International Journal of Computer Vision
https://doi.org/10.1007/s11263-022-01676-8

A Realism Metric for Generated LiDAR Point Clouds

Larissa T. Triess[1,2] · **Christoph B. Rist**[1] · **David Peter**[1,3] · **J. Marius Zöllner**[2,4]

Received: 13 February 2022 / Accepted: 28 July 2022

Abstract
A considerable amount of research is concerned with the generation of realistic sensor data. LiDAR point clouds are generated by complex simulations or learned generative models. The generated data is usually exploited to enable or improve downstream perception algorithms. Two major questions arise from these procedures: First, how to evaluate the realism of the generated data? Second, does more realistic data also lead to better perception performance? This paper addresses both questions and presents a novel metric to quantify the realism of LiDAR point clouds. Relevant features are learned from real-world and synthetic point clouds by training on a proxy classification task. In a series of experiments, we demonstrate the application of our metric to determine the realism of generated LiDAR data and compare the realism estimation of our metric to the performance of a segmentation model. We confirm that our metric provides an indication for the downstream segmentation performance.

Keywords Metric · Point cloud · LiDAR · Realism · Adversarial learning · Local features · Semantic segmentation

1 Introduction

Simulations and generative models, such as **G**enerative **A**dversarial **N**etworks (GANs), are often used to synthesize realistic training data samples to improve the performance of perception networks (Park et al., 2019; Xu et al., 2021; Löhdefink & Fingscheidt, 2022; Li et al., 2022). Assessing the realism of such synthesized samples is a crucial part of the process. This is usually done by experts, a cumbersome and time consuming approach. Though a lot of work has been conducted to determine the quality of generated images (Goodfellow et al., 2014; Salimans et al., 2016; Theis et al., 2016; Heusel et al., 2017; Lehmann & Romano, 2006), little work is published about how to quantify the realism of point clouds (Shu et al., 2019; Triess et al., 2021b). Visual inspection of such data is expensive and not reliable given that the interpretation of 3D point data is rather unnatural for humans. Because of their subjective nature, it is difficult to compare generative approaches with a qualitative measure. This work closes the gap and introduces a quantitative evaluation for LiDAR point clouds.

In recent years, a large amount of evaluation measures for GANs emerged (Borji, 2019). Many of them are image-specific and cannot be applied to point clouds. Existing work on generating realistic LiDAR point clouds mostly relies on qualitative measures to evaluate the generation quality. Alternatively, some works apply annotation transfer (Sallab et al., 2019) or use the *Earth Mover's Distance* as an evaluation criterion (Caccia et al., 2019). However, these methods require either annotations associated with the data or a matching target, i.e.Ground Truth, for the generated sample. Both are often not feasible when working with large-scale data generation or transfer learning setups.

One main application of data generation is to train downstream perception models, i.e.segmentation or detection models that make use of the generated data. Here it is crucial

Communicated by Juergen Gall.

✉ Larissa T. Triess
larissa.triess@mercedes-benz.com

Christoph B. Rist
christoph_bernd.rist@mercedes-benz.com

David Peter
david.peter@bosch.com

J. Marius Zöllner
zoellner@fzi.de

[1] Mercedes-Benz AG, Stuttgart, Germany

[2] Karlsruhe Institute of Technology, Karlsruhe, Germany

[3] Robert-Bosch GmbH, Stuttgart, Germany

[4] Research Center for Information Technology, Karlsruhe, Germany

Published online: 22 September 2022

to reduce the domain gap between generated data and target data on which the trained perception model is applied (Triess et al., 2021a). Therefore, the performance of the trained perception model itself can be used as an indication for the realism of the data. However, using this as a proper metric is impractical since it requires to re-train the target network on multiple versions of the data to evaluate their realism. A solution is a metric that can determine the realism of the data already while training the generative model.

To address this need, our previous work (Triess et al., 2021b) proposes a reliable metric that gives a quantitative estimate about the realism of generated LiDAR data. Fig. 1 shows the concept of the metric as a distance measure in high-dimensional feature space. The metric is trained to learn relevant features via a proxy classification task. To avoid learning global scene context, we use hierarchical feature set learning to confine features locally in space. To discourage the network from encoding dataset-specific information, we use an adversarial learning technique which enables robust quantification of unseen data distributions. In this work, we extend our previous approach (Triess et al., 2021b) with evaluations on the influence of data realism on segmentation performance and add additional ablations of the adversarial training. In summary, our contributions are:

- We present a learning-based quantitative metric to measure the realism of LiDAR point clouds.
- We use an adversarial learning technique to suppress irrelevant features, such that the metric can be applied to unseen data.
- In experiments on generated LiDAR data, we analyze the relationship between data realism and downstream perception performance. We show that our metric is a good indicator for the resulting perception performance.

2 Related Work

First, this section discusses GAN evaluation measures and their applicability to generated LiDAR data. Second, we give a brief overview on metric learning.

2.1 GAN Evaluation Measures

A considerable amount of literature deals with how to evaluate generative models and proposes various evaluation measures. The most important ones are summarized in extensive survey papers (Lucic et al., 2018; Xu et al., 2018; Borji, 2019). They can be divided into two major categories: qualitative and quantitative measures.

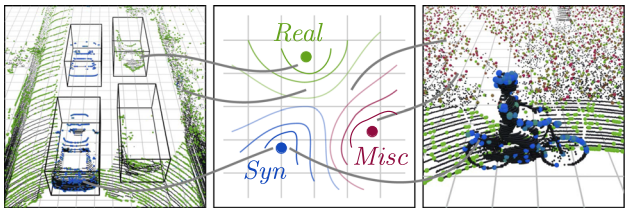


Fig. 1 Proposed approach: The realism measure has a tripartite understanding of the 3D-world (middle). The left and right image show the color-coded metric scores for query points on two example scenes. Both scenes are from the real-world dataset KITTI (*Real*) and are augmented with dynamic objects from the simulated CARLA dataset (*Syn*). The left image shows inserted cars from CARLA (left) next to real KITTI cars (right). The right image demonstrates the metric results for a synthetic bicycle-and-person object in a KITTI scene. Additionally, the terrain in the background is distorted with noise, which is detected as *Misc*

2.1.1 Qualitative Evaluation

Qualitative evaluation (Goodfellow et al., 2014; Huang et al., 2017; Zhang et al., 2017; Srivastava et al., 2017; Lin et al., 2018; Chen et al., 2016; Mathieu et al., 2016) uses visual inspection of a small collection of examples by humans and is therefore of subjective nature. It is a simple way to get an initial impression of the performance of a generative model but cannot be performed in an automated fashion. In other previous work, we use the **M**ean **O**pinion **S**core (MOS) testing to verify the realism of generated LiDAR point clouds (Triess et al., 2019). It was previously introduced in (Ledig et al., 2017a) to provide a qualitative measure for realism in RGB images. In contrast to (Ledig et al., 2017a), where untrained people were used to determine the realism, (Triess et al., 2019) requires LiDAR experts for the testing process to assure a high enough sensor domain familiarity of the test persons. This makes the process even more time-consuming and expensive. Furthermore, the subjective nature of qualitative measures in general makes it difficult to compare performances across different works, even when a large inspection group, such as Mechanical Turk, is used. Therefore, quantitative metrics are crucial.

2.1.2 Quantitative Evaluation

Quantitative evaluation is performed over a large collection of examples, often in an automated fashion. Table 1 categorizes a number of quantitative GAN measures into six categories according to their properties.
Feature-based (Salimans et al., 2016; Gurumurthy et al., 2017; Heusel et al., 2017; Che et al., 2017; Zhou et al., 2018; Shu et al., 2019): Feature-based metrics measure the realism of the data by computing a distance in high-dimensional feature spaces. The **I**nception **S**core (IS) (Salimans et al., 2016) and the ***F**réchet **I**nception **D**istance* (FID) (Heusel et al., 2017) are the two most popular metrics and extract their fea-

Table 1 GAN evaluation measures: This table categorizes GAN evaluation measures and states their most important pros and cons according to our application

Category	Metric Examples	⊕	⊖
Feature-based	IS (Salimans et al., 2016), Modified IS (Gurumurthy et al., 2017), Mode Score (Che et al., 2017), AM Score (Zhou et al., 2018), FID (Heusel et al., 2017), FPD (Shu et al., 2019)	used in many papers with pre-trained models available	based on features from non-LiDAR datasets (i.e.ImageNet (Deng et al., 2009) and ShapeNet (Chang et al., 2015))
Distribution-based	Average Log-Likelihood (Goodfellow et al., 2014; Theis et al., 2016), Coverage (Tolstikhin et al., 2017), MMD (Gretton et al., 2012; Achlioptas et al., 2018), BPT (Arora et al., 2018), NDB (Richardson & Weiss, 2018)	independent of data modality, capture sample diversity and mode collapse	manual checkpoint selection, no absolute measure, (additional visual inspection)
Classification	Wasserstein Critic (Arjovsky et al., 2017), Classification Performance (Radford et al., 2016; Isola et al., 2017), Boundary Distortion (Santurkar et al., 2018), C2ST (Lehmann & Romano, 2006), AAD (Yang et al., 2017)	independent of data modality	freshly trained discriminators for each test on held-out data, no absolute measure
Output Comparison	IRP (Wang et al., 2016), Reconstruction Error (Xiang & Li, 2017)	independent of data modality, per-sample score	high run-time because of nearest neighbor matching
Model Comparison	GAM (Im et al., 2016), TWRSK (Olsson et al., 2018), NRDS (Zhang et al., 2018)	compare different GAN models against each other	labor intensive, high complexity
	Precision, Recall, F1 Score	simple and fast to compute	only relative performance of discriminator to generator
Low-Level Statistics	SSIM (Wang et al., 2004), PSNR, sharpness, contrast, mean power spectrum	simple and fast to compute	specific for camera images, no higher-level information

tures from the ImageNet dataset (Deng et al., 2009). This makes them exclusively applicable to camera image data. The ***F**réchet **P**oint **C**loud **D**istance* (FPD) (Shu et al., 2019) is applicable to single-object point clouds, as it is based on features from the PointNet dataset (Charles et al., 2017). In contrast to our method, these measures require labels on the target domain to train the feature extractor, cannot handle variable sized point clouds, and do not provide local scores. Further, it is only possible to compare a sample to one particular distribution and therefore makes it difficult to obtain a reliable measure on unseen data.

Distribution-based (Goodfellow et al., 2014; Theis et al., 2016; Tolstikhin et al., 2017; Gretton et al., 2012; Achlioptas et al., 2018; Arora et al., 2018; Richardson & Weiss, 2018): Most distribution-based measures are independent of the data modality and thus can be used to evaluate GANs operating on point clouds. They successfully capture the sample diversity and mode collapse of the model, but cannot determine the realism of a single sample. Most approaches are labor intensive as they require manual checkpoint selection and several runs over the test data.

Classification (Arjovsky et al., 2017; Radford et al., 2016; Isola et al., 2017; Santurkar et al., 2018; Lehmann & Romano, 2006; Yang et al., 2017): Another common approach is to use classification networks to assess the quality of GAN outputs. Two-Sample Test (C2ST), for example, assesses whether two samples are drawn from the same distribution. This requires freshly trained discriminators for each test on a held-out subset of the data.

Output comparison (Wang et al., 2016; Xiang & Li, 2017): Among others, computing reconstruction errors is one common method to assess generated data. For point clouds, EMD and ***C**hamfer's **D**istance* (CD) are often used, as they can operate in a permutation-invariant fashion. These metrics also serve as a basis for some distribution-based measures, such as coverage or **M**inimum **M**atching **D**istance (MMD) (Achlioptas et al., 2018). Caccia et al. (2019) use EMD and CD directly as a measure of reconstruction quality on entire scenes captured with a LiDAR scanner. However, this is only applicable to paired translation GANs or supervised approaches, because it requires a known target to measure the reconstruction error.

Model comparison (Im et al., 2016; Olsson et al., 2018; Zhang et al., 2018): There exist two types of model comparison techniques. The first includes simple metrics that capture the performance of the discriminator relative to the current

state of the generator. The other type focuses on the evaluation of sample diversity and comparison between several GAN architectures. However, these measures are labor intensive and of high complexity as they often require several network combinations and trainings.
Low-level statistics (Khrulkov et al., 2018; Wang et al., 2004): Computing low-level statistics of the underlying data is easy and fast. However, statistics like **S**tructural **S**imilarity **I**ndex **M**easure (SSIM), **P**eak **S**ignal-to-**N**oise **R**atio (PSNR), sharpness, or contrast are specific for RGB images and not capable to capture higher-level information.
This work aims at providing a practical quantitative metric to determine the realism of individual generated samples via learned features. Therefore, we consider our proposed method as a combination of the following categories: feature-based, distribution-based, and output comparison.

2.2 Metric Learning

The goal of deep metric learning is to learn a feature embedding, such that similar data samples are projected close to each other while dissimilar data samples are projected far away from each other in the high-dimensional feature space. Common methods use siamese networks trained with contrastive losses to distinguish between similar and dissimilar pairs of samples (Chicco, 2021). Thereupon, triplet loss architectures train multiple parallel networks with shared weights to achieve the feature embedding (Hoffer & Ailon, 2015; Dong & Shen, 2018). This work uses an adversarial training technique to push features in a similar or dissimilar embedding.

3 Method

3.1 Objective and Properties

The aim of this work is to provide a method to estimate the level of realism for arbitrary LiDAR point clouds. We design the metric to learn relevant realism features directly from distributions of real-world data. The output of the metric can then be interpreted as a distance measure between the input and the learned distribution in a high dimensional space.

Based on the discussed aspects of existing point cloud and GAN measures, we expect a useful LiDAR point cloud metric to be:
Quantitative: The realism score is a quantitative measure that determines the distance of the input sample to the internal representation of the learned realistic distribution. The score S^{Real} has well defined lower and upper bounds that reach from 0 (unrealistic) to 1 (realistic).
Universal: The metric has to be applicable to any LiDAR input and therefore must be independent from any application or task. This means no explicit ground truth information, such as class labels or bounding boxes, is required.
Transferable: The metric must give a reliable and robust prediction for all inputs, independent of whether the data distribution of the input sample is known by the metric or not. This makes the metric transferable to new and unseen data.
Local: The metric should be able to compute spatially local realism scores for smaller regions within a point cloud. These scores can then be combined with additional information, such as motion, semantics, or distance to provide a detailed analysis of the data. The metric is also expected to focus on identifying the realism of the point cloud properties while ignoring global scene properties as much as possible to reduce domain biases.
Flexible: Point clouds are usually sets of un-ordered points with varying size. Therefore, it is crucial to have a processing that is permutation-invariant and independent of the number of points to process.
Simple: Easy applicability and a fast computation time allows the metric to run in parallel to the training of a neural network for LiDAR data generation. This enables monitoring the realism of the generated sample during the training of the network.

We implement our metric in such a way that the described properties are fulfilled. To differentiate the metric from a GAN discriminator, we emphasize that a discriminator is not *transferable* to unseen data, since it recognizes only one specific data distribution to be realistic.

3.2 Architecture

Figure 2 shows the architecture of our approach. The following describes the components and presents how each part is designed to contribute towards achieving the desired metric properties. The underlying idea of the metric design is to compute a distance measure between different data distributions of *realistic* and *unrealistic* LiDAR point cloud compositions. The network learns features indicating realism from data distributions by using a proxy classification task. Specifically, the network is trained to classify point clouds from different datasets into three categories: *Real*, *Syn*, *Misc*. The premise is the possibility to divide the probability space of LiDAR point clouds into those that derive from real-world data (*Real*), those that derive from simulations (*Syn*), and all the others (*Misc*), e.g.distorted or randomized data. Refer to Fig. 1 for an impression. By acquiring the prior information about the tripartite data distribution, the metric does not require any target information or labels for inference.

The features are obtained with hierarchical feature set learning, explained in Sect. 3.2.1. Section 3.2.2 outlines our adversarial learning technique.

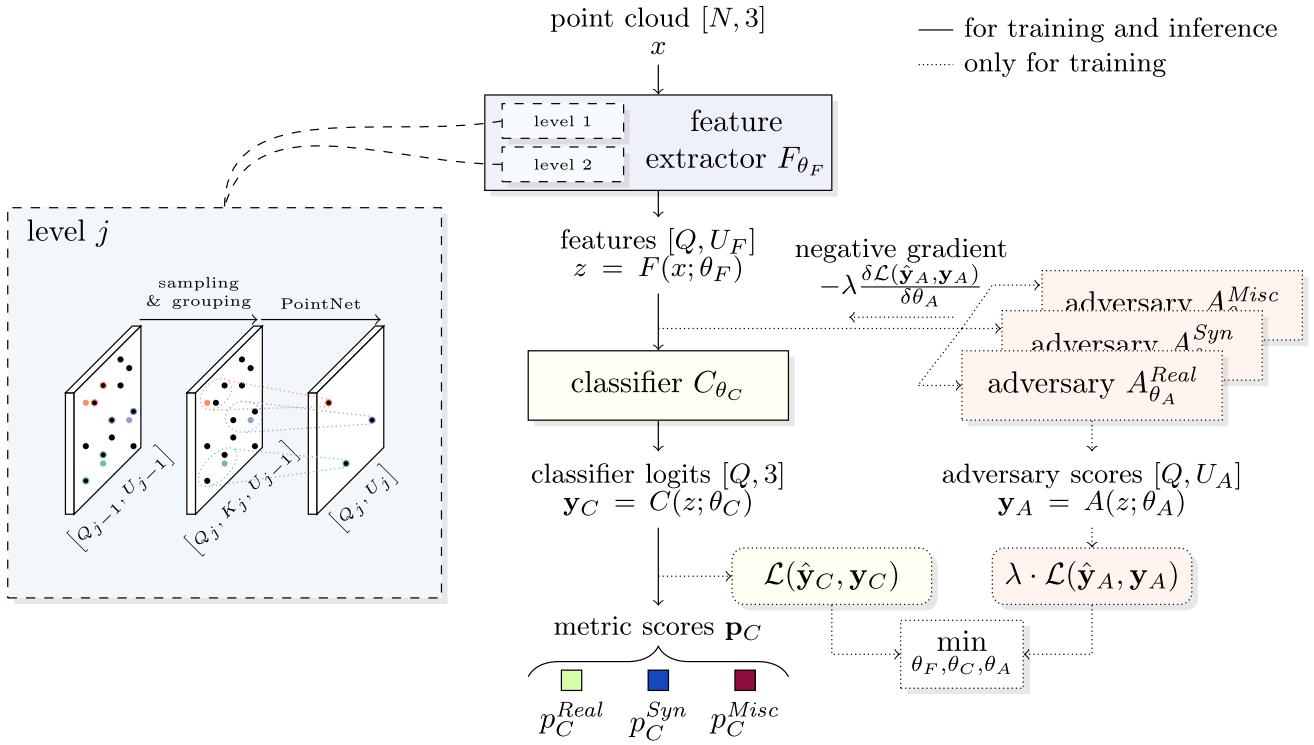


Fig. 2 Architecture The feature extractor F_{θ_F} uses hierarchical feature set learning from PointNet++ (Qi et al., 2017) to encode information about each of the Q query points and their K nearest neighbors. The neighborhood features z are then passed to the classifier C_{θ_C} which outputs probability scores $\mathbf{p}_C$ for each category (*Real*, *Syn*, *Misc*). In training, z is fed to the adversaries A_{θ_A}, which output probability scores $\mathbf{p}_A$ for each dataset of their respective category. For the classifier and all three adversaries a multi-class cross-entropy loss is minimized. For C to perform as good as possible while A should perform as bad as possible, the gradient is inverted between the adversarial input and the feature extractor (Beutel et al., 2017). λ is a factor that regulates the influence of the adversarial loss, weighting the ratio of accuracy versus fairness. In our experiments we use a factor of $\lambda = 0.3$

3.2.1 Feature Extractor

The blue parts of Fig. 2 visualize the PointNet++ (Qi et al., 2017) concept of the feature extractor F_{θ_F}. It has two abstraction levels, sampling $Q_1 = 2048$ and $Q_2 = 256$ query points with $K_1 = 20$ and $K_2 = 10$ nearest neighbors (KNN), respectively. Keeping the number of neighbors and abstraction levels low limits the network to only encode information about *local* LiDAR-specific statistics instead of global scenery information. On the other hand, the high amount of query points helps to cover many different regions within the point cloud and guarantees the *local* aspect of our method. In contrast to PointNet++, we use KNN search instead of radius search to find the neighboring points. PointNet++ was proposed for point clouds from the ShapeNet dataset (Chang et al., 2015), which have uniformly sampled points on object surfaces. In LiDAR point clouds, points are not uniformly distributed and with increasing distance to the sensor, also the distance between neighboring points increase. Therefore, we found KNN search more practical to obtain meaningful neighborhoods in LiDAR scans compared to radius search.

In each abstraction level, we use a 3-layer MLP with filter sizes of [64, 64, 128] and [128, 128, 256], respectively. This results in the neighborhood features $z = F(x, \theta_F)$ of size $[Q, U_F]$ with $U_F = 256$ features for each of the $Q = 256$ query points. The features z are then fed to a densely connected classifier C_{θ_C} (yellow block). It consists of a hidden layer with 128 units, to which 50% dropout is applied during training, and the output layer with U_C units.

The classifier output is a probability vector $\mathbf{p}_{C,q} = \text{softmax}(y_C) \in [0, 1]^{U_C}$ per query point q. The vector has $U_C = 3$ entries for each of the categories *Real*, *Syn* and *Misc*. The component $p^{Real}_{C,q}$ quantifies the degree of realism in each local region q. The scores $\mathbf{S} = \frac{1}{Q} \sum_q \mathbf{p}_{C,q}$ for the entire scene are given by the mean over all query positions. Here, S^{Real} is a measure for the degree of realism of the entire point cloud. A score of 0 indicates low realism while 1 indicates high realism.

3.2.2 Adversarial Training

To obtain a *transferable* metric network, our metric leverages a concept often used to design fair network architectures or domain losses (Beutel et al., 2017; Raff & Sylvester, 2018). The idea is to force the feature extractor to encode only information into the latent representation z that is relevant for the realism estimation. This means, we actively discourage the feature extractor from encoding information that is specific to the distribution of a single dataset. In other words—using fair networks terminology (Beutel et al., 2017)—we treat the concrete dataset name as a sensitive attribute. With this procedure we can improve the generalization ability towards unknown data.

To achieve this behavior, we add a second output path for adversarial learning that consists of one adversary A_{θ_A} for each category (see orange parts in Fig. 2). Each of the adversaries predicts classification probabilities for all the datasets in their respective category. To simplify the following explanation, we assume there is only one adversary. The architecture of the adversary is identical to the one of the classifier, except for the number of units in the output layer U_A, which depends on the number of training datasets for the respective category ($U_A^{Real} = 2$, $U_A^{Syn} = 2$, $U_A^{Misc} = 3$). Following the designs proposed in (Beutel et al., 2017; Raff & Sylvester, 2018), we train all network components by minimizing the losses for both heads, $\mathcal{L}_C = \mathcal{L}\left(\mathbf{y}_C, \hat{\mathbf{y}}_C\right)$ and $\mathcal{L}_A = \mathcal{L}\left(\mathbf{y}_A, \hat{\mathbf{y}}_A\right)$, but reversing the gradient in the path between the adversary input and the feature extractor. The goal is for C to predict the category $\mathbf{y}_C$ and for A to predict the dataset $\mathbf{y}_A$ as good as possible, but for F to make it hard for A to predict $\mathbf{y}_A$. Training with the reversed gradient results in F encoding as little information as possible for predicting $\mathbf{y}_A$. The training objective is formulated as

$$\begin{aligned}\min_{\theta_F,\theta_C,\theta_A} \; & \mathcal{L}\Big(C\big(F(x;\theta_F);\theta_C\big), \hat{y}_C\Big) \\ & + \mathcal{L}\Big(A\big(J_\lambda[F(x;\theta_F)];\theta_A\big), \hat{y}_A\Big)\end{aligned} \tag{1}$$

with θ being the trainable variables and J_λ a special function

$$J_\lambda[F] = F \quad \text{but} \quad \nabla J_\lambda[F] = -\lambda \cdot \nabla F \tag{2}$$

such that the forward pass is an identity function while the gradient is inverted in the backward pass while training. The factor λ determines the ratio of accuracy and fairness.

In the applications of the related literature (Beutel et al., 2017; Raff & Sylvester, 2018), the sensitive attribute and the requested attribute are often correlated but have no direct coupling. In our case, this would mean that different data samples from the same dataset could belong to multiple categories. But this is not the case, instead samples from one dataset always belong to the same category. Therefore, our sensitive attribute, the dataset, always directly determines the requested attribute, the category. A single adversary would now suppress all information of the sensitive attribute, thus also suppresses important information to obtain the requested attribute which then leads to unwanted decline in classifier performance. Therefore, a separate adversary for each category is needed, such that only the sensitive information regarding the dataset is suppressed, while keeping the requested information about the category intact. The adversaries $A : \{A^{Real}, A^{Syn}, A^{Misc}\}$ have the trainable variables $\theta_A : \{\theta_A^{Real}, \theta_A^{Syn}, \theta_A^{Misc}\}$. Each adversary outputs estimates for only the datasets of their respective category. This forces the feature extractor to encode only common features within one category, while not removing important features from other categories. The loss is now defined as $\mathcal{L}_A = \mathcal{L}_{A^{Real}} + \mathcal{L}_{A^{Syn}} + \mathcal{L}_{A^{Misc}}$.

4 Experimental Setup

4.1 Datasets

Table 2 shows the datasets used for this work. We use two different groups of datasets, one that is used to train and evaluate the metric while the other group is only used for evaluation. With the strict separation of training and evaluation datasets, additionally to the training and test splits, we demonstrate that our method is a useful measure on unknown data distributions. In both cases alike, the datasets stem from one of three categories: *Real*, *Syn*, *Misc*.

Within the *Real* category, publicly available real-world datasets are used for training (KITTI, nuScenes) and evaluation (PandaSet). For *Syn*, we use the CARLA simulator where we implement the sensor specifications of a Velodyne HDL-64 sensor to create ray-traced range measurements. GeoSet is the second dataset in this category. Here, simple geometric objects, such as spheres and cubes are randomly scattered on a ground plane in three dimensional space and ray-traced in a scan pattern. Additionally, we augment the synthetic data with little noise at training time, such that they are not trivially distinguishable from the other categories. For evaluation, we use the GTAV-LiDAR dataset (Hurl et al., 2019), which contains simulated LiDAR samples from the video game Grand Theft Auto V (GTA V). It has a large detailed world with realistic graphics, which provides a diverse data collection environment.

Finally, we add a third category, *Misc*, to allow the network to represent meaningless data distributions, as they often occur during GAN trainings or sensor failures. Therefore, *Misc* contains randomized data that is generated at training time. Misc 1 and Misc 2 are generated by linearly increasing the depth over the rows or columns of a virtual LiDAR scanner, respectively. Misc 3 is a simple Gaussian noise with

Table 2 Datasets: The table lists the datasets for each category

	Dataset	Samples	Train.	Eval.
Real	KITTI (Geiger et al., 2013)	18,329	✓	✓
	nuScenes (Caesar et al., 2020)	28,130	✓	✓
	PandaSet (Scale, 2020)	–	×	✓
Syn	CARLA (Dosovitskiy et al., 2017)	106,503	✓	✓
	GeoSet	18,200	✓	✓
	GTAV-LiDAR (Hurl et al., 2019)	–	×	✓
Misc	Misc 1,2,3	∞	✓	✓
	Misc 4	–	×	✓

The two rightmost columns show whether the dataset is used to train or evaluate the metric model. The number of samples used for testing is 1000 for all datasets. The number of training samples is listed in the middle column

varying standard deviations. Misc 4 is only used for evaluation and is created by setting patches of varying height and width of the LiDAR depth projection to the same distance. Varying degrees of Gaussian noise are added to the Euclidean distances of Misc {1, 2, 4}.

In addition to the training data listed in the tables, we use 1000 samples from a different split of each dataset to obtain our evaluation results. No annotations or additional information are required to train or apply the metric, all operations are based on the xyz coordinates of the point clouds.

4.2 Up-sampling models

We use the task of up-sampling to demonstrate the application of our metric. Up-sampling is a type of domain adaptation, where the source domain is the low resolution data and the target domain is the high resolution data. In contrast to more complex adaptations, such as simulation-to-real or sensor-to-sensor setups, we can focus on evaluating the actual data realism instead of additional domain gaps introduced by scene content. However, this is still a complex task, since the model must understand the scene in order to synthesize realistic high-resolution LiDAR outputs. This makes it an ideal testing candidate for our realism metric.

In Sect. 6 we compare the realism of generated samples from five different up-sampling methods to the target high-resolution. The generation process is based on cylindrical depth projections of the LiDAR point clouds, as proposed in (Triess et al., 2019). We compare two traditional methods, i.e.nearest neighbor and bilinear interpolation, and three learning-based methods. The generator of all three learning-based methods is adapted from the SRGAN architecture (Ledig et al., 2017a). One version is trained with an $\mathcal{L}_1$-loss, another with $\mathcal{L}_2$-loss, and the GAN uses an adversarial loss. The GAN discriminator is also adapted from (Ledig et al., 2017a). We conduct the experiments for 4× up-sampling in the vertical dimension. Implementation and training details can be found in the appendix.

4.3 Baselines

As baselines for our metric, we report the reconstruction errors of the up-sampled data. These errors can serve as an indication of the generation quality, but are usually not suitable as a metric for synthesized data, since they require a target sample. In our case, this target is the original high-resolution sample from which we generate the low-resolution sample as input to the up-sampling network. We compute the ***C**hamfer's **D**istance* (CD), ***M**ean* **A**bsolute Error (MAE), and **M**ean **S**quared **E**rror (MSE) between the predicted point cloud P^p and the target P^t. For CD, the point clouds are considered as un-ordered sets $P = \{p\}$, such that

$$d_{CD}(P^p, P^t) = \frac{1}{|P^p|} \sum_{p^p \in P^p} \min_{p^t \in P^t} \|p^p - p^t\|_2 + \frac{1}{|P^t|} \sum_{p^t \in P^t} \min_{p^p \in P^p} \|p^t - p^p\|_2 \tag{3}$$

while for MAE $= \|p^t_{ij} - p^p_{ij}\|_1$ and MSE $= \|p^t_{ij} - p^p_{ij}\|_2$, the point clouds are arranged as projected images $P = \{p_{ij}\}$ with the indices i and j for the respective row and column of the projection. Typical GAN evaluation measures for point cloud generation are Coverage (Tolstikhin et al., 2017) and MMD (Gretton et al., 2012). Both are based on finding the best match between the generated and the target point cloud. We can assume that the best match is always the original high-resolution image of the same scene, then the metrics simplify to Cov ≈ 1.0 and MMD ≈ d_{CD} due to our paired translation. Therefore, we do not report these metrics additionally to the reconstruction errors in the evaluation section.

4.4 Semantic Segmentation

The key application for our metric is to evaluate the generation capabilities of generative models to improve downstream perception. This enables checkpoint selection or early stopping of GAN trainings under the assumption that better data leads to better perception models. We investiagate this in our application experiments. Using the up-sampling models from Sect. 4.2, we transform data from the source (low-resolution) to the target (high-resolution) domain. This step generates pseudo-datasets of different quality for each method. We then use these pseudo-datasets to train semantic segmentation models which are finally evaluated on the target domain. It is expected that if the metric ranks the realism of a generated dataset higher than another one, training with this data also leads to better segmentation performance on the target domain. This is because the data is— per metric— more realistic, i.e.the domain gap is smaller (Triess et al., 2021a).

As a segmentation model, we use SqueezeSegV2 (Wu et al., 2019) and RangeNet21 (Milioto et al., 2019). Instead of the original 19 classes, we combine some of them and only predict 9 classes. Details on the architecture and training can be found in the appendix.

5 Metric Evaluation

5.1 Balance between Accuracy and Fairness

First, the metric has to be calibrated by choosing the correct factor λ of the adversarial loss during training. This is an important property which controls the ratio between accuracy and fairness. A well chosen factor will maximize the difference between a high classifier accuracy and a low adversary accuracy.

Figure 3 shows the classifier accuracy in black and the adversary accuracy in brown (weighted sum over the three category adversaries, shown as dashed lines). With increasing λ, the adversarial accuracy decreases slowly, while the classification accuracy suddenly drops. This happens because the classifier gradients are overruled by the reversed gradients of the adversary, hindering it from train properly. Interestingly, the adversarial part of the *Real* category is significantly more influenced by λ than those of the other two. One reason might be that the *Real* datasets in themselves are already very diverse, especially compared to the *Syn* or *Misc* datasets. The number of different sceneries is higher, but the most variance is caused by more diverse appearance of the same object types (e.g.pedestrians) and the additional sensor noise, which is not present in the *Syn* datasets. This makes it hard for the model to extract only realism relevant features in form of common information from the *Real* datasets while not removing any other relevant information. Thus, the model

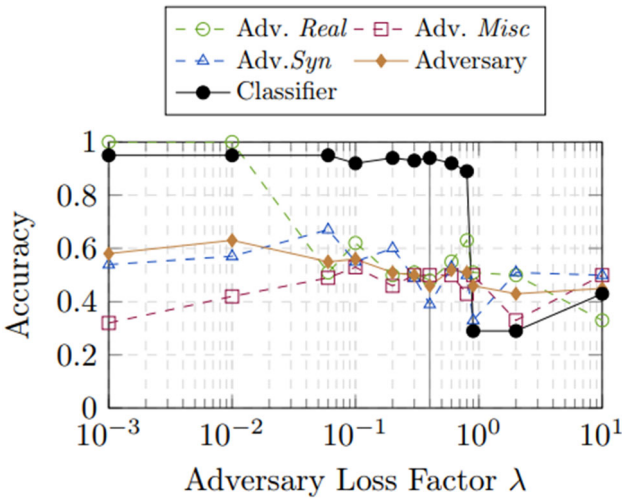


Fig. 3 Accuracy versus Fairness: Accuracy of classifier and adversaries over the loss factor λ. At small λ, the classification accuracy is high which means good performance. However, adversary accuracy is also quite high (at least for *Real*) which means no fairness in this part. With increasing λ the network gets fairer while maintaining its high level of classification accuracy. At a certain point the network becomes unstable and deteriorates into chance level performance in the classifier

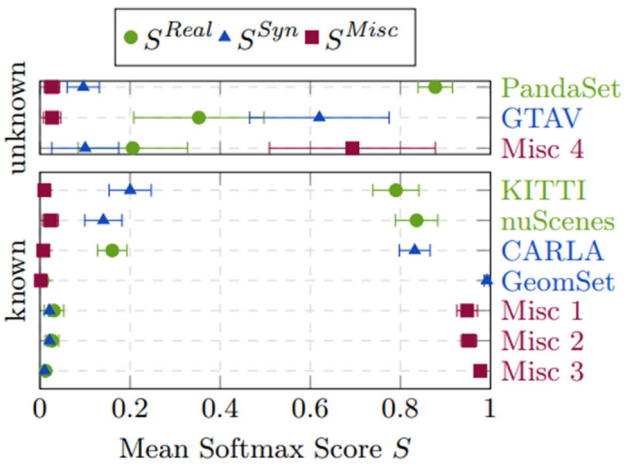
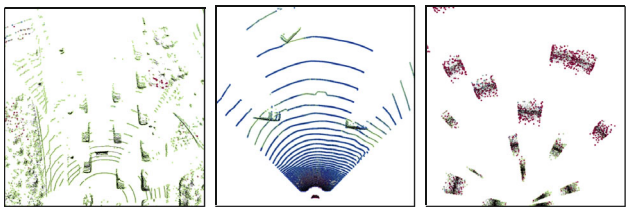


Fig. 4 Metric results: Shown is the metric output S for *Real*, *Syn*, and *Misc* on different datasets. The lower part shows the results for the test split of the known datasets, while the upper part depicts one unknown dataset from each category. The color of the dataset name indicates the respective category

requires more pressure in form of higher λ to accomplish this challenging task for the *Real* category, compared to *Syn* and *Misc*, where it is easier to extract common information while not removing any other relevant information.

We use a factor of $\lambda = 0.3$ for all further experiments in this paper (indicated by the gray vertical line). Here, the classifier has a good performance (93%) while the adversary operates slightly above chance level (50%).



(a) PandaSet **(b)** GTAV **(c)** Misc 4

Fig. 5 Qualitative performance on unknown data: The figure shows the metric results on three unknown datasets. **a** shows the PandaSet dataset as an example for *Real*. **b** shows the GTAV dataset for *Syn*. The overall high *Real* scores seem to be caused by regions that contain cars. **c** shows an example for the Misc 4 dataset

5.2 Overall Dataset Results

We run our metric network on the evaluation datasets, as well as on the test split of the training datasets. Figure 4 shows the mean of the metric scores S for each of the three categories. The known datasets (lower part) clearly achieve well-separated scores and predict their respective category, e.g.CARLA is classified with a high *Syn* score.

We obtain notable results on the unknown datasets (upper part). Qualitative example frames are depicted in Fig. 5. The *Real* dataset PandaSet behaves similar to the two known *Real* datasets, KITTI and nuScenes. This shows that the metric focused to encode realism relevant features from KITTI and nuScenes, such that PandaSet is easily categorized as such as well. The randomly generated Misc 4 dataset is correctly located within the *Misc* category, however with higher deviations in the scores, leading to *Misc* scores around 70% and *Real* scores around 20%. The deviations are caused by the high variance that was used to generate this dataset, where some regions have slightly higher *Real* or *Syn* scores.

The *Syn* dataset GTAV has a slightly different behavior. Here, S^{Syn} is around 60%, while the score for *Real* is around 35% and the deviation from those mean values is quite large. The reason for these high deviations and therefore lower *Syn* scores is a systematic behavior of the metric caused by the data distribution. Figure 5b shows that the high *Real* scores mainly stem from regions containing vehicles. GTAV has more detailed car models than CARLA which therefore appear almost like real vehicles in the point cloud. This example clearly demonstrates the benefit of the locality aspect of our metric which enables such detailed investigations.

5.3 Adversary Ablation

The proposed approach uses the adversarial loss to embed features for *Real*, *Syn*, and *Misc* while at the same time omit dataset-specific information as far as possible. To demonstrate the feature encoding behavior, we train additional

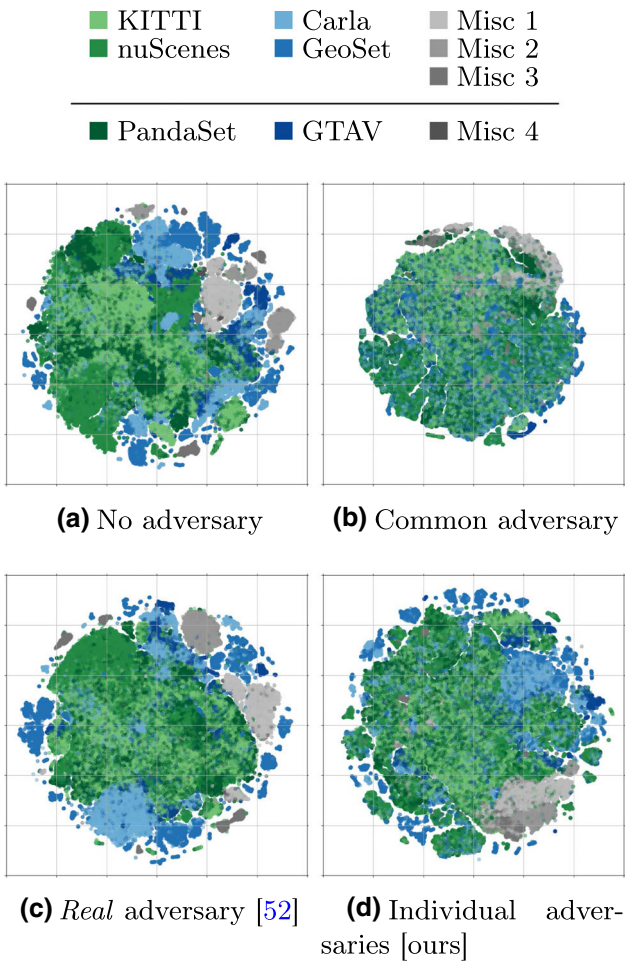


Fig. 6 Learned feature embedding: Shown are the t-SNE plots for the feature embedding z of four versions of the adversary configuration for the otherwise identical metric network. In **a** the model is trained without an adversary. **b** shows the features when a single adversary is used for training. **c** visualizes the features of our previous method (Triess et al., 2021b) that only used an adversary for the *Real* category. **d** depicts our approach, where one adversary per category is trained

metric networks with varying adversary configurations and visualize the learned features on the validation data.

Figure 6 shows plots of the t-SNE of the neighborhood features z. t-SNE is a dimensionality reduction method that tries to map data from a high dimension (z vector) to a low dimension (2D image) space while minimising information loss. Close points in the image have similar representations in z. Each metric category is represented by a different color, while the individual datasets are of different shades of this color. The darkest colors belong to the unknown datasets that were never seen by the metric network at training time, i.e.PandaSet, GTAV, Misc 4. We include them for demonstration purposes regarding the *transferability* to unseen data.

The two extreme cases of the configuration form Fig. 6a and b. Figure 6a represents the metric as a simple classifier without an adversary, where each shade of each color forms

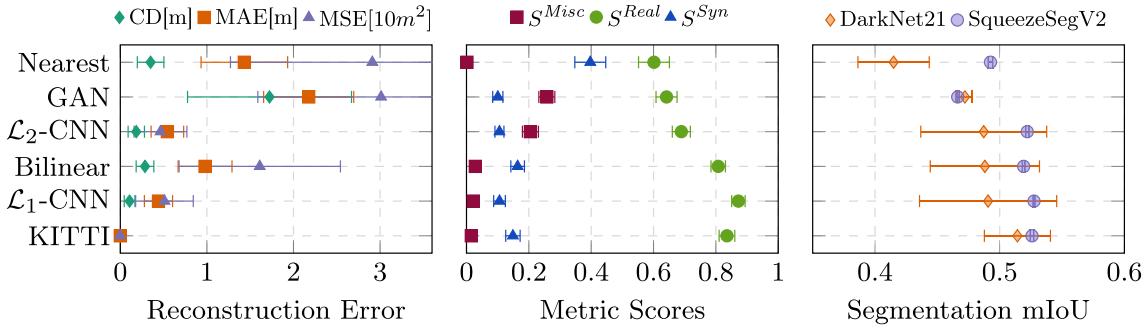


Fig. 7 Metric scores for up-sampling methods: The vertical axis lists five methods to perform 4× LiDAR scan up-sampling and the high-resolution target data ("KITTI"). The left plot shows the reconstructions errors of different baseline measures. The middle plot shows the three parts of our realism measure. The right plot shows the semantic segmentation results on the original KITTI dataset of a segmentation model trained with the data generated from the respective row. The methods are ordered from top to bottom by increasing human visual judgment ratings

their own clusters with little overlap to others. This means the features of each dataset are distinct and make it hard for the metric to estimate a reasonable score for unseen datasets. Figure 6b, on the other hand, uses one common adversary which leads to decreased classifier accuracy since features from all sources are forced into a common representation. This can be observed by the mixed colors with no clusters, not even between categories.

A useful metric requires a mix of the two versions above, where features of one category are similar and features from different categories are dissimilar. Therefore, we propose to use per-category adversaries. In our previous work (Triess et al., 2021b) the adversary was only applied for *Real*, as depicted in Fig. 6c. In this work we use one adversary for each category, as represented by Fig. 6d. In both cases the green colors of the *Real* datasets are clearly mixed, while at the same time being sufficiently distinguishable from the blue or gray clusters. However, our per-category approach (Fig. 6d) also shows mixed features among the blue and gray points, whereas our previous approach shows more distinct clusters. This is especially visible for *Misc*, where Fig. 6c has one cluster for each shade but our method better combines them.

Further, the feature visualization shows that the unknown dataset PandaSet is fully integrated into the *Real* cluster for our method, as opposed to when using no adversary. The clusters of the unknown GTAV dataset mostly overlap with *Syn*, but also partially with *Real*. This aligns with the metric results that we saw previously for GTAV, where parts of the data containing vehicles appear quite realistic.

We conduct the adversary ablation only qualitatively, because it is not possible to compare the quantitative scores of the different versions. A metric trained as in Fig. 6b could have a different allocation of scores in range [0, 1] than a metric as in Fig. 6d.

6 Metric Application

In this section we demonstrate how our realism measure ranks different datasets generated by neural networks. We then compare these results to our baseline evaluation measures (introduced in Sect. 4.3), and analyse the resulting performance of a segmentation network.

Figure 7 is divided into three parts horizontally. The leftmost plot shows the baseline metrics, the middle shows the results of our metric network, the rightmost plot shows the segmentation performance. The vertical axis on the left lists five different versions of KITTI data, generated as explained in Sect. 4.2. For the displayed segmentation results, different versions of the same model were trained with each of the datasets and then evaluated on the original KITTI data.

The realism score for the original KITTI is displayed for reference and has reconstruction errors of zero. The methods are ranked from top to bottom by increasing realism as approximately perceived by humans.[1] In general, the baseline metrics show a tendency but no clear correlation to the degree of realism and struggle to produce an unambiguous ordering of the methods. Our realism score, on the other hand, sorts the up-sampling methods according to human visual judgment. These results align with the ones in (Triess et al., 2019), which shows that a low reconstruction error does not necessarily imply high realism in the generated outputs. This is the main reason for the emergence of perceptual losses in recent years (Johnson et al., 2016; Ledig et al., 2017b).

The upper row of Fig. 8 shows an example scene for all up-sampling versions with their obtained scores. The $\mathcal{L}_1$-CNN produces an almost perfect version of the original high-resolution data, only with some noise at object boundaries.

[1] It is not clear how to rank the *nearest neighbor interpolation* here, since its appearance is completely different to the others. Therefore we simply placed it according to its metric score.

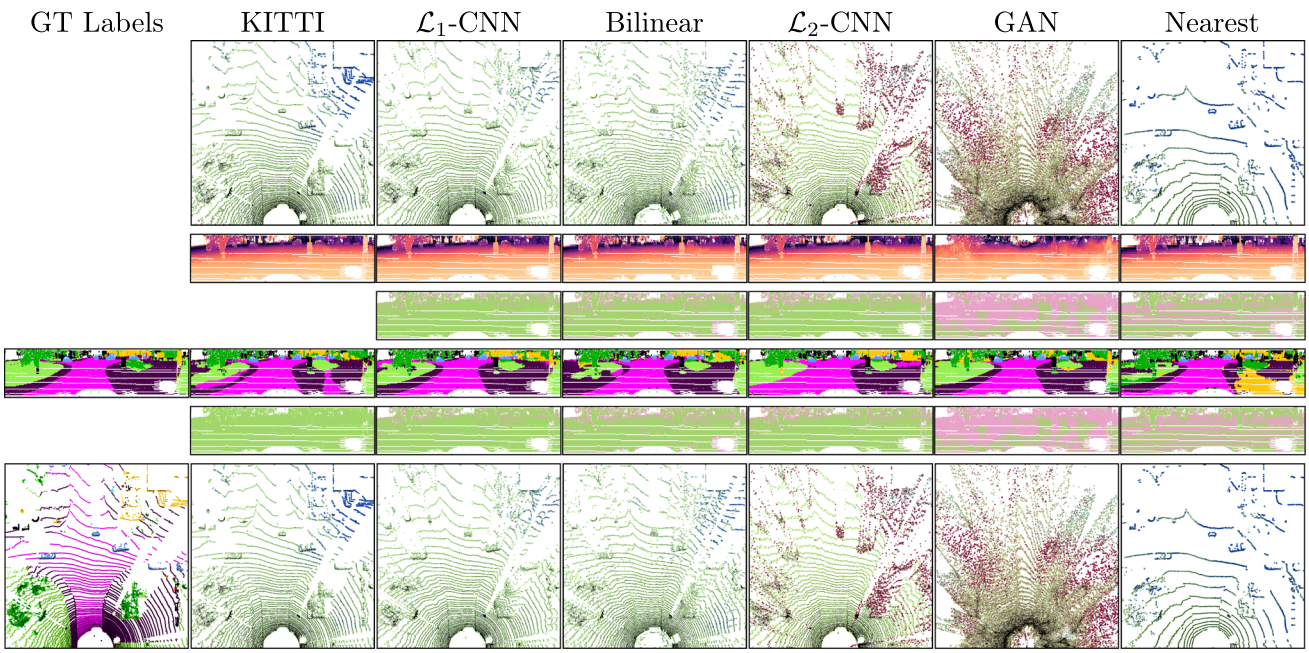


Fig. 8 Qualitative up-sampling and segmentation results: The first row shows the metric results on an up-sampled KITTI scene. The original scan is shown in column "KITTI". The colors are soft interpolations of *Real*, *Syn*, and *Misc*. The second row shows the color-coded depth projection of the point cloud. The third row shows the relative error between the generated sample and the original high resolution sample from column "KITTI". A pixel is green if the error is 0% and pink if the error is higher than 10%, all values in between are linearly interpolated. The fourth and sixth row show the segmentation results of a model trained on the respective up-sampled data. A legend of the semantic colors is provided in Table 3. The ground truth semantic labels are shown in the leftmost column. For better comparison, the fifth column shows correctly classified pixels in green and wrong classifications in pink. The visualized sample is from the validation split and was neither used to train the metric, nor the segmentation network

Table 3 Semantic segmentation performance: The table lists the evaluation results of the DarkNet21 model for point-wise semantic segmentation

KITTI Version	accuracy	mean IoU	person	two-wheeler	large-vehicle	vehicle	road	sidewalk	terrain	construction	vegetation
Class frequency $\frac{\text{train}}{\text{val}}$			$\frac{0.05}{0.16}$	$\frac{0.06}{0.12}$	$\frac{0.22}{0.10}$	$\frac{4.4}{6.2}$	$\frac{21.7}{18.9}$	$\frac{14.5}{12.1}$	$\frac{07.7}{12.9}$	$\frac{21.3}{14.3}$	$\frac{26.8}{30.3}$
Nearest	76.0	41.5	13.7	3.3	*0.5*	*89.0*	78.6	47.6	24.5	56.9	71.8
GAN	81.1	47.2	*16.5*	**9.3**	0.2	88.0	**86.5**	**71.1**	62.5	69.1	79.5
L2-CNN	82.8	48.7	13.5	1.9	**1.3**	**90.0**	82.7	62.0	66.1	69.8	79.1
Bilinear	83.2	48.8	12.5	4.5	0.3	88.6	84.5	67.4	*69.0*	*72.0*	*81.1*
L1-CNN	*83.5*	*49.1*	10.9	2.1	0.2	86.2	*84.7*	*68.1*	65.4	71.3	79.5
Original	**84.9**	**51.4**	**20.7**	*6.5*	*0.5*	87.0	84.5	67.7	**69.2**	**73.7**	**82.6**

For each row, the model is trained on the respective dataset which corresponds to a high-resolution KITTI variation generated from low-resolution data. The evaluation results are all reported on the validation split of the original KITTI data. All numbers in the table are given in %. Best results are shown in bold, second best in *italic*

Bilinear interpolation works very well on large surfaces, but produced single noise points especially in regions where the LiDAR usually receives no return, e.g.windows. The $\mathcal{L}_2$-CNN can reconstruct the outlines of the scene, but suffers from high noise throughout the entire point cloud. Similarly, the up-sampling GAN suffers from high noise, but often is not able to reconstruct the outlines of the scene and forms random point clusters instead of clear objects. The nearest neighbor interpolation causes vertically stretched objects, which works

fine for walls, poles, and other vertical objects, but fails for the ground.

These differences also cause different behavior in downstream perception in the target domain when the generated data is used for training. The rightmost plot in Fig. 7 shows the overall results, while Table 3 shows class-wise results. Additionally, the bottom row of Fig. 8 visualizes segmented example point clouds produced by the models trained with the respective data. Both segmentation models show similar trends for the order of the up-sampling methods as the realism metric. The slightly higher *Real* score for $\mathcal{L}_1$-CNN than for the original KITTI data can also be seen in the segmentation score of the SqueezeSegV2 model, but is neither significant nor does it behave in the same way for DarkNet21. Also the SqueezeSegV2 behavior on the nearest neighbor up-sampling is not equal to those of DarkNet21 and the metric. It can be assumed that two effects lead to this different behavior: First, as mentioned in footnote 1, it is not clear how exactly the nearest neighbor interpolation should be judged in terms of realism. Second, SqueezeSegV2 exhibits almost no variance on its performance scores. The combination of these two effects could cause the difference in behavior, but it is not quite clear how and therefore needs further investigation which is left for future work.

The class-wise results in Table 3 show that $\mathcal{L}_2$-CNN and GAN achieve quite good results for dynamic objects. At the same time, it is very hard to tell which of the point clusters in the 3D visualization of Fig. 8 belong to these objects. This raises the question why training with this highly distorted data achieves such good performance in the target domain. The question can be answered by looking at the projected LiDAR scan. Here it becomes visible that even regions that suffer from high noise can still be approximately detected by their edge outlines in the projection. The third row of Fig. 8 shows the point-wise relative error between the generated and the target point cloud with the error being clipped to a maximum of 10%. Even for the appearing noisy $\mathcal{L}_2$-CNN version, relative errors are quite low and therefore outlines are clearly visible in the depth projection (second row). We find that this is an indication that the segmentation model is not influenced by local noise perturbations, but rather learns a more generalized appearance of the object shapes.

7 Discussion

Our experiments show a correlation between measured training data realism and final perception performance. Qualitatively however, the segmentation performance seems to be less affected by reduced point cloud realism than expected by judging from the 3D images. We believe that this is caused by the selected architectures of the up-sampling and segmentation models. The segmentation networks operate on the 2D projections of the point clouds which is similar to the projection space used for up-sampling. Even though objects are blurred and unrecognizable when the GAN up-sampling is displayed as raw 3D data, objects shapes are still detectable on the 2D projections. We make two considerations from this observation:

First, visual judgment is highly dependent on the chosen data representation and their visualization. This is an important reason to use such a quantitative metric as ours on a large amount of data. Second, we believe that our metric might be more reliable to estimate the performance of downstream tasks operating on 3D space.

A major concept to keep in mind is the difference between domain gap and realism. If the task to solve is to train a method for KITTI-to-nuScenes adaptation, then both the target and the source domain are *Real*. Our metric can be used to rule out any unrealistic data compositions that form in the transition between those two datasets, e.g.while training a domain adaptation method. However, if the method is just outputting an identity function, the realism would be at maximum, while the domain gap still causes bad perception performance in the target domain. Therefore, tasks are only in parts dependent on the realism of the data and domain gaps have to be measured differently.

8 Conclusion

This paper presented a novel metric to quantify the degree of realism of local regions in LiDAR point clouds. Through adversarial learning, we obtain a feature encoding that is able to adequately capture data realism more generally instead of focusing on dataset-specific characteristic. In extensive experiments, we demonstrated the reliability and applicability of our metric on unseen data. The predictions of our method correlate well with visual judgment, unlike reconstruction errors serving only as a proxy for realism. In addition, we investigated the influence of data realism on a downstream perception task.

Future work includes to design a generative model that uses synthetic data, e.g.CARLA, as input to generate realistic real-world data, e.g.KITTI. Our metric is used in this setup to find the optimal point where the generated data actually improves the downstream perception performance of a segmentation or object detection model.

Funding Open Access funding enabled and organized by Projekt DEAL.

Table 4 Network architecture: Detailed network architecture and input format definition. The ID of each row is used to reference the output of the row. ↑ indicates that the layer directly above is an input. N denotes the number of LiDAR measurements. Q_j are the number of query points at abstraction level j. K_j are the number of nearest neighbors to search at abstraction level j. U are the number of output units of the classifier and the adversaries

ID	Inputs	Operation	Output shape	Description
1	LiDAR	x, y, z	$[N \times 3]$	Position of each point relative to sensor origin
Feature extractor: abstraction module 1				
2	↑, Q_1	Farthest point sampling	[2048]	Indices of Q_1 query points
3	1, ↑	Group	$[2048 \times 3]$	Grouped sampled points
4	1, 2, K_1	Nearest neighbor search	$[2048 \times 20]$	Indices of the K_1 nearest neigbors per query
5	1, 2, ↑	Group	$[2048 \times 20 \times 3]$	Grouped neighborhoods
6	↑	Neighborhood normalization	$[2048 \times 20 \times 3]$	Translation normalization towards query point
7	↑	(Conv+LeakyReLU) ×2	$[2048 \times 20 \times 64]$	Kernel size 1×1, stride 1
8	↑	Conv+LeakyReLU	$[2048 \times 20 \times 128]$	Kernel size 1×1, stride 1
9	↑	ReduceMax	$[2048 \times 128]$	Maximum over neighborhood features
Feature extractor: abstraction module 2				
10	3, Q_2	Farthest point sampling	[256]	Indices of Q_2 query points
11	3, 10, K_2	Nearest neighbor search	$[256 \times 10]$	Indices of the K_2 nearest neighbors per query
12	3, 10, ↑	Group	$[256 \times 10 \times 3]$	Grouped neighborhoods
13	↑	Neighborhood normalization	$[256 \times 10 \times 3]$	Translation normalization towards query point
14	9, 11	Group	$[256 \times 10 \times 128]$	Grouped features
15	13, ↑	Concat features	$[256 \times 10 \times 131]$	Grouped features with xyz
16	↑	(Conv+LeakyReLU) ×2	$[256 \times 10 \times 128]$	Kernel size 1×1, stride 1
17	↑	Conv+LeakyReLU	$[256 \times 10 \times 256]$	Kernel size 1×1, stride 1
18	↑	ReduceMax	$[256 \times 256]$	Maximum over neighborhood features → latent representation z
Classifier/adversary				
19	↑	Dense+LeakyReLU	$[256 \times 128]$	
20	↑	Dropout	$[256 \times 128]$	Dropout ratio 50%
21	↑	Dense	$[256 \times U_{C,A}]$	Output logits vector $y_{C,A}$
22	↑	Softmax	$[256 \times U_{C,A}]$	Output probability vector $p_{C,A}$

Appendix A Metric Implementation Details

Table 4 lists all layers, inputs, and operations of our **D**eep **N**eural **N**etwork (DNN) architecture. We use TensorFlow to implement online data processing, neural network weight optimization, and network inference. The implementation is oriented on the original PointNet++ implementation (Qi et al., 2017).[2] The Adam optimizer is used for optimization. We use an initial learning rate of $1e^{-3}$ with exponential warm-up and decay.

The classifier outputs the scores for each of the $U_C = 3$ categories, namely *Real*, *Syn*, *Misc*. The adversary for *Real* has $U_A^{Real} = 2$ output channels, for KITTI and nuScenes. The *Syn* adversary outputs $U_A^{Syn} = 2$ scores for CARLA and GeoSet. For the *Misc* category, the respective adversary has

[2] PointNet++ code https://github.com/charlesq34/pointnet2.

Table 5 SRGAN Generator Architecture: Detailed network architecture and input format definition of the SRGAN generator (Ledig et al., 2017a)

ID	Inputs	Operation	Output shape	Description
Input features from LiDAR scan				
1	LiDAR	x, y, z	$[N \times 3]$	Position of each point relative to sensor origin
2	↑	Projection $(x, y, z) \rightarrow (r, \varphi, \theta)$	$[H, W, 1]$	Cylindrical depth projection r with θ over H and φ over W
Residual blocks				
3	↑	Conv+ParametricReLU	$[H, W, 64]$	Kernel size 9×9, stride 1
4	↑	Conv+BN+ParametricReLU	$[H, W, 64]$	Kernel size 3×3, stride 1
5	↑	Conv+BN	$[H, W, 64]$	Kernel size 3×3, stride 1
6	↑, 3	Add	$[H, W, 64]$	Element-wise addition
7	↑	Repeat steps (4-6)	$[H, W, 64]$	$\times 16$ repetition of residual blocks
8	↑	Conv+BN	$[H, W, 64]$	Kernel size 3×3, stride 1
9	↑, 3	Add	$[H, W, 64]$	Element-wise addition
Super-resolution blocks				
10	↑	Conv	$[H, W, 256]$	Kernel size 3×3, stride 1
11	↑	SubpixelShuffle	$[2 \cdot H, W, 128]$	Reshape by moving values from the channel dimension to the spatial dimension
12	↑	ParametricReLU	$[2 \cdot H, W, 128]$	
13	↑	Repeat steps (10-12)	$[f_{\text{up}} \cdot H, W, 128]$	$\times \log_2 f_{\text{up}}$ repetition with f_{up} being the desired up-sampling factor, i.e. $f_{\text{up}} = \{2, 4, 8\}$
14	↑	Conv	$[f_{\text{up}} \cdot H, W, 1]$	Kernel size 9×9, stride 1

The ID of each row is used to reference the output of the row. ↑ indicates that the layer directly above is an input. N denotes the number of measured LiDAR points. H denotes the number of layers in the LiDAR sensor and W are the number of layer pulses fired per 360° revolution. The cylindrical depth projection is either retrieved directly from the raw image of the sensor or with a back-projection by computing (r, φ, θ) from (x, y, z). Missing measurements are set to a constant distance in the dense projection and are masked in the loss computation

Table 6 SRGAN discriminator architecture: Detailed network architecture and input format definition of the SRGAN discriminator (Ledig et al., 2017a)

ID	Inputs	Operation	Output shape	Description
1	LiDAR	r^{gt} or r^{hr}	$[f_{\text{up}} \cdot H, W, 1]$	High-resolution cylindrical depth projection
Conv blocks				
2	↑	Conv+LeakyReLU	$[f_{\text{up}} \cdot H, W, 64]$	Kernel size 3×3, stride 1
3	↑	Conv+BN+LeakyReLU	$[\frac{f_{\text{up}}}{2} H, \frac{1}{4} W, 64]$	Kernel size 5×5, strides 2×4
4	↑	Conv+BN+LeakyReLU	$[\frac{f_{\text{up}}}{2} H, \frac{1}{4} W, 128]$	Kernel size 3×3, stride 1
5	↑	Conv+BN+LeakyReLU	$[\frac{f_{\text{up}}}{4} H, \frac{1}{8} W, 128]$	Kernel size 3×3, stride 2
6	↑	Conv+BN+LeakyReLU	$[\frac{f_{\text{up}}}{4} H, \frac{1}{8} W, 256]$	Kernel size 3×3, stride 1
7	↑	Conv+BN+LeakyReLU	$[\frac{f_{\text{up}}}{4} H, \frac{1}{16} W, 256]$	Kernel size 3×3, strides 1×2
8	↑	Conv+BN+LeakyReLU	$[\frac{f_{\text{up}}}{4} H, \frac{1}{16} W, 512]$	Kernel size 3×3, stride 1
9	↑	Conv+BN+LeakyReLU	$[\frac{f_{\text{up}}}{8} H, \frac{1}{32} W, 512]$	Kernel size 3×3, stride 2
Reduction				
10	↑	Flatten	$[\frac{f_{\text{up}}}{2} \cdot H \cdot W]$	
11	↑	Dense+LeakyReLU	[1024]	
12	↑	Dense	[1]	

The input to the network is either the ground truth r^{gt} or the prediction from the generator r^{hr}

Table 7 Class mapping: This table shows the detailed class label mapping of the original dataset label ids to our custom mapping used for the segmentation experiments

Learned Classes	KITTI	nuScenes	CARLA
Unlabeled (0)	Unlabeled (0) outlier (1) on-rails (16, 256) other-vehicle (20, 259) other-structure (52) other-object (99)	Noise (0) animal (1) personal-mobility (5) stroller (7) wheelchair (8) barrier (9) debris (10) pushable-pullable (11) trafficcone (12) bicycle-rack (13) ambulance (19) police (20) trailer (22) other (29) ego-vehicle (31)	Unlabeled (0) other (3)
Person (1)	Person (30, 254) bicyclist (31, 253) motorcyclist (32, 255)	Adult (2) child (3) construction-worker (4) police-officer (6)	Pedestrian (4) rider (13)
Two-wheeler (2)	Bicycle (11) motorcycle (15)	Bicycle (14) motorcycle (21)	Two-wheeler (14)
Large-vehicle (3)	Bus (13, 257) truck (18, 258)	Bus (15, 16) construction vehicle (18) truck (23)	–
Vehicle (4)	Car (10, 252)	Car (17)	Car (10)
Road (5)	Road (40) parking (44) other-ground (49) lane-marking (60)	Driveable-surface (24) flat-other (25)	Road-line (6) road (7)
Sidewalk (6)	Sidewalk (48)	Sidewalk (26)	Sidewalk (8)
Terrain (7)	Terrain (72)	Terrain (27)	Terrain (15)
Construction (8)	Building (50) fence (51) pole (80) traffic-sign (81)	Manmade (28)	Building (1) fence (2) pole (5) wall (11) traffic-sign (12)
Vegetation (9)	Vegetation (70) trunk (71)	Vegetation (30)	Vegetation (9)

$U_A^{Misc} = 3$ outputs, for Misc 1,2,3. Implementation-wise, all adversaries have the full seven output channels for all datasets. The category split is implemented as a class weighting when computing the loss from the adversary output, such that the loss becomes zero if the input does not origin from within the respective category. We found this the easiest and most stable way to implement the desired behavior in TensorFlow graph mode.

Appendix B Up-Sampling Models

This section gives additional details on the up-sampling experiments for metric verification of Sec. 4.4 in the main paper. The up-sampling process is based on cylindrical depth projections of the LiDAR point clouds. Only the vertical resolution of the LiDAR images is enhanced. The bilinear interpolation is a traditional approach for which we directly used the resize method from TensorFlow (`tf.image.resize(images, size, method= ResizeMethod.BILINEAR)`). For all other experiments, we used the generator from the SRGAN architecture (Ledig et al., 2017a) and for the GAN experiments, also the discriminator architecture. After being processed by the super-resolution networks, the generated point clouds are converted back into lists of points and are fed to the metric network for realism judgement.

Table 5 lists all layers, inputs, and operations of the SRGAN generator architecture. In the $\mathcal{L}_{\{1,2\}}$-CNN trainings, a weighted $\mathcal{L}_\alpha$ loss is minimized. The objective is formulated as

$$\min_{\theta_G} \mathcal{L}_\alpha = \min_{\theta_G} \frac{1}{\alpha|\gamma|} \sum_{(i,j)\in\gamma} \left| r_{i,j}^{\text{gt}} - r_{i,j}^{\text{hr}} \right|$$

with the set of measured points γ, and r^{gt} being the high-resolution Ground Truth target and r^{hr} the prediction

$$r^{\text{hr}} = G_{\theta_G}\left(r^{\text{lr}}\right)$$

from the low-resolution input r^{lr}.

Table 6 lists all layers, inputs, and operations of the SRGAN discriminator architecture. Here, an adversarial loss,

defined as

$$\min_{\theta_G} \max_{\theta_D} \left\{ \log \left[D_{\theta_D} \left(r^{\text{gt}} \right) \right] + \log \left[1 - D_{\theta_D} \left(G_{\theta_G} \left(r^{\text{lr}} \right) \right) \right] \right\}$$

is minimized. The Adam optimizer is used for optimization with an initial learning rate of $1e^{-3}$.

Appendix C Segmentation

For both segmentation models, DarkNet21 and SqueezeSegV2, we use the original PyTorch implementation of Milioto et al. (Milioto et al., 2019)[3]. For our experiments, we modified the class labels to have the same classes for all used datasets. Table 7 shows the label mapping from the original dataset to our custom label set for all three datasets.

Appendix D Additional Results

Additionally to Fig. 4, Table 8 provides class-wise metric results for the datasets with semantic labels.

Table 8 Class-wise metric results: Shown are the class-wise averages of the metric output *S* for *Real*/*Misc*/*Syn*

	KITTI	nuScenes	CARLA
Person	.98/.00/.02	.75/.00/.25	.14/.00/.86
Two-wheeler	.96/.00/.04	.63/.00/.37	.05/.00/.95
Large-vehicle	–/ –/ –	.68/.00/.31	–/ –/ –
Vehicle	.92/.00/.08	.81/.00/.19	.07/.00/.93
Road	.91/.00/.09	.85/.00/.15	.02/.00/.98
Sidewalk	.88/.00/.12	.85/.00/.14	.08/.00/.92
Terrain	.93/.03/.05	.77/.04/.19	.36/.00/.64
Construction	.62/.00/.38	.76/.01/.22	.06/.00/.94
Vegetation	.87/.00/.13	.82/.07/.11	.05/.00/.95
Total	.80/.00/.20	.85/.02/.13	.16/.00/.84

In line "total", also scores with unknown semantic labels are included

References

Achlioptas, P., Diamanti, O., Mitliagkas, I. & Guibas, L. (2018). Learning representations and generative models for 3D point clouds. In *Proceedings of the international conference on learning representations (ICLR) workshops*.

Arjovsky, M., Chintala, S. & Bottou, L. (2017). Wasserstein GAN.

Arora, S., Risteski, A. & Zhang, Y. (2018). Do GANs learn the distribution? Some theory and empirics. In *Proceedings of the international conference on learning representations (ICLR)*.

Beutel, A., Chen, J., Zhao, Z. & Chi, E. H. (2017). Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations. In *Workshop on fairness, accountability, and transparency in machine learning*.

Borji, A. (2019). Pros and cons of GAN evaluation measures. In *Computer vision and image understanding (CVIU),* (pp. 41–65).

Caccia, L., van Hoof, H., Courville, A. & Pineau, J. (2019). Deep generative modeling of LiDAR Data. In *Proceedings of the IEEE international conference on intelligent robots and systems (IROS),* (pp. 5034–5040).

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 IEEE conference on computer vision and pattern recognition (CVPR),* (pp. 11618–11628).

Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L. & Yu, F.(2015). ShapeNet: An information-rich 3D model repository.

Charles, R. Q., Su, H., Kaichun, M. & Guibas, L.J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR),* (pp. 77–85).

Che, T., Li, Y., Jacob, A.P., Bengio, Y. & Li, W. (2017). Mode regularized generative adversarial networks. In *Proceedings of the international conference on learning representations (ICLR)*.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I. & Abbeel, P. (2016). InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems (NIPS)*.

Chicco, D. (2021). Siamese neural networks: an overview. In *Artificial neural networks,* (pp. 73–94).

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR),* (pp. 248–255).

Dong, X. & Shen, J. (2018). Triplet loss in siamese network for object tracking. In *Proceedings of the European conference on computer vision (ECCV),* (pp. 472–488).

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. & Koltun, V. (2017). CARLA: An open urban driving simulator. In *Proceedings of the 1st annual conference on robot learning,* (pp. 1–16).

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR), 32*(11), 1231–1237.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems (NIPS)*.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B. & Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research (JMLR),* (pp. 723–773).

Gurumurthy, S., Sarvadevabhatla, R. K. & Babu, R. V. (2017). DeLiGAN: Generative adversarial networks for diverse and limited data. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR),* (pp. 4941–4949).

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B. & Hochreiter, S.(2017). GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems (NIPS),* (pp. 6629–6640).

Hoffer, E. & Ailon, N. (2015). Deep metric learning using triplet network. In *Similarity-based pattern recognition (SIMBAD),* (pp. 84–92).

Huang, X., Li, Y., Poursaeed, O., Hopcroft, J. & Belongie, S. (2017). Stacked generative adversarial networks. In *Proceedings IEEE conference on computer vision and pattern recognition (CVPR),* (pp. 1866–1875).

[3] Code: https://github.com/PRBonn/lidar-bonnetal.

Hurl, B., Czarnecki, K. & Waslander, S.L. (2019). Precise Synthetic Image and LiDAR (PreSIL) Dataset for autonomous vehicle perception.

Im, D.J., Kim, C.D., Jiang, H. & Memisevic, R. (2016). Generating images with recurrent adversarial networks.

Isola, P., Zhu, J. Y., Zhou, T. & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*.

Johnson, J., Alahi, A. & Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of the European conference on computer vision (ECCV)*.

Khrulkov, V. & Oseledets, I. V. (2018). Geometry score: A method for comparing generative adversarial networks. In *Proceedings of the international conference on machine learning (ICML)*.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR),* (pp. 105–114).

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z. & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR),* (pp. 105–114).

Lehmann, E. L., & Romano, J. P. (2006). *Testing statistical hypotheses*. Springer.

Li, D., Ling, H., Kim, S. W., Kreis, K., Barriuso, A., Fidler, S. & Torralba, A. (2022). BigDatasetGAN: Synthesizing ImageNet with pixel-wise annotations. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*.

Lin, Z., Khetan, A., Fanti, G. & Oh, S. (2018). PacGAN: The power of two samples in generative adversarial networks. In *Advances in neural information processing systems (NIPS),* (pp. 324–335).

Löhdefink, J. & Fingscheidt, T. (2022). Improving performance of semantic segmentation CycleGANs by noise injection into the latent segmentation space.

Lucic, M., Kurach, K., Michalski, M., Gelly, S. & Bousquet, O. (2018). Are GANs created equal? A large-scale study. In *Advances in Neural Information Processing Systems (NIPS),* (pp. 698–707).

Mathieu, M.F., Zhao, J.J., Zhao, J., Ramesh, A., Sprechmann, P. & LeCun, Y. (2016). Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural information processing systems (NIPS),* (pp. 5047–5055).

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

Olsson, C., Bhupatiraju, S., Brown, T., Odena, A. & Goodfellow, I. (2018). Skill rating for generative models.

Park, T., Liu, M., Wang, T. & Zhu, J. (2019). Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*.

Qi, C. R., Yi, L., Su, H. & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems (NIPS)*.

Radford, A., Metz, L. & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Raff, E. & Sylvester, J. (2018). Gradient reversal against discrimination: A fair neural network learning approach. In *Proceedings of IEEE international conference on data science and advanced analytics (DSAA),* (pp. 189–198).

Richardson, E. & Weiss, Y. (2018). On GANs and GMMs. In *Advances in neural information processing systems (NIPS),* (pp. 5852–5863).

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X. & Chen, X. (2016). Improved techniques for training GANs. In *Advances in neural information processing systems (NIPS),* (pp. 2234–2242).

Sallab, A. E., Sobh, I., Zahran, M. & Essam, N. (2019). LiDAR Sensor modeling and Data augmentation with GANs for Autonomous driving. In *Proceedings of the international conference on machine learning (ICML) workshops*.

Santurkar, S., Schmidt, L. & Madry, A. (2018). A classification-based study of covariate shift in GAN distributions. In *Proceedings of the international conference on machine learning (ICML),* (pp. 4480–4489).

Scale AI: PandaSet (2020), https://pandaset.org

Shu, D., Park, S. W. & Kwon, J. (2019). 3D Point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE international conference on computer vision (ICCV),* (pp. 3858–3867).

Srivastava, A., Valkov, L., Russell, C., Gutmann, M. U. & Sutton, C. (2017). VEEGAN: Reducing mode collapse in GANs using implicit variational learning. In *Advances in neural information processing systems (NIPS)*

Theis, L., van den Oord, A. & Bethge, M. (2016). A note on the evaluation of generative models. In *Proceedings of the international conference on learning representations (ICLR)*.

Tolstikhin, I. O., Gelly, S., Bousquet, O., Simon-Gabriel, C. J. & Schölkopf, B. (2017). AdaGAN: Boosting generative models. In *Advances in neural information processing systems (NIPS),* (pp. 5424–5433).

Triess, L. T., Dreissig, M., Rist, C. B. & Zöllner, J. M. (2021). A survey on deep domain adaptation for LiDAR perception. In *Proceedings of IEEE intelligent vehicles symposium (IV) workshops*.

Triess, L. T., Peter, D., Baur, S. A., & Zöllner, J. M. (2021). Quantifying point cloud realism through adversarially learned latent representations. In *Proceedings of the German conference on pattern recognition (GCPR)*.

Triess, L. T., Peter, D., Rist, C. B., Enzweiler, M. & Zöllner, J. M. (2019). CNN-based synthesis of realistic high-resolution LiDAR data. In *Proceedings of IEEE intelligent vehicles symposium (IV),* (pp. 1512–1519)

Wang, Y., Zhang, L. & van de Weijer, J. (2016). Ensembles of generative adversarial networks. In *Advances in neural information processing systems (NIPS) workshops*.

Wang, Z., Bovik, A., Sheikh, H., & Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transaction on Image Processing (TIP), 13*(4), 600–612.

Wu, B., Zhou, X., Zhao, S., Yue, X. & Keutzer, K. (2019). SqueezeSegV2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud. In *Proceedings of IEEE international conference on robotics and automation (ICRA)*.

Xiang, S. & Li, H. (2017). On the effects of batch and weight normalization in generative adversarial networks.

Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F. & Weinberger, K.Q. (2018). An empirical study on evaluation metrics of generative adversarial networks.

Xu, Y., He, F., Du, B., Zhang, L. & Tao, D. (2021). Self-ensembling GAN for cross-domain semantic segmentation.

Yang, J., Kannan, A., Batra, D. & Parikh, D. (2017). LR-GAN: Layered recursive generative adversarial networks for image generation. In *Proceedings of the international conference on learning representations (ICLR)*.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X. & Metaxas, D. (2017). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision (ICCV),* (pp. 5908–5916).

Zhang, Z., Song, Y. & Qi, H. (2018). Decoupled learning for conditional adversarial networks. In *Proceedings of the IEEE winter conference on applications of computer vision (WACV),* (pp. 700–708).

Zhou, Z., Cai, H., Rong, S., Song, Y., Ren, K., Zhang, W., Wang, J. & Yu, Y. (2018). Activation maximization generative adversarial nets. In *Proceedings of the international conference on learning representations (ICLR)*.