International Journal of Computer Vision
https://doi.org/10.1007/s11263-023-01914-7

3D Adversarial Augmentations for Robust Out-of-Domain Predictions

Alexander Lehner[1,2] · Stefano Gasperini[1,2] · Alvaro Marcos-Ramiro[2] · Michael Schmidt[2] · Nassir Navab[1] · Benjamin Busam[1] · Federico Tombari[1,3]

Received: 17 December 2022 / Accepted: 15 September 2023

Abstract
Since real-world training datasets cannot properly sample the long tail of the underlying data distribution, corner cases and rare out-of-domain samples can severely hinder the performance of state-of-the-art models. This problem becomes even more severe for dense tasks, such as 3D semantic segmentation, where points of non-standard objects can be confidently associated to the wrong class. In this work, we focus on improving the generalization to out-of-domain data. We achieve this by augmenting the training set with adversarial examples. First, we learn a set of vectors that deform the objects in an adversarial fashion. To prevent the adversarial examples from being too far from the existing data distribution, we preserve their plausibility through a series of constraints, ensuring sensor-awareness and shapes smoothness. Then, we perform adversarial augmentation by applying the learned sample-independent vectors to the available objects when training a model. We conduct extensive experiments across a variety of scenarios on data from KITTI, Waymo, and CrashD for 3D object detection, and on data from SemanticKITTI, Waymo, and nuScenes for 3D semantic segmentation. Despite training on a standard single dataset, our approach substantially improves the robustness and generalization of both 3D object detection and 3D semantic segmentation methods to out-of-domain data.

Keywords Adversarial augmentation · 3D point cloud · Robustness · Domain generalization · Out-of-domain

1 Introduction

Reliable understanding of the surroundings in general settings is crucial for high automation (Kilic et al., 2021; Lehner et al., 2022; Mirza et al., 2022). However, current methods lack the necessary robustness and generalization capabilities to properly tackle unexpected events in safety-critical applications, such as autonomous driving and robotics (Baier et al., 2019). Deploying state-of-the-art approaches directly to real-world problems raises a set of issues which go beyond solving the task on a public dataset.

In these setups, characteristics such as robustness and strong generalization become of utmost importance to circumvent dangerous consequences (Gasperini et al., 2023a). As challenging scenes can drastically hinder performance (Zhao et al., 2022), particularly crucial is a model's ability to robustly generalize to unseen scenarios, e.g., out-of-domain and long tail samples, as well as to adverse illumination and weather conditions (Gasperini et al., 2023b), e.g., at night and with rain. Various categories of works are aimed at mitigating these issues, including domain adaptation (Yi et al., 2021), domain generalization (Wang et al., 2021), simulations (Beery et al., 2020), estimating the uncertainty (Gasperini et al., 2021a), and generating adversarial examples (Tu et al., 2020).

Due to the difficulty of capturing corner cases and challenging situations from the long tail of the data distribution, training datasets cannot contain all possible scenarios (Lehner et al., 2022) and typically include mostly average cases. This forces to design robust methods that work effectively not only on the available training data distribution,

Communicated by Bernhard Egger.

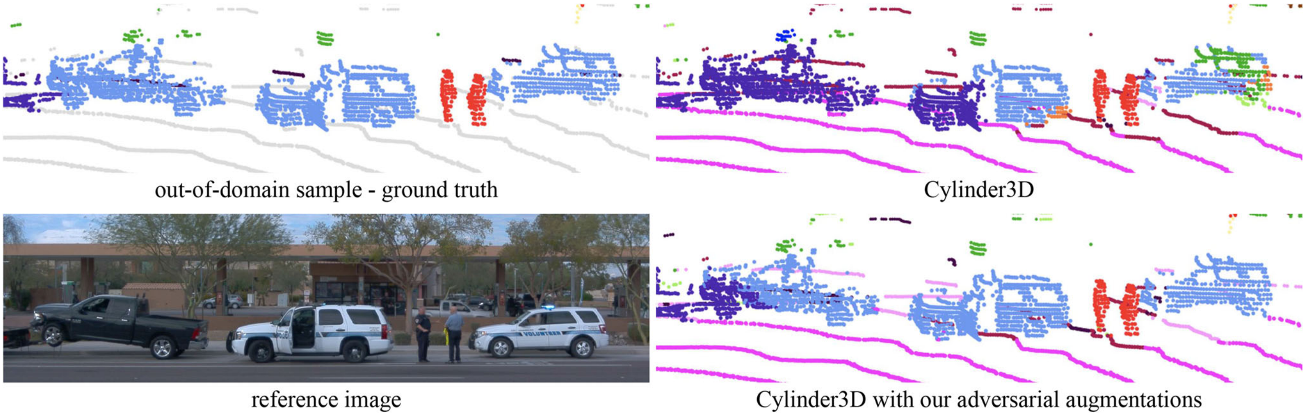
Alexander Lehner and Stefano Gasperini have contributed equally to this work.

✉ Alexander Lehner
alexander.lehner@tum.de

✉ Stefano Gasperini
stefano.gasperini@tum.de

[1] CAMP, Technical University of Munich, Munich, Germany
[2] BMW Group, Munich, Germany
[3] Google Inc., Zurich, Switzerland

Published online: 18 October 2023

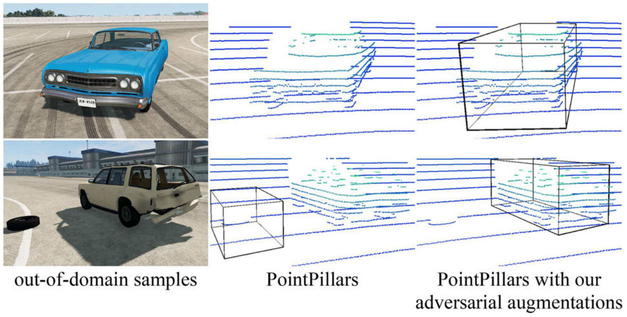

out-of-domain sample - ground truth

Cylinder3D

reference image

Cylinder3D with our adversarial augmentations

Fig. 1 3D semantic segmentation predictions of Cylinder3D (Zhu et al., 2021a) trained on SemanticKITTI (Behley et al., 2019) with and without our adversarial augmentations. This figure shows a challenging out-of-domain sample from Sun et al. (2020), including three large cars, of which a pickup truck being towed and a large SUV with an open door. Due to the domain gap, the standard Cylinder3D could not correctly segment any of the three vehicles in the scene. Instead, by plausibly expanding the available training data, our domain generalization method allowed the model to improve the segmentation of the out-of-domain cars. The road and other classes were ignored in the transfer to Waymo due to misaligned definitions across the datasets


out-of-domain samples

PointPillars

PointPillars with our adversarial augmentations

Fig. 2 3D object detection predictions of PointPillars (Lang et al., 2019) trained on KITTI (Geiger et al., 2012), without and with our adversarial augmentations, on out-of-domain cars from the proposed CrashD dataset. The models were transferred without any fine-tuning. Due to the unusual shapes of the vehicles compared to those in KITTI, the standard PointPillars could not detect them. By expanding the training distribution, our adversarial augmentations allowed the model to detect both the old car and the damaged one. *Images used with courtesy of BeamNG GmbH*

but also on rare unseen and unknown samples (Gasperini et al., 2023a; Jung et al., 2021b). Since rare samples are typically unavailable at training time and might not be part of the same distribution, as shown in Figs. 1 and 2, state-of-the-art approaches tend to fail with out-of-domain samples, thereby proving the need for more robust solutions (Gasperini et al., 2023a; Mirza et al., 2022).

Although many works addressed part of these concerns on 2D image data (Qiao et al., 2020; Beery et al., 2020; Gasperini et al., 2023a; Hendrycks et al., 2021a; Mirza et al., 2022; Postels et al., 2019), these issues remain mostly open for 3D point clouds (Lehner et al., 2022). Compared to using 2D data alone, 3D sensors (e.g., LiDAR and ToF cameras) offer an extra layer of redundancy and robustness, which is highly valuable in safety-critical settings.

In 3D semantic segmentation, methods assign a known class to each input point. Since 3D approaches heavily rely on the geometric relationship between 3D points, non-standard and out-of-domain objects, such as those in Fig. 1, can be easily assigned to the wrong class, thereby posing a safety threat if not properly taken into account. Moreover, since *softmax* highly promotes the most probable class, current approaches tend to be extremely confident even with out-of-distribution samples, or when delivering wrong predictions (Gawlikowski et al., 2021), worsening the problem. As challenging scenarios can naturally occur in the real world (Hendrycks et al., 2021b), robustness against them is a fundamental characteristic to ensure the safe deployment of a model.

To avoid the difficult and expensive collection of long tail samples for training, challenging scenarios can be designed artificially via adversarial attacks applied to readily available data (Goodfellow et al., 2015). These adversarial examples expose the vulnerabilities of a model and can be incorporated in the training data to improve its robustness via adversarial augmentation (Lehner et al., 2022; Li et al., 2022a). While existing methods have generated adversarial examples to improve robustness against out-of-domain data for 3D object detection (Tu et al., 2020; Lehner et al., 2022), this remains still unexplored for 3D semantic segmentation.

In this work, we extend the capabilities of a model to out-of-domain data. We achieve this by enriching the training data with adversarial examples, which we make plausible via constraints for sensor-awareness and smooth deformations. First, we learn a small set of vectors, applicable to the entire dataset, to generate challenging samples. Then, we retrain the model while deforming available objects with our adversarial vectors, and integrate these adversarial examples as data augmentation. Thanks to difficult and plausible deformations, we substantially improve the generalization to challenging out-of-domain data, without using any extra information. The main contributions of this work can be summarized as follows:

- We raise awareness on natural adversarial examples, such as damaged and rare cars.
- We propose a sensor-aware and sample-independent adversarial augmentation method for domain generalization on 3D point clouds.
- We specialize our domain generalization approach for both 3D object detection and 3D semantic segmentation, making it the first effectively working on both tasks.
- We strengthen the model's decision boundaries via targeted and untargeted adversarial augmentation.
- We explain the impact of our method by analyzing the vectors learned for the two tasks.
- We publicly release CrashD: a dataset containing rare out-of-domain and long tail cars.

Differently from the conference version of this work (Lehner et al., 2022), here we: (1) extend our method to 3D semantic segmentation, which requires a different architecture and adversarial loss compared to 3D object detection (Sect. 3.1); (2) generate our adversarial vectors both to resemble specific classes of choice (i.e., targeted attack) and also in an untargeted fashion; (3) assess the impact of the LiDAR intensity signal on the performance degradation of a model when transferring to different sensors (Sect. 5.4); (4) generate adversarial intensity signals within our augmentations to mitigate the domain gap (Sect. 3.1.2); (5) generate adversarial augmentations of multiple object classes (Sect. 3.2.1); (6) eliminate the need for 3D bounding boxes exploiting point-level annotations; (7) analyze how our adversarial augmentations alter the decision boundaries of the model (Sect. 5.1); (8) assess the impact of our augmentations on the robustness against transformations of the input (Sect. 5.6); (9) analyze the differences between adversarial vectors learned for different tasks as well as their impact on the models (Sect. 5.7); (10) train our vectors against a different architecture (Cylinder3D by Zhu et al., 2021a); and (11) evaluate our method on three additional public datasets.

2 Related Work

Our work focuses on adversarial augmentation for both 3D semantic segmentation and 3D object detection. In this section, first, we provide an overview of 3D semantic segmentation methods (Sect. 2.1), then we go through approaches addressing generalization across both tasks (Sect. 2.2). Finally, we explore adversarial methods, focusing on 3D point clouds and semantic segmentation (Sect. 2.3).

2.1 3D Semantic Segmentation

Depending on the input representation, semantic segmentation methods for LiDAR point clouds can be grouped in projection-based and point-based. The former project 3D point clouds to a regular grid, which allows to use standard convolutions. Regular grids can be voxels to use 3D convolutions (Zhou & Tuzel, 2018), or pixels via spherical projections to use common and fast 2D convolutions well-known from the image domain (Milioto et al., 2019; Wang et al., 2018). Voxels could be cubic (Zhou & Tuzel, 2018), vertical square columns as in PointPillars (Lang et al., 2019), or cylinder partitions as in Cylinder3D (Zhu et al., 2021a). Specifically, when partitioning the point cloud, Cylinder3D takes into account the increasing sparsity at higher distances, by means of distance-dependent voxel sizes. Instead, point-based methods operate directly on the 3D point clouds, by extracting features exploiting the geometrical relationship between neighboring points. PointNet was the first in this category (Qi et al., 2017), using multilayer perceptrons and aggregating the extracted features through a global max-pooling. Other works, such as KPConv (Thomas et al., 2019), focused on creating new operations dedicated to points. More recently, a variety of hybrid approaches tried to exploit the benefits of both projection and point-based categories (Tang et al., 2020; Xu et al., 2021).

Other recent works explored the use of various aids, such as 2D data, attention mechanisms, sequential data and contrastive learning. 2DPASS (Yan et al., 2022) exploits 2D information to improve the representation learning on LiDAR point clouds. $(AF)^2$-S3Net (Cheng et al., 2021) uses attention to capture fine details in LiDAR semantic segmentation. Chen et al. (2021) exploited sequential data to segment moving objects in LiDAR point clouds. SegContrast (Nunes et al., 2022) improves representation learning in a self-supervised fashion via a contrastive loss, to drive similar structures of LiDAR point clouds towards each other in the embedding space. LESS (Liu et al., 2022) aims at reducing human annotation for semantic segmentation of LiDAR point clouds, by leveraging geometrical patterns towards an heuristic pre-segmentation. While 3D semantic segmentation assigns a semantic class to each point in input, 3D panoptic segmentation takes an extra step by additionally segmenting the points belonging to individual instances (Gasperini et al., 2021c; Marcuzzi et al., 2022; Razani et al., 2021).

In this work, we focus on 3D semantic segmentation and base our experiments on Cylinder3D (Zhu et al., 2021a), which we retrain with the only modification of including our generated adversarial examples at training time, to improve its generalization to out-of-domain data.

2.2 Improving Generalization

Generalizing to unseen data is highly desirable for any learning-based model (Wang et al., 2021), particularly in unconstrained real-world settings, such as autonomous driving. Unseen data comprises any sample that is not part of the training set, including data both in-domain (e.g., validation set) and out-of-domain (e.g., unseen categories). Depending on the task and domain, a wide variety of techniques can be used to improve generalization and robustness, such as domain-robust sensor signals (Gasperini et al., 2021b), multi-modal fusion concepts (Jung et al., 2021a), and robust 3D descriptors (Tombari et al., 2010). Domain generalization aims at improving the performance on a target domain (e.g., data captured by a different sensor), without using any information about it Wang et al. (2021). Instead, domain adaptation exploits knowledge about the target data (Wang et al., 2020b; Yi et al., 2021). Domain generalization methods can be categorized in two groups: those acting on the model itself, and those operating on the input data.

Among the former category, regularizing the model is commonly done to reduce overfitting (Srivastava et al., 2014) or address domain generalization (Balaji et al., 2018). Gasperini et al. (2021a) found uncertainty estimation beneficial to reduce the domain gap between different data distributions. Since estimating the uncertainty provides information about the knowledge boundaries of a model, it is helpful to segment unknown objects belonging to unseen categories as well (Gasperini et al., 2023a). Furthermore, search algorithms were used to find new architectures that improve robustness (Mok et al., 2021).

Generalization can be improved also by manipulating the input data. Albuquerque et al. (2020) explored pretraining and multi-task learning to improve results on out-of-distribution data. Moreover, adding synthetic samples can strengthen the performance on rare classes (Beery et al., 2020). Data augmentation (Summers & Dinneen, 2019; Zhang et al., 2021b; Hendrycks et al., 2021a; Nekrasov et al., 2021) is part of this category as well. Among these, adversarial approaches extend the training data with altered inputs learned in an adversarial fashion as a way to improve generalization (Volpi et al., 2018; Tu et al., 2020; Qiao et al., 2020; Lehner et al., 2022).

The method we propose in this work addresses domain generalization (i.e., does not use any target information) and belongs to the data category, specifically to the adversarial approaches, detailed in Sect. 2.3.

2.2.1 Generalization in 3D Object Detection

In the context of generalization, some works addressed the task of 3D object detection. In the image domain, Simonelli et al. (2020) created virtual views normalizing the objects with respect to their distance. By doing so, they were able to generalize better to samples at different depths. Tu et al. (2020) improved the generalization towards cars with roof-mounted objects, by using adversarial examples on LiDAR point clouds. With LISA, Kilic et al. (2021) targeted adverse weather conditions with a physics-based simulator for LiDAR point clouds. They generated data and included it during training to improve the model robustness in challenging conditions. Wang et al. (2020b) explored domain adaptation to bridge the gap between LiDAR point clouds containing cars with different sizes, due to the distributions of vehicles in different countries (e.g., Germany and USA).

Generalization is also the focus of our work, where we explore adversarial augmentation for 3D object detectors on point clouds.

2.2.2 Generalization in 3D Semantic Segmentation

With the increasing interest of expanding the applicability of learning-based systems, several researchers aimed to improve generalization and robustness of 3D semantic segmentation approaches. PCT (Xiao et al., 2022) mitigates the reality gap from synthetic LiDAR data for semantic segmentation. Yi et al. (2021) explored domain adaptation for semantic segmentation of LiDAR point clouds. They used a network to recover the underlying point cloud surface, from which they transferred labels across different LiDAR sensors. They also executed an experiment without using target knowledge, thus performing domain generalization. Mix3D (Nekrasov et al., 2021) is a data augmentation technique for semantic segmentation aimed at generalizing beyond contextual priors of the training set. It works by combining two augmented scenes, thereby generating novel out-of-context environments. Li et al. (2022b) improved robustness by means of a test time augmentation strategy within a knowledge distillation framework, incorporating a Transformer-based voxel feature extractor. The recent 3DLabelProp (Sanchez et al., 2022) was the first method released for domain generalization in 3D semantic segmentation. Specifically, they assessed the generalization ability of state-of-the-art approaches and proposed a method exploiting both spatial and temporal information. By relying on the previous frames from a sequence, as well as the current input, they were able to increase the robustness of the output.

In this work, we improve the model robustness and generalization through adversarial augmentation. Unlike domain adaptation approaches which use knowledge about the target domain (Yi et al., 2021), similarly to 3DLabelProp we perform domain generalization, thus use solely information about the source domain. Moreover, our method operates on individual frames, instead of sequences as 3DLabelProp (Sanchez et al., 2022). Therefore, our method differs from all existing approaches. Ours is also the first general approach shown to work effectively on both 3D object detection and 3D semantic segmentation.

2.3 Adversarial Examples

Adversarial examples are input modifications purposely designed to lead a model to wrong predictions (Szegedy et al., 2014; Goodfellow et al., 2015). They are generated against a trained model (e.g., to reduce its accuracy) and are typically transferable to other models as well (Lehner et al., 2022). A multitude of works already explored adversarial examples in the image domain (Carlini & Wagner, 2017; Moosavi-Dezfooli et al., 2016; Papernot et al., 2016; Yuan et al., 2019; Xiao et al., 2018), where small pixel perturbations fool target models. Alaifari et al. (2019) deformed images via sample-specific adversarial vector fields (i.e., sets of vectors anchored to a set of points in a given space). Wang et al. (2020a) proposed adversarial morphing fields to alter image pixels spatially. However, adversarial examples are still mostly unexplored on point clouds, especially those captured by 3D sensors (e.g., LiDAR and ToF camera).

2.3.1 Adversarial Methods for Point Clouds

Adversarial methods for 3D point clouds can be divided into three categories: generation if they add points, removal if they remove points, and perturbation if they only move points.

Generation and Removal Xiang et al. (2019) were the first to work on adversarial point clouds and proposed a series of methods, some of which added points. Cao et al. (2019b) added adversarial objects to LiDAR point clouds. Similarly, Tu et al. (2020) added adversarial meshes on the roof of cars. A different set of works investigated sensor attacks, adding points with a spoofing device (Cao et al., 2019a). Instead, removal methods learn to discard critical points in an adversarial fashion (Yang et al., 2019).

Perturbation Xiang et al. (2019) proposed the first two perturbation methods. For the iterative gradient L2 attack, they adapted PGD from the image domain (Madry et al., 2018), optimizing for a minimal deformation constrained by the L2 norm. They also proposed the Chamfer attack, using the Chamfer distance between the original and the deformed object to reduce the perceptibilty of the attack (Liu et al., 2020). Our method is similar to the iterative gradient L2 attack, but we do not learn a dedicated vector for every point of each sample. Instead, we learn one sample-independent vector field (i.e., operating on the whole dataset) and introduce further constraints to improve the plausibility of our perturbations. Liu et al. (2020) explored more noticeable perturbations than the ones of Xiang et al. (2019). They produced continuous shapes by altering neighboring points accordingly. Cao et al. (2021) generated adversarial objects and then 3D printed them to fool multi-modal (LiDAR and camera) object detectors.

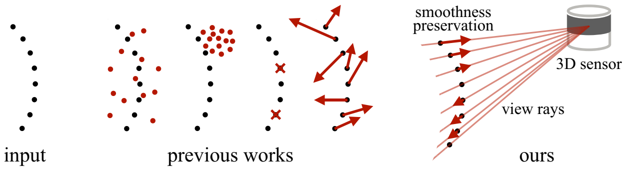


Fig. 3 Adversarial perturbations introduced by previous works, compared to ours. While others add points, remove them, or shift them with minor constraints, our approach is sensor-aware and only slides points along the sensor view ray, thereby preserving occlusions. Additionally, our method produces adversarial examples with smooth surfaces, by considering the movement of the neighboring points when computing the shift for each point

Generalization Most works on adversarial point clouds were proposed targeting the ModelNet dataset (Xiang et al., 2019; Liu et al., 2020; Hamdi et al., 2020), which features a set of synthetic 3D point clouds with various object shapes. As the samples of ModelNet were not captured by a 3D sensor, these pioneering works often produce unrealistic outputs (Xiang et al., 2019; Liu et al., 2020). In fact, these approaches were not intended to improve the generalization of the models, but rather set the basis for adversarial attacks on point clouds (Xiang et al., 2019). Additionally, they are all sample-specific, making their applicability limited in practice, as they would require to be optimized independently for each object they are applied on Xiang et al. (2019), Liu et al. (2020) and Hamdi et al. (2020). Instead, Tu et al. (2020) assessed the impact on 3D object detection of meshed objects (e.g., canoes and couches) synthesized on top of a car roof within a LiDAR scene. Moreover, they attacked these meshes and used them to defend the detector. By doing so, they improve robustness and generalization to unseen cars with roof-mounted objects.

Our work is significantly different from all sample-specific methods (Alaifari et al., 2019; Xiang et al., 2019; Liu et al., 2020; Yang et al., 2019). In fact, we construct a single set of vectors. Similarly to Tu et al. (2020), we aim to improve the generalization to out-of-domain data. However, unlike them, as can be seen in Fig. 3, we do not add any points, making ours a perturbation method. Additionally, by not making any assumptions on the object nor the kind of sensor, our method has a wider applicability than that of Tu et al. (2020). Moreover, we preserve the plausibility of our adversarial examples by considering occlusion

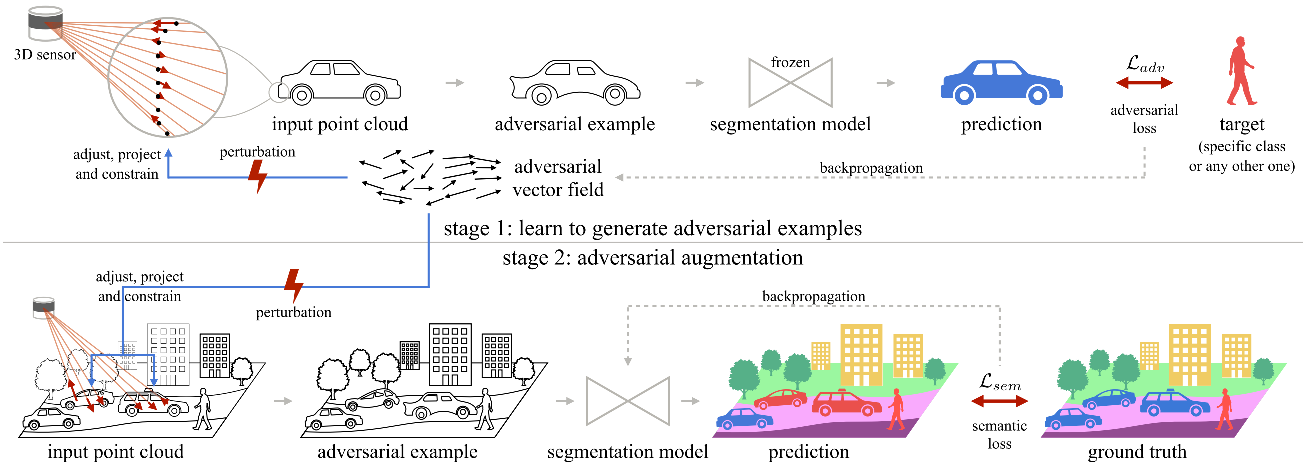


Fig. 4 Overview of the proposed method. In the first stage (top), we learn a vector field by iteratively deforming objects to minimize the adversarial loss $\mathcal{L}_{adv}$ against a frozen model. The applied deformations are plausible thanks to a series of constraints (e.g., moving points only along their sensor view ray), and they can either target a specific different class (e.g., *person* as in the figure), or remain untargeted to any other class. In the second stage (bottom), the vector field learned in the first stage is deployed throughout the scenes for adversarial augmentation. The augmented scenes are used during the training of a new robust segmentation model, thereby improving generalization to unseen objects. While the diagram exemplifies the process for 3D semantic segmentation, the same method can be similarly applied to other tasks, such as 3D object detection

constraints, which were ignored so far, and making our perturbations sensor-aware. We achieve this by shifting points only along the sensor ray. Furthermore, our method differs from previous works as it generates adversarial examples via transferable learned vector fields.

In the conference version of this work (Lehner et al., 2022) we first deformed objects to fool a 3D object detector, then incorporated these adversarial examples when training a new detector (adversarial augmentation). By doing so, we substantially improved its robustness and generalization to out-of-domain samples. In this work, we aim at extending this to 3D semantic segmentation. Moreover, we bring a series of improvements towards out-of-domain generalization, such as targeted adversarial augmentation, as well as new insights on our method and domain generalization via thorough analysis.

2.3.2 Adversarial Methods for Semantic Segmentation

The majority of work exploiting adversarial examples (e.g., as adversarial attacks or for data augmentation) has focused either on the imaging domain (Luo et al., 2021; Abdollahi et al., 2021; Arnab et al., 2018) or on different tasks, such as classification (Li et al., 2022a) or object detection (Tu et al., 2020; Lehner et al., 2022). Instead, only few methods were proposed for semantic segmentation of point clouds. Among these, AttAN (Zhang et al., 2021a) explores adversarial learning to improve the predictions by correcting noisy ones, while (Zhu et al., 2021b) proposed an adversarial attack in which they carefully added real objects in a LiDAR scene, to fool state-of-the-art segmentation methods.

Our work generates adversarial examples and performs adversarial augmentation for 3D semantic segmentation, substantially improving the generalization to out-of-domain data by expanding the available training set with hard and plausible examples.

3 Method

Our method is based on deforming point clouds to improve generalization and robustness against natural object variations and out-of-domain data. We achieve this via adversarial augmentation. We apply our approach both on 3D object detection and 3D semantic segmentation. As shown in Fig. 4, our method is based on a vector field learned in an adversarial fashion. After training the vectors against a frozen target model (Sect. 3.1), we freeze the vector field and apply it to objects in the available training data. The same vector field can be applied to any seen or unseen object. To do so, we first scale the vector field to match the target object dimensions. Then, we constrain the points movement to preserve the object shape and occlusions, while making the deformations sensor-aware (Sect. 3.2). Our vector fields are class specific and we use them to deform all objects of their class. Such deformed objects are adversarial examples, which we integrate during training as data augmentation (Sect. 3.3).

3.1 Adversarial Vector Field

Since the goal is fooling the detector/classifier by perturbing a point cloud without adding nor removing points, vectors are convenient in this setting, because they represent shifts in the point cloud directly. Additionally, using vectors allows for both compactness and transferability, as the same learned vector field can be applied to any target object.

Construction To create a lattice of uniformly spaced 3D vectors within a 3D bounding box, we discretize the space of a default bounding box B_o with a step size t. This generates root coordinates $\boldsymbol{f}$ in 3D space, each of which is assigned an empty vector $\boldsymbol{v} = (x, y, z)$. The default bounding box B_o is defined by its width w, height h, length l, orientation angle α, and its center $\varsigma = (x, y, z)$.

3.1.1 Adversarial Losses

For 3D Object Detection To suppress irrelevant bounding box proposals and focus on the ones that are most likely to be accurate, we use a binary cross entropy loss following the method described by Tu et al. (2020). Let $\mathcal{Q}$ be the set of relevant proposals, where each proposal q has a confidence score s. A proposal is considered relevant if its prediction confidence score $s > 0.1$. We minimize s by weighting it according to the 3D IoU with the ground truth q^*. The objective is defined as:

$$\mathcal{L}_{adv.od} = \sum_{q,s \,\in\, \mathcal{Q}} -\text{IoU}(q^*, q) \log(1 - s). \tag{1}$$

During training, minimizing $\mathcal{L}_{adv.od}$ optimizes the vector fields to reduce the confidence score of the prediction of the detector. As the optimization converges, this causes the detector to either miss the object or predict a misaligned bounding box.

For 3D Semantic Segmentation Specifying our method to the task of 3D semantic segmentation, we explore two alternative configurations, namely untargeted and targeted adversarial augmentations. As for 3D object detection, we first learn a set of vector fields, via a loss function which depends on the configuration, as described below. While augmenting with untargeted adversarial examples aims at strengthening the weakest decision boundaries of a model, using targeted ones gives the option of reinforcing a specific class boundary of interest.

Untargeted Loss for 3D Semantic Segmentation The untargeted attack aims to change the model's prediction ρ of each point to any wrong class. This attack is trained by maximising the cross-entropy loss which would be normally minimized when training the segmentation model. Let C be the number of classes, $\gamma_{\boldsymbol{p},c}$ be the binary indicator (0 or 1) if class label c is the correct classification for its points $\boldsymbol{p}$ and $\rho_{\boldsymbol{p},c}$ the predicted probability of all points $\boldsymbol{p}$ belonging to class c. The objective function to be minimized for the untargeted vectors is defined as the opposite of the standard cross-entropy loss as:

$$\mathcal{L}_{adv.ssu} = \sum_{c=1}^{C} \gamma_{\boldsymbol{p},c} \log(\rho_{\boldsymbol{p},c}). \tag{2}$$

So the goal is to fool the model into wrongly classifying as many points as possible. The easiest way to minimize $\mathcal{L}_{adv.ssu}$ is attacking the weakest decision boundary for each point. This loss formulation aims at globally degrading the performance of the model, also beyond the points being perturbed directly (i.e., belonging to the class of interest). Therefore, it can shift points to change the predictions of other points of which it does not change the location.

Targeted Loss for 3D Semantic Segmentation The targeted attack aims at changing the model's predictions to a specific class. So instead of maximizing the loss of the correct class as in the untargeted setting, here we minimize the loss for a different specific class of choice $\hat{c}$. Let $\boldsymbol{p}_{c^*}$ be the points of the class c^* whose points are perturbed (i.e., adversarial class), and $\rho_{\boldsymbol{p},c^*}$ the model's predictions for these points. Given $\gamma_{\hat{c}}$ as the binary indicator for the chosen target class $\hat{c}$, the objective function to be minimized for our targeted vectors is defined as:

$$\mathcal{L}_{adv.sst} = -(\gamma_{\hat{c}} \log(\rho_{c^*})). \tag{3}$$

This formulation forces the adversarial vectors to cross the decision boundary between the correct class c^* and the target class $\hat{c}$. Unlike the untargeted loss $\mathcal{L}_{adv.ssu}$ which aimed at reducing the IoU across all classes, by acting upon the points of the adversarial class, the targeted loss $\mathcal{L}_{adv.sst}$ ignores the predictions on all points other than those of the adversarial class c^*.

Training Procedure To train the vector field, we apply the same vectors to all target objects in every scene in the training set, and minimize the adversarial loss across the entire dataset. This process iteratively updates the vectors, resulting in different deformations of the target objects and ultimately leading to different predictions. As the chosen adversarial loss $\mathcal{L}_{adv}$ smoothly converges, the performance of the model, against which the vector field is optimized, decreases. Once the vectors have been trained, they can be used for data augmentation to improve the performance of a new model.

3.1.2 LiDAR Intensity

For models using the LiDAR intensity (also called reflectivity, or reflectance), such as Cylinder3D (Zhu et al., 2021a), we perturb not only the 3D location of the LiDAR points, but also their intensity signal τ. In these cases, we add an additional dimension to our adversarial vectors, making them 4-dimensional, i.e., 3 spatial coordinates plus intensity, as $\boldsymbol{v} = (x, y, z, \tau)$. By doing so, the adversarial loss affects not only the 3D location of the points, but also their intensity values. Therefore, the same loss functions $\mathcal{L}_{adv}$ can be applied to learn these 4D vectors.

3.2 Point Cloud Perturbation

To apply a vector field, we first scale it to fit the target object's size. We then manipulate the points using these vectors and constrain their movement as described in the following sections.

3.2.1 Anchor Points

By being sample-independent, our vectors must maintain a relative spacing between them. Therefore, each vector needs to be anchored to a certain point, which can be either part of the point cloud, or not. This is crucial both when learning and when applying the vectors, otherwise it would be unclear how to move which points. In the conference version of this work, where we focused only on 3D object detecion (Lehner et al., 2022), we anchored the vectors to a grid formed within a reference bounding box B_o. Specifically, we relied on ground truth 3D bounding boxes to learn and apply our vector fields as augmentation. However, for 3D semantic segmentation, 3D bounding boxes may not be available. A naive approach without utilizing 3D boxes could employ a large vector field that covers the entire scene, which could be active only on the points belonging to a certain adversarial class (e.g., *car*). This has severe drawbacks: it would require a very high amount of vectors, and the vectors could overfit due to the limited samples available at certain locations within the scene (e.g., at far distances), reducing their efficacy. Instead, we address the lack of ground truth 3D bounding boxes with two alternative solutions: an off-the-shelf 3D object detector to predict 3D bounding boxes, and axis-aligned bounding boxes around the points belonging to each instance, which are even simpler to obtain.

Predicted Bounding Boxes Using 3D bounding boxes when applying our method on either 3D object detection or 3D semantic segmentation has the benefit of following a similar pipeline for both tasks. While ground truth bounding boxes are readily available on datasets designed for 3D object detection (e.g., KITTI Geiger et al., 2012), they may be unavailable for semantic data (e.g., SemanticKITTI Behley et al., 2019). In the latter case, they can be obtained with an off-the-shelf detector (e.g., Deng et al., 2021). However, the effectiveness of our method would then depend on the performance of the 3D object detector. To mitigate this dependency, we discard false positives by ensuring that all boxes contain points annotated with the correct class. When learning to deform, false negatives can be problematic because if too few samples are detected, the vectors may overfit. When performing adversarial augmentation, too many false negatives render the augmentation useless as it would be applied only on a limited number of samples, which would not allow for improved generalization to out-of-domain data. We use predicted boxes for the popular *car* class. Instead, for smaller classes, such as *person*, the performance of 3D object detectors is not as reliable (also due to the KITTI dataset (Geiger et al., 2012) missing annotations for *person* and *cyclist*), requiring different solutions.

Axis-Aligned Boxes Using axis-aligned boxes instead of predicted boxes implies having access to point-level instance annotations. We construct an axis-aligned box around the points belonging to each instance. Compared to predicted boxes, using ground truth instance annotations guarantees the correct coverage of all objects. The drawback lies in having larger grids (e.g., when the object is rotated by 45° with respect to the ground axes), which causes a sparser distribution of vectors on the points to be perturbed. However, this strategy has the advantage that when applying the vector fields all objects can be deformed, assuming that their points are annotated. Since existing 3D object detectors for LiDAR data mainly focus on the *car* class, which is the one having the most accurate annotations in KITTI (Geiger et al., 2012), their performance on other classes is sub-optimal. Therefore, for classes other than *car* (e.g., *person*) we opt for axis-aligned boxes. Since these boxes have no direction indication, we extract a pseudo orientation, by considering the shorter side $min = (w, l)$. Such ambiguity prevents the vector fields to specialize to certain orientations. Nevertheless, we deal with the direction ambiguities as described in Sect. 3.2.3.

3.2.2 Plausibility Constraints

Optical Ray Consistency To improve the generalization ability of our models and to ensure that the deformations take into account the physical constraints of the sensor, we use a sensor model that allows 3D points to be moved only along the optical ray. To do this, we first compute the ray $\boldsymbol{u}_i$ between the sensor and each point $\boldsymbol{p}_i$, which determines the direction in which each point can be moved. Then, we calculate the deformation vectors $\boldsymbol{r}_i$ for each point $\boldsymbol{p}_i$ by projecting its nearest vector $\boldsymbol{v}_i$ onto the ray $\boldsymbol{u}_i$. This means that each point can only be moved by the vector $\boldsymbol{r}_i$.

Regularizing the Deformations To limit the amount by which the points can be perturbed, we restrict the vectors $\boldsymbol{v}_i$ to have a maximum magnitude of ϵ according to the standard PGD L_∞ attack (Madry et al., 2018). We also ensure that the resulting deformed shape has smooth surfaces by sampling multiple neighboring vectors and using them to move a given

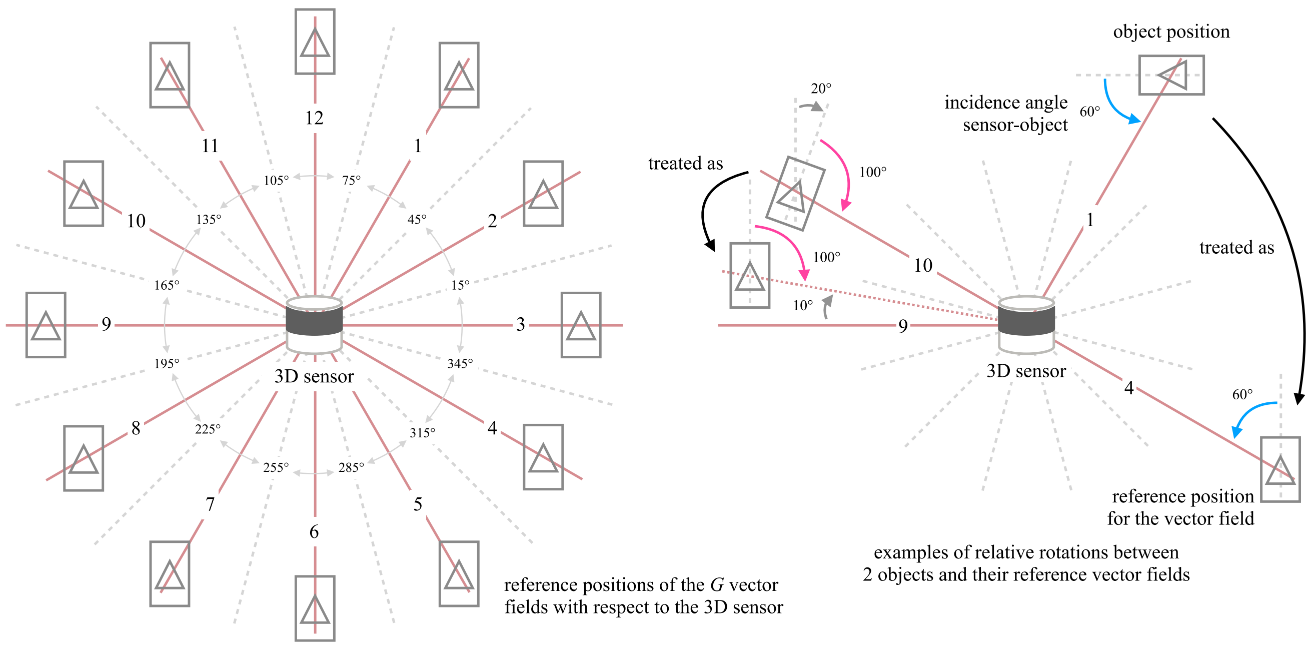

12
11
1
10
2
9
3
8
4
7
5
6
105°
75°
135°
45°
165°
15°
195°
345°
225°
315°
255°
285°
3D sensor
reference positions of the G vector fields with respect to the 3D sensor
object position
incidence angle sensor-object
60°
20°
treated as
100°
100°
10°
10
1
9
3D sensor
4
treated as
60°
reference position for the vector field
examples of relative rotations between 2 objects and their reference vector fields

Fig. 5 Representation of the relative rotations between the 3D sensor and the objects. The triangle inside each box points towards the direction of the object (e.g., a *car* driving away from the sensor at position 12, left). We use $G = 12$ vector fields, one for each of the 12 positions in the scheme (left). The red sensor rays are positioned at the β_g angles, such as 0, 30, and 60°. As we consider relative rotations, objects pointing in other directions are treated as axis-aligned objects at different positions (left) depending on their incidence angles from the sensor rays. For example (right), the object at position 1 pointing left is hit by the sensor ray with an incidence angle of 60° (blue), so it will be treated by the 4-th vector field, which will keep its incidence angle unchanged whilst rotating the object by 90° clockwise such that it points forward. Moreover, objects at angles that do not match the G reference positions are considered at intermediate positions according to their rotation angles. So the object at position 10 rotated clockwise by 20° is treated by the 9-th vector field at an angle that is 10° clockwise from the reference position 9, thereby keeping the incidence angle unchanged at 100° (pink) (Color figure online)

3D point. For each j-th vector among the k nearest neighbors, we calculate the Euclidean distance d_{ij} between the point $\boldsymbol{p}_i$ and its nearest vector $\boldsymbol{v}_{ij}$ from the vector field. The final position of each point is determined by weighing the deformation vectors $\boldsymbol{r}_{ij}$ with their corresponding distances d_{ij} and summing them together:

$$\boldsymbol{m}_i = \frac{\sum_{j=1}^{k} \frac{1}{d_{ij}} \boldsymbol{r}_{ij}}{\sum_{j=1}^{k} \frac{1}{d_{ij}}} \tag{4}$$

This allows for a smoother transition in depth between neighboring points, as vectors with opposite directions would cancel each other out and result in little or no movement of the affected point. All together, this helps to preserve the smoothness of the deformed shape and reduce the amount of irregular deformations.

Regularizing the Intensity Shift When using the intensity as extra input signal, the forth dimension of each vector is used to increase or decrease its value τ. Since the intensity is a scalar, it is independent from the optical ray direction. However, we constrain the adversarial shift Λ on the intensity values τ for each point $\boldsymbol{p}_i$ by a maximum of ψ. To ensure smooth intensity changes on neighboring points we also weight the final intensity perturbation similarly to Eq. 4. The resulting intensity value is then clipped within the range [0, 1].

The effect of our attack is illustrated in Fig. 10, where it can be seen how compared to the sample-specific Chamfer attack (Liu et al., 2020), ours preserves the overall object shape, moving the points only slightly.

3.2.3 Relative Rotation

While it is possible to use a single vector field for all objects of a certain class in the entire dataset, this would lead to rather small and generic deformations. By fitting the same vector field to all objects, its vectors need to point in all directions to be able to shift points across the highly diverse object poses with respect to the capturing sensor. This wide variability of directions reduces its efficacy, resulting in small deformations. We circumvent this problem and allow for a larger degree of alignment between neighboring vectors, by learning G different vector fields according to the relative rotation of the object to the sensor along yaw. Specifically, as shown in Fig. 5 from a top-down view, we divide the surrounding 360° angle around the vertical axis centered at the 3D sensor in $G = 12$ slices of 30° each, starting from 15°. By doing so, each vector field g focuses on perturbing only objects visible from ± 15° around β_g, which is the reference angle for the vector field g (shown in Fig. 5 by the red sensor rays). In particular, as illustrated in the figure, for every object we consider its incidence angle to the sensor, as well as its orientation angle in the coordinate system of reference. As shown in the right of the figure, we rotate each object to make it axis-aligned and orient it towards the front of the sensor (i.e., forward with respect to the ego vehicle), then we position it around the sensor such that its incidence angle remains unchanged. Depending on which g of the G positions the rotated axis-aligned and forward-facing object is located, we select the corresponding vector field g to perturb it. This is described in Fig. 5 via two examples.

When using ground truth or predicted 3D bounding boxes (Sect. 3.2.1), the object orientation is available. This allows the 12 vector fields to specialize on different instances, such that each can be specific to objects having points located in certain regions of the default bounding box B_0. This means applying each vector field g only to the objects appearing in its corresponding slice around β_g, as shown in the figure. However, without oriented 3D bounding boxes, but using axis-aligned boxes determined by the point-level annotations (Sect. 3.2.1), it is not possible to establish the orientation of the objects. Therefore, with $G = 12$, opposite pairs of vector fields would overlap, e.g., 1 and 7 in Fig. 5. To avoid this issue, with axis-aligned boxes we consider only $G = 6$ vector fields (1–6), and use them at both their nominal positions and the corresponding opposite ones.

3.3 Adversarial Data Augmentation

After learning to deform objects in the first stage (Fig. 4), we use the vector field as adversarial augmentation. In this phase, we train a model for a downstream task (e.g., 3D object detection or 3D semantic segmentation) and use the adversarial vectors to perturb the scenes in input as data augmentation. This increases the robustness of the model and also its generalization capability, as we expand the training data in a plausible way thanks to our combination of adversarial examples and constraints (e.g., sliding points only along their sensor view ray). Since the learned perturbations are structurally-consistent, they are better suited than standard augmentations (e.g., scaling, flip, rotation) to resemble rare out-of-domain object shapes, such as cars from a different country (Wang et al., 2020b).

Instead of applying the same set of vectors over the entire dataset, we increase the diversity of the training data by using N different vector fields for each of the G rotations (Sect. 3.2). These $N \times G$ vector fields are learned independently from one another and they are randomly initialized. During training of the model for the downstream task, we randomly select one object in the scene and we deform it with a randomly chosen vector field out of the N available ones for its relative rotation.

We do not augment all instances in the scene to let the model learn unperturbed objects as well as deformed ones. Moreover, thanks to the N different sets of vectors, the same object can be deformed differently at different epochs as different vectors are chosen. Furthermore, the same vector field does not deform all objects the same way, as it gets scaled and rotated to match the object it is applied on. Then, since the 3D points do not lie on the same grid as the vectors, their shift depends on how close they are from which vector. By considering neighboring vectors and the relative distances of each point to the closest vectors, even slightly different 3D locations of the points result in different deformations.

Overall, we introduce a wide variability and randomness in the training process which aid generalization and prevent overfitting to specific deformations, as demonstrated in the next section through extensive experiments.

4 Experimental Setup

4.1 Datasets

We conducted our experiments on seven different datasets. Six of them are autonomous driving datasets with available LiDAR data: KITTI (Geiger et al., 2012), the Waymo Open Dataset both for semantic segmentation and object detection (Sun et al., 2020), our proposed synthetic CrashD, SemanticKITTI (Behley et al., 2019), and nuScenes (Caesar et al., 2020). Additionally, in our previous conference publication we applied our method also on the indoor SUN RGB-D dataset (Song et al., 2015), to demonstrate the wide applicability of our approach, also to time-of-flight (ToF) cameras. All datasets used in this work are openly available, provided by the respective authors cited in the reference section.

4.1.1 3D Object Detection Datasets

We report on four different datasets for 3D object detection. **KITTI** (Geiger et al., 2012) is a popular 3D object detection benchmark recorded in Germany. We adopted a standard split (Lang et al., 2019), which comprises 3712 training and 3769 validation LiDAR point clouds, where we used the *car* class, reporting on the standard *easy*, *moderate* and *hard*. We evaluated models trained on KITTI for 3D object detection on Waymo and our CrashD (without any fine-tuning) to assess the generalization capability of the models to out-of-domain data, particularly critical for autonomous driving. The **Waymo** Open Dataset (Sun et al., 2020) is a challenging large-scale collection of real scenes recorded in various locations of the USA. It is highly diverse with different weather and illumination conditions, such as rain and night. Specifically, we evaluated on the official validation set. For LiDAR 3D object detection, we trained solely on KITTI, transferred the models to Waymo and CrashD, and considered only the *car* class.

Proposed CrashD Dataset To estimate the generalizability of a model on rare out-of-domain data, in our previous conference publication we openly released a synthetic dataset, which we called CrashD (Lehner et al., 2022). This dataset was designed as out-of-domain test benchmark for 3D object detectors trained on KITTI (Geiger et al., 2012), Waymo (Sun et al., 2020) or similar datasets. The proposed CrashD includes various types of rare cars with different shapes, such as old, sports, and damaged. While if included at all these categories are normally only long tail samples in standard datasets, in CrashD half of the cars are damaged, and half have an unusual shape (e.g., classic cars). The other half of each category is undamaged, or normal, serving as control group to assess the performance gap of a model between standard cars and rare out-of-domain ones. Specifically, the crashes were individually generated with a realistic physics simulator (Maul et al., 2021). We generated various types of crashes and we distinguished them depending on the intensity, namely *light*, *moderate*, *hard*, as well as the kind of damage: *clean* (i.e., undamaged), *linear* (i.e., frontal or rear), and *t-bone* (i.e., lateral). We randomly and automatically generated 15,340 scenes and captured them with a 64-beam LiDAR, which we configured to mimic the one in KITTI. Each scene features between 1 and 5 cars, with visible damages (*crash* set). Then we collected the *clean* set after repairing the vehicles and placing them at the same locations. Overall, CrashD contains 46,936 cars. CrashD is available for download through the dataset website https://crashd-cars.github.io/. Further details can be found in our conference publication and its supplementary material (Lehner et al., 2022).

4.1.2 3D Semantic Segmentation Datasets

We report results for 3D semantic segmentation on three different autonomous driving datasets. **SemanticKITTI** (Behley et al., 2019) is a popular 3D semantic segmentation benchmark recorded in Germany. Behley et al. annotated with semantic classes the LiDAR point clouds from the KITTI odometry task. SemanticKITTI includes point-level annotations for 23,201 full 360° scans for training (including validation) and 20,351 for testing. We followed the convention, using sequence 08 as validation set (i.e., 4071 samples), and the remaining annotated sequences for training (i.e., 19,130 samples). We evaluated the models across the standard 19 classes, 8 of which are objects. As for 3D object detection, we are interested in assessing the generalization and robustness of the models, so we include two additional datasets as transfers (without any fine-tuning). The **Waymo** Open Dataset (Sun et al., 2020) was recently extended with point-level annotations for a subset of the 3D object detection scans. The 3D semantic segmentation dataset includes labels for 23 different classes. **nuScenes** (Caesar et al., 2020) is another large scale autonomous driving dataset. It contains around 15 h of driving data collected in Boston and Singapore, with diverse traffic scenarios (e.g., both left and right hand drive). Similarly to Waymo, it includes more challenging weather and lighting conditions (e.g., rain and night) compared to SemanticKITTI, making it a difficult out-of-domain test for our models. For Waymo and nuScenes, we evaluated on their validation sets.

4.1.3 Domain Generalization

As we did not train our models on Waymo, nuScenes or CrashD, but only used these as generalization tests after training solely on KITTI (detection) SemanticKITTI (segmentation), various challenges arised when transferring. KITTI and SemanticKITTI were captured with the same sensors and we designed CrashD to mimic the LiDAR from KITTI. However, the sensors used in the other transfer datasets are rather different, thereby providing highly different 3D point clouds. Scans from Waymo are 50% denser than those of KITTI, and the field of view is narrower on Waymo. Instead, nuScenes was captured with a 32-beam LiDAR, delivering significantly sparser point clouds than those captured with the 64-beam sensor of SemanticKITTI. For object detection, we report on the proposed CrashD to show the model performance at the long tail of the data distribution, featuring rare out-of-domain samples, such as damaged and rare cars. Instead, Waymo is used to demonstrate the capabilities on challenging real-world data from a different country captured by a different 3D sensor. For the same reasons, for semantic segmentation, we conducted experiments on Waymo and nuScenes. However, we used both in order to assess the models performance with denser and sparser point clouds compared to those used for training. Overall, the difficult set of transfers included in this work demonstrates the ability of the models to generalize beyond their training specifications.

Semantic Classes for Transfers Since the annotation specifications adopted for each dataset are different, we had to map the classes accordingly. This was not an issue for object detection (Lehner et al., 2022) as we focused on the *car* class, which is available on all datasets used. Instead, for semantic segmentation, similarly to previous works (Yi et al., 2021), we could not evaluate on all classes. In fact, segments such as *road* were annotated differently and had to be ignored for the transfer: Waymo (Sun et al., 2020) included also the driveways connecting parking lots with the road over sections of sidewalks, while SemantiKITTI did not. Nevertheless, on

SemanticKITTI all our models (including Cylinder3D) were trained on the 19 classes available. When transferring we considered only the compatible classes: 12 for Waymo and 8 for nuScenes. Instead, to compare with Yi et al. (2021), we used their class mapping: 2 classes for Waymo and 10 for nuScenes. The details of the mappings can be found in our Supplementary Material.

4.1.4 Annotation Requirements

As described in Sect. 3.2.1, it is crucial to associate the adversarial vectors to a region of the point cloud. We did this via bounding boxes. Therefore, for object detection training our method does not require any extra annotations, other than the 3D bounding boxes which are readily available for training a detector. Instead, a semantic segmentation dataset does not necessarily include bounding box annotations. We circumvent this in 2 alternative ways: via an off-the-shelf 3D object detector, or via axis-aligned bounding boxes. The former strategy implies that the detector has been trained with 3D bounding box annotations on a different dataset. The latter requires point-level annotations of instances, which may not be available. Exploring both strategies ensures the applicability of our method. Conversely, at inference time our adversarially augmented models do not have any requirements in terms of annotations, making them as applicable as standard models.

4.2 Evaluation Metrics

We evaluated the 3D object detection performance on the standard **AP**, with a 3D IoU threshold of 0.7 for KITTI and CrashD, 0.5 for Waymo. To measure the quality of the adversarial perturbations for object detection we followed Tu et al. (2020) using the attack success rate (**ASR**) metric. It measures the percentage of objects that become false negatives after undergoing an adversarial alteration. For the ASR, we considered an object detected if its 3D IoU was larger than 0.7. Instead, for 3D semantic segmentation, we evaluated on the common **mIoU** as the mean over the classes IoUs. For the adversarial attacks, we computed the IoU on the adversarial examples generated from the validation set.

4.3 Network Architectures

For 3D semantic segmentation we used Cylinder3D (Zhu et al., 2021a), which divides the point cloud into voxels with distance-dependent sizes. For 3D object detection, we used three different detectors. PointPillars (Lang et al., 2019) voxelizes the scene in vertical columns (i.e., pillars) from the bird's eye view, using PointNet (Qi et al., 2017) for feature extraction. Part-A^2 Net (Shi et al., 2020) is an extension of PointRCNN that predicts intra-object part locations for improved accuracy. VoteNet (Qi et al., 2019) is based on PointNet++ and Hough voting. While PointPillars and Part-A^2 Net are mostly used for autonomous driving settings, VoteNet is used indoor.

4.4 Implementation Details

For the *car* class we constructed each vector field within B_o with $w = 1.8$ m, $h = 1.6$ m, $l = 4.6$ m and a step size of $t = 20$ cm resulting in 1656 vectors per vector field. If not stated otherwise, we grouped objects by relative rotations with $G = 12$ groups, and set $N = 6$. For the *person* class we used axis-aligned boxes with the dimensions $w = 0.54$ m, $h = 1.7$ m, $l = 0.66$ m and a step size of $t = 5$ cm resulting in 5036 vectors per vector field. Due to the ambiguous direction of the axis-aligned boxes, we only use $G = 6$ vector fields, each at its position and the opposite one with respect to the sensor. During the perturbation stage, we moved points according to their $k = 2$ nearest vectors and deformed only along the sensor ray. For the PGD optimization, we used Adam with a learning rate of 0.05 for object detection and 0.01 for semantic segmentation. The distance threshold was set to $\epsilon = 30$ cm and $\psi = 0.3$ for the intensity. Each vector was randomly initialized form a uniform distribution with values between -1 and 1 cm.

For the second stage (i.e., adversarial augmentation, training of the 3D object detection or the 3D semantic segmentation model), we used all the same hyperparameters and configurations provided by the authors of each model, apart from adding our adversarial examples at training time (Sect. 3.2). We trained all models using PyTorch. For 3D object detection we used the MMDetection3D framework (Contributors, 2020), while for 3D semantic segmentation we used the code provided by Zhu et al. (2021a). All models were trained on a single NVIDIA Tesla V100 32GB GPU.

While for object detection we did not use the LiDAR intensity signal, all segmentation models took it in input, unless otherwise noted (e.g., Table 5). Specifically, the intensity values of SemanticKITTI are provided by the authors in the range [0, 1]. Conversely, Waymo intensity values are unbounded, and those of nuScenes are in the range [0, 255]. Since we trained all semantic models on SemanticKITTI, we scaled the intensities of the other datasets to match the same range. For Waymo, we followed Zhu et al. (2021a) adjusting them with *tanh*. For nuScenes, we normalized them to [0, 1].

4.5 Prior Works and Baselines

We compared our work with other adversarial methods, which we used to generate adversarial examples for augmenting the training data. For 3D object detection, all models were applied on PointPillars (Lang et al., 2019), unless otherwise noted. Instead, for 3D semantic segmentation, all were based on the Cylinder3D framework (Zhu et al., 2021a). We selected Cylinder3D thanks to its strong performance on the popular LiDAR semantic segmentation benchmarks (i.e., SemanticKITTI and nuScenes).

To represent point perturbation methods, we used the iterative gradient L2 (Xiang et al., 2019) and the Chamfer attack (Liu et al., 2020). For adversarial generation, we used (Xiang et al., 2019) adding 10%, while for removal we used (Yang et al., 2019) removing 10% of the object points. We applied these adversarial approaches for both 3D object detection and 3D semantic segmentation.

For a fair comparison, all models were trained on the same dataset split: for object detection we used the KITTI split used by Lang et al. (2019); for semantic segmentation we used the standard split of SemanticKITTI (Behley et al., 2019). Moreover, for all approaches we used $\epsilon = 30$ cm, $\psi = 0.3$ and we altered the point clouds as data augmentation with the same settings as for our approach (i.e., random selection of one object per scene to augment).

For 3D semantic segmentation, we compared ours with the domain adaptation method of Yi et al. (2021), which was designed to address the disparities between different LiDAR sensors. Furthermore, for 3D object detection we combined our approach with the domain adaptation statistical normalization (SN) strategy of Wang et al. (2020b). Following them, after computing the average box dimensions in the target datasets (i.e., Waymo and CrashD), we scaled the source (i.e., KITTI) point clouds within the ground truth boxes accordingly and fine-tuned the trained 3D object detector with this altered target-aware source data.

5 Quantitative Results

In this section, first, we discuss the performance of adversarial methods towards out-of-domain generalization (Sect. 5.1), then adversarial methods as attacks (Sect. 5.2) and the trade-off between attack strength and generalization (Sect. 5.3). Furthermore, we explore the benefits of the LiDAR intensity signal (Sect. 5.4), explaining its impact by comparing its distribution across classes and datasets (Sect. 5.5). Then, we assess the robustness of our approach against varying intensity inputs (Sect. 5.6). Moreover, we compare the vectors learned for different tasks and their impact on adversarial augmentation across tasks (Sect. 5.7), we compare our approach with domain adaptation (Sect. 5.8) and combine with it too (Sect. 5.9), as well as with standard data augmentations (Sect. 5.12). Additionally, we discuss the impact of our vectors across different architectures (Sect. 5.10), their benefits in terms of robustness at further distances (Sect. 5.11), and also the effect of different data annotations (Sect. 5.13). Finally, we discuss the impact of our constraints in an ablation study (Sect. 5.14). Additional results can be found in the supplementary material of this work, as well as in our conference publication and its supplementary material (Lehner et al., 2022).

Throughout the results section, we use consistent IDs denoting trained models and vectors, to help associating different experiments and evaluations. Within each line of a table, all results across different datasets are obtained by the same model trained only on KITTI or SemanticKITTI (depending on the task), and identified by the ID, except for the attack performance, which is always computed on the baseline (i.e., PointPillars or Part-A^2 for object detection, and Cylinder3D for semantic segmentation).

5.1 Adversarial Methods and Generalization

5.1.1 Semantic Segmentation

Generalization Data As described by previous works (Wang et al., 2020b; Gasperini et al., 2021b), nuScenes and Waymo feature rather different scenarios compared to those of KITTI or SemanticKITTI. Both transfers are particularly challenging due to the different countries in which the data was captured, leading to different vehicles and street layouts. An additional major challenge is due the different LiDAR sensors used to capture the point clouds. Compared to SemanticKITTI, Waymo has a 50% higher point density and a narrower field of view (Sun et al., 2020), while nuScenes is significantly sparser, especially at further distances. Therefore, evaluating the transfer from SemanticKITTI to Waymo and nuScenes (without fine-tuning) assesses the benefits of the adversarial deformations towards the segmentation of previously unseen and out-of-domain real objects and scenes. In these settings, preserving the plausibility of the perturbations is crucial to improve the generalization of a model by expanding its training data distribution without introducing samples that are too far from it. Our method addresses this via plausibility constraints (e.g., sensor-awareness) and adversarial examples.

Compared Methods Table 1 shows the comparison between our approach and related adversarial methods towards out-of-domain generalization for 3D semantic segmentation. All models are applied on Cylinder3D (Zhu et al., 2021a). In particular, we report other adversarial perturbation methods used as adversarial augmentations, namely the iterative gradient L2 (Xiang et al., 2019) and the Chamfer attack (Liu et al., 2020), adversarial generation (Xiang et al., 2019), as well as adversarial removal (Yang et al., 2019).

Results Although none of the adversarial augmentations improved the in-domain mIoU (i.e., on SemanticKITTI), our approach was the only one to improve upon the baseline on the transfers to both Waymo and nuScenes. Our method applied on the *car* class in an untargeted fashion (u.c) reached the highest mIoU and IoU for *car*, in both Waymo and

Table 1 Comparison of 3D semantic segmentation models trained on SemanticKITTI (Behley et al., 2019) towards out-of-domain data (without any fine-tuning), namely Waymo (Sun et al., 2020) and nuScenes (Caesar et al., 2020) validation sets

ID	Attack	Method	SemanticKITTI				→ Waymo				→ nuScenes			
			mIoU	*car*	*pers.*	*bicyc.*	mIoU	*car*	*pers.*	*bicyc.*	mIoU	*car*	*pers.*	*bicyc.*
c.o	–	Cylinder3D	**64.39**	96.02	73.84	47.15	29.43	72.10	6.69	5.68	29.97	66.04	0.00	0.89
s.l	a.l*	iterative gradient L2*	56.30	77.22	72.45	47.39	24.86	24.48	8.93	1.07	26.15	17.93	0.12	0.25
s.c	a.c*	Chamfer attack*	58.36	83.72	64.94	47.85	27.21	35.48	20.61	4.55	22.40	12.33	0.00	0.20
s.g	a.g*	adversarial generation*	62.94	95.51	**75.87**	47.86	26.70	64.19	5.10	**8.31**	28.05	66.15	0.02	0.67
s.r	a.r*	adversarial removal*	63.43	95.86	74.27	48.56	24.29	70.73	4.44	5.45	29.48	64.22	0.05	0.73
u.c	$N \times$a.u.c	[ours] untargeted *car*	63.85	96.03	77.11	33.41	**37.74**	**82.38**	17.13	5.04	**31.55**	**71.92**	0.06	0.53
t.o	$N \times$a.t.o	[ours] targeted *oth.v.*	64.15	**96.66**	72.88	49.21	30.39	75.70	9.88	3.66	29.57	66.65	0.00	0.00
t.b	$N \times$a.t.b	[ours] targeted *bicycle*	62.75	95.58	74.49	**54.27**	33.43	76.14	**23.12**	7.11	30.06	66.64	0.00	**1.87**
t.m	$N \times$a.t.m	[ours] targeted *motorc.*	60.80	96.41	72.68	41.63	30.28	76.03	20.31	3.28	29.82	68.40	0.00	1.40
u.p	$N \times$a.u.p	[ours] untargeted *pers.*	63.12	96.35	75.82	50.11	33.85	74.47	22.22	4.24	29.44	60.59	**0.17**	0.58

Bold values indicate the best-performing results

Each method applies a data augmentation, and apart from the baselines (first two lines) all others are adversarial augmentation approaches. All models are based on Cylinder3D (Zhu et al., 2021a). Underlined values represent the classes that are directly affected by the adversarial attacks. →: transfer from SemanticKITTI. *: sample-specific. *pers.*: *person*; *bicyc.*: *bicycle*; *oth.v.*: *other-vehicle*; *motorc.*: *motorcycle*

Table 2 Comparison of 3D object detection methods trained on KITTI (Geiger et al., 2012) towards out-of-domain data (without any fine-tuning), namely Waymo validation set (Sun et al., 2020) and our CrashD datasets

ID	Architecture	Method	KITTI				→ Waymo	→ CrashD			
			AP					AP *normal*		AP *rare*	
			easy	*mod.*	*hard*	ASR	AP	*clean*	*crash*	*clean*	*crash*
p.n	PointPillars	no augm.	70.00	61.88	56.23	–	30.68	1.79	0.93	3.92	2.33
p.s		no obj. sampl.	83.83	74.14	68.30	–	37.85	50.36	36.44	28.70	20.02
p.p		PointPillars	**88.24**	77.11	74.55	–	40.86	65.20	43.67	34.14	22.48
p.l		iter. grad. L2 *	86.24	76.92	73.84	*95.9	39.86	58.65	41.86	35.92	23.69
p.c		Chamfer att.*	87.15	77.05	74.07	***99.8**	40.54	56.84	39.56	36.29	24.73
p.g		advers. gener.*	86.12	76.39	73.18	*91.6	40.55	57.75	38.03	35.73	24.18
p.r		advers. remov.*	86.51	76.85	74.04	*86.1	40.32	66.52	48.88	41.42	28.10
p.o		[ours]	87.05	**77.13**	**75.55**	63.4	**44.61**	**67.95**	**52.87**	**43.40**	**30.37**
d.a		SN dom. adapt.	–	–	–	–	49.27	79.42	72.59	60.53	48.23
d.a.o		[ours] + SN	–	–	–	–	**51.32**	**92.14**	**87.28**	**86.26**	**76.42**
p.a	Part-A^2	Part-A^2	89.60	79.16	78.52	–	49.76	83.05	63.25	74.03	52.33
p.a.o		[ours]	**89.65**	**79.26**	**78.62**	50.5	**56.08**	**88.80**	**73.80**	**81.10**	**61.34**

Bold values indicate the best-performing results

Each method applies a different data augmentation (for adversarial ones ASR is measured on their adversarial examples), or performs domain adaptation (only SN (Wang et al., 2020b)), resulting in the reported APs. Only the *car* class is considered. →: transfer from KITTI. *: sample-specific, so adversarial examples are tailored to the samples being evaluated (i.e., validation set)

nuScenes. Specifically, on Waymo, the IoU of our untargeted approach was higher than that of the standard Cylinder3D (c.o) by over 10 (+14%) for *car* and over 8 (+28%) on average on all classes (mIoU). On the sparser point clouds of nuScenes, it was higher by 6 (+9%) on car and 1.6 (+5.3%) on average. This shows the effectiveness of our adversarial augmentations to significantly improve the robustness against challenging out-of-domain data.

The IoU scores obtained by the standard Cylinder3D on nuScenes (Caesar et al., 2020) were particularly low. For *person*, the standard Cylinder3D scored 0.00. This can be attributed to the highly sparse point clouds of nuScenes, which did not have enough points on smaller classes (e.g., *person* and *bicyle*) to be accurately processed by Cylinder3D trained on the denser SemanticKITTI. An additional reason is the different LiDAR intensity distributions (Sect. 5.5). Since the architecture on which every method in the table is based upon (i.e., Cylinder3D) performed so poorly on nuScenes, the performance gains on this dataset were not as significant as on Waymo. Nevertheless, our untargeted *car* configuration (u.c) outperformed the standard Cylinder3D and all other adversarial approaches for both mIoU and the reference class *car*.

Sample-specific perturbation approaches (Chamfer and the iterative gradient L2 attacks), by introducing severely altered samples, which are too far from the existing training data, worsened the generalization performance when used for adversarial augmentation. Instead, adversarial generation and removal are less disruptive, by acting upon fewer points (i.e., 10%). Nevertheless, unlike our method, they also worsened the IoU on the reference class *car*.

Targeted Adversarial Augmentations While our untargeted augmentations (u.c) performed better overall, especially in the transfers to Waymo and nuScenes, targeted ones can be deployed to strengthen a specific class boundary. Depending on the specific use cases, errors may be valued differently for different classes. For an autonomous vehicle, confusing *vegetation* with *trunk* may not be as severe as confusing *car* with *road*, or *car* with *bicycle*. This is linked with the impact that such errors have on downstream tasks (e.g., trajectory prediction and path planning). Therefore, it may be of interest to strengthen specific class boundaries of a model to avoid severe confusions (e.g., *car-bicycle*), even at the cost of weakening a different boundary (e.g., *car-other-vehicle*). As shown in Table 1 our targeted adversarial augmentations offer this valuable option. In the table we report various augmentations based on targeted adversarial attacks on the points of the *car* class towards the corresponding targeted class (e.g., *bicycle*). Among our models, augmenting *car* points while targeting *bicycle* led to the highest IoUs for the *bicycle* class, both in- and out-of-domain, showing the impact of our targeted techniques to alter specific decision boundaries of the model. The other targeted augmentations acted on different decision boundaries, depending on their target class. For example, targeting *other-vehicle* improved the in-domain IoU for *other-vehicle* by 9.3 points over the standard Cylinder3D, reaching 63.8.

Class *Person* Table 1 reports also our untargeted adversarial augmentations for the class *person*. Given that this class has significantly less points than *car* (477K compared to 30.8M on the validation set of SemanticKITTI), both our adversarial vectors and our augmentations operate on a substantially smaller set, which can hinder the effectiveness, compared to *car*. Nevertheless, despite the reduced data available, our adversarial augmentations improved the generalization both in-domain and out-of-domain (Waymo), compared to the standard Cylinder3D. Especially for the *person* class, the IoU improved by 2 points on SemanticKITTI, and increased by a substantial 3.3x on Waymo. These results show the effectiveness of our approach on a class different than *car*, while operating without 3D bounding boxes.

5.1.2 Object Detection

Generalization Data Similarly to Table 1 for 3D semantic segmentation, Table 2 compares our method with other adversarial approaches for 3D object detection. Here, we apply ours and baseline approaches on PointPillars (Lang et al., 2019). As for 3D semantic segmentation, none of the adversarial approaches improved the in-domain AP on KITTI compared to the standard PointPillars. However, our adversarial augmentations significantly improved the performance on out-of-domain data. As previously shown by Wang et al. (2020b), the transfer from KITTI to Waymo is particularly difficult due to the different shapes and sizes of the vehicles, which depend on the country where the data was collected (i.e., Germany and USA, respectively), as well as the different street layouts and urban landscapes. Moreover, the Waymo point clouds are 50% higher in density and its LiDAR sensor has a narrower field of view (Sun et al., 2020). Depending on the global markets, the distribution of vehicle types varies significantly. This is due to certain manufacturers selling different models in different markets, as well as their market penetration, and the preferences and needs of the local population. Therefore, this challenging transfer evaluates the quality of the adversarial perturbations compared to the real vehicles found in different countries.

Results On Waymo, while all other adversarial approaches underperformed the standard PointPillars, our method outperformed it by over 9%. Moreover, our adversarial augmentations brought a 13% improvement on Part-A^2 (Shi et al., 2020). This shows the effectiveness of our approach towards challenging out-of-domain data also for 3D object detection, thanks to the combination of hard examples with plausible deformations. The right columns of Table 2 are dedicated to the results on the proposed CrashD dataset. Despite transferring the models from KITTI, the AP on *clean normal* cars remained relatively high. This can be attributed to the simplicity of such samples. However, the detection performance dropped significantly when the exact same cars at the same locations were damaged (*crash*).

This perfomance gap shows how far damaged cars are from the training distribution of KITTI, making them natural adversarial examples. The gap increased even further with *rare* cars (i.e., old and sports cars), highlighting again the gap from the source domain (i.e., KITTI) and *normal* vehicles. For these reasons, the most difficult samples of CrashD were those combining both out-of-domain aspects (i.e., rarity and damage) into the *rare crash* group. The AP of PointPillars on this group reduced relatively from *normal clean* by 66%.

Despite the challenges, our approach significantly outperformed on all transfers and categories the standard object detectors (i.e., PointPillars and Part-A^2), as well as all other adversarial methods. Removing points (Yang et al., 2019) was the only adversarial method among the baselines

Table 3 Efficacy of adversarial methods as attacks on 3D semantic segmentation

ID	Ad.Cl.	Tar.Cl.	Adversarial Attack	mIoU	*car*	*pers.*	*bicyc.*	*oth.v.*	*motorc.*	*truck*
(c.o)	–	–	(Cylinder3D)	64.39	96.02	73.84	47.15	54.42	69.11	88.96
a.l*	*car*	any	iterative grad. L2 *	**41.16**	**11.18**	70.30	**18.88**	38.10	47.10	**3.30**
a.c*		any	Chamfer attack *	50.92	17.13	73.10	46.56	50.70	52.94	25.88
a.g*		any	advers. generation *	64.06	95.73	73.81	46.96	53.94	67.97	88.23
a.r*		any	advers. removal *	64.38	95.71	73.86	47.13	54.56	69.02	89.11
a.u.c	*car*	any	[ours] untargeted	53.52	62.59	68.98	36.80	16.79	46.47	49.22
a.u.-		any	[ours] untarg.-10%	54.65	66.01	68.06	37.86	18.09	52.02	55.25
a.u.a		any	[ours] untar.ax-alg.	61.29	89.02	71.43	46.16	38.58	65.19	80.38
a.t.b		*bicyc.*	[ours] tar. *bicyc.*	53.89	68.74	67.11	33.11	20.20	45.08	52.07
a.t.o		*oth.v.*	[ours] tar. *oth.v.*	54.61	65.91	68.96	38.10	**16.43**	52.04	53.88
a.t.m		*motorc.*	[ours] tar. *motorc.*	55.62	75.79	68.40	42.07	25.25	**27.46**	61.20
a.t.t		*truck*	[ours] tar. *truck*	55.01	76.19	71.19	39.70	22.89	49.62	40.39
a.u.p	*pers.*	any	[ours] untargeted	60.68	96.01	**30.92**	46.07	54.50	68.55	89.00

Bold values indicate the best-performing results

All IoUs are computed on the predictions of Cylinder3D (c.o) given deformed point clouds of the validation set of SemanticKITTI by means of each adversarial method. Therefore, a lower IoU translates in a more effective adversarial attack. All methods attack both the 3D location of the points and their intensity values. The adversarial class (Ad.Cl.) indicates the class of which points were altered (e.g., *car*). The target class (Tar.Cl.) shows the class that the adversarial methods aimed to switch the predictions of the adversarial class to. Any as Tar.Cl. from *car* (Ad.Cl.) means altering *car* points in an untargeted fashion, while having *truck* as target means perturbing the point clouds such that the model predicts *truck* for *car* points. *: by being sample-specific, the attack had to be tuned on the same samples on which the method is evaluated (i.e., validation set). Underlined numbers highlight the targeted classes, or the untargeted ones

which improved the generalization on CrashD. This could be attributed to its preservation of the overall point clouds, which is an aspect in common with our approach. Nevertheless, removing points adversarially did not improve the generalization to Waymo, which contains denser point clouds and more challenging real scenes. It also did not improve for 3D semantic segmentation.

Similarly to 3D semantic segmentation, the improved generalization can be attributed to the effectiveness of our adversarial augmentations to expand the training data distribution with difficult and plausible examples. Thanks to the added diversity, our method managed to mitigate the domain gap, as shown by the results in the tables. Overall, the results across Tables 1 and 2 demonstrate the effectiveness of our adversarial augmentations on two highly different tasks, namely 3D semantic segmentation and 3D object detection.

5.2 Adversarial Methods as Attacks

In Table 3 we compare the effectiveness of the methods as adversarial attacks for 3D semantic segmentation, in terms of the change in IoU when using their adversarial examples as input for Cylinder3D compared to the untouched inputs.

5.2.1 Semantic Segmentation, Untargeted Attacks

Applied on cars, the iterative gradient L2 (Xiang et al., 2019) and the Chamfer attack (Liu et al., 2020) were able to majorly reduce the IoU on *car* (Table 3). Being sample-specific, their deformations had to be learned directly on the data on which they are applied, and in this case also evaluated (i.e., validation set of SemanticKITTI). The same holds true for adversarial generation (Xiang et al., 2019) and removal (Yang et al., 2019), except that they were not effective attacks as they could not reduce the *car* IoU. This is due to semantic segmentation being a dense task, requiring a prediction for each point. Therefore, removing or adding a few critical points is not enough to lower the IoU as all points count equally towards the metric. In comparison, considering 3D object detection, few points can have a larger impact as they can shift the predicted bounding box (Table 2), thereby reducing the IoU with the ground truth box until it is below the threshold for the AP (e.g., 0.7). Being sample-independent and learned on the training set, the proposed method is not an attack as effective as the iterative gradient L2 or the Chamfer attacks at reducing the *car* IoU. Nevertheless, our untargeted attack (a.u.c) lowered the *car* IoU by over 33 points. In fact, the proposed method is not meant to render the objects unrecognizable. Instead, we aim to perturb them while preserving their overall shape. As shown in our conference publication (Lehner et al., 2022), strong attacks do not lead to improved generalization, which is the aim of this work. Furthermore, in Table 3 we also report the results of our untargeted attack on the *person* class (a.u.p). This effectively lowered the IoU on *person* by 43 points and did not change the *car* IoU with respect to Cylinder3D (c.o). Comparing the *person* attack with the untargeted attack on *car* (a.u.c) shows the effectiveness of our approach, also without using 3D bounding boxes (Sect. 3.2.1).

5.2.2 Semantic Segmentation, Targeted Attacks

In this work, we explored also targeted adversarial attacks and augmentations. The purpose of the targeted attacks in this context is to test a specific class boundary, rather than limiting the attack to the weakest boundary (i.e., untargeted). We then strengthen this boundary with our targeted adversarial augmentations. With reference to Table 3, all our targeted attacks not only correctly reduced the *car* IoU, but also that of their respective targeted class (e.g., *motorcycle*). Specifically, each targeted attack reached the lowest IoU in its class (underlined) among all our attacks, showing their effectiveness. Our untargeted attack often aimed at confusing *car* with *other-vehicle* and *motorcycle*, both decreasing significantly, while not as much with *person*. Therefore, the IoU gap between targeting *other-vehicle* (a.t.o) and not targeting any specific class (untargeted) is rather small for *other-vehicle*. This can be attributed to the *car-other-vehicle* decision boundary being weaker than others for the Cylinder3D model examined. However, comparing the performance of Cylinder3D on the standard data (c.o), the perturbed point clouds untargeted (a.u.c) and the targeted ones on *bicycle*, *motorcycle*, or *truck* shows a clear difference on the targeted classes, especially for *motorcycle*.

5.2.3 Object Detection, Untargeted Attacks

As seen for semantic segmentation, also for object detection our approach is not an attack as strong as the sample-specific ones. This is shown in terms of ASR in Table 2, where we compared our sample-independent vector fields to their sample-specific point-to-point deformations. As for semantic segmentation, their perturbations were trained directly on the validation set of KITTI, on which the ASR was evaluated. Nevertheless, a very high score on ASR means that the objects became unrecognizable. This goes against aiding generalization, which is the goal of our method.

5.2.4 Considerations on Attack Efficacy Versus Generalization

Unlike adversarial approaches designed as attacks, we do not aim to have the model fully miss our adversarial examples (high ASR). Instead, we want to deform them to increase the robustness on out-of-domain data. Therefore, the adversarial examples need to be altered enough to expand the training data distribution, but not too far from it to prevent confusion for the model. We balance these aspects via sample-independent adversarial perturbations and constraining the deformations. The adversarial attack generates hard examples, while the sample-independence and the constraints (e.g., sensor-awareness) mitigate their strength and preserve their plausibility. As demonstrated throughout this work, this combination is effective to improve generalization and robustness.

5.2.5 Impact of Data Annotations

In Table 3, we explore the impact of the availability of data annotations on our adversarial attack. We explore two different strategies: an off-the-shelf 3D object detector deployed on the training data, and point-level instance annotations. In the latter case, we create axis-aligned bounding boxes around the 3D points constituting an instance. The axis-aligned one prevents the vector fields to specialize on certain object orientations, since such orientation is unavailable, causing sub-optimal performance (a.u.a) compared to using an off-the-shelf detector (a.u.c). In the table, we also assess the effect of the quality of the detector's predictions on our attack. We do so by randomly ignoring 10% of its bounding boxes. As this impacts directly the performance on the *car* class, the attack is not as effective (a.u.-), whilst managing to significantly reduce the IoU on *car* by 30 IoU. In Table 10, we show the impact of this on generalization.

5.3 Specificity-Generalization Trade-Off

As seen in Sects. 5.1 and 5.2, strong adversarial examples do not necessarily translate into successful adversarial augmentations. This is because if the attack is too strong (e.g., Chamfer), the generated samples are too far from the existing data distribution, which causes the model to learn objects that are not useful towards the task at hand, thereby degrading the performance. Such extreme transformation can be seen in Fig. 10. Therefore, there is a trade-off between an attack strength and its benefits to improve generalization via adversarial augmentation. While this trade-off is evident for sample-specific approaches (Sects. 5.1, 5.2), it is less trivial how it can arise in sample-independent settings, such as ours, where the attack strength is inherently mitigated by not being sample-specific. Towards this end, in this section we explore this trade-off in the context of our method, as we purposely make our attack overfit on fewer samples.

Object Detection Table 4 shows that by varying the amount of considered relative rotations G, a trade-off arises between generalization, attack specificity (i.e., attack strength on individual samples by overfitting to the training data), and storage (i.e., amount of vectors). $G = 12$ offers a good balance. Instead, taking G to the extreme and having one vector field for each instance of the dataset, ours would become sample-specific, inheriting the weaker generalization capabilities of prior works (Liu et al., 2020; Yang et al., 2019), as shown already at $G = 360$. However, while the

Table 4 Our adversarial augmentation method for 3D object detection applied on PointPillars (Lang et al., 2019) trained on KITTI with varying amounts of relative rotations G

ID	# G	KITTI		→ Waymo	# vectors
		ASR ↑	*mod.*	AP	
p.u	1	55.08	**77.32**	40.43	**10K**
p.o	12	**63.37**	77.13	**44.61**	120K
p.f	360	44.84	77.06	40.30	3.6M

Bold values indicate the best-performing results

→: transfer without fine-tuning

Table 5 Impact of the LiDAR intensity signals on 3D semantic segmentation models trained on SemanticKITTI, both in-domain and when transferring (without fine-tuning) on out-of-domain data

ID	Method	SemK attack		SemanticKITTI				→ Waymo				→ nuScenes			
		mIoU	*car*	mIoU	*car*	*pers.*	*bike*	mIoU	*car*	*pers.*	*bike*	mIoU	*car*	*pers.*	*bike*
c.n	Cylinder3D w/o intensity	–	–	59.23	**96.23**	68.67	**32.61**	37.39	63.74	37.95	4.05	32.11	56.17	0.04	0.35
u.n	[ours] w/o intensity	56.63	86.03	**59.40**	95.73	**70.35**	26.53	**40.36**	**70.70**	**52.73**	**7.57**	**34.30**	**63.13**	**1.02**	**2.52**
c.o	Cylinder3D w/ intensity	–	–	**64.39**	96.02	73.84	**47.15**	29.43	72.10	6.69	**5.68**	29.97	66.04	0.00	**0.89**
u.c	[ours] w/ intensity	**53.25**	**62.61**	63.85	**96.03**	**77.11**	33.41	**37.74**	**82.38**	**17.13**	5.04	**31.55**	**71.92**	**0.06**	0.53

Bold values indicate the best-performing results

All models are based on Cylinder3D (Zhu et al., 2021a)

sample-specific methods needed to be trained on the validation set, allowing for high ASRs (Table 2), our vectors were learned on the training set. So with high G, ours overfitted on the training data, which emerges by evaluating on the validation set. Our augmentation strategy learns only 1656 3D vectors to perturb objects. However, by training with $G = 12$ and $N = 6$ (p.o, our standard configuration), the amount of vectors increased to 120K. Conversely, the sample-specific iterative gradient L2 (Xiang et al., 2019) and the Chamfer (Liu et al., 2020) attacks required 10.9M and 12.6M vectors for training and validation sets respectively. This shows the easy applicability of our approach.

5.4 Impact of LiDAR Intensity

5.4.1 Effect In-Domain

In Table 5 we assess the impact of using the LiDAR intensity (also called reflectivity) as extra input for each 3D point. Removing the intensity significantly reduced the mIoU and the IoU on certain classes, such as *person* and *bicycle*, but not on *car*. On Waymo, models not using intensity outperformed the ones relying on intensity by up to 8 mIoU points for the baseline Cylinder3D (Zhu et al., 2021a). This gap can be attributed to the different sensors used to captured SemanticKITTI and Waymo, which implies different reflectivity measurements. Therefore, for this transfer, not taking the intensity as extra input is advantageous. Nevertheless, our adversarial augmentation significantly improved the predictions both with or without intensity, especially on *car* and *person*.

5.4.2 Effect on Generalization and Robustness

A trade-off exists between in- and out-of-domain which depends on the use of the LiDAR intensity values (Table 5). Since the intensity is a valuable input signal, the in-domain mIoU is higher when integrating it. However, it is better to avoid using it when transferring to different sensors. Most 3D semantic segmentation experiments in this work used the intensity, because using it maximizes the outcome on the data distribution which is available at the time of development (i.e., in-domain, SemanticKITTI). Instead, we consider the transfer datasets (i.e., out-of-domain, Waymo, nuScenes) as real-world test scenarios, which include data that is not available during the development and training of the method. In realistic scenarios it is not always known on which exact data a model will be applied on.

While different intensity values would arise only by deploying the model on data captured by a different LiDAR sensor, it is possible for example that a trained model gets deployed on data from the fleet of vehicles of a newer generation which includes a different sensor configuration. Therefore, robustness against varying intensity values would mitigate the domain gap introduced by the different sensor and potentially avoid to collect the data again with the newer sensor, simplifying the process.

By generating adversarial examples which alter not only the 3D points (Lehner et al., 2022; Xiang et al., 2019; Liu et al., 2020; Tu et al., 2020), but also the LiDAR intensity values, our adversarial augmentation approach increases the robustness of the model and makes it less reliant on it. As

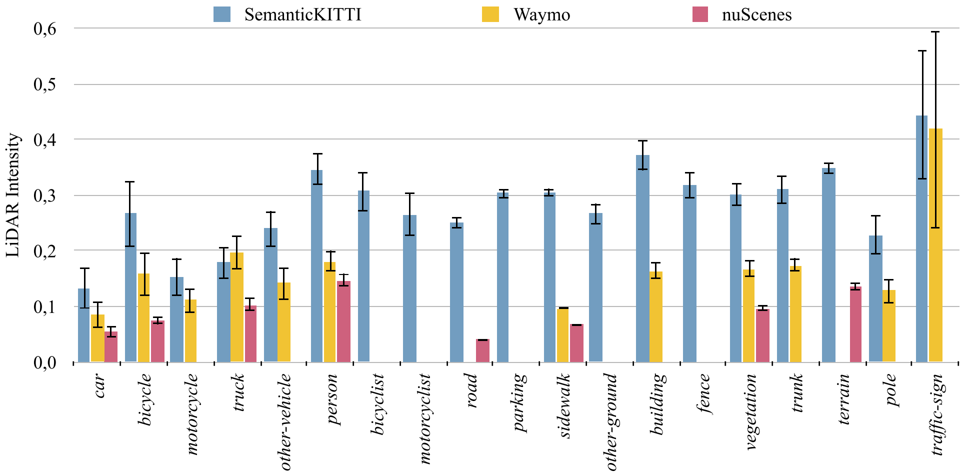


Fig. 6 Plot of the mean LiDAR intensity values for each semantic class (x-axis) across the validation sets of the three semantic datasets used, namely SemanticKITTI (Behley et al., 2019), Waymo (Sun et al., 2020), and nuScenes (Caesar et al., 2020). Each error bar represents the standard deviation within that class and dataset. The classes naming and definition are aligned with those of SemanticKITTI (i.e., 19 classes). Not all classes are represented for the other datasets, due to the non-overlapping definitions with SemanticKITTI (e.g., *building* has no dedicated class in nuScenes). Therefore, the naming follows the convention of SemanticKITTI

shown in Table 5, adding the intensity does not degrade the IoUs of ours as much as for the standard Cylinder3D, thereby effectively mitigating the sensor change.

5.4.3 Effect on the Attacks

Comparing the attack performances with and without the intensity (SemK attack in Table 5), confirms that the LiDAR intensity plays a major role for Cylinder3D. On the reference *car* class, the purely geometrical attack without intensity generated adversarial examples which were not particularly challenging for the model, as shown by the relatively high IoU on *car* after the attack (SemK attack). Nevertheless, using the geometrical 3D vectors that generated these examples for augmentations significantly improved the performance on out-of-domain data. On Waymo, our approach increased the *car* IoU by 11% without intensity (3D vectors), and 14% with intensity (4D vectors). On nuScenes, by 12% without, and 8% with intensity. In fact, as described in Sect. 5.1, weaker attacks (e.g., without intensity) can still lead to valuable augmentations.

5.5 LiDAR Intensity Distributions

Comparing the distributions of the intensity values for each class across the three datasets shows a significant difference. We report this in Fig. 6.

5.5.1 Benefits

First, by looking at the intensity values of SemanticKITTI (Fig. 6), it is evident how useful the intensity signal can be to identify the semantic classes. For SemanticKITTI, the distribution of the class *car* is significantly different from that of *building*, making it easier to separate the two classes. Nevertheless, it should be noted that the LiDAR intensity changes significantly depending on the distance. For SemanticKITTI, between 50 and 60 ms, the mean intensity of *building* is the same as the overall mean of *car*.

5.5.2 Different Sensors

While for the most part in SemanticKITTI increasing the distance causes a reduction of the LiDAR intensity values (Fig. 6), this is not the case for Waymo and nuScenes, where the reflectivity is larger at higher distances. This, together with the misaligned distributions shown in Fig. 6, makes it even more challenging to transfer from SemanticKITTI to the other datasets when taking the intensity as input.

5.5.3 Different Ways to Treat the Values

Only Waymo provides the raw intensity signals, while SemanticKITTI and nuScenes provide them already scaled, within [0,1] and [0,255] respectively. It is unclear how the scaled values were obtained for SemanticKITTI and nuScenes. To mitigate the impact of the pre-processing of each dataset, we further scale the nuScenes intensity values to [0,1], and apply *tanh* to those of Waymo to constrain them within [0,1] as well. The resulting distributions are shown in Fig. 6.

5.5.4 Effect on Our Attacks

Our adversarial augmentations perturb also the intensity signals. As we limited the maximum shift of intensity to 0.3, in theory this allowed to transition from the mean *car* value (lowest for SemanticKITTI), to the mean *traffic-sign* value (highest). However, 0.3 prevented from fully representing the the intensity distribution of *traffic-sign* starting from *car*. Therefore, while an untargeted attack on *car* could focus on weaker boundaries (both semantically and in terms of intensity values), such as *truck* and *other-vehicle*, not all targeted attacks can be very effective due to the intensity shift. For example, *car* and *traffic-sign* are rather far (both semantically and in terms of intensity values), rendering such targeted attack more difficult to achieve. For these reasons, we focused our targeted attacks on traffic participants other than *car*, such as *other-vehicle* and *bicycle*.

5.6 Robustness to Changing LiDAR Intensity Values

In Table 6 we assess the increase of robustness by our adversarial augmentations when changing the LiDAR intensity values, while leaving the 3D points unchanged. We do so by comparing our method with Cylinder3D and applying various transformations to the intensity values in input. We report such results both in-domain (i.e., SemanticKITTI) and out-of-domain (i.e., Waymo). The models evaluated in the table were trained only on SemanticKITTI, and our adversarial augmentations were applied only on *car* points in an untargeted fashion, leaving all the other points unchanged compared to the baseline, which also applied standard augmentations (e.g., rotation).

5.6.1 Impact of Using the LiDAR Intensity

As discussed, the intensity is a powerful input signal, which can help distinguishing the semantic classes. This is why models trained with it are particularly sensitive to changes of its values, which caused the transfer on Waymo to be more effective when training without intensity (Table 6). The table shows how much Cylinder3D relies on the reflectivity value, when trained both with and without our adversarial augmentations. Our method outperforms the standard Cylinder3D across all changes applied, proving the added robustness of our augmentations. The gap is most extreme in the uniform random case ranging from 0 to 1 (all possible intensity values): on SemanticKITTI our method achieved a 6.2x higher IoU on *car* (70.0 compared to 11.3) and 2.5x higher mIoU (21.6–8.7); on Waymo ours reached a 76x higher IoU on *car* (68.8–0.9) and 4x higher mIoU (23.1–5.8). However, it is the complete lack of intensity values (all 0) that shows how much each model relies on the intensity signal after training with it. In this case, for the reference class *car*, the IoU of our approach dropped only by 13 for SemanticKITTI and 7 for Waymo, compared to the IoU of Cylinder3D dropping by 73 and 19 respectively.

5.6.2 Benefits of Our Method

Similarly to the improved robustness on different vehicle shapes thanks to geometrical perturbations (e.g., rare and damaged, with CrashD and Waymo for 3D object detection), the reason for the significant improvements shown in Table 6 is that, when augmenting, our method alters also the intensity values in an adversarial fashion. Our augmented model learns to be less reliant on the reflectivity input, allowing it to generalize better to different values (e.g., uniform random). On the reference class *car*, despite the extreme intensity changes applied in input, our model always reached in- or out-of-domain a remarkable IoU of at least 68.82 (lowest reached with uniform random [0,1], on Waymo). Conversely, Cylinder3D trained without our adversarial augmentations was unable to deliver satisfactory predictions, with the *car* IoU dropping as low as 0.91 (lowest reached with uniform random [0,1], on Waymo).

The benefits of our approach are evident also considering the in-domain performance alone (Table 6). On the standard SemanticKITTI (none), the baseline achieved a higher mIoU than our method. However, any transformation applied to the intensity values introduced major changes, with ours always outperforming Cylinder3D by significant margins.

5.6.3 Side-Effect on Other Classes

Despite our adversarial modifications were only applied on *car* points in an untargeted way, as shown in Table 6 they had a major effect on other classes as well, such as for *person* and *bicycle*. For *person*, our method improved the IoU by up to 40x on Waymo (reached with uniform random [0,1]) and by up to 11x on SemanticKITTI (reached with all 0). Analogously, for *bicycle*, despite performing worse than Cylinder3D both in- and out-of-domain without modifying the intensity values in input (none), our approach improved the IoU by up to 7x on SemanticKITTI (reached with Gaussian) and 3.3x on Waymo (reached with uniform random [0,1]) when altering the intensity values.

Table 6 Detailed impact of the LiDAR intensity as extra input on 3D semantic segmentation

LiDAR Intensity Transformation	Dataset	ID	Method	mIoU	*car*	*pers.*	*bike*
none	SemanticKITTI	c.o	Cylinder3D	**64.39**	96.02	73.84	**47.15**
		u.c	[ours]	63.85	**96.03**	**77.11**	33.41
	→ Waymo	c.o	Cylinder3D	29.43	72.10	6.69	**5.68**
		u.c	[ours]	**37.74**	**82.38**	**17.13**	5.04
all 0	SemanticKITTI	c.o	Cylinder3D	12.27	23.15	0.61	0.01
		u.c	[ours]	**31.09**	**83.13**	**6.77**	**0.25**
	→ Waymo	c.o	Cylinder3D	22.75	53.19	0.05	**0.15**
		u.c	[ours]	**32.43**	**75.43**	**1.49**	0.01
Gaussian noise (std: 0.3)	SemanticKITTI	c.o	Cylinder3D	15.59	61.91	26.84	1.61
		u.c	[ours]	**32.51**	**77.98**	**35.06**	**11.50**
	→ Waymo	c.o	Cylinder3D	21.64	62.18	13.66	6.16
		u.c	[ours]	**34.22**	**70.01**	**35.77**	**6.77**
uniform random [0, 1]	SemanticKITTI	c.o	Cylinder3D	8.73	11.33	**20.53**	2.24
		u.c	[ours]	**21.61**	**69.98**	8.89	**7.91**
	→ Waymo	c.o	Cylinder3D	5.76	0.91	0.31	0.59
		u.c	[ours]	**23.10**	**68.82**	**12.50**	**1.95**
uniform random noise +[0, +0.3]	SemanticKITTI	c.o	Cylinder3D	28.21	89.40	35.98	**24.30**
		u.c	[ours]	**44.88**	**94.76**	**49.16**	23.52
	→ Waymo	c.o	Cylinder3D	35.36	65.39	37.75	12.29
		u.c	[ours]	**42.31**	**80.64**	**48.48**	**12.36**
uniform random noise +[−0.3, +0.3]	SemanticKITTI	c.o	Cylinder3D	41.94	92.89	35.30	10.21
		u.c	[ours]	**54.21**	**95.09**	**68.93**	**27.63**
	→ Waymo	c.o	Cylinder3D	32.11	70.45	19.85	8.13
		u.c	[ours]	**41.02**	**81.09**	**36.06**	**8.36**
random shift ±0.3	SemanticKITTI	c.o	Cylinder3D	17.54	71.99	27.36	1.62
		u.c	[ours]	**35.93**	**81.06**	**40.58**	**10.95**
	→ Waymo	c.o	Cylinder3D	27.05	61.96	33.64	**8.12**
		u.c	[ours]	**38.71**	**76.21**	**46.83**	6.92

Bold values indicate the best-performing results for each transformation and Dataset

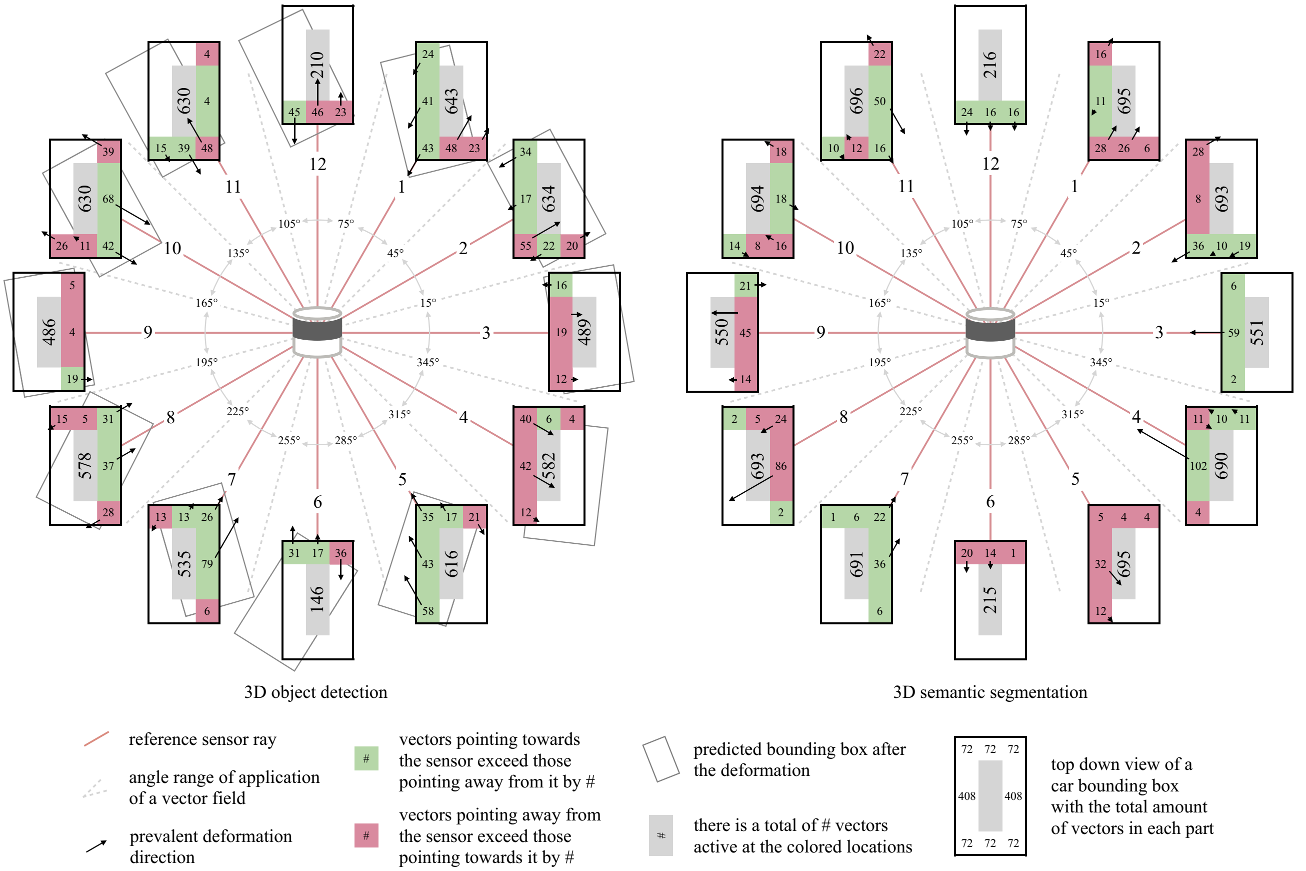
Various transformations are applied to the intensity values (first column), leading to the reported IoUs. All models are trained on SemanticKITTI and based on the Cylinder3D architecture (Zhu et al., 2021a), with [ours] trained with our untargeted adversarial augmentations only on *car* points, also for the intensity values (restricted to a maximum perturbation of 0.3). Across the various datasets and input configurations, a total of only two models is evaluated in this table: one for the standard Cylinder3D (c.o), and one trained with our adversarial augmentation method (u.c)

5.6.4 In-Domain and Out-of-Domain

Furthermore, while the mIoU gap between in- and out-of-domain is relatively large without modifying the intensity values, this gap shrinks and even inverts when applying many of the modifications shown in Table 6. With Gaussian noise, both Cylinder3D and our approach achieved a higher mIoU on Waymo than SemanticKITTI, albeit on a different number of classes. Apart from the constrained uniform random noises, all other modifications made our method achieve a higher mIoU on Waymo than SemanticKITTI. This can be attributed to the denser point cloud in Waymo compared to SemanticKITTI, which allows to extract better geometrical features when LiDAR intensity values become unreliable due to the changes.

5.6.5 Reducing the Domain Gap

Interestingly, in Table 6 there are random modifications that significantly improved the transfers to Waymo in terms of mIoU (up to +6, compared to the unmodified intensity inputs). This was the case of the constrained uniform random noise settings (+[0, +0.3] and +[−0.3, +0.3]), for both Cylinder3D and our model. This improvement can be attributed to the transformation bringing the intensity distribution of Waymo closer to that of the standard SemanticKITTI. Therefore, the random transformation


12
210
45 46 23
11
630
4
4
15 39 48
1
643
24
41
43 48 23
10
630
39
68
26 11 42
2
634
34
17
55 22 20
9
486
5
4
19
3
489
16
19
12
8
578
15 5 31
37
28
4
582
40 6 4
42
12
7
535
13 13 26
79
6
5
616
35 17 21
43
58
6
146
31 17 36
105°
75°
135°
45°
165°
15°
195°
345°
225°
315°
255°
285°
216
24 16 16
696
22
50
10 12 16
695
16
11
28 26 6
694
18
18
14 8 16
693
28
8
36 10 19
550
21
45
14
551
6
59
2
693
2 5 24
86
2
690
11 10 11
102
4
691
1 6 22
36
6
695
5 4 4
32
12
215
20 14 1
reference sensor ray
angle range of application of a vector field
prevalent deformation direction
#
vectors pointing towards the sensor exceed those pointing away from it by #
#
vectors pointing away from the sensor exceed those pointing towards it by #
predicted bounding box after the deformation
#
there is a total of # vectors active at the colored locations
72 72 72
408
408
72 72 72
top down view of a car bounding box with the total amount of vectors in each part

3D object detection

3D semantic segmentation

Fig. 7 Comparison of adversarial vector fields learned for 3D object detection (left) and 3D semantic segmentation (right). For each task, $G = 12$ vector fields are displayed according to their relative rotation from the sensor. All are trained to reduce the performance on the *car* class. The color in each area indicates the prevalent direction of deformation: green means that the majority of vectors points towards the sensor, while red means away from it. This is emphasized by the black arrows, which are aligned to the sensor rays and have a magnitude proportionate to the number inside the corresponding colored areas. This number indicates how many more vectors point in that direction compared to those facing the opposite one, after being projected along the sensor view ray (those with magnitudes lower than their random initialization are ignored). Object detection vectors often aim at rotating the predicted bounding box, which is represented in grey as an estimation, thereby reducing the AP. They achieve this by pushing one corner of the car and pulling the opposite one. Instead, segmentation vectors do not exhibit this behavior, because rotating the objects does not reduce the semantic IoU. In comparison, they need to alter more points, which makes them significantly more active, as shown by the number in the middle of each vector field

reduces the intensity gap shown in Fig. 6 between the two datasets, as Waymo typically displayed a lower intensity compared to SemanticKITTI. For the same reasons, ours benefitted also from the random shift ±0.3 in terms of mIoU.

5.7 Cross-Task Vectors Comparison

Figure 7 shows an analysis of the vector fields learned for each of the two tasks. In this section we discuss the effects of the tasks on the vector fields.

5.7.1 Object Detection

For 3D object detection, the adversarial loss aims at reducing the confidence and the IoU of the predicted box with respect to the ground truth box. The vectors can achieve this by simply making the detector believe that the object is rotated. Specifically, as shown in Fig. 7, the vectors learned to pull a corner of the car towards the sensor (green) and push the opposite one away (red). This changes the orientation of the surface on which the detector bases its regression of the rotation. Therefore, the detector predicts a rotated box, which significantly reduces the IoU. This affects the AP as the amount of boxes exceeding the IoU threshold decreases. This behavior is visible across most of the $G = 12$ relative positions, particularly evident at 6 and 12. All other positions also exhibit the same actions, apart from 4. At 4, all corners push the points away. Nevertheless, this also reduces the box IoU as it shifts all points in the same direction.

5.7.2 Semantic Segmentation

Interestingly, this rotatory pattern is mostly absent on the vector fields learned for 3D semantic segmentation, which

Table 7 Comparison on 3D semantic segmentation of adversarial vectors learned for 3D object detection and 3D semantic segmentation

ID	Method	SemK attack		SemanticKITTI				→ Waymo				→ nuScenes			
		mIoU	*car*	mIoU	*car*	*pers.*	*bike*	mIoU	*car*	*pers.*	*bike*	mIoU	*car*	*pers.*	*bike*
c.n	Cylinder3D	–	–	59.23	**96.23**	68.67	**32.61**	37.39	63.74	37.95	4.05	32.11	56.17	0.04	0.35
u.d	[ours] for obj.det.	58.54	94.47	58.43	95.73	**70.90**	17.19	36.28	**70.76**	30.71	5.99	32.27	62.49	0.09	1.83
u.n	[ours] for sem.seg.	**56.63**	**86.03**	**59.40**	95.73	70.35	26.53	**40.36**	70.70	**52.73**	**7.57**	**34.30**	**63.13**	**1.02**	**2.52**

Bold values indicate the best-performing results

All models are based on Cylinder3D (Zhu et al., 2021a) and do not use the LiDAR intensity

Table 8 Comparison on 3D object detection of adversarial vectors learned for 3D object detection (Lehner et al., 2022) and 3D semantic segmentation

ID	Method	KITTI				→ Waymo	→ CrashD			
		AP					AP *normal*		AP *rare*	
		easy	*mod.*	*hard*	ASR	AP	*clean*	*crash*	*clean*	*crash*
p.p	PointPillars	**88.24**	77.11	74.55	–	40.86	65.20	43.67	34.14	22.48
p.e	[ours] for sem.seg.	86.88	75.98	68.73	17.2	40.70	59.20	42.82	39.23	28.15
p.o	[ours] for obj.det.	87.05	**77.13**	**75.55**	**63.4**	**44.61**	**67.95**	**52.87**	**43.40**	**30.37**

Bold values indicate the best-performing results

All models are based on PointPillars (Lang et al., 2019)

exhibit a rather different behavior (Fig. 7). This can be explained by considering the two different tasks, adversarial losses and metrics involved. For object detection, the metric is based on the IoU between the predicted box and the corresponding ground truth. AP is relatively discrete, as it is computed by counting positive and negative predictions. Moreover, an object is considered missed if the IoU is lower than a certain threshold (e.g., 0.7).

5.7.3 AP Versus IoU

While pretending to rotate the object can be detrimental for the AP and lead to a successful adversarial attack for object detection, it is not as useful for semantic segmentation. Semantic segmentation is a dense task, demanding a prediction for each point. Every prediction and every point count towards the IoU, which makes it rather smooth compared to the AP. Therefore, to change the predicted class of a 3D point, the vectors here cannot simply rotate the object, but need to make it look as if it belonged to a different class. This makes it significantly harder for semantic segmentation compared to object detection.

5.7.4 Number of Activated Vectors

Semantic segmentation activated more vectors than object detection, because it required to move more points to fool the model and reduce the IoU. This is shown in Fig. 7 through the values in the middle of each position. For semantic segmentation, a maximum of only 6 vectors within a vector field remained inactive (reached at position 4: 690 out of 696), for a total of 7079 active vectors out of 7104. Conversely, for object detection up to 118 vectors remained inactive (reached at position 7: 578 out of 696), for a total of 6179 active out of 7104. Therefore, semantic segmentation activated 900 more vectors, equivalent to 11% for the 12 positions, or a relative 15% increase over object detection. For this analysis, as highlighted in the figure by the colors, we considered visible vectors from the sensor, and we regarded as inactive those having a magnitude lower than that of their random initialization.

5.7.5 Impact of *G* on the Vectors

Figure 7 visually explains also the reasons why with $G = 1$, i.e., using 1 vector field for the entire dataset, the vectors can reach only sub-optimal results (Table 4). Although with $G = 1$ the vectors can still learn to give the illusion of a rotated object (i.e., having all corners alternating between pulling and pushing the points), they cannot achieve the diversity and specificity shown in the figure with $G = 12$, and they cannot be as directed as those with $G = 12$. By being effective in all positions and distances, the vector field with $G = 1$ needs to contain vectors pointing in all directions. This limits their efficacy as their magnitude decreases when projected to the sensor view rays. Therefore, the shifts and rotations applied with $G = 1$ cannot be as strong as those with $G = 12$ at all positions.

5.7.6 Transferring the Attack to a Different Task

The difference between the vector fields learned for the two tasks is shown also in Tables 7 and 8. Interestingly, augmenting with the vectors learned for object detection (Lehner et al., 2022) when training a semantic segmentation model delivered strong predictions for the reference class *car*, performing on par or even better than the semantic vectors, also when transferring to Waymo (Table 7). Although the detection vectors lowered the mIoU, this shows the flexibility of our approach even across tasks. Augmenting PointPillars with the vectors learned for segmentation underperformed the baseline on *car*, while they significantly improved the AP on the challenging *rare* samples of CrashD (Table 8).

This difference between the vectors learned for the two tasks could be attributed to the application of the detection vectors resulting in pseudo rotated cars, instead of the semantic vectors trying to resemble a different class. Augmenting with the detection vectors, thanks to the pseudo rotatory behavior shown in Fig. 7, is likely to improve the regression of the bounding box rotation. Since estimating the rotation correctly is crucial to exceed the IoU threshold with the ground truth box, improving the regression of the rotation translates in better box IoU on challenging cases (e.g., AP on Waymo in Table 8). Instead, by not exhibiting that rotatory behavior, the semantic vectors are likely to not improve the rotation regression (Table 8). On the proposed CrashD, the semantic vectors underperformed the detection baseline on *normal* cars, but significantly outperformed it on difficult *rare* ones, resembling long tail samples, both damaged and undamaged.

Detection Vectors for Segmentation As described in Sect. 5.3, for our adversarial augmentation method the ability to generalize is related to the performance of the adversarial attacks on which the augmentation is based upon. The detection vectors on the semantic model delivered poor attack performance (SemK attack), with only a slight decrease in IoU (Table 7). This means that the examples generated by perturbing *car* points with the detection vectors were not challenging for the semantic model (i.e., Cylinder3D). Although this would hint towards the futility of these vectors in semantic settings, using them as augmentation delivers significant improvements on *car* segments on both Waymo and nuScenes. Therefore, despite being weak as attacks, the detection vectors successfully regularized the training of Cylinder3D by augmenting the training data beyond standard augmentations.

Segmentation Vectors for Detection The semantic vectors delivered a stronger cross-task attack, with an ASR of 17.2 against PointPillars (Table 8). This can be attributed to the semantic ones trying to shift the points to resemble a different class, instead of rotating the object. As the object detector has no explicit semantic understanding beyond its reference class *car*, augmenting with the semantic vectors probably expanded the training data in a direction which is not beneficial for the detector, causing confusion. Nevertheless, optimizing the adversarial vectors for each task independently leads to more significant deformations targeted around the weaknesses of the respective model and task combination, allowing for more meaningful adversarial augmentations, which lead to better overall performance (mIoU and AP).

5.7.7 Considerations on Adversarial Augmentation

The essence of data augmentation is enriching the training data. Whether adversarial (e.g., semantic vectors in Table 7), or not (e.g., detection vectors, which were learned for a different task and model, so they lost their adversarial properties), data augmentation is crucial for domain generalization, to tackle challenging out-of-domain samples. In particular, we believe that data augmentation is most effective when it manages to push the boundaries of the training distribution towards corner cases which are not present in the original training set.

While this expansion could be easily achieved via extreme transformations of the input, it is not necessarily beneficial for the model and may even harm its performance, as seen for the sample-specific adversarial approaches (Sect. 1). The difficulty is preserving the usefulness of the augmented samples towards the task at hand and real-world data. This is where our plausibility constraints (Sect. 3.2.2) play a fundamental role. Together with our generated examples, which represent hard cases by being adversarial, the constraints determine a good balance between expanding the training data and preserving plausibility of the new samples. When using the vectors across different tasks, the plausibility is still preserved as all constraints are applied, but the generated examples do not represent hard cases as the vectors were optimized for a different task. This ultimately reduces the degree of expansion of the training distribution given by the augmentations, compared to using purposed vectors optimized for the same task.

5.8 Comparison with Domain Adaptation

Our approach does not use any information about the target data (e.g., Waymo and nuScenes), making it a domain generalization method. Conversely, domain adaptation bridges the domain gap by exploiting knowledge about the target data, such as the average object size (Wang et al., 2020b).

Semantic Segmentation In Table 9 we show a comparison with the domain adaptation work for LiDAR point clouds of Yi et al. (2021), who exploited the different sensors used to captured the data. Compared to all other experiments, where we reported on all matching classes, in these experiments we use the setup of Yi et al. (2021), who used 2 classes for Waymo, namely *person* and the super-class *vehicle*, and 10 for nuScenes. However, while we trained our models on all 19 classes of SemanticKITTI, Yi et al. (2021) trained only on the 10 classes they considered for nuScenes, making it

Table 9 Comparison with domain adaptation (DA) for 3D semantic segmentation

ID	Method	DA	→ Waymo (2 cl.)		→ nuScenes (10 cl.)	
			mIoU	rel.change	mIoU	rel.change
c.l.b	Complete&Label baseline	no	46.3	–	27.9	–
c.l	Complete&Label	yes	52.0	+12.3%	**31.6**	**+13.3%**
c.n	Cylinder3D	no	51.8	–	25.8	–
u.n	[ours]	no	**63.4**	**+22.4%**	28.0	+8.5%

Bold values indicate the best-performing results

Models transferred to Waymo and nuScenes after training without LiDAR intensity on SemanticKITTI with 10 classes (Complete&Label (Yi et al., 2021)) and 19 classes (ours). The evaluation follows the class setup of Yi et al. (2021). Due to the major differences between Complete&Label and our approach, also in terms of baselines, we report the relative improvements (rel.change)

easier for them to transfer their models. Due to the performance discrepancy of their baseline compared to ours (i.e., Cylinder3D), we consider the relative improvements of each approach.

Despite not using any target information, our adversarial augmentation approach outperformed the domain adaptation method of Yi et al. (2021) by a significant margin on Waymo. Ours improved by 22% over our baseline (i.e., Cylinder3D), while theirs improved by 12% over their baseline. Both baselines were trained on SemanticKITTI and transferred without any fine-tuning nor domain adaptations. However, their models were trained only on the 10 classes being evaluated, compared to ours trained on the full set of 19 classes, which made it more challenging for ours. Transferring to nuScenes, due to the sparsity of its point clouds compared to the source data (i.e., SemanticKITTI), Cylinder3D was unable to deliver satisfactory results, underperforming the baseline of Yi et al. (2021). Over the baselines, ours improved by 8.5%, compared to their 13.3% increase. Therefore, using knowledge about the target domain (i.e., nuScenes) was helpful here.

Moreover, our method does not alter the entire point clouds, but only a single object per scene, always belonging to the same class (i.e., *car* in this case). Instead, domain adaptation approaches, such as that of Yi et al. (2021) act upon the whole 3D point clouds. This allows them to improve over all classes, while ours mainly focuses on a single class (e.g., *car*). This could be an extra reason why ours outperformed on the 2 class setting (i.e., Waymo), but not with 10 classes (i.e., nuScenes). Nevertheless, on Waymo, without using any knowledge about the target data nor sensor, ours outperformed a recent domain adaptation technique designed to be robust across different LiDAR sensors (Yi et al., 2021). This confirms the benefits of our approach towards robustness and out-of-domain data.

5.9 Combination with Domain Adaptation

By addressing domain generalization, our approach does not use any target information. Therefore, ours is not alternative to domain adaptation methods (Wang et al., 2020b; Yi et al., 2021), which make use of target data. However, similarly to other data augmentation strategies, our approach can be combined with domain adaptation techniques.

Object Detection As shown in Table 2, such combination further boosts the performance on challenging out-of-domain data. By altering the objects size via the statistical normalization (SN) of Wang et al. (2020b), the AP on Waymo increased. Constrained by the high amount of false positives and negatives, when combined with SN, ours retained a margin of over 2% compared to PointPillars with SN. Moreover, the AP on CrashD improved significantly across all categories, especially for the hardest *rare crash* group. The results show how, despite a substantial increase in AP from PointPillars (Lang et al., 2019), SN alone did not reach the full potential of the detector. Only when combined with ours, the AP doubled (*normal crash*) and more than tripled (*rare crash*) over PointPillars, without using any extra information about the target. This shows the benefit of this combination, and reiterates the added value of incorporating adversarially deformed objects via data augmentation to improve generalization to out-of-domain samples.

5.10 Different 3D Object Detectors

In Table 2, we also compare the performance of our augmentations when paired with different 3D object detectors, namely PointPillars (Lang et al., 2019) and Part-A^2 (Shi et al., 2020). Remarkably, using the proposed adversarial augmentation improved the AP of Part-A^2 on Waymo by a large margin. The superiority of Part-A^2 over the other detector can be attributed to its part-awareness (Shi et al., 2020), which might have set its focus on the most relevant object parts (e.g., wheels) and their relationships to identify cars also in out-of-domain settings. Adding our adversarial deformations significantly improved the generalization of both detectors to out-of-domain data, despite training our vector fields solely against PointPillars, and transferring them to Part-A^2. This shows the wide applicability and transferrability of our techniques.

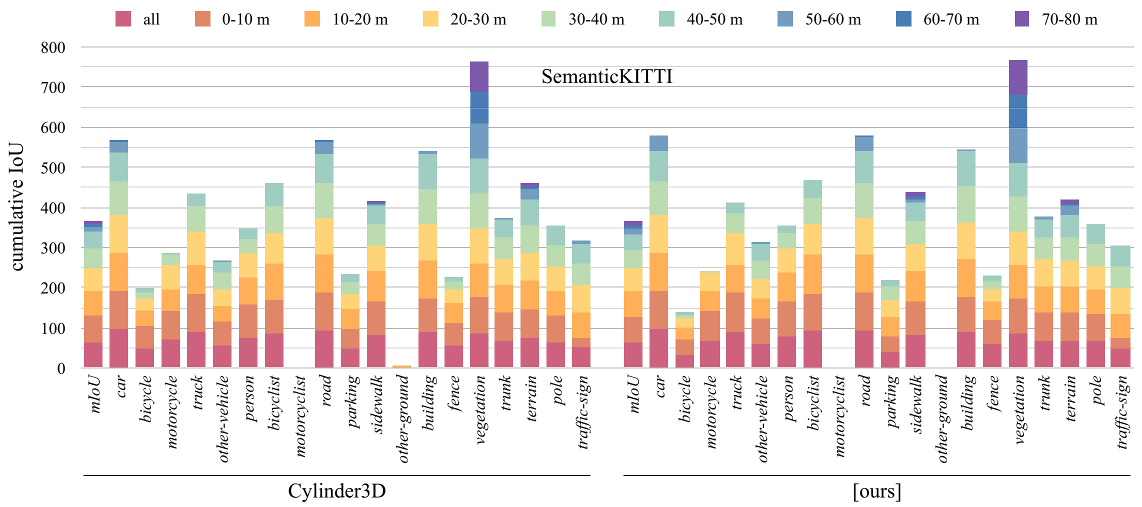


Fig. 8 Evaluation by distance on the validation set of SemanticKITTI (Behley et al., 2019). Predictions of Cylinder3D (Zhu et al., 2021a) trained without (left, with ID c.0) and with (right, with ID u.c) our adversarial augmentations. Both models use the intensity signal. Our approach augments only *car* points in an untargeted fashion. The plot shows stacked bars (i.e., cumulative IoU) and the colors represent different distance bins, as indicated in the legend above

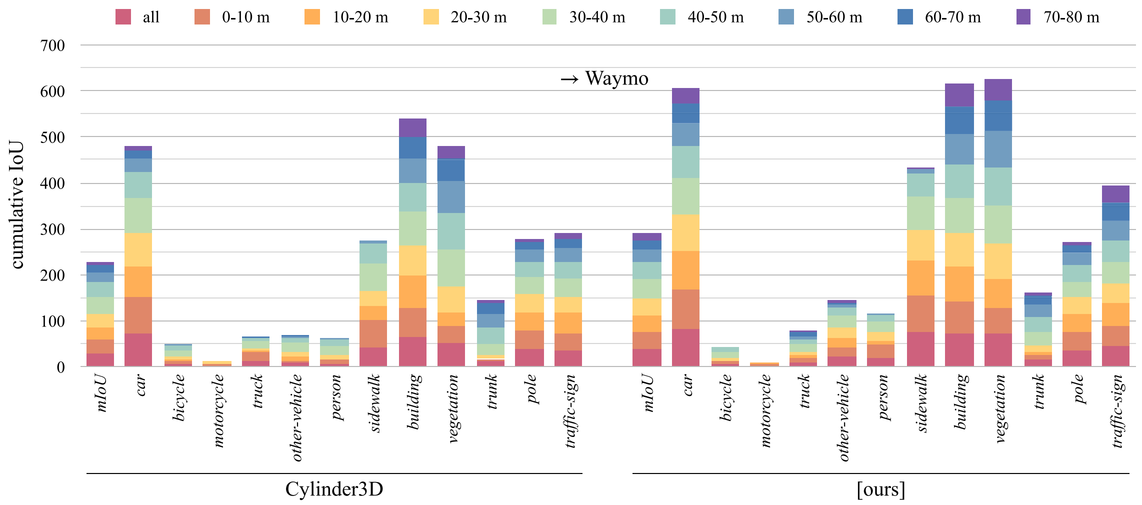


Fig. 9 Evaluation by distance on the transfer to the validation set of Waymo (Sun et al., 2020). Predictions of Cylinder3D (Zhu et al., 2021a) trained on SemanticKITTI without (left, with ID c.0) and with (right, with ID u.c) our adversarial augmentations. Both models use the intensity signal. Our approach augments only *car* points in an untargeted fashion. The plot shows stacked bars (i.e., cumulative IoU) and the colors represent different distance bins, as indicated in the legend above

5.11 Robustness at Further Distances

Semantic Segmentation In Figs. 8 and 9 we compare the IoUs obtained by Cylinder3D with those from our method at varying distances from the LiDAR sensor. Figure 8 reports the results on SemanticKITTI, while Fig. 9 those of the transfer to Waymo. Towards this end, we first computed the distance in 3D of each point to the LiDAR sensor, we clustered them in 8 bins ranging 10 ms each, and finally computed the IoUs separately for each bin.

Figure 8 shows that our method performed similarly to the baseline (i.e., Cylinder3D) on SemanticKITTI across various distances, with only minor differences. This is reflected by the similar mIoUs in Table 1. However, Fig. 9 shows a different outcome for the challenging out-of-domain Waymo. Despite reaching similar IoUs at closer distances, our adversarial augmentations delivered better predictions, especially further from the sensor, i.e., more challenging sparser areas, where the LiDAR intensity follows an opposite trend to that of SemanticKITTI (Sect. 5.5). This superiority is shown by

Table 10 Comparison of 3D semantic segmentation models trained on SemanticKITTI (Behley et al., 2019) and transferred to Waymo (Sun et al., 2020) and nuScenes (Caesar et al., 2020) validation sets (without fine-tuning)

ID	Attack	Method	SemanticKITTI				→ Waymo				→ nuScenes			
			mIoU	*car*	*pers.*	*bicyc.*	mIoU	*car*	*pers.*	*bicyc.*	mIoU	*car*	*pers.*	*bicyc.*
c.o	–	Cylinder3D	**64.39**	96.02	73.84	47.15	29.43	72.10	6.69	5.68	29.97	66.04	0.00	**0.89**
u.a	$N\times$a.u.a	[ours] untar.ax.alg. *car*	61.82	**96.30**	75.21	**50.67**	28.38	74.18	12.21	**6.49**	27.75	64.88	0.04	0.58
u.-	$N\times$a.u.-	[ours] untar.-10% *car*	63.92	96.16	74.02	42.04	36.65	81.25	13.15	6.17	30.42	70.01	**0.06**	0.39
u.c	$N\times$a.u.c	[ours] untargeted *car*	63.85	96.03	**77.11**	33.41	**37.74**	**82.38**	**17.13**	5.04	**31.55**	**71.92**	**0.06**	0.53

Bold values indicate the best-performing results

This evaluation assesses the impact of the available data annotation on our method. With u.- we randomly removed 10% of the bounding boxes predicted by the off-the-shelf 3D detector, so our attack was learned on 10% less objects, and our model was learned augmenting 10% less objects. Instead, with u.a we explore the impact of using axis-aligned bounding boxes instead of predicted 3D bounding boxes. The u.c model used throughout this work used all the boxes predicted by the off-the-shelf detector

the amount of blue and purple in the plot, which correspond to the furthest bins, between 60 and 70 ms, and between 70 and 80, respectively. Especially for *car*, *building*, and *vegetation*, our method managed to deliver good quality predictions, even beyond 70 ms. Instead, the standard Cylinder3D between 70 and 80 ms could only recognize a few *car* points, with an IoU of 6.7, compared to 32.9 of our approach. This confirms the robustness of our method, also at high distances.

5.12 Combination with Data Augmentations

Being an adversarial data augmentation, our approach is not alternative to other augmentation techniques, but can be applied in combinations with others, e.g., Mix3D (Nekrasov et al., 2021).

Object Detection As shown in Table 2, on KITTI removing all augmentations had a major impact on the AP. For PointPillars (Lang et al., 2019), not using augmentations (no augm.) drastically reduced the APs, especially on CrashD at IoU 0.7. Instead, at IoU 0.5, the AP on *normal clean* was 65.59, with the standard PointPillars reaching 98.91. Adding common augmentations (no obj. sampl., e.g., flip and rotation) increased the APs, but introducing the popular object sampling (Lang et al., 2019) (PointPillars) improved them even further. Nevertheless, applying our adversarial augmentations on *car* points on top of the standard augmentations significantly improved the IoUs on out-of-domain data. This shows the compatibility of our adversarial approach with other data augmentation techniques.

5.13 Impact of Data Annotations

In Table 10, we explore the impact of the availability of data annotation on our method. As described in Sect. 3.2.1, our method requires references on which to apply the vector field. Specifically, we used 3D bounding boxes to match the objects with the vector fields. While bounding box annotations are often available, this may not always be the case (e.g., SemanticKITTI). To circumvent this, we explore two different strategies: using an off-the-shelf 3D object detector deployed on the training data, or using point-level instance annotations and wrapping each instance in an axis-aligned bounding box.

In the table, we report the effect of both strategies. While the axis-aligned is inherently sub-optimal due to the lack of orientation information, preventing the vector fields from specializing to the object viewing angles, it still brings improvements on the *car* class over the baseline both in-domain and on Waymo. Instead, using the off-the-shelf 3D detector delivers superior performance, especially out-of-domain. In the table, we also explore the impact of the quality of the predictions of the off-the-shelf detector. We do so by considering 10% less of its bounding box predictions, selected randomly. Remarkably, the performance degrades only slightly for the *car* class, showing the robustness of our method with respect to its assumptions in terms of data annotations.

5.14 Ablation Study on Deformation Constraints

Semantic Segmentation As we introduced sensor-awareness and surface smoothness constraints to our deformations, we investigate their impact in terms of generalization to out-of-domain data. In Table 11, we report this comparison on 3D semantic segmentation when limiting the deformations to $\epsilon = 30$ cm. It can be seen that not learning the perturbations (i.e., not adversarial), but applying all our constraints (all constr. no learn) is already an effective augmentation technique, as it improved on the reference *car* class, also when transferring to Waymo. Instead, removing all constraints, but learning the vector fields (unleash) led to the strongest adversarial attack (i.e., lowest IoUs after the attack). However, the lack of constraints did not allow for robust generalization, reducing the mIoU in-domain, and the *car* class on Waymo. When deforming with sensor-awareness (ray constraint), the attack lost effectiveness, but in-domain mIoU

Table 11 Ablation on the deformation constraints imposed by our method on 3D semantic segmentation, compared to Cylinder3D (Zhu et al., 2021a)

ID	Method	SemK attack		SemanticKITTI				→ Waymo				→ nuScenes			
		mIoU	*car*	mIoU	*car*	*pers.*	*bike*	mIoU	*car*	*pers.*	*bike*	mIoU	*car*	*pers.*	*bike*
c.o	Cylinder3D	–	–	**64.39**	96.02	73.84	47.15	29.43	72.10	6.69	**5.68**	29.97	66.04	0.00	**0.89**
u.l	all constr. no learn	63.92	95.77	62.07	**96.21**	75.11	48.85	28.87	78.76	12.54	4.12	27.85	63.81	0.02	0.35
u.u	unleash	**44.79**	**27.99**	61.64	96.07	72.56	46.58	30.16	69.22	13.32	5.97	30.52	63.28	0.00	0.04
u.r	ray constraint	52.77	61.78	62.37	95.76	71.48	**49.26**	28.61	69.35	12.46	4.30	27.41	66.59	**0.13**	0.23
u.c	full	53.25	62.61	63.85	96.03	**77.11**	33.41	**37.74**	**82.38**	**17.13**	5.04	**31.55**	**71.92**	0.06	0.53

Bold values indicate the best-performing results

All models are based on Cylinder3D, use LiDAR intensity as input, and our adversarial augmentations are untargeted on the *car* class. SemK attack represents the IoUs obtained by Cylinder3D after perturbing the input point clouds with the various configurations

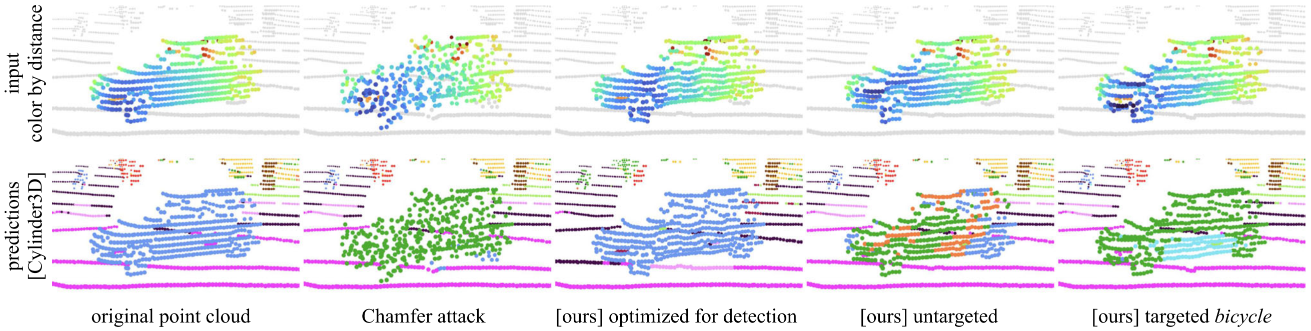


Fig. 10 Perturbations and predictions for 3D semantic segmentation. The first row shows crops of the input point clouds from SemanticKITTI (Behley et al., 2019), with a car whose points are colored according to their distance from the LiDAR sensor. Along the row, each point cloud is perturbed by a different adversarial method optimized on *car*, namely the sample-specific Chamfer attack (Liu et al., 2020), and our approach in three different configurations: optimized for 3D object detection, untargeted, and targeted towards *bicycle* (turquoise in the predictions). The second row shows the predictions of Cylinder3D (Zhu et al., 2021a) on the perturbed point clouds of the first row. All adversarial methods included the intensity signal, except for our vectors optimized for object detection, which did not include it. Therefore, the evaluated model evaluated in the second row takes in input also the intensity values (ID: c.o), except for the one predicting on the point cloud perturbed by the detection vectors (ID: c.n)

increased. With full, we added the constrain on the surface smoothing (Sect. 3.2.2), which allowed for superior transfer capabilities, thanks to the improved plausibility of the deformations.

6 Qualitative Results

6.1 3D Semantic Segmentation, Adversarial Examples

Figure 10 shows the impact of the perturbations applied by our method compared to sample-specific adversarial attacks, represented by the Chamfer attack (Liu et al., 2020). As discussed in Sects. 5.1 and 5.2, by being sample-specific (i.e., instance-specific in our setup), prior works deliver very strong attacks thanks to their highly noticeable perturbations. These substantial perturbations bring their adversarial examples too far from the training distribution (second column of the figure), making the samples unrecognizable. In fact, the attacked car is predicted as *vegetation* (green). However, for the same reasons, augmenting with the point clouds produced by the Chamfer attack (or other sample-specific methods) carries no benefits in terms of generalization, as the data will likely not contain objects of the same category resembling the perturbed objects. Therefore, augmenting with such samples confuses the model.

6.1.1 Plausibility

As we focus on improving generalization and robustness to out-of-domain data, our adversarial examples need to be plausible, yet difficult enough for the model to expand the training data distribution. This plausibility can be seen in Fig. 10 via the significantly less noticeable perturbations applied by our method (from the third to the last columns), compared to the Chamfer attack. Our constraints (e.g., sensor-awareness) mitigated the deformations leading to adversarial examples which still resemble an object of the same class (i.e., *car*). As our plausible adversarial examples are different from the existing training data, they could be similar to real-world out-of-distribution objects (e.g., a damaged car). Therefore, using them as augmentation positively enriches the training data and improves the model generalization and robustness.

6.1.2 Targeted/Untargeted and Cross-Task

In Fig. 10, we compare the effect on the point clouds of three variants of our adversarial method on semantic segmentation, namely the untargeted vectors for the *car* class (a.u.c), the targeted vectors from *car* to *bicycle* (a.t.b), and the vectors optimized for object detection and applied on semantic segmentation. Although the vectors optimized against the 3D detector PointPillars altered the points visually similar to the ones trained for semantic segmentation (targeted and untargeted), their perturbations did not affect Cylinder3D, which managed to accurately segment the deformed car. Conversely, by being optimized for the same task, both our untargeted and targeted vectors perturbed the point cloud in a way that made Cylinder3D wrongly segment the car. On the untargeted adversarial example Cylinder3D predicted a mix of *vegetation* (green), *fence* (orange), and *car* (blue). Instead, on the targeted example Cylinder3D predicted a mix of *bicycle* (turquoise) and *vegetation* (green), as if it recognized a bicycle leaning against a bush, which is rather common in SemanticKITTI (Behley et al., 2019). By augmenting with an untargeted sample as the one shown in the figure, the model learns to better distinguish the *car* class from the others. Instead, with a targeted sample, the network improves its decision boundary between the two specific classes, i.e., *car* and *bicycle* as shown. The figure shows both the effectiveness of our adversarial examples, as well as the difference between our targeted and untargeted approach.

6.1.3 Limitations

As we aimed at preserving the comparability between our detection and semantic vectors for cross-task comparisons, we did not selectively perturb only the points in the 3D bounding box corresponding to the class of interest (i.e., *car*). Instead, we perturb all the points inside the box. This is visible in Fig. 10 on a portion of the road next to the car, which was shifted by the vectors. Furthermore, by relying on predicted bounding boxes for cars, we could not perturb points outside the boxes, despite being part of the same instance. In the example shown in Fig. 10, the off-the-shelf detector used (Deng et al., 2021) did not fully detect the car, and left a few points outside the box. This is visible on the roof line and along other lines of points on the side of the car. Therefore, these points were left unperturbed by all methods, including the Chamfer attack (which was applied on the same objects). Nevertheless, in the adversarial examples those points were anyways wrongly segmented by Cylinder3D.

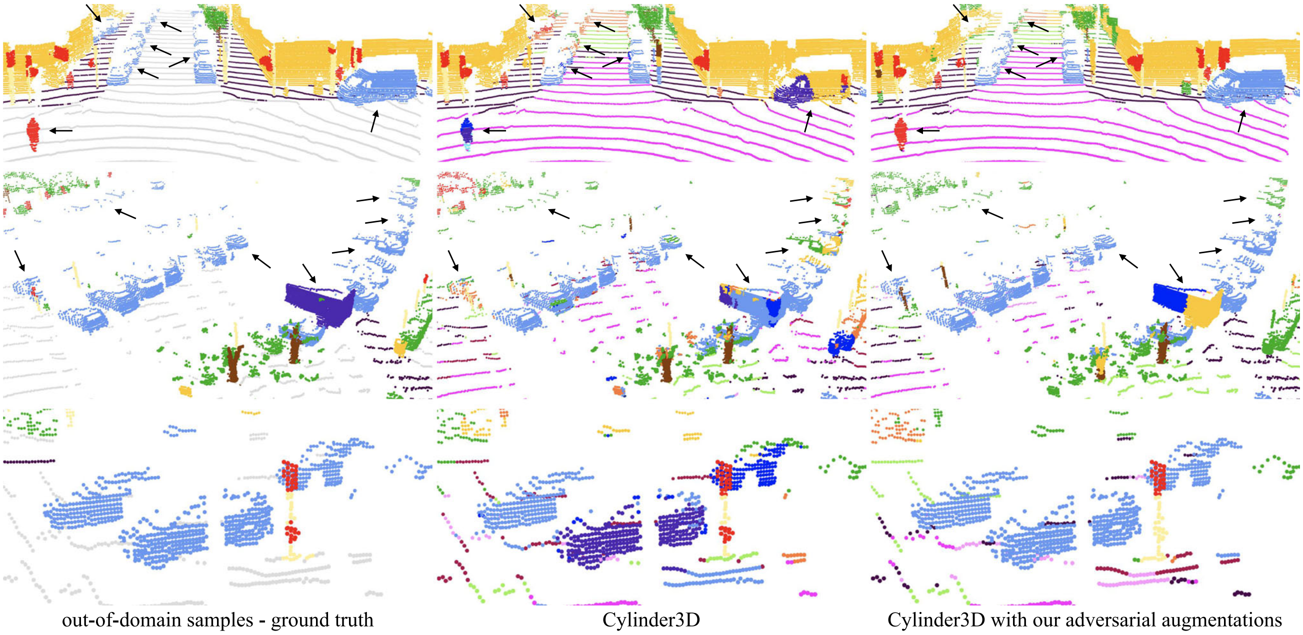

out-of-domain samples - ground truth
Cylinder3D
Cylinder3D with our adversarial augmentations

Fig. 11 3D semantic segmentation predictions on challenging out-of-domain samples from Waymo (Sun et al., 2020). Both models are based on Cylinder3D (Zhu et al., 2021a), trained only on SemanticKITTI, and transferred to Waymo without fine-tuning. Two Cylinder3D models are compared: without (c.o) and with (u.c) our adversarial augmentations. Black arrows highlight some of the objects segmented wrongly by the baseline. The road and other classes were ignored in the transfer to Waymo due to misaligned definitions across the datasets

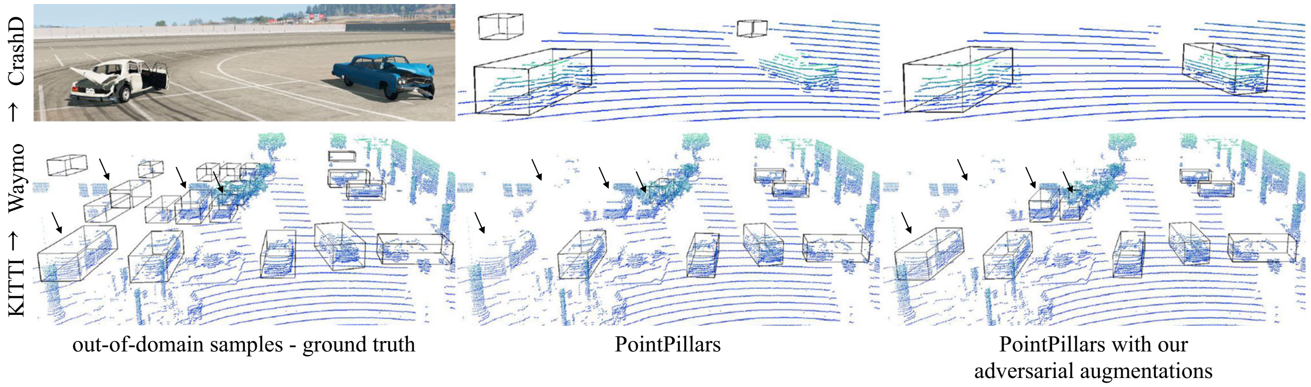

CrashD
KITTI → Waymo
out-of-domain samples - ground truth
PointPillars
PointPillars with our adversarial augmentations

Fig. 12 3D object detection predictions on challenging out-of-domain samples from the proposed CrashD dataset (top) and Waymo (Sun et al., 2020) (bottom). Both models are based on PointPillars (Lang et al., 2019), trained only on KITTI, and transferred to CrashD and Waymo without fine-tuning. Two PointPillars models are compared: without (p.p) and with (p.o) our adversarial augmentations. Black arrows highlight some of the objects segmented wrongly by the baseline

6.2 3D Semantic Segmentation, Robustness and Generalization

Figure 11 shows the semantic predictions of Cylinder3D trained with and without our adversarial augmentations. Both models are trained on SemanticKITTI, and transferred to the point clouds of Waymo shown in the figure. As the adversarial examples used to train our model were only on cars, we focus on the *car* class. Due to the large domain gap between the two datasets, neither of the models was able to properly segment all cars in the scenes. However, while ours missed only the cars with few points, the standard Cylinder3D could not segment several visible ones, confusing them with *truck*, *building*, *vegetation*, and other classes. The superior performance of our model across the various scenes confirms the added robustness and generalization of our method towards challenging out-of-domain data.

6.3 3D Object Detection

In Fig. 12 we compare the transfer from KITTI to CrashD and Waymo (Sun et al., 2020) in terms of 3D bounding boxes predicted by PointPillars (Lang et al., 2019) trained with and without our adversarial augmentations. For the proposed CrashD, our approach correctly identified both damaged cars, without false positives. The figure testifies also the intensity of the *hard* damages of CrashD, and how challenging these vehicles are for a 3D detector compared to standard cars (e.g., KITTI), resembling natural adversarial examples. For the transfer from KITTI to Waymo, both the baseline and our augmented model had difficulty detecting all cars in the scene, especially those with fewer points in the parking lot on the left of the crops. However, the standard PointPillars ignored also 3 recognizable cars featuring lots of points. Instead, albeit leaving room for improvement, ours could recognize them.

We refer to the **Supplementary Material** for more results on indoor settings, the IoUs on the complete set of classes, as well as details on the class mappings for the transfers to Waymo and nuScenes.

7 Conclusion

In this work we presented a flexible adversarial augmentation approach, which improves generalization and robustness across multiple 3D tasks. By expanding the available training data with plausible adversarial examples, our augmentations increase safety by addressing challenging long tail and out-of-domain samples without the burden of capturing them in the real-world. As extensive experiments across multiple tasks and datasets showed its benefits and flexibility, we believe our method constitutes a valuable step towards safe and robust perception for high automation systems.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11263-023-01914-7.

Acknowledgements This work was partly sponsored by the German Federal Ministry for Economic Affairs and Energy (Grant Number $19A19005B$), through the VDA KI-Absicherung project.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Abdollahi, A., Pradhan, B., Sharma, G., Maulud, K. N. A., & Alamri, A. (2021). Improving road semantic segmentation using generative adversarial network. *IEEE Access, 9*, 64381–64392.

Alaifari, R., Alberti, G. S., & Gauksson, T. (2019). ADef: An iterative algorithm to construct adversarial deformations. In *Proceedings of the international conference on learning representations*.

Albuquerque, I., Naik, N., Li, J., Keskar, N., & Socher, R. (2020). Improving out-of distribution generalization via multi-task self-supervised pretraining. *arXiv preprint* arXiv:2003.13525

Arnab, A., Miksik, O., & Torr, P. H. (2018). On the robustness of semantic segmentation models to adversarial attacks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 888–897).

Baier, L., Jöhren, F., & Seebacher, S. (2019). Challenges in the deployment and operation of machine learning in practice. In *European conference on information systems (ecis)*.

Balaji, Y., Sankaranarayanan, S., & Chellappa, R. (2018). Metareg: Towards domain generalization using meta-regularization. *Advances in Neural Information Processing Systems, 31*, 998–1008.

Beery, S., Liu, Y., Morris, D., Piavis, J., Kapoor, A., Joshi, N., Meister, M., & Perona, P.,(2020). Synthetic examples improve generalization for rare classes. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 863–873).

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., & Gall, J. (2019). SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9297–9307).

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., & Beijbom, O. (2020). NuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11621–11631).

Cao, Y., Wang, N., Xiao, C., Yang, D., Fang, J., Yang, R., Chen, Q. A., Liu, M., & Li, B. (2021). Invisible for both camera and LiDAR: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *Proceedings of the IEEE symposium on security and privacy* (pp. 176–194).

Cao, Y., Xiao, C., Cyr, B., Zhou, Y., Park, W., Rampazzi, S., Chen, Q. A., Fu, K., & Mao, Z. M. (2019a). Adversarial sensor attack on LiDAR-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security* (pp. 2267–2281).

Cao, Y., Xiao, C., Yang, D., Fang, J., Yang, R., Liu, M., & Li, B. (2019b). Adversarial objects against lidar-based autonomous driving systems. *arXiv preprint* arXiv:1907.05418

Carlini, N., & Wagner, D. (2017). Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE symposium on security and privacy* (pp. 39–57).

Chen, X., Li, S., Mersch, B., Wiesmann, L., Gall, J., Behley, J., & Stachniss, C. (2021). Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data. *IEEE Robotics and Automation Letters, 6*(4), 6529–6536.

Cheng, R., Razani, R., Taghavi, E., Li, E., & Liu, B. (2021). (AF)2-S3Net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12547–12556).

Contributors, M. (2020). MMDetection3D: Open-MMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d.

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., & Li, H. (2021). Voxel R-CNN: Towards high performance voxel-based 3D object detection. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 35, pp. 1201–1209).

Gasperini, S., Haug, J., Mahani, M.-A.N., Marcos- Ramiro, A., Navab, N., Busam, B., & Tombari, F. (2021). CertainNet: Sampling-free uncertainty estimation for object detection. *IEEE Robotics and Automation Letters, 7*(2), 698–705.

Gasperini, S., Koch, P., Dallabetta, V., Navab, N., Busam, B., & Tombari, F. (2021b). R4Dyn: Exploring radar for self-supervised monocular depth estimation of dynamic scenes. In *Proceedings of the IEEE international conference on 3d vision (3dv)* (pp. 751–760).

Gasperini, S., Mahani, M.-A.N., Marcos-Ramiro, A., Navab, N., & Tombari, F. (2021). Panoster: End-to-end panoptic segmentation of LiDAR point clouds. *IEEE Robotics and Automation Letters, 6*(2), 3216–3223.

Gasperini, S., Marcos-Ramiro, A., Schmidt, M., Navab, N., Busam, B., & Tombari, F. (2023a). Segmenting known objects and unseen unknowns without prior knowledge. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 19321–19332).

Gasperini, S., Morbitzer, N., Jung, H., Navab, N., & Tombari, F. (2023b). Robust monocular depth estimation under challenging conditions. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 8177–8186).

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., & Shahzad, M. (2021). A survey of uncertainty in deep neural networks. *arXiv preprint* arXiv:2107.03342 .

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3354–3361).

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *Proceedings of the international conference on learning representations*.

Hamdi, A., Rojas, S., Thabet, A., & Ghanem, B. (2020). AdvPC: Transferable adversarial perturbations on 3D point clouds. *Proceedings of the European conference on computer vision* (pp. 241–257).

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., & Song, D., (2021). The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 8340–8349).

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., & Song, D. (2021). Natural adversarial examples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 15262–15271).

Jung, H., Brasch, N., Leonardis, A., Navab, N., & Busam, B. (2021a). Wild ToFu: Improving range and quality of indirect time-of-flight depth with RGB fusion in challenging environments. In *2021 international conference on 3d vision* (3dv) (pp. 239–248).

Jung, S., Lee, J., Gwak, D., Choi, S., & Choo, J. (2021b). Standardized Max Logits: A simple yet effective approach for identifying unexpected road obstacles in urban-scene segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 15425–15434).

Kilic, V., Hegde, D., Sindagi, V., Cooper, A. B., Foster, M. A., & Patel, V. M. (2021). LiDAR Light Scattering Augmentation (LISA): Physics-based simulation of adverse weather conditions for 3D object detection. *arXiv preprint* arXiv:2107.07004 .

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). PointPillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12697–12705).

Lehner, A., Gasperini, S., Marcos-Ramiro, A., Schmidt, M., Mahani, M. A. N., Navab, N., Busam, B., & Tombari, F. (2022). 3D-V field: Adversarial augmentation of point clouds for domain generalization in 3D object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 17295–17304).

Li, G., Xu, G., Qiu, H., He, R., Li, J., & Zhang, T. (2022a). Improving adversarial robustness of 3D point cloud classification models. In *European conference on computer vision* (pp. 672–689).

Li, J., Dai, H., & Ding, Y. (2022b). Self-distillation for robust LiDAR semantic segmentation in autonomous driving. In *European conference on computer vision* (pp. 659–676).

Liu, D., Yu, R., & Su, H. (2020). Adversarial shape perturbations on 3D point clouds. In *Proceedings of the European conference on computer vision* (pp. 88–104).

Liu, M., Zhou, Y., Qi, C. R., Gong, B., Su, H., & Anguelov, D. (2022). LESS: Label-efficient semantic segmentation for LiDAR point clouds. In *European conference on computer vision* (pp. 70–89).

Luo, Y., Liu, P., Zheng, L., Guan, T., Yu, J., & Yang, Y. (2021). Category-level adversarial adaptation for semantic segmentation using purified features. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*, 3940–3956.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In *Proceedings of the international conference on learning representations*.

Marcuzzi, R., Nunes, L., Wiesmann, L., Vizzo, I., Behley, J., & Stachniss, C. (2022). Contrastive instance association for 4D panoptic segmentation using sequences of 3D LiDAR scans. *IEEE Robotics and Automation Letters, 7*(2), 1550–1557.

Maul, P., Mueller, M., Enkler, F., Pigova, E., Fischer, T., & Stamatogiannakis, L. (2021). BeamNG.tech technical paper.

Milioto, A., Vizzo, I., Behley, J., & Stachniss, C. (2019). RangeNet++: Fast and accurate LiDAR semantic segmentation. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 4213–4220).

Mirza, M. J., Micorek, J., Possegger, H., & Bischof, H. (2022). The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14765–14775).

Mok, J., Na, B., Choe, H., & Yoon, S. (2021). AdvRush: Searching for adversarially robust neural architectures. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12322–12332).

Moosavi-Dezfooli, S., Fawzi, A., & Frossard, P. (2016). DeepFool: A simple and accurate method to fool deep neural networks. In *IEEE conference on computer vision and pattern recognition* (pp. 2574–2582). IEEE.

Nekrasov, A., Schult, J., Litany, O., Leibe, B., & Engelmann, F. (2021). Mix3D: Out-of context data augmentation for 3D scenes. In *2021 international conference on 3d vision (3dv)* (pp. 116–125).

Nunes, L., Marcuzzi, R., Chen, X., Behley, J., & Stachniss, C. (2022). SegContrast: 3D point cloud feature representation learning through self-supervised segment discrimination. *IEEE Robotics and Automation Letters, 7*(2), 2116–2123.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. In *Proceedings of the IEEE European symposium on security and privacy* (pp. 372–387).

Postels, J., Ferroni, F., Coskun, H., Navab, N., & Tombari, F. (2019). Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2931–2940).

Qi, C. R., Litany, O., He, K., & Guibas, L. J. (2019). Deep Hough voting for 3D object detection in point clouds. In *Proceedings of the IEEE international conference on computer vision*.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 652–660).

Qiao, F., Zhao, L., & Peng, X. (2020). Learning to learn single domain generalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12556–12565).

Razani, R., Cheng, R., Li, E., Taghavi, E., Ren, Y., & Bingbing, L. (2021). GP-S3Net: Graphbased panoptic sparse semantic segmentation network. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 16076–16085).

Sanchez, J., Deschaud, J.-E., & Goulette, F. (2022). Domain generalization of 3D semantic segmentation in autonomous driving. *arXiv preprint* arXiv:2212.04245 .

Shi, S., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). From points to parts: 3D object detection from point cloud with part-aware and part aggregation network. In *IEEE transactions on pattern analysis and machine intelligence*.

Simonelli, A., Bulo, S. R., Porzi, L., Ricci, E., & Kontschieder, P. (2020). Towards generalization across depth for monocular 3D object detection. In *Proceedings of the European conference on computer vision* (pp. 767–782).

Song, S., Lichtenberg, S. P., & Xiao, J. (2015). SUN RGB-D: A RGB-D scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 567–576).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research, 15*(1), 1929–1958.

Summers, C., & Dinneen, M. J. (2019). Improved mixed-example data augmentation. In *Proceedings of the IEEE winter conference on applications of computer vision* (pp. 1262– 1270).

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., & Vasudevan, V. (2020). Scalability in perception for autonomous driving: Waymo open dataset. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2446–2454).

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. In *Proceedings of the international conference on learning representations*.

Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., & Han, S. (2020). Searching efficient 3d architectures with sparse point-voxel convolution. |In *European conference on computer vision* (pp. 685–702).

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., & Guibas, L.J. (2019). KPConv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6411–6420).

Tombari, F., Salti, S., & Di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *European conference on computer vision* (pp. 356–369).

Tu, J., Ren, M., Manivasagam, S., Liang, M., Yang, B., Du, R., Cheng, F., & Urtasun, R. (2020). Physically realizable adversarial examples for LiDAR object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13713–13722).

Volpi, R., Namkoong, H., Sener, O., Duchi, J., Murino, V., & Savarese, S. (2018). Generalizing to unseen domains via adversarial data augmentation. In *Proceedings of the international conference on neural information processing systems* (pp. 5339–5349).

Wang, J., Lan, C., Liu, C., Ouyang, Y., & Qin, T. (2021). Generalizing to unseen domains: A survey on domain generalization. in *Proceedings of the international joint conference on artificial intelligence* (pp. 4627–4635).

Wang, R., Juefei-Xu, F., Guo, Q., Huang, Y., Xie, X., Ma, L., & Liu, Y. (2020a). Amora: Blackbox adversarial morphing attack. In *Proceedings of the ACM international conference on multimedia* (pp. 1376–1385).

Wang, Y., Chen, X., You, Y., Li, L. E., Hariharan, B., Campbell, M., Weinberger, K. Q., & Chao, W. L., (2020b). Train in Germany, test in the USA: Making 3D object detectors generalize. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11713–11723).

Wang, Y., Shi, T., Yun, P., Tai, L., & Liu, M. (2018). PointSeg: Real-time semantic segmentation based on 3D LiDAR point cloud. *arXiv preprint* arXiv:1807.06288 .

Xiang, C., Qi, C.R., & Li, B. (2019). Generating 3D adversarial point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9128–9136).

Xiao, A., Huang, J., Guan, D., Zhan, F., & Lu, S. (2022). Transfer learning from synthetic to real LiDAR point cloud for semantic segmentation. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 36, pp. 2795–2803).

Xiao, C., Li, B., Zhu, J.-y., He, W., Liu, M., & Song, D. (2018). Generating adversarial examples with adversarial networks. In *Proceedings of the international joint conference on artificial intelligence* (pp. 3905– 3911).

Xu, J., Zhang, R., Dou, J., Zhu, Y., Sun, J., & Pu, S. (2021). RPVNet: A deep and efficient range-point-voxel fusion network for LiDAR point cloud segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 16024–16033).

Yan, X., Gao, J., Zheng, C., Zheng, C., Zhang, R., Cui, S., & Li, Z. (2022). 2DPASS: 2D priors assisted semantic segmentation on LiDAR point clouds. In *Proceedings of the European conference on computer vision* (pp. 677–695).

Yang, J., Zhang, Q., Fang, R., Ni, B., Liu, J., & Tian, Q. (2019). Adversarial attack and defense on point sets. *arXiv preprint* arXiv:1902.10899 .

Yi, L., Gong, B., & Funkhouser, T. (2021). Complete & Label: A domain adaptation approach to semantic segmentation of LiDAR point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 15363–15373).

Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems, 30*(9), 2805–2824.

Zhang, G., Ma, Q., Jiao, L., Liu, F., & Sun, Q. (2021a). AttAN: Attention adversarial networks for 3D point cloud semantic segmentation. In *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence* (pp. 789–796).

Zhang, L., Deng, Z., Kawaguchi, K., Ghorbani, A., & Zou, J. (2021b). How does mixup help with robustness and generalization? In *Proceedings of the international conference on learning representations*.

Zhao, B., Yu, S., Ma, W., Yu, M., Mei, S., Wang, A., He, J., Yuille, A., & Kortylewski, A. (2022). OODCV: A benchmark for robustness to out of- distribution shifts of individual nuisances in natural images. In *European conference on computer vision* (pp. 163–180).

Zhou, Y., & Tuzel, O. (2018). VoxelNet: End-to end learning for point cloud based 3D object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4490–4499).

Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., & Lin, D. (2021a). Cylindrical and asymmetrical 3D convolution networks for lidar segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9939–9948).

Zhu, Y., Miao, C., Hajiaghajani, F., Huai, M., Su, L., & Qiao, C. (2021b). Adversarial attacks against LiDAR semantic segmentation in autonomous driving. In *Proceedings of the 19th ACM conference on embedded networked sensor systems* (pp. 329–342).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.