learning useful features

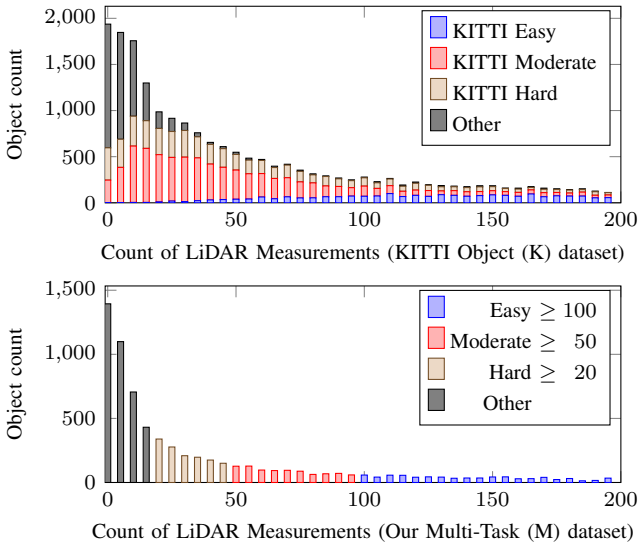


Fig. 5: **Number of LiDAR Measurements on class 'car'** We estimate that the difficulty to detect an object in a LiDAR scan has an inverse correlation to the number of LiDAR measurements on the respective object.

and therefore improve the final performance after fine-tuning for the target data. However, it is challenging to discover a large-scale dataset which has a superset of the desired tasks annotated to be used for multi-task training. Therefore we use an alternating training scheme to train our architecture on two or more heterogeneous datasets each with its own input data and label domain (Figure 4). We switch the input data and respective output task after each training batch so that every optimization step alternates between all datasets. The idea is to leverage the backpropagation process to combine features from the individual training sets. This allows us to include both tasks from the supporting datasets in our pre-training. Our network architecture facilitates this scheme by employing fairly small task-specific output heads.

D. Training and architecture parameters

We use the Adam optimizer [30] and implement an axis-aligned 2D non-maximum-suppression on the bird view image. When training the voxel grid extent is $80.0\,\mathrm{m} \times 80.0\,\mathrm{m} \times 10.0\,\mathrm{m}$ in front of the sensor with a voxel size of $0.2\,\mathrm{m} \times 0.2\,\mathrm{m} \times 0.4\,\mathrm{m}$ for the x, y, and z axis respectively. Our input representation choice makes it easy to apply spatial augmentations to the input data. We apply a random uniform rotation drawn from the interval $[-45.0°, 45.0°]$ around the upward pointing z-axis of the sensor coordinate system, a small uniform translation from the interval $[-0.1\,\mathrm{m}, 0.1\,\mathrm{m}]$ and Gaussian noise on the point coordinates with a mean $\mu = 0.0\,\mathrm{m}$ and variance $\sigma = 0.02\,\mathrm{m}^2$. The per-task uncertainty weights $\sigma_{\mathrm{RPN,Reg,Segm}}$ are initialized with 1.0.

IV. Experimental Evaluation

A. Datasets

To train our multi-task network we have manually annotated scans from Velodyne-VLP32 recordings with ground truth labels for 3D object detection and for semantic segmentation. We will refer to this dataset as *LiDAR Multitask* (M) dataset. Our dataset features diverse traffic scenes including city and country roads (see Figure 6 for example frames). Both annotation types are only available within a FOV of approx. 90° of a front-facing RGB-camera because these images are necessary to guide the annotators.

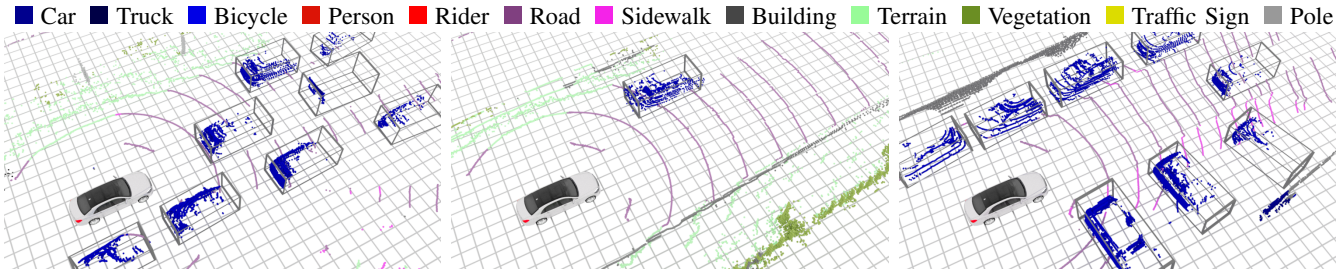
We aim to improve the performance of our multi-task setup by using two already available datasets for pre-training. As supporting datasets we use the KITTI Object Dataset [31] and the semantic segmentation dataset of [21]. The latter is auto-generated by transferring an RGB-image segmentation from a trained RGB-CNN onto LiDAR points using the sensors' calibration. This approach allows for effortless generation of arbitrary amounts of training data for LiDAR semantic segmentation, which is of high quality but still not as accurate as manually annotated data [21].

The sizes and training splits of the datasets are listed in Table II. Each dataset holds 360° LiDAR scans recorded from a vehicle on public roads. The mounting position of the LiDAR scanner in datasets S and M varies between the roof and the hood of the recording vehicle. Our specialized multi-task dataset is smaller than the two supporting datasets. These datasets have the benefit of either being publicly available (KITTI) or consisting of labels without the burden of manual annotation effort. Adversely, every supporting dataset only features one of our two desired output tasks.

Domain adaptation is particularly challenging because of the different type of LiDAR sensors used, i.e. Velodyne HDL64 (KITTI) and Velodyne VLP32 (datasets M and S). The main difference being the doubled number of layers in the HDL64 model. Additionally, the HDL64 features a uniform distribution of its layers over the elevation angle, whereas the VLP32 concentrates its layers around the horizon when mounted with the rotating axis pointing upwards. The much larger sampling density of the HDL64 is clearly visible in Figure 2. The different distribution of layers results in different sampling characteristics. Notable in our setup is the high sampling density of the HDL64 directly in front of

Dataset [LiDAR Sensor]	Annotated Tasks	#Annotated Frames		
		Train.	Val.	Test
KITTI Object (**K**) [HDL-64]	Object Detection	3712	3769	7518
LiDAR Semantic (**S**) [VLP-32]	Semantic Segmentation	340 000	12 261	22 983
LiDAR Multitask (**M**) [VLP-32]	Detection + Segmentation	1047	226	441

TABLE II: **LiDAR Datasets**: To train for the multi-task problem, we manually annotated a small multi-task dataset (M) with both 3D objects and point-wise LiDAR semantic segmentation. To boost performance, we make use of two supporting datasets for pre-training: A semantic segmentation dataset (S) [21] auto-generated from RGB-segmentation and the KITTI Object Detection dataset (K) [31] annotated with 3D objects.



(a) Waiting at traffic light (b) On highway (c) Parked cars in narrow residential street

Fig. 6: **Qualitative results of our fine-tuned architecture** in various traffic situations. Notice that all predictions outside of the RGB camera's field of view of about 90 degrees towards the ego-vehicle's front are generated by our model without having seen training data in these regions.

the ego-vehicle compared to recordings of the VLP32.

B. Evaluation Metrics

We use the well-established average precision (AP) metric to rate the performance of the 3D detection task for the object class 'car' in our experiments. A detection is accepted as a true positive if it has an intersection-over-union with the ground truth bounding box greater than 70% in bird view perspective. Naturally, some cars in a dataset are easier to detect than others. For that purpose the KITTI Benchmark categorizes each box into one of three different difficulty classes (easy, moderate, hard) based on the level of truncation, occlusion and height in the image. We refer to [31] for details. This ranking is biased towards RGB-image based object detectors: Objects that are very close to the ego vehicle tend to get classified as difficult because of a large truncation in the camera image.

For LiDAR-based detection we propose to instead use the number of LiDAR measurements within the object's bounding box as a difficulty measure. This way the impact of distance and occlusions is covered in a straightforward manner. We analyzed the distribution of KITTI's difficulty classes in objects with a certain number of measurements, see Figure 5. We chose to use 20-49 (hard), 50-99 (moderate), and 100+ (easy) LiDAR points per object as underlying criteria to establish difficulty classes. In doing so, the overall assignment of objects to difficulty classes has a similar distribution as for the KITTI dataset. Note that AP scores are still difficult to compare between datasets because the difficulty to detect an object might depend on various factors.

Performance of the semantic task is rated using the standard mean intersection-over-union (mIoU) metric over the set of individual LiDAR points. We train the network on 12 different semantic classes based on Cityscapes [32] which were slightly tailored to the properties of a LiDAR sensor as defined in [21].

C. Experimental Results

We first train our CNN-architecture on the support datasets K and S individually and combined using our proposed alternating training scheme. This gives us three sets of weights (K, S, and K+S) to initialize our architecture for the fine-tuning step on the target LiDAR multi-task dataset.

Method	Training Dataset	Detection AP (KITTI difficulties)			Segmentation mIoU
		Hard	Moderate	Easy	
MultiTask	K	**0.692**	**0.711**	**0.815**	-
MultiTask	S	-	-	-	**0.579**
MultiTask	K+S	0.678	0.708	0.791	0.569

TABLE III: **Performances in multi-task and single-task setup:** Multi-task training (last row) reaches comparable performance levels as single-task training (first two rows), using the identical underlying network architecture.

Upon convergence on the multi-task dataset we select the training state with the highest performance on the validation set and report the performance metrics on the test split of our datasets S and M. AP scores on the KITTI Dataset are reported on the validation split using the evaluation code of the original authors as supplied by the KITTI development kit. We use the KITTI object dataset split proposed by [33]. We will refer to the AP score of the *hard* category when comparing detection results.

The results of our study on pre-training are listed in Table IV. The multi-task (M) section includes test performances for both tasks and all three pre-training settings compared to a training from scratch. Every setting is tested with and without the influence of data augmentation. The increased number of training setup and consistent results give us additional confidence in our training and evaluation methodology.

Training our multi-task architecture for both tasks without initialization only slightly decreases the performance of the individual tasks as listed in Table III. This gives us confidence that we found a deep network architecture that works well for the multi-task problem. Providing 3D objects and semantics in a single run makes the processing time overhead negligible since every new task only requires to add an additional network head, c.f. Figure 3.

Using network weights of a pre-training as initialization improves the final performance on the respective task, as shown in Table IV. This is true for pre-training on a different sensor (compare AP 0.603 with 0.724 in detection accuracy when pre-training on KITTI (K) and for training on a larger dataset with a shifted label domain (compare mIoU 0.670 with 0.694 in segmentation when pre-training on Semantic

Pre-training Dataset	Augm.	Multi-Task (M)					KITTI (K)			Semantic (S)
		Detection AP [Our difficulties]				Segm. mIoU	Detection AP [KITTI difficulties]			Segm. mIoU
		All	Hard	Moderate	Easy		Hard	Moderate	Easy	
No pre-training	✓	0.481	0.603	0.678	0.695	0.670	0.118	0.109	0.090	0.512
	✗	0.364	0.462	0.534	0.551	0.520	0.111	0.107	0.106	0.413
KITTI Object (K)	✓	0.582	0.724	0.814	0.833	0.464	0.340	0.305	0.298	0.380
	✗	0.522	0.657	0.743	0.757	0.246	0.489	0.479	0.559	0.221
LiDAR Semantic (S)	✓	0.322	0.416	0.476	0.501	0.691	0.045	0.046	0.046	0.539
	✗	0.129	0.154	0.174	0.176	0.676	0.012	0.011	0.004	0.531
KITTI Object (K) *and* LiDAR Semantic (S)	✓	**0.603**	**0.748**	**0.820**	**0.848**	**0.695**	0.486	0.496	0.580	0.545
	✗	0.554	0.706	0.775	0.786	0.685	0.457	0.435	0.482	0.551

TABLE IV: **Ablation Study on different pre-training setups** for our multi-task problem. The performances on the original support datasets used for pre-training after fine-tuning on the target dataset M are listed on the right-hand side.

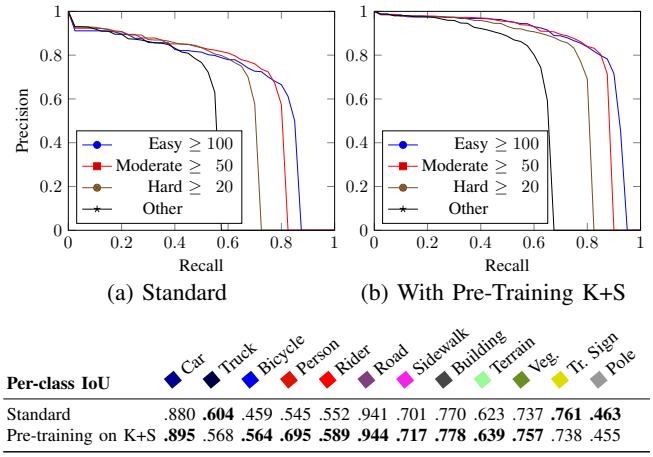


Fig. 7: **Performance gain in both tasks through pre-training:** Our approach for pre-training on K+S datasets improves both detection mAP (top) and segmentation mIoU (bottom) at the same time.

(S). However, when pre-training on a single task, the other task's performance is significantly reduced after finetuning. Compare AP 0.603 with 0.416 in detection accuracy with weights from Semantic and mIoU 0.670 with 0.464 left after using weights from KITTI.

Using the network weights from a pre-training with our proposed alternating training methodology between both tasks solves the issue of the performance drop in segmentation when pre-training on object detection and vice versa. Both tasks perform better at the same time compared to a training from scratch. This is achieved without access to a support dataset featuring the same set of multi-task annotations. Detection accuracy even improves notably compared to the pre-training with only KITTI data (AP 0.748 vs. 0.724) and segmentation performance gains slightly (mIoU 0.695 vs. 0.691). The overall improvement for both tasks is detailed in Figure 7.

Augmenting the dataset with spatial transformations improves the final performance for both tasks when dealing with limited training data. In Figure 8 we show that the positive effects of pre-training and augmentation add up for an additional performance boost. Both techniques combined allow to reach feasible performance with only few training samples and a simple training procedure. The performances on the target dataset also improve consistently in all pre-training settings (Table IV).

D. Further Analysis and Discussion

The right-hand part of Table IV details the performances of the detector on the original support datasets after fine-tuning on the target dataset. Without fine-tuning the detection rate on a different LiDAR sensor is poor reaching only an AP of 0.118 on KITTI compared to 0.692 when training on the KITTI dataset. The label domain shift between manually annotated multi-task semantic and the transferred labels in the semantic dataset (S) is small enough to reach an acceptable performance (mIoU 0.512 vs. 0.579). Later layers of deep neural network learn more high-level and therefore task-specific features. This is concordant with our findings that pre-training on single task for a multi-task fine-tuning is problematic. The task which is already pre-trained becomes dominant and hurts the other task's performance notably which cannot benefit from more low-level features.

The performance on the support datasets considerably drops in all setups when fine-tuning the network to the different target domain. What stands out is that this performance drop actually becomes substantially smaller when pre-training is performed on both datasets K+S with augmentation enabled. In this case the network yields a remaining

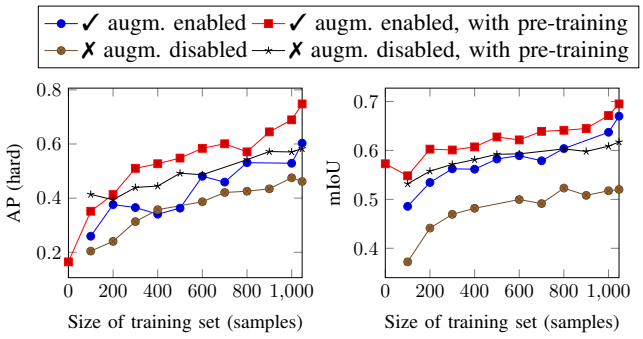


Fig. 8: **Performance over training dataset size:** Comparing the performance gains by augmenting and pre-training for a varying size of samples for training. Note that each data point on the x-axis represents a new training of our CNN.

detection performance on KITTI of AP 0.486 (K+S) vs. 0.340 (K) and segmentation performance of mIoU 0.545 (K+S) vs. 0.539 (S). We reached the overall best performance when pre-training on two diverse data domains in an alternating way (K+S). The additional observation of a mitigated performance drop on the support datasets indicates that a different quality of the learned features in pre-training has been achieved leading to better generalization of the desired tasks. This manifests in the resulting object detector achieving a object detection performance of AP 0.748 on the VLP32 and AP 0.486 on the HDL64 sensor simultaneously.

V. Conclusion

This paper presented a training methodology for heterogeneous datasets that improves performance of a LiDAR-based multi-task detection system. We proposed a novel multi-task CNN architecture and demonstrated its suitability for both object detection and semantic segmentation without much additional overhead. Our extensive experiments prove that our method is effective and enables the usage of other LiDAR-datasets even if a specific set of annotations is required for the target task. In addition we take our experimental results as an indication that our strategy enables the CNN to learn features that generalize better between varied LiDAR input domains. The described approach is task agnostic and could be applied to enable or boost performance of different LiDAR multi-task problems.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, doi: 10.1007/s11263-015-0816-y, 2015.

[2] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[3] G. Csurka, "A comprehensive survey on domain adaptation for visual applications," *Advances in Computer Vision and Pattern Recognition*, no. 9783319583464, pp. 1–35, doi: 10.1007/978-3-319-58347-1, 2017.

[4] B. Li, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," *Robotics Science and Systems*, doi: 10.15607/RSS.2016.XII.042, 2016.

[5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] A. Behl, D. Paschalidou, S. Donné, and A. Geiger, "Pointflownet: Learning representations for 3d scene flow estimation from point clouds," *CoRR*, vol. abs/1806.02170, 2018.

[8] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," in *The European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[10] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," doi: arXiv:1812.05784v1, 2018.

[11] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "SPLATNet: Sparse lattice networks for point cloud processing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2530–2539.

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," " *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 5099–5108.

[14] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[15] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *CoRR*, vol. abs/1801.07829, 2018.

[16] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3D Graph Neural Networks for RGBD Semantic Segmentation," *IEEE International Conference on Computer Vision (ICCV)*, pp. 5209–5218, 2017.

[17] B. Li, "3D Fully Convolutional Network for Vehicle Detection in Point Cloud," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-September, pp. 1513–1518, doi: 10.1109/IROS.2017.8205955, 2017.

[18] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7482–7491.

[19] M. Teichmann, M. Weber, J. M. Zllner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving." *CoRR*, vol. abs/1612.07695, 2016.

[20] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[21] F. Piewak, P. Pinggera, M. Schäfer, D. Peter, B. Schwarz, N. Schneider, D. Pfeiffer, M. Enzweiler, and J. M. Zöllner, "Boosting lidar-based semantic labeling by cross-modal training data generation." in *ECCV Workshops*, 2018.

[22] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328.

[23] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 512–519, doi: 10.1109/CVPRW.2014.131, 2014.

[24] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2017-January, pp. 5168–5177, doi: 10.1109/CVPR.2017.549, 2017.

[25] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "EuroCity Persons: A Novel Benchmark for Person Detection in Traffic Scenes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, doi: 10.1109/TPAMI.2019.2897684, 2019.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, 2015, pp. 448–456.

[27] S. Xiang and H. Li, "On the effect of batch normalization and weight normalization in generative adversarial networks," *CoRR*, vol. abs/1704.03971, 2017.

[28] Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu, "Adaptive Batch Normalization for practical domain adaptation," *Pattern Recognition*, vol. 80, pp. 109–117, doi: 10.1016/j.patcog.2018.03.005, 2018.

[29] Velodyne. Velodyne LiDAR. [Online]. Available: https://velodynelidar.com/

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

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

[32] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[33] X. Chen and Y. Zhu, "3D Object Proposals for Accurate Object Class Detection," *Advances in Neural Information Processing Systems*, pp. 1–9, doi: 10.1109/ICRA.2016.7487486, 2015.