LIDAR DOMAIN ADAPTATION

—

AUTOMOTIVE 3D SCENE UNDERSTANDING

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

von der KIT-Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation

von

M.Sc. Larissa T. Triess

Tag der mündlichen Prüfung:	20.12.2022
Referent:	Prof. Dr. J. Marius Zöllner
Korreferent:	Prof. Dr. rer. nat. Markus Enzweiler

ABSTRACT

Environment perception and scene understanding play essential roles in autonomous vehicles. A vehicle needs to be aware of the geometry and semantics of its surroundings to predict the behavior of other traffic participants and localize itself in the drivable space to navigate properly. Today, virtually all modern perception systems for automated driving are based on deep learning methods. They require tremendous amounts of data with matching annotations to be trained. Obtaining the data is relatively easy, since it only requires a vehicle equipped with the correct sensors to drive around. However, generating annotations is a very time consuming and expensive process. To make this challenge even more difficult, autonomous vehicles are required to operate virtually anywhere (e.g. Europe and Asia, rural and urban) and anytime (e.g. day and night, summer and winter, rain and fog). This requires the data to cover an even larger amount of different scenarios and domains.

In practice it is not feasible to collect and annotate data for such a variety of domains. However, training only with data from one domain will result in bad performance in a different target domain, due to domain gaps in the data. For a safety-critical application this is not acceptable. The field of domain adaptation introduces methods that help to close these domain gaps without the usage of annotations from the target domain and thus work towards designing scalable perception systems. The majority of domain adaptation works focus on two-dimensional camera perception. In autonomous vehicles, however, the three-dimensional scene understanding is essential, for which **Li**ght **D**etection **a**nd **R**anging (LiDAR) sensors are used widely nowadays.

This dissertation addresses domain adaptation for LiDAR perception from multiple perspectives. First, a number of techniques are introduced that improve the performance and run-time of semantic segmentation systems. The obtained insights are integrated into the perception model which is used throughout this dissertation to evaluate the effectiveness of the proposed domain adaptation approaches. Second, existing approaches are discussed and gaps in research are presented by a formulation of open research questions. To answer some of these questions, this dissertation presents a novel quantitative LiDAR metric. This metric allows to estimate the realism of LiDAR data, which is decisive for the performance of a perception system. Thus, the metric is used to evaluate the quality of LiDAR point clouds that are generated for the purpose of domain mapping, where data is transferred from one domain to another. This allows to re-use labels obtained from a source do-

main in the target domain without any additional annotations. In a different take on domain adaptation, this dissertation proposes a novel method that uses the geometry of the scene to learn domain-invariant features. The geometric information helps to improve the domain adaptation capabilities of the segmentation model and obtains state-of-the-art performance without any additional overhead in the inference. In the end, a novel method to generate semantically meaningful object shapes from continuous descriptions is proposed, which can – with additional work – be used to augment scenes to improve the detection capabilities of the models. To summarize, this dissertation presents a comprehensive framework for domain adaptation and semantic segmentation of LiDAR point clouds in the context of autonomous driving.

ZUSAMMENFASSUNG

Umgebungswahrnehmung und Szeneverständnis spielen bei autonomen Fahrzeugen eine wesentliche Rolle. Ein Fahrzeug muss sich der Geometrie und Semantik seiner Umgebung bewusst sein, um das Verhalten anderer Verkehrsteilnehmer:innen vorherzusagen und sich selbst im fahrbaren Raum zu lokalisieren, um somit richtig zu navigieren. Heutzutage verwenden praktisch alle modernen Wahrnehmungssysteme für das automatisierte Fahren tiefe neuronale Netze. Um diese zu trainieren, werden enorme Datenmengen mit passenden Annotationen benötigt. Die Beschaffung der Daten ist relativ unaufwendig, da nur ein mit den richtigen Sensoren ausgestattetes Fahrzeug herumfahren muss. Die Erstellung von Annotationen ist jedoch ein sehr zeitaufwändiger und teurer Prozess. Erschwerend kommt hinzu, dass autonome Fahrzeuge praktisch überall (z.B. Europa und Asien, auf dem Land und in der Stadt) und zu jeder Zeit (z.B. Tag und Nacht, Sommer und Winter, Regen und Nebel) eingesetzt werden müssen. Dies erfordert, dass die Daten eine noch größere Anzahl unterschiedlicher Szenarien und Domänen abdecken.

Es ist nicht praktikabel, Daten für eine solche Vielzahl von Domänen zu sammeln und zu annotieren. Wenn jedoch nur mit Daten aus einer Domäne trainiert wird, führt dies aufgrund von Unterschieden in den Daten zu einer schlechten Leistung in einer anderen Zieldomäne. Für eine sicherheitskritische Anwendung ist dies nicht akzeptabel. Das Gebiet der sogenannten Domänenanpassung führt Methoden ein, die helfen, diese Domänenlücken ohne die Verwendung von Annotationen aus der Zieldomäne zu schließen und somit auf die Entwicklung skalierbarer Wahrnehmungssysteme hinzuarbeiten. Die Mehrzahl der Arbeiten zur Domänenanpassung konzentriert sich auf die zweidimensionale Kamerawahrnehmung. In autonomen Fahrzeugen ist jedoch das dreidimensionale Verständnis der Szene essentiell, wofür heutzutage häufig LiDAR-Sensoren verwendet werden.

Diese Dissertation befasst sich mit der Domänenanpassung für LiDAR-Wahrnehmung unter mehreren Aspekten. Zunächst wird eine Reihe von Techniken vorgestellt, die die Leistung und die Laufzeit von semantischen Segmentierungssystemen verbessern. Die gewonnenen Erkenntnisse werden in das Wahrnehmungsmodell integriert, das in dieser Dissertation verwendet wird, um die Wirksamkeit der vorgeschlagenen Domänenanpassungsansätze zu bewerten. Zweitens werden bestehende Ansätze diskutiert und Forschungslücken durch die Formulierung von offenen Forschungsfragen aufgezeigt. Um einige dieser Fragen zu beantworten, wird in

dieser Dissertation eine neuartige quantitative Metrik vorgestellt. Diese Metrik erlaubt es, den Realismus von LiDAR-Daten abzuschätzen, der für die Leistung eines Wahrnehmungssystems entscheidend ist. So wird die Metrik zur Bewertung der Qualität von LiDAR-Punktwolken verwendet, die zum Zweck des Domänenmappings erzeugt werden, bei dem Daten von einer Domäne in eine anderen übertragen werden. Dies ermöglicht die Wiederverwendung von Annotationen aus einer Quelldomäne in der Zieldomäne. In einem weiteren Feld der Domänenanpassung wird in dieser Dissertation eine neuartige Methode vorgeschlagen, die die Geometrie der Szene nutzt, um domäneninvariante Merkmale zu lernen. Die geometrischen Informationen helfen dabei, die Domänenanpassungsfähigkeiten des Segmentierungsmodells zu verbessern und ohne zusätzlichen Mehraufwand bei der Inferenz die beste Leistung zu erzielen. Schließlich wird eine neuartige Methode zur Erzeugung semantisch sinnvoller Objektformen aus kontinuierlichen Beschreibungen vorgeschlagen, die – mit zusätzlicher Arbeit – zur Erweiterung von Szenen verwendet werden kann, um die Erkennungsfähigkeiten der Modelle zu verbessern. Zusammenfassend stellt diese Dissertation ein umfassendes System für die Domänenanpassung und semantische Segmentierung von LiDAR-Punktwolken im Kontext des autonomen Fahrens vor.

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to Prof. Dr. J. Marius Zöllner for accepting me as a Ph.D. student in his research group at Karlsruhe Institute of Technology. Thank you for guiding me through this work and providing valuable feedback to my research. I am especially grateful for the shift of the weekly doctoral colloquium to online mode, such that I had the chance to participate in every meeting from then on.

Next, I would like to thank Prof. Dr. rer. nat. Markus Enzweiler for providing me with the opportunity to conduct my research in the LiDAR Perception Group at Mercedes-Benz R&D and for acting as second reviewer of this dissertation. Thank you for encouraging me to write my first paper, for providing insightful comments to my work, and for being a great mentor.

My sincere and deepest gratitude goes to Dr. rer. nat. David Peter, my technical adviser at Mercedes-Benz R&D. You were my mentor in many aspects of the Ph.D. studies and beyond. Thank you for being encouraging and inspiring, for having an open ear to my ideas and problems, and your dedication in the days and nights before submission deadlines.

My thanks also go to the remaining colleagues of the LiDAR Perception Group and the Scene Understanding Team at Mercedes-Benz, especially to my office room mates Christoph Rist, Stefan Baur, and David Emmerichs. Thank you for all the valuable discussions and your support. Thank you even more for creating a great atmosphere that made our working hours enjoyable and for becoming friends.

Eventually, I would like to thank everyone else who supported me emotionally, let them be friends or family. My biggest gratitude goes to you Christoph for not just being a colleague that taught me a lot, but also for sharing your love with me. In the office you supported me on a technical level and at home you supported me emotionally. Thank you for always being there. I am very much looking forward to being married to you.

PUBLICATIONS

JOURNALS

L. T. Triess, C. B. Rist, D. Peter, and J. M. Zöllner. A Realism Metric for Generated LiDAR Point Clouds. In *International Journal of Computer Vision (IJCV)*, 2022.

CONFERENCE PROCEEDINGS

L. T. Triess, A. Bühler, D. Peter, F. B. Flohr, J. M. Zöllner. Point Cloud Generation with Continuous Conditioning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022.

L. T. Triess, D. Peter, J. M. Zöllner. Semi-Local Convolutions for LiDAR Scan Processing. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2021.

L. T. Triess, D. Peter, S. A. Baur, and J. M. Zöllner. Quantifying point cloud realism through adversarially learned latent representations. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, 2021.

L. T. Triess, M. Dreissig, C. B. Rist, and J. M. Zöllner. A Survey on Deep Domain Adaptation for LiDAR Perception. In *Proc. IEEE Intelligent Vehicles Symposium (IV) Workshops*, 2021.

L. T. Triess, D. Peter, C. B. Rist, and J. M. Zöllner. Scan-based Semantic Segmentation of LiDAR Point Clouds: An Experimental Study. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2020.

L. T. Triess, D. Peter, C. B. Rist, M. Enzweiler, and J. M. Zöllner. CNN-based synthesis of realistic high-resolution LiDAR data. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2019.

SUPERVISED THESES

B. Johannsen. *Domain Invariant Feature Learning for Cross-Sensor Semantic Segmentation of Lidar Point Clouds by Leveraging the underlying Scene Geometry*. Master thesis, University of Stuttgart, 2022.

A. Bühler. *GAN-based Synthesis of 3D Point Clouds conditioned on Continuous Object Dimensions*. Master thesis, University of Stuttgart, 2021.

PATENTS

L. T. Triess, C. B. Rist. Computer-implementiertes Verfahren zum semantischen Segmentieren und computerimplementiertes Verfahren zum Trainieren eines computer-implementierten Algorithmus zur Bestimmung einer Szenensegmentierung. Filed: 2022/06/28. DE. Patent DE102022002324 (A1), 2022. Patent pending.

L. T. Triess, C. B. Rist. Verfahren zum semantischen Segmentieren von ersten Sensordaten eines ersten Sensortyps. Filed: 2021/05/21. DE. Patent DE102021002684 (A1), 2021. Patent pending.

L. T. Triess, D. Peter. Verfahren zum Transformieren von Sensordaten. Filed: 2021/05/21. DE. Patent DE102021002689 (A1), 2021. Patent pending.

L. T. Triess, D. Peter. Verfahren zur Generierung realistischer Karten von Strahlausfällen in simulierten LiDAR-Daten. Publication Date: 2021/07/08. DE. Patent DE102021002559 (A1), 2021.

L. T. Triess, D. Peter. Verfahren zur automatischen Erkennung und Lokalisierung von Anomalien in mittels eines Lidarsensors erfassten Daten. Publication Date: 2021/04/15. DE. Patent DE102021001043 (A1), 2021.

L. T. Triess, D. Peter. Verfahren zum Trainieren von einem neuronalen Netzwerk einer elektronischen Recheneinrichtung eines Kraftfahrzeugs. Publication Date: 2021/04/15. DE. Patent DE102021000803 (A1), 2021.

L. T. Triess. Verfahren zur Transformation erfasster Sensordaten aus einer ersten Datendomäne in eine zweite Datendomäne. Publication Date: 2020/10/01. DE. Patent DE102020001541 (A1), 2020.

L. T. Triess, D. Peter. Verfahren zur Verarbeitung von Lidarsensordaten. Publication Date: 2020/01/02. DE. Patent DE102019003621 (A1), 2020.

ACRONYMS

2D	**2**-**D**imensional
AcGAN	**A**uxiliary **c**lassifier **G**enerative **A**dversarial **N**etwork
ADAS	**A**dvanced **D**river **A**ssistance **S**ystems
AdvEnt	**Adv**ersarial **Ent**ropy Minimization
BEV	**B**ird's-**E**ye **V**iew
BN	**B**atch **N**ormalization
CcGAN	**C**ontinuous **c**onditional **G**enerative **A**dversarial **N**etwork
CD	*Chamfer's Distance*
CNN	**C**onvolutional **N**eural **N**etwork
COV	**Cov**erage
DA	**D**omain **A**daptation
DNN	**D**eep **N**eural **N**etwork
ECA	**E**uclidean **C**orrelation **A**lignment
EMD	*Earth Mover's Distance*
FCN	**F**ully **C**onvolutional **N**etwork
FPD	*Fréchet Point Cloud Distance*
GAN	**G**enerative **A**dversarial **N**etwork
GCA	**G**eodesic **C**orrelation **A**lignment
GCN	**G**raph **C**onvolution **N**etwork
GPS	**G**lobal **P**ositioning **S**ystem
HDL-64	Velodyne **HDL-64**
ICP	**I**terative **C**losest **P**oint
JSD	*Jensen-Shannon Divergence*
KDE	**K**ernel **D**ensity **E**stimation
KNN	**K**-**N**earest-**N**eighbors
LiDAR	**Li**ght **D**etection **a**nd **R**anging
LReLU	**L**eaky **Re**ctified **L**inear **U**nit
MAE	**M**ean **A**bsolute **E**rror
MinEnt	**Min**imal-**Ent**ropy Correlation Alignment
mIoU	**m**ean **I**ntersection **o**ver **U**nion
MLP	**M**ulti-**L**ayer **P**erceptron
MOS	**M**ean **O**pinion **S**core
MSE	**M**ean **S**quared **E**rror
RADAR	**Ra**dio **D**etection **a**nd **R**anging

ReLU	**Re**ctified **L**inear **U**nit
RGB	**R**ed-**G**reen-**B**lue
SLC	**S**emi **L**ocal **C**onvolution
t-SNE	**t**-Distributed **S**tochastic **N**eighbor **E**mbedding
VAE	**V**ariational **A**uto **E**ncoder
VLP-32	Velodyne **VLP-32**

CONTENTS

1

INTRODUCTION

CONTENTS

Automated driving will change the mobility of the future. This requires a robust, reliable, and scalable environment perception system. Among others, the system needs to operate in multiple domains, e.g. different weather scenarios, countries, or daytime. Therefore, this dissertation investigates domain adaptation methods, involving data generation and evaluation for semantic scene understanding. This chapter motivates the research topic and discusses the goals of this dissertation. It starts with an introduction of automated driving, environment perception, and domain adaptation in section 1.1. Section 1.2 gives an outline of the dissertation goals. Eventually, the structure of the dissertation is presented in section 1.3 with its main contributions in section 1.4.

1.1 MOTIVATION

AUTONOMOUS DRIVING Mobility is a key aspect of life and has taken a big leap forward for humans with the invention of the modern automobile [Benz 1886]. Ever since, the number of vehicles worldwide has increased steadily. The higher dissemination of motor vehicles and the associated increase in driven kilometers comes at the cost of increased road accident fatalities [Statistisches Bundesamt 2020]. As a consequence, passive and active safety systems have been invented, installed, and made mandatory for new vehicles. While passive safety systems, such as airbags or seatbelts, protect the occupants during a crash, active safety systems aim to prevent or mitigate the crash. For these **A**dvanced **D**river **A**ssistance **S**ystems (ADAS) to work, the vehicles are equipped with a variety of sensors, such as camera, microphone, **Ra**dio **D**etection **a**nd **R**anging (RADAR), and only recently also with **Li**ght **D**etection **a**nd **R**anging (LiDAR). These sensors provide a detailed mapping of the surrounding which is then interpreted in efficient scene understanding systems.

Besides safety, comfort nowadays plays a major role in the further development of mobility. Being able to relax or work while

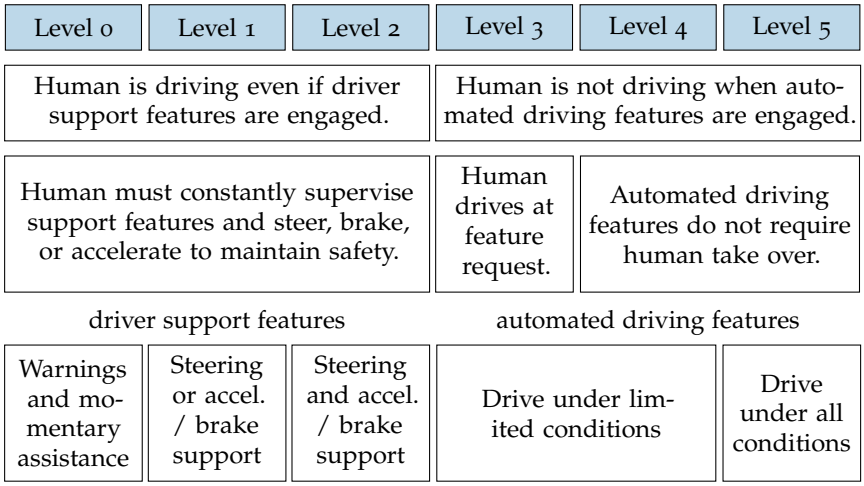


Figure 1.1: **SAE J3016 Levels of Driving Automation**: SAE J3016 defines a taxonomy of six levels of driving automation. They range from Level 0 (no driving automation) to Level 5 (full driving automation) in the context of motor vehicles and their operation on roadways. The graphic is inspired by [SAE International 2021].

stuck in heavy traffic is one of several desires that drive the development of automated driver systems. These systems are divided into six levels of automation. Fig. 1.1 shows the levels of automation defined by SAE J3016 [SAE International 2021]. The lower three levels account for driver support features with the human constantly driving. The upper three levels have automated driving features which require a human only, if a feature requests the human to take over, or not at all. Only recently, the first official level 3 system in the world was approved by the German Federal Motor Transport Authority [Mercedes-Benz Group AG 2021]. The methods developed in this dissertation aim to enable automated driving features from level three upwards.

SCENE UNDERSTANDING The field of scene understanding addresses automatic understanding of sensor data from a semantic and geometric perspective. Assigning meaning to raw sensor data or parts of it is typically done in form of predefined class labels. Fig. 1.2 shows an example for environment perception via a LiDAR sensor. Here, both object bounding boxes with semantic classes and point-wise semantic class labels are visualized. Bounding boxes are most commonly used to classify and localize objects within a scene. They are typically defined as the smallest possible bounding box around an object of interest augmented with a class label and a confidence score. Bounding box object detection only works for countable classes, such as cars and pedestrians, but individual instances of road or terrain are not well defined. To account for these classes, point-wise semantic segmentation is better suited. The task is to

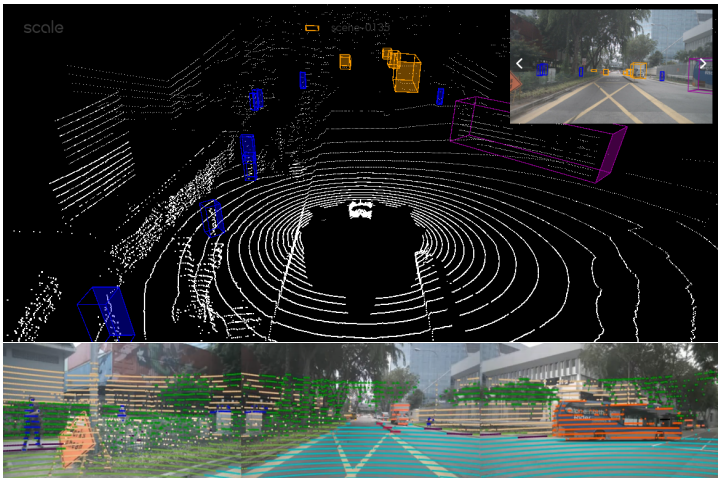


Figure 1.2: **Example for Environment Perception**: The vehicle uses camera and LiDAR to scan the environment. In the top, bounding boxes for 3D object detection are visualized in both the camera image and the LiDAR point cloud. The bottom image shows point-wise semantic segmentation of the point cloud, projected into the camera view. The visualization is of the nuScenes dataset [Caesar et al. 2020].

assign a semantic class label to each point measured by the LiDAR scanner. Objects that do not well fit a bounding box thus can be accurately classified.

DOMAIN ADAPTATION Nowadays, virtually all methods for automotive scene understanding rely on deep learning [Zhou and Tuzel 2018; Lang et al. 2019; Milioto et al. 2019; Sirohi et al. 2022]. This requires huge amounts of data and corresponding annotations. Acquiring these annotations is time consuming, expensive, and error prone. Since autonomous vehicles are exposed to a variety of domains, such as different weather conditions, geographic regions, illumination situations, and seasons, the data needs to cover all these scenarios. Given the complexity of the manual annotation process in combination with an endless variation of domains, it is impossible to use human annotated data and have a scalable system. Furthermore, fast development cycles of the system additionally introduce hardware changes, such as sensor types and vehicle setups, and might require knowledge transfer from simulation. To enable scalable automated driving, it is therefore crucial to address these domain shifts in a robust and efficient manner. These techniques are typically called domain adaptation.

1.2 DISSERTATION GOALS

There is a large interest and progress in robust and reliable 3D environment perception, such as semantic segmentation or object

detection, in the research community. Nowadays, data often is of higher importance than neural network architecture design, as it can have larger impacts on the system behavior. Automated vehicles are exposed to a number of different data domains in terms of weather situations, geography, street signs and rules, sensor types and mounting positions, and even transition from simulation to real-world. Therefore, scalable automation systems that can handle these domain shifts are essential but not yet at the desired level of maturity. An efficient data handling combined with a domain knowledge transfer can help to construct such systems.

To this end, this dissertation first identifies the decisive factors of LiDAR perception, where the data can be identified as one important aspect. Some concepts, such as domain mapping, heavily rely on automatic data generation to bridge domain gaps. Since the generated data influences the final performance of the model, this dissertation investigates the correlation between the realism of the data and the resulting perception performance. Therefore, a suitable realism measure for the LiDAR point clouds is required. Besides data generation, the transfer of knowledge between different data domains, is another essential factor to construct scalable systems. Regarding the aforementioned points, this dissertation identifies and addresses the following research questions:

- What are the decisive factors for successful LiDAR perception? (Chapter 3)
- What are important domain gaps for LiDAR perception and how are they approached in the literature? (Chapter 4)
- Is it possible to estimate the realism of LiDAR point clouds, and if so how? (Chapter 5)
- Does the realism of generated LiDAR data directly correlate with the performance of perception systems? (Chapter 5, Chapter 6)
- Is it possible to generate realistic LiDAR data and use it efficiently for domain mapping applications? (Chapter 6)
- Can transfer of geometric scene knowledge help to learn semantics for another domain? (Chapter 7)

While investigating these open items throughout the dissertation, a common procedure is followed. A thorough literature review discusses existing solutions, then identifies the open issues, and finally addresses these by improving upon existing methods or introducing new concepts. This is done by keeping the particular challenges in an automotive environment, as discussed above, in mind and work towards a pipeline that generates and evaluates LiDAR data that is used to deliver robust and reliable semantic scene information.

1.3 DISSERTATION OUTLINE

Fig. 1.3 shows the structure of this dissertation. Chapter 2 starts with an overview of related work and technical background with respect to the scope of the dissertation. The discussion is conducted from a high-level point of view and only provides a general understanding for the following topics, as more specific related literature is addressed in the individual technical chapters. The final application for all proposed methods in this dissertation is the perception model itself. Therefore, chapter 3 investigates various influences on semantic segmentation performance and proposes a final model that has faster run-time and lower capacity demands but similar performance to a baseline model. The insights of this chapters are used to create the validation module in the following domain adaptation chapters. An extensive overview on deep domain adaptation for LiDAR perception is provided in chapter 4. Besides presenting several domain adaptation mechanisms and tasks, the chapter also introduces relevant open research questions in the field. Among others, the necessity for a reliable metric for generated LiDAR point clouds is identified for the case of domain mapping. This need is addressed in chapter 5 which presents a novel approach to quantify the realism of LiDAR point clouds. The metric is applied in chapter 6, where data generation is used for domain mapping to address the applications of *sensor-to-sensor* and *sim-to-real* adaptation. Additionally, a study on the relationship between data realism and semantic segmentation performance is provided. Chapter 7 tackles the more general application of *dataset-to-dataset* adaptation by using self-supervised geometry information to learn domain-invariant features. Another aim of data generation, besides using it for domain mapping, is to enhance existing training data, for example by augmenting it with synthetically generated objects. Therefore, chapter 8 introduces a **G**enerative **A**dversarial **N**etwork (GAN) method to generate single object point clouds with continuous parameters for specific object features. Eventually, the dissertation is concluded in chapter 9 with a discussion and an outlook for future work.

1.4 CONTRIBUTIONS

The contributions of this dissertation are as follows:

IMPROVED SEMANTIC SCENE SEGMENTATION Semantic segmentation models as proposed in literature are more often optimized for benchmark performance instead of real-world applications. This often leads to very powerful but at the same time slow and hardware demanding systems. Chapter 3 reduces the memory demands of an existing method and proposes a new data representation com-

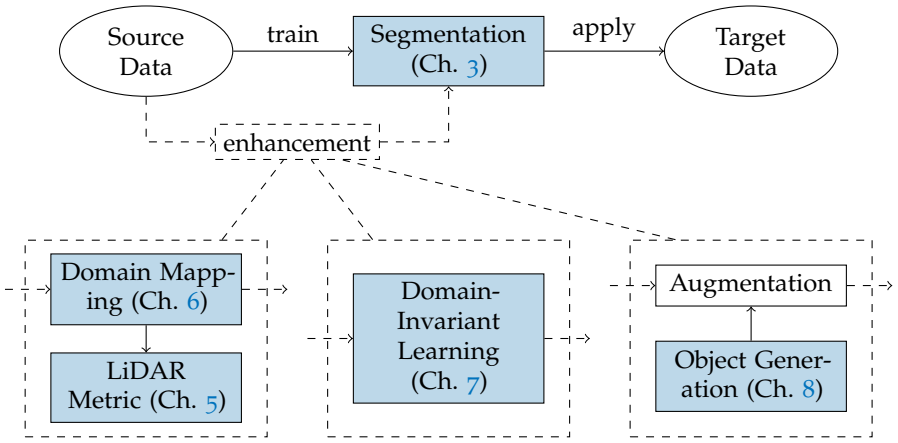


Figure 1.3: **Structure of Dissertation**: Perception models are trained on source data and are then deployed on target data. This dissertation aims to make this process more robust in a scalable system. Chapter 6 uses the method of domain mapping, where data from a source domain is translated to a target domain. Since this requires realistic data generation, chapter 5 proposes a metric to quantify the realism of LiDAR point clouds and includes a study on their effect on perception performance. Chapter 7 proposes a new method that leverages domain-invariant feature learning to account for domain shifts. Data augmentation is another approach to make deployment of perception systems more robust. Chapter 8 proposes a novel method to generate custom-fit object shapes that can be used for data augmentation. Further methods for better segmentation performance are investigated in chapter 3, whereas chapter 4 gives a detailed overview and discussion on domain adaptation for LiDAR applications.

bined with an enhanced processing scheme which leads to a smaller and faster network with no reduction in perception performance. The insights of this chapter are used for the segmentation models throughout this dissertation.

3D DATA GENERATION Training effective perception systems requires large amounts of data with high-quality annotations. A variety of possibilities exist to generate additional training data, such as using simulation frameworks, augmenting existing data, or transferring data from other domains. However, most of these methods are extensively investigated for **2-D**imensional (2D) images, but not for 3D LiDAR data. Therefore, generating realistic LiDAR point clouds is still an open topic. Chapter 6 and chapter 8 propose several 3D data generation methods for different use-cases within autonomous driving using generative models. The models are designed to make use of existing data and work towards improving performance on downstream semantic segmentation tasks. Experiments show that the effectiveness of this approach is highly dependent on the size of the domain gap.

LIDAR EVALUATION MEASURE Data with the right quality and context is crucial to train and improve deep learning based perception systems. However, judging the realism of individual samples can be tedious and time consuming, especially for complex data structures, such as point clouds. Chapter 5 presents a novel approach to quantify the realism of local regions in LiDAR point clouds. The resulting metric can assign a realism score to samples without requiring any task specific annotations and shows reliable interpolation capabilities between data with varying degree of realism. Chapter 6 additionally investigates the connection between the realism of the point clouds and the resulting downstream semantic segmentation performance. This work is first to provide a system that rates the realism of LiDAR data and also compares it with respect to the expected perception performance. The proposed system therefore provides vital insights into the relevance of data realism and generation.

DOMAIN-INVARIANT FEATURE LEARNING VIA GEOMETRIC CUES Using LiDAR sensors with different specifications causes domain shifts, even if the sensors sample the same scene from the same position. Therefore, *sensor-to-sensor* adaptation can be solved by reconstructing the underlying geometry of the scene. However, existing approaches produce additional overhead by explicitly constructing this canonical domain, which leads to a slow runtime at inference which is not real-time applicable. Chapter 7 proposes a novel method that leverages the underlying geometry of the scene implicitly. The geometry of both the source and target domains are represented

in a common compressed feature space, such that learned semantics from the source domain can be applied to the target domain. The resulting model shows that geometric information can help to learn semantic features and can therefore be used to learn domain-invariant features for semantic segmentation tasks.

2

PRELIMINARIES IN 3D POINT CLOUD PROCESSING AND DEEP LEARNING

CONTENTS

This chapter gives a general introduction to the basic building blocks used in this dissertation. The sections contain additional references for more information and the reader is referred to [Goodfellow et al. 2016] for an excellent overview on the field of deep learning. The chapter is structured as follows: Section 2.1 explains the basic operating principles of a LiDAR sensor. Section 2.2 gives a brief overview on common LiDAR perception tasks and data representations. Section 2.3 introduces generative models which are used throughout this work.

2.1 LIDAR SENSORS

2.1.1 *Measurement Principle*

LiDAR sensors provide information about the distance, position, and reflective intensity of an object. A LiDAR consists of one or multiple LASER systems that emit light pulses. Fig. 2.1 shows that the LASER light is emitted by the sensor (red) and is then reflected by a target and returned to a receiver unit (blue). The distance r to the target can then be computed with the time of flight Δt and the speed of light c

$$r = \frac{\Delta t \cdot c}{2} \quad . \tag{2.1}$$

This measurement principle is similar to those of RADAR sensors. Additionally, the sensor provides an estimate of the reflectivity of an object by measuring the amount of the received light (blue part of

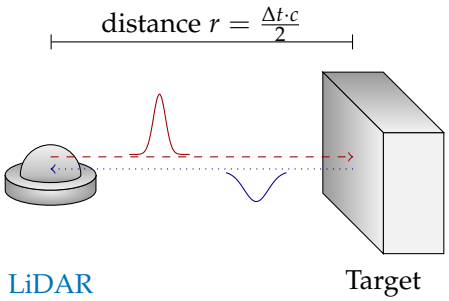


Figure 2.1: **LiDAR Measurement Principle**: The sensor emits a LASER light (red) which is reflected by the target and returned to the receiver. The amount of received light (blue) is measured and used as an estimate of the reflectivity of the object.

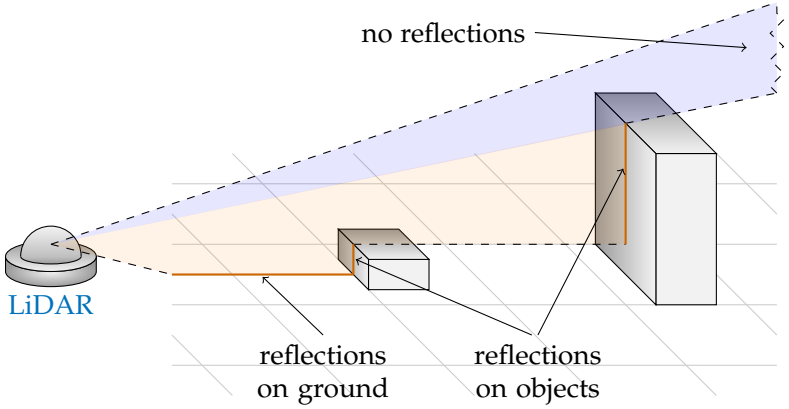


Figure 2.2: **LiDAR Scene Scanning**: The sensor emits an array of light impulses that create the vertical **F**ield **O**f **V**iew. Some of the beams are reflected by the ground or objects (orange), while others are not reflected at all (blue).

fig. 2.1). Retro-reflective surfaces, such as traffic signs or some road markings, result in higher reflective measurements.

There are multiple ways in which a LiDAR sensor can fail to provide a point distance measurement. First, the maximum distance is limited due to beam divergence and atmospheric absorption. Second, outgoing lasers pulses might hit specular reflective surfaces and never return to the sensor. Third, the laser might not be pointed towards an object at all, but towards the sky as visualized in fig. 2.2.

2.1.2 *Rotating LiDAR Systems*

There exists a multitude of LiDAR systems that possess different scan patterns, illumination procedures, pulse techniques, and field of view characteristics. In automotive applications – or more specifically in this work – mainly rotating LiDAR systems are used. They are also used in the most important public datasets, e.g. *KITTI* [Geiger

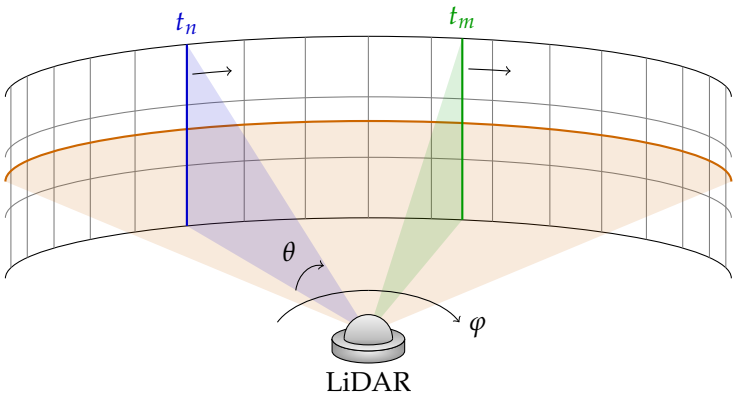


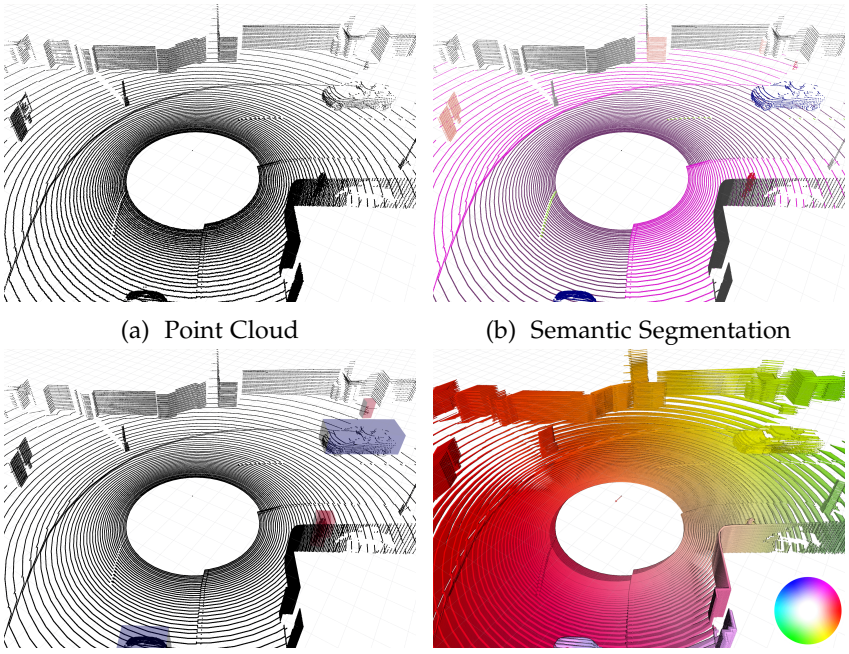
Figure 2.3: **LiDAR Scanning Pattern**: Rotating LiDAR sensors periodically scan the environment at their current orientation defined by an elevation θ_i and an azimuth φ_j angle. The indices i and j represent the possible discrete orientations and construct the sensor image.

et al. 2012], *SemanticKITTI* [Behley et al. 2019], and *nuScenes* [Caesar et al. 2020].

These sensors possess multiple stacked send and receive modules that rotate around a common vertical axis. While rotating, each LASER module periodically measures the distance r_{ij} at its current orientation (see fig. 2.3). This provides a constant stream of measurement points, which are typically cut after each full 360° rotation and are then referred to as frames or scans. The orientation in each scan is described by an elevation angle θ_i and an azimuth angle φ_j (additional offsets to the top and side from the rotation axis are neglected in the following). The indices $i = 1...H$ and $j = 1...W$ represent the possible discrete orientations and construct the sensor image in $\mathbb{R}^{H \times W \times C}$. Here, H corresponds to the number of vertically stacked modules, i.e. layers, of the sensor, while W results from the pulse frequency of the sensor and its revolution speed (usually between 10 Hz and 20 Hz). The number of channels C is determined by the information that is provided by the sensor, e.g. $C = 2$ for distance and reflectivity.

In automotive applications, the sensor is usually not static, but moves with the ego-vehicle. This causes the reference point for each measurement along the azimuth angle to be different. However, a scan representation assumes a common reference point for all data points within the scan. If the ego-motion of the vehicle is known, the data can be transferred to such a common reference point. The ego-motion can be computed from precise position measurements obtained by **G**lobal **P**ositioning **S**ystem (GPS).

The 3D data of a LiDAR scan is also called point cloud. Generic point clouds are usually represented as unordered sets of points in $\mathbb{R}^{N \times (x,y,z)}$ with the number of points N. This is also a common way



(a) Point Cloud (b) Semantic Segmentation

(c) Object Detection (d) Motion Estimation

Figure 2.4: **Example Scene with Task Annotations**: Shown is a recorded scene (a) and the respective annotations for three different perception tasks (b) - (d). Each color in (b) represents one semantic class, such as road (purple) or car (blue). The same colors are used for the object bounding boxes in (c), including only cars (blue) and pedestrians (red). The colors of the arrows in (d) indicate the direction of the motion (see color-wheel in the bottom right corner). The length of the arrows correspond to the respective velocity. The ego-vehicle is placed in the middle of the circle with no measurements.

to store LiDAR point clouds. The Cartesian coordinates x, y, z can easily be computed with spherical-to-Cartesian mapping

$$\begin{pmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{pmatrix} = r_{ij} \cdot \begin{pmatrix} \cos\varphi_j \cdot \sin\theta_i \\ \sin\varphi_j \cdot \sin\theta_i \\ \cos\theta_i \end{pmatrix} . \tag{2.2}$$

Since point sets are the more general representation for this kind of 3D data, this work often refers to the image representation as a cylindrical projection of the point cloud. Note that this projection is a lossless representation of the 3D data, as there are no mutual point occlusions. Since all orientation angles are known, the image can always be transformed back to an approximation of the 3D point cloud with spherical-to-Cartesian mapping.

2.2 DEEP LEARNING FOR LIDAR PERCEPTION

LiDAR sensors are used in autonomous vehicles to obtain precise distance measurements of the 3D surrounding and extract high-level information about the underlying scenery to solve a number of different tasks. Fig. 2.4 shows three of the most common LiDAR perception tasks. These include point-wise semantic segmentation (fig. 2.4b), bounding-box object detection (fig. 2.4c), and point-wise motion estimation (fig. 2.4d). Segmentation tasks may also include instance segmentation, panoptic segmentation, and part segmentation [Milioto et al. 2019; Xu et al. 2020; Sirohi et al. 2022], while object detection and scene flow estimation can be combined to solve tracking tasks [Zhou and Tuzel 2018; Lang et al. 2019; Liu et al. 2019]. Another field involves estimating unseen parts of the scenery or completing sparse data, which is usually referred to as scene completion [Rist et al. 2020; Agia et al. 2020]. This dissertation mainly uses semantic segmentation to evaluate the proposed approaches.

With the availability of large-scale datasets and sufficient computational resources, **D**eep **N**eural **N**etworks (DNNs) are nowadays at the core of almost all state-of-the-art scene understanding methods. For an excellent overview on the field of deep learning, the reader is referred to [Goodfellow et al. 2016].

2.2.1 *Data Representation*

As of today, no single method to represent 3D point clouds prevailed. A detailed overview on point cloud representation and related architectures for 3D data is given in a survey by Guo et al. [2021]. The networks used for point-wise semantic segmentation can be divided into two categories:

Projection-based networks: multi-view [Lawin et al. 2017; Boulch et al. 2017], spherical [Wu et al. 2018, 2019; Milioto et al. 2019], and volumetric [Meng et al. 2019; Rethage et al. 2018; Graham et al. 2018] representations.

Point-based networks: point-wise **M**ulti-**L**ayer **P**erceptrons (MLPs) [Qi et al. 2017a,b; Zhao et al. 2019b], convolution-based [Hua et al. 2018; Thomas et al. 2019; Wang et al. 2018], and graph-based [Landrieu and Simonovsky 2018; Wang et al. 2019a] networks.

2.2.2 *Semantic Segmentation*

Semantic segmentation is a crucial part of detailed scene understanding. **F**ully **C**onvolutional **N**etworks (FCNs) marked the breakthrough for RGB image segmentation in deep learning research [Shelhamer et al. 2017]. They use memory efficient filter kernels by convolving

over the dense 2D regular grid of the images. Introduction of dilated convolutions combined with conditional random fields improved the prediction accuracy [Chen et al. 2017a; Yu and Koltun 2016; Krähenbühl and Koltun 2011]. Gains on speed were mainly achieved with encoder-decoder architectures that fuse feature maps of higher layers with spatial information from lower layers or approaches that combine image features from multiple refined paths [Badrinarayanan et al. 2017; Lin et al. 2020].

For point-wise segmentation of 3D data, many approaches evolved from their 2D ancestors by using projection-based intermediate representations of the data. However, crucial modifications to the respective network architectures had to be introduced to fit the needs of projected data [Wu et al. 2018; Piewak et al. 2018]. Only since the release of *SemanticKITTI* [Behley et al. 2019], a large scale dataset of real-world driving scenarios with point-wise semantic annotations of LiDAR scans is publicly available to facilitate the development of point-wise semantic segmentation algorithms.

2.3 DEEP GENERATIVE MODELS

Deep learning models can be divided into two categories, discriminative models and generative models [Ng and Jordan 2001]. A discriminative model is a model of the conditional probability $P(Y \mid X = x)$ of the target Y, given an observation x. Examples are segmentation models as discussed in section 2.2.2. A generative model, on the other hand, is a model of the conditional probability $P(X \mid Y = y)$ of the observable X, given a target y. Popular deep generative models include **V**ariational **A**uto **E**ncoders (VAEs) [Kingma and Welling 2014], GANs [Goodfellow et al. 2014], auto-regressive models, and diffusion models [Dhariwal and Nichol 2021]. This dissertation mainly focuses on GANs.

2.3.1 *Generative Adversarial Networks*

Fig. 2.5 shows the basic concept of a vanilla GAN architecture, as introduced by Goodfellow et al. [2014]. The network consists of two models, the generator G and the discriminator D. The generator $G(z;\theta_G)$ is a differentiable function with parameters θ_G that maps input noise $p_z(z)$ to data space. The differentiable function of the discriminator $D(x;\theta_D)$ outputs a scalar that represents the probability that x came from the data distribution p_{data} rather than the generator's distribution p_G. The networks are trained, such that

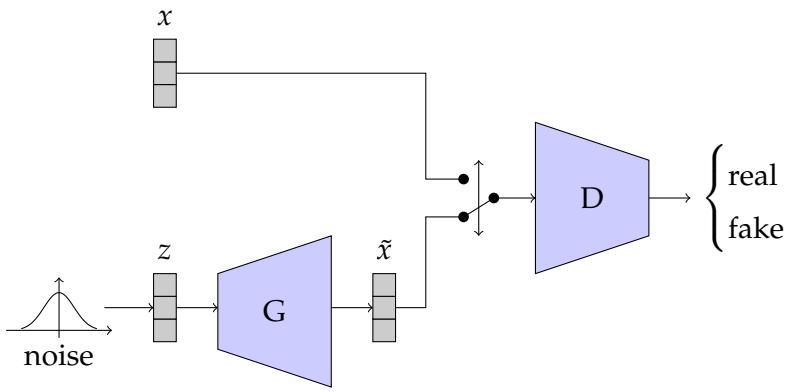


Figure 2.5: **Generative Adversarial Network (GAN)**: Basic structure of a GAN consisting of two adversarial subnetworks, the generator G and the discriminator D [Goodfellow et al. 2014]. The switch symbol indicates the alternating training process, where either a real sample x or a generated sample $\tilde{x}$ is fed to the discriminator.

D and G follow a two-player minimax game with value function $V(G, D)$:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z)))] \quad . \tag{2.3}$$

In other words, the aim of the discriminator is to determine whether the sample x stems from the generator distribution p_G or the data distribution p_{data}. At the same time, the generator tries to trick the discriminator into believing that its generated sample $G(z)$ stems from the data distribution p_{data}. With the GAN objective $V(G, D)$ and the alternating updates of D and G, both networks lead to the improvement of the opponent's performance.

2.3.2 *Conditional GANs*

Often it is required to not simply imitate a specific data distribution, but also to condition the generated samples on certain classes or styles that occur within the distribution. The vanilla GAN architecture can easily be extended to a conditional model where both the generator and the discriminator are conditioned on the label y [Mirza and Osindero 2014]. Fig. 2.6 shows that the discriminator now receives x or $\tilde{x}$ with y as inputs, while the generator combines the prior input noise $p_z(z)$ and y in a joint hidden representation. The objective function $V(G, D)$ from eq. (2.3) is now extended to

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x \mid y)] + \mathbb{E}_{z \sim p_z(z)} [\log (1 - D(G(z \mid y)))] \quad . \tag{2.4}$$

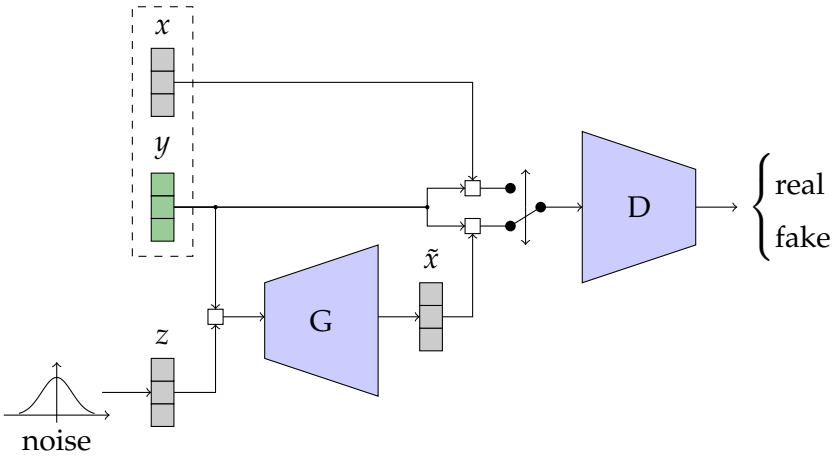


Figure 2.6: **Conditional Generative Adversarial Network (cGAN)**: Structure of the basic version of a cGAN [Mirza and Osindero 2014]. The real data comes with matching labels y (e.g. classes or styles) for each sample x. The generator G now creates fake samples $\tilde{x}$ conditioned on the label y. The discriminator D always receives a sample and its corresponding label. The combination of sample and label has many possibilities, a simple version is to simply concatenate the features (here abstracted with the white squares).

Conditional GANs soon reached popularity in image-to-image translation [Isola et al. 2017] and accelerated research for many mapping scenarios for camera images. CycleGAN [Zhu et al. 2017] additionally introduces inverse mappings that are enforced by cycle consistency losses. This solves the issue of solving an underconstrained mapping problem and the model does not require any paired samples for training. The introduction of the cycle consistency constrain opened the doors to a variety of domain mapping use-cases. CyCADA [Hoffman et al. 2018] is a cycle-consistent adversarial domain adaptation model that adapt representations at both pixel-level and feature-level.

3 SCAN-BASED SEMANTIC SEGMENTATION

CONTENTS

Autonomous vehicles need to have a semantic understanding of the three-dimensional world around them in order to reason about their environment. State of the art methods use deep neural networks to predict semantic classes for each point in a LiDAR scan. A powerful and efficient way to process LiDAR measurements is to use two-dimensional, image-like projections (section 2.2.1). This chapter contains a comprehensive experimental study of image-based semantic segmentation architectures for LiDAR point clouds. Various techniques are introduced that boost the performance and improve runtime as well as memory constraints.

This chapter is adapted from [Triess et al. 2020] and [Triess et al. 2021c] and contains verbatim quotes of these works.

3.1 OVERVIEW

Many state-of-the-art semantic segmentation approaches make use of traditional two-dimensional **C**onvolutional **N**eural **N**etworks (CNNs) by using the cylindrical image representations of the point clouds [Behley et al. 2019; Wu et al. 2018; Piewak et al. 2018]. However, most of these works use ego-motion corrected point clouds that lead to systematic point occlusions in the image projection which impacts the semantic segmentation performance in those regions (cf. fig. 3.1a). Therefore, this chapter proposes a scan unfolding method for *KITTI* [Geiger et al. 2012] that features less projection artifacts than those currently used in literature. Further, the scan unfolding allows for the application of a periodic padding scheme

that provides context at the horizontal field-of-view boundaries and can be propagated through the entire network.

Additionally, this chapter shows that the spatial stationary assumption of convolutions is still applicable to inputs with varying statistical properties over parts of the data, such as projected LiDAR scans. These data structures exhibit similar features as aligned images for which locally connected layers have been introduced [Taigman et al. 2014]. At the time of writing, the introduction of **S**emi **L**ocal **C**onvolutions (SLCs) showed that weight sharing convolutions stay the most powerful tool for semantic segmentation. Nowadays transformers [Vaswani et al. 2017; Dosovitskiy et al. 2021] perform en-par or better than many of the existing convolution based networks.

In order to surpass the current baseline of a specific metric, the networks tend to become bigger in terms of more free parameters. This can result in a declined generalization capacity, since the network rather "remembers" than "learns". Further, the architectures require more resources in terms of memory and runtime in both training and inference. Especially for autonomous vehicles it is vital that the components match specific resource constrains and are operable in real-time. The experiments show that at the expanse of very little accuracy, the resource requirements of the models can be heavily decreased.

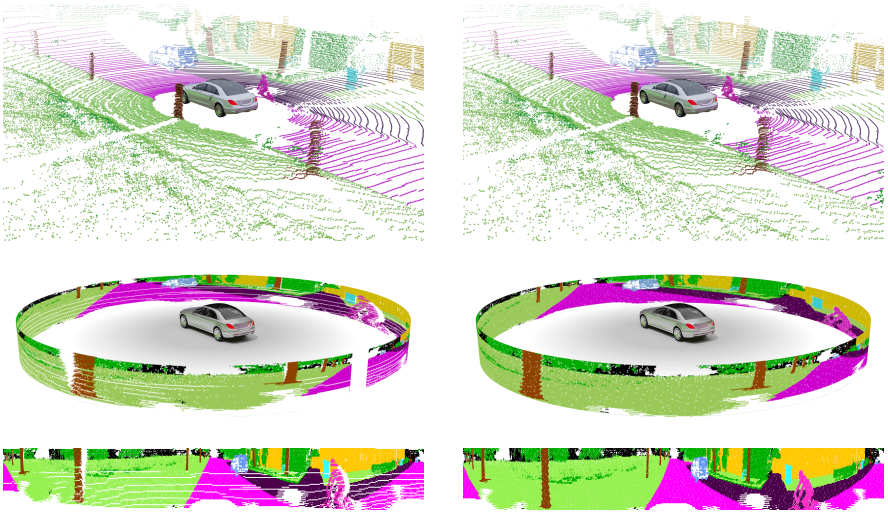
The main contributions of this chapter are:

- a comprehensive study on training techniques for real-world image-based semantic segmentation architectures
- a proposal for dense scan unfolding on *KITTI* and a cyclic padding mechanism for horizontal field-of-view context
- introduction of **S**emi **L**ocal **C**onvolutions, a layer with weight-sharing along only one of the two spatial dimensions

3.2 EFFICIENT NETWORK CONFIGURATION

Semantic segmentation is an important component of 3D scene understanding, where the goal is to augment precise geometric information from the LiDAR sensor with a detailed semantic interpretation. Behley et al. [Behley et al. 2019] showed that projection-based networks outperform state-of-the-art point-based networks for point-wise semantic segmentation on LiDAR point clouds. Therefore, this dissertation uses the spherical projection as the representation of the LiDAR data for the segmentation task.

The following introduces three methods (section 3.2.1) and presents their results in section 3.2.2. Finally, the insights are combined to produce a final model with improved performance and runtime (section 3.2.3).



(a) Ego-motion corrected projection method [Milioto et al. 2019]

(b) Scan unfolding method [Triess et al. 2020]

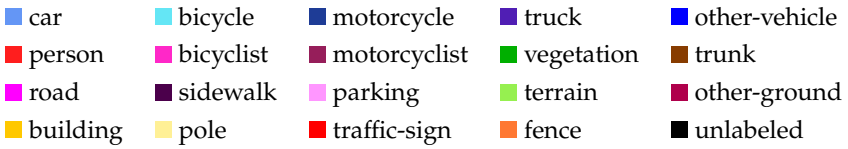


Figure 3.1: **Cylindrical Point Cloud Projection**: (a) Correcting for ego-motion leads to a projection that suffers from systematic point occlusions as some 3D points are projected into occupied pixels. Hidden points can not provide any information to the network and may not be accurately classified. (b) The proposed scan unfolding method provides a dense projection without systematic discretization artifacts. The proposed cyclic padding mechanism provides context at the horizontal field-of-view boundaries by closing the gap in the cylindrical projection (top right). The figure is adapted from [Triess et al. 2020].

3.2.1 *Method*

The method of this section is composed of three parts. Section 3.2.1.1 proposes an efficient scan-unfolding that enables using a more efficient padding scheme within the network, called cycle padding, as proposed in section 3.2.1.2. Section 3.2.1.3 introduces two loss functions for the task of semantic segmentation, which are then compared in the experiment section.

3.2.1.1 *Scan Unfolding*

Section 2.1.2 explained the working mechanism of LiDAR sensors and how to transform between the 3D point clouds and their corresponding 2D cylindrical projections. Openly available datasets usually provide the data as lists of Cartesian coordinates which requires a back-projection into the image-like structure for projection-based

networks [Geiger et al. 2013; Caesar et al. 2020]. However, this is sometimes not trivial, as ego-motion correction is applied to the data or missing measurements are not represented in the list-like structure. Fig. 3.1 shows two different projection schemes: ego-corrected projection and the proposed scan unfolding.

EGO-MOTION CORRECTED PROJECTION The projection shown in fig. 3.1a is a proxy representation by Milioto et al. [2019]. It suffers from mutual point occlusions due to the ego-motion correction of the data and leaves large areas without data (white pixels).

SCAN UNFOLDING Fig. 3.1b depicts a projection with reduced mutual point occlusions, thus minimizing the loss of information. The scan unfolding method is designed to be a proxy representation of the original raw sensor data. The algorithms leverages information about this particular sensor layout. For details on the proposed projection algorithm, see appendix A.1.

3.2.1.2 *Cyclic Padding*

When convolving over data in a CNN, the input is usually padded in order to match the desired output shape. Since LiDAR measurements represent a constant stream of data along the horizontal axis of the projections, the precise padding would take snippets from the previous and subsequent 360° scan in time. This is not practical when training the network and not applicable at inference time. However, using the scan-based projection it becomes possible to implement a cyclic padding strategy by basically taking the values from the opposite side of the range image. Due to the cylindrical projection of the scan, a closed 360° view is formed (see fig. 3.1b). This can be propagated through the entire network.

3.2.1.3 *Loss Functions*

For semantic segmentation tasks, the multi class cross-entropy

$$CE(\hat{y}, y) = -\sum_{i,c} \hat{y}_c^i \log y_c^i \tag{3.1}$$

is the most-often used loss function [Good 1956]. Here, $\hat{y}_c^i$ is the one-hot encoded ground truth distribution for class c at pixel position i, while y_c^i is the corresponding softmax prediction.

The performance of such systems is usually evaluated with the Jaccard Index over all classes [Jaccard 1901], which is often referred to as **m**ean **I**ntersection **o**ver **U**nion (mIoU). In order to reach high mIoU values, the cross-entropy is minimized over training. However, the loss does not directly reflect the inverse of the metric.

In order to directly maximize the mIoU, it is possible to use the Dice coefficient which measures the similarity between two samples [Sorensen 1948; Dice 1945]. The soft Dice loss can be written as

$$DL(\hat{y}, y) = 1 - \frac{1}{C} \sum_c \frac{2 \sum_i \hat{y}_c^i y_c^i}{\sum_i (\hat{y}_c^i)^2 + \sum_i (y_c^i)^2} \tag{3.2}$$

where C is the total number of classes.

3.2.2 *Evaluation*

The basis of the experiments in this section is the RangeNet implementation of Milioto et al. [2019]. Note that the proposed models are compared against a slightly modified version of RangeNet, referred to as RangeNet* (R*), which omits x, y, and z as input channels. Both version are benchmarked against each other and show no significant difference in the resulting metric results. The second column of table 3.1 shows the baseline results of RangeNet*.

The following shows experiments that vary the network parameters (section 3.2.2.1), the loss function (section 3.2.2.2), and the scan construction (section 3.2.2.3).

3.2.2.1 *Network Parameters*

Larger networks tend to be more prone to overfitting. RangeNet with its 50.4 million trainable parameters is also affected by this. Fig. 3.2 and table 3.2 show the performance of the network for a decreasing number of trainable parameters by adapting the filter sizes within the convolutions. A large reduction of parameters, causes the performance to decrease only slightly. Further, it can be observed that the smaller networks generalize better due to decreased overfitting. With a reduction to only 10% of the original number of parameters, the model (in the center) still reaches 96% of the performance while, at the same time, decreasing the inference time of the network to one third.

3.2.2.2 *Loss Functions*

The third column of table 3.1 shows that replacing cross-entropy loss with Dice loss increases the mIoU by 3.2%. Class-wise the two losses show distinguished quality. Dice loss reaches better performances on classes bicycle, bicyclist, pole, traffic-sign, and trunk. Cross-entropy, on the other hand, performs better on motorcycle, parking, and person. If IoU is the metric to reflect the desired quality in a network performance, it is advisable to use Dice loss instead of cross-entropy. It has the advantage of directly maximizing the metric as opposed to cross-entropy.

Table 3.1: **Semantic segmentation performance**: This table shows experimental results for a subset of the proposed techniques and compares them with RangeNet* (R*). Note that the numbers deviate from the ones published in [Milioto et al. 2019], as numbers are reported on the validation dataset instead of the test dataset. The table is based on [Triess et al. 2020].

	Baseline	Ablations				Combined
Base Network	R*	R*	R*	R*	R*	D
Dice Loss		✓			✓	✓
Scan Unfolding			✓	✓	✓	✓
Cyclic Padding				✓	✓	✓
Inference $\left[\frac{\text{ms}}{\text{frame}}\right]$	74.3	74.3	74.3	74.3	74.3	**30.9**
Mean IoU [%]	46.7	48.2	47.5	47.9	**48.5**	48.2
Bicycle	23.0	24.3	23.9	23.1	22.1	25.5
Car	91.0	92.0	90.7	92.1	93.3	91.1
Motorcycle	31.8	28.1	37.6	32.3	26.0	25.6
Truck	29.5	39.5	31.3	35.5	29.3	38.8
Other-Vehicle	29.6	25.6	24.9	22.8	21.9	21.7
Person	26.2	17.5	22.9	24.9	15.3	23.0
Bicyclist	48.4	55.6	53.0	51.5	41.8	48.6
Motorcyclist	0.0	0.0	0.0	0.0	0.2	0.0
Road	92.9	92.4	93.2	94.8	93.1	93.1
Sidewalk	78.9	78.5	79.2	79.9	77.7	77.9
Parking	41.5	36.9	43.2	43.0	38.1	43.3
Other-Ground	0.4	0.0	0.3	0.3	0.7	0.5
Building	82.1	81.9	83.5	84.2	82.1	82.9
Fence	49.7	48.3	51.2	49.4	50.1	55.9
Traffic-Sign	25.7	34.6	25.8	25.4	38.2	37.3
Pole	36.1	47.2	36.2	36.3	45.8	48.6
Trunk	42.9	53.5	45.9	47.9	49.9	48.3
Vegetation	82.7	84.0	84.0	84.1	84.2	83.3
Terrain	75.5	75.0	75.4	77.2	74.3	70.8

* drop x, y, and z channels from the input as experiments show that these features do not influence the performance in a significant way

Table 3.2: **Performance for different network sizes**: We report the mean value of training and validation mIoU as well as the respective standard deviation ($\pm x$). The table is based on [Triess et al. 2020].

Network	A	B	C	D	RangeNet*
Number of parameters	0.4M	1.3M	4.2M	12.7M	50.4M
Filter size config.	32, 32, 32, 32, 32, 32	32, 48, 64, 64, 64, 64	32, 48, 64, 96, 128, 256	32, 48, 64, 128, 256, 512	32, 64, 128, 256, 512, 1024
Train mIoU [%]	39.2 ± 0.5	45.6 ± 1.0	52.0 ± 1.3	54.1 ± 3.2	59.7 ± 4.1
Val mIoU [%]	38.7 ± 0.6	41.7 ± 5.1	43.5 ± 2.5	44.7 ± 1.2	46.4 ± 0.7
Inference time [ms]	20.5	22.1	23.9	30.9	74.3

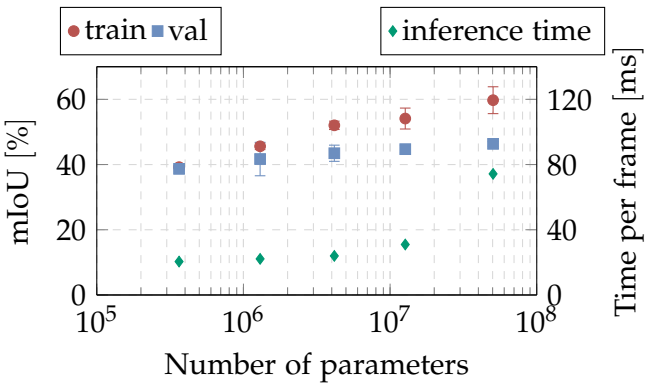


Figure 3.2: **Overfitting**: A significant overfitting gap is present for networks at RangeNet size. The effect only vanishes when reducing the number of parameters by two orders of magnitude. The figure is based on [Triess et al. 2020].

3.2.2.3 *Scan construction*

This section compares the ego-motion corrected projection with the proposed scan unfolding method in two otherwise identical settings. The former uses the ego-motion corrected data from *SemanticKITTI*, while the latter uses the raw data obtained from *KITTI*. The point-wise annotations are identical for both. However, note the target segmentation might differ depending on the occlusions that arise from the projection. Table 3.1 shows the validation results for our scan unfolding method (second column in the ablation section) in comparison to RangeNet* using the ego-motion corrected data. The scan unfolding achieves a gain of 1.7% in mIoU. Classes with small or thin objects, such as bicyclist or trunk, benefit especially. This can be attributed to the differences in projection for foreground objects, as highlighted in fig. 3.1.

In addition, zero padding is replaced with the proposed cyclic padding strategy in all convolution layers. The results are listed in the fifth column of table 3.1. Exploiting the cycle consistency of the scan renders beneficial for the performance but does not generate a substantial boost. It can be concluded that this is a more accurate padding scheme than the default zero-padding for 360°scans.

3.2.3 *Summary*

In a final experiment, the above insights are used to combine the components that generated a positive effect on the segmentation accuracy. Table 3.1 shows that combining Dice loss and the scan unfolding method with cyclic padding reaches the best performance. These settings are also tested on a smaller network D (see table 3.2) which achieves a higher segmentation score than the plain version of the much larger RangeNet*. The inference time of this model is less than half of the time of the bigger model.

3.3 SEMI LOCAL CONVOLUTION

This section introduces a new base layer for neural networks, specifically designed to process LiDAR depth projections. First, section 3.3.1 gives a brief overview on weight sharing in convolution layers, before introducing the new layer type in section 3.3.2. Experiments and discussions on the results are presented in section 3.3.3 and section 3.3.4, respectively.

3.3.1 *Weight Sharing in Convolution Layers*

Convolution layers apply a filter bank on their input. The filter weights are shared over all spatial dimensions, meaning that for ev-

ery location in the feature map the same set of filters are learned. The re-usability of weights causes a significant reduction in the number of parameters compared to fully connected layers. This allows deep convolutional neural networks to be trained successfully, in turn leading to a substantial performance boost in many computer vision applications. The underlying premise of convolutional methods is that of translational equivariance, i.e. that features that have been learned in one region of the image are useful in other regions as well.

For applications such as face recognition which deal with aligned data, locally connected layers have proved to be advantageous [Gregor and LeCun 2010; Huang et al. 2012; Taigman et al. 2014]. These layers also apply a filter bank. Contrary to convolutional layers, weights are not shared among the different locations in the feature map, allowing different sets of filters to be learned for every location in the input.

The spatial stationary assumption of convolutions does not hold for aligned images due to different local statistics in distant regions of the image. In a projected LiDAR scan, the argument holds true for sensors that are mounted horizontally. Each horizontal layer is fixed at a certain vertical angle. As the environment of the sensor is not invariant against rotations around this axis, this leads to different distance statistics in each vertical layer. Prior to [Triess et al. 2020], applying locally connected filters on point cloud projections has not been investigated.

3.3.2 *Method*

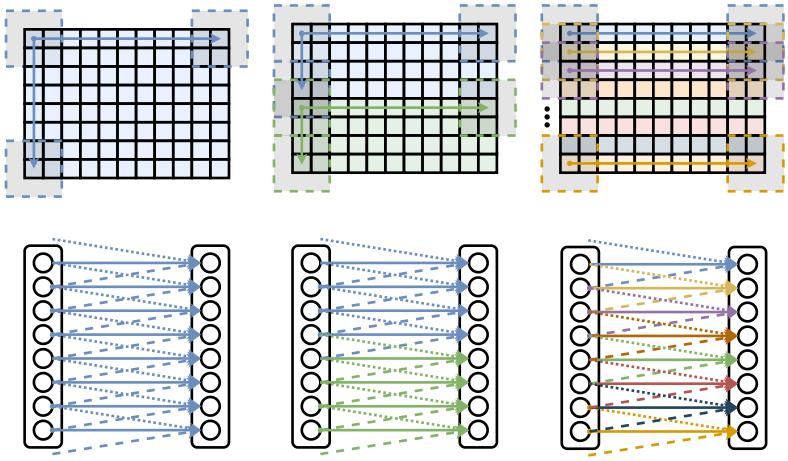
In order to introduce **S**emi **L**ocal **C**onvolutions (SLCs), consider an input feature map x with shape $[H_x, W_x, C_x]$, representing a cylindrical projection with height H_x, width W_x, and C_x channels. The output of the layer is another feature map y with shape $\left[H_y, W_y, C_y\right]$. In the following, without loss of generality, x is considered to be padded such that $H_y = H_x$ and $W_y = W_x$.

A normal convolution layer has a kernel k of shape $\left[I, J, C_x, C_y\right]$, with the spatial sizes I, J and the filter sizes C_x, C_y. The output of such a layer is

$$y_{h,w,c_y} = \sum_{c_x} \sum_{i} \sum_{j} k_{i,j,c_x,c_y} \cdot x_{h-i,w-j,c_x} \tag{3.3}$$

where the sum over i (and similarly for j) is appropriately restricted to the range $-\lfloor I/2 \rfloor \dots \lfloor I/2 \rfloor$.

In a SLC layer, the kernel has multiple components for different parts along the vertical axis of the input as illustrated in fig. 3.3 (note that the concept can also be applied to the horizontal direction).



(a) $\alpha = 1$ (b) $\alpha = 2$ (c) $\alpha = H_x = 8$

Figure 3.3: **Semi Local Convolution (SLC)**: (a) Illustration of a normal convolution for an input tensor of shape $[H_x, W_x] = [8, 11]$. A 3×3 sliding kernel is represented by the gray rectangle. Weight sharing is active across the full image. This is a special case of SLC with $\alpha = 1$. (b) SLC with $\alpha = 2$. Weights are shared in the upper and lower half of the input, respectively. This allows the network to learn different kernels depending on the horizontal position in the input image. (c) For $\alpha = H_x$, weight sharing along the vertical dimension is completely turned off, weights are only shared horizontally. Different filters can be learned for each individual vertical position. The figure is based on [Triess et al. 2020].

With the number of components $\alpha \in \mathbb{N}$ with $1 \leq \alpha \leq H_x$, the kernel has a shape of $\left[I, J, C_x, C_y, \alpha\right]$. The output of the SLC is then given by

$$y_{h,w,c_y} = \sum_{c_x} \sum_{i} \sum_{j} k_{i,j,c_x,c_y,\alpha_h} \cdot x_{h-i,w-j,c_x} \tag{3.4}$$

where $\alpha_h = \lfloor h/H \cdot \alpha \rfloor$ selects the respective filter-component depending on the vertical position h.

For $\alpha = H$, there is no weight sharing along the vertical axis, a new filter is used for every single data row. A regular convolution, as defined in eq. (3.3), is obtained with $\alpha = 1$. For values in between, the degree of weight sharing can be adapted to the desired application.

3.3.3 *Evaluation*

This section investigates the introduction of SLC layers in various experiments. The implementation of Milioto et al. [2019] is used for the experiments. All results are reported on the validation split of the SemanticKITTI dataset [Behley et al. 2019]. The input to the network is a two-channel image with the projected depth measurements and the respective reflecivity values.

Table 3.3: **Overall Results**: Shown is the semantic segmentation performance over 19 object classes for two different base networks on the validation split of the SemanticKITTI dataset. Each base network is augmented with the proposed SLC layer and tested for different values of α, i.e. the number of vertical filters within the convolution. All convolution layers, except input and output layer, are replaced within the network. The table is based on [Triess et al. 2021c].

Network	Metric	# vertical filters α						
		1	2	4	8	16	32	64
DarkNet21 [Milioto et al. 2019]	Accuracy	**0.83**	0.82	0.81	0.79	0.75	0.73	-
	mIoU	**0.36**	0.34	0.32	0.30	0.27	0.26	-
SqueezeSegV2 [Wu et al. 2019]	Accuracy	**0.84**	0.82	0.81	0.80	0.76	0.75	0.71
	mIoU	**0.36**	0.35	0.31	0.30	0.27	0.26	0.25

Table 3.3 shows the overall results for two different base networks. First, it shows that SLCs do not outperform normal convolutions. Second, the performance decreases with increasing α. Third, the above two points apply to both base networks, even though DarkNet21 has approximately 24.7M trainable parameters, whereas SqueezeSegV2 only has approximately 928.5k parameters.

When α increases with a factor of 2, then also the number of trainable parameters increases by a factor of 2 (approximately)[1]. Therefore, a second row of experiments is conducted, where the number of trainable parameters in the base network are decreased by decreasing the number of filters in each layer. Naturally, we expect a network with lower capacity to perform worse. Table 3.4 shows the semantic segmentation accuracy and mIoU performance for the DarkNet21 model. The entries with the gray background mark those networks that have approximately the same number of trainable variables as the base network, since the modification in α and output filter size cancel each other out. Here again, an increase in α leads to decreased performance. One has to note, that even if the number of parameters is the same for the gray cells, the increase in α only leads to more capacity over the spatial dimension of the feature maps, whereas larger output filters in general lead to more capacity over the depth of the network for the entire spatial extent.

3.3.4 *Discussion*

The experiments show that SLCs are not able to outperform normal convolutions and performance usually decreases with increasing α. This effect is stronger for networks with a large number of parame-

1 DarkNet21 with $\alpha = 64$ is too large for a single GPU, therefore no results are reported

Table 3.4: **Scaled Network Performance**: This table shows the semantic segmentation performance of the DarkNet21 [Milioto et al. 2019] model (Accuracy / mIoU). Over the columns of the table, the number of vertical filters within each SLC layer are increased. Over the rows of the table, the number of overall output filters for each layer are decreased, i.e. 2 means multiplying the number of filters by a factor of $\frac{1}{2}$ which results in $\frac{1}{4}$ of the original trainable variables. The gray cells mark those that contain configurations where the increase in α is neutralized with the decrease of filter sizes and thus results in approximately the same number of trainable parameters. The table is based on [Triess et al. 2021c].

	1	2	4	8	16	32	64
1	0.83 / 0.36	0.82 / 0.34	0.81 / 0.32	0.79 / 0.30	0.75 / 0.27	0.73 / 0.26	-
2	0.83 / 0.35	0.81 / 0.32	0.83 / 0.33	- / -	- / -	- / -	0.70 / 0.24
4	0.82 / 0.34	- / -	0.77 / 0.30	0.76 / 0.28	0.74 / 0.25	- / -	0.72 / 0.24
8	0.77 / 0.31	- / -	- / -	0.74 / 0.27	0.72 / 0.25	0.71 / 0.24	0.71 / 0.23

ters. Therefore, it can be assumed that normal convolution layers of adequate capacity can already handle the different statistical properties across the vertical spatial dimension. This claim is supported by the findings of Kayhan and van Gemert [2020] who show that convolutional layers exploit absolute spatial location. Therefore, CNNs are in fact not translation invariant which means a network with sufficient capacity is able to learn even differing statistics over the vertical dimension of such a LiDAR scan.

3.4 CONCLUSION

This chapter presented an experimental study on projection-based semantic segmentation of LiDAR point clouds. The experiments show that carefully chosen loss functions and input data representations can lead to a boost in semantic segmentation performance. The proposed scan unfolding method is preferred over the cylindrical projection of ego-motion corrected data. In the case of single-frame processing, it can be combined with a cyclic padding mechanism which leads to another small improvement. The scan unfolding and cyclic padding are applied to the LiDAR projections for all further experiments in this dissertation.

However, using the proposed SLCs instead of normal convolutions decreases segmentation performance, especially when decreasing the amount of weight sharing. Since normal convolution layers can already exploit spatial location information within the network, it is not necessary to explicitly address the large difference in appearance along the vertical axis of a LiDAR scan in a special layer.

The chapter also demonstrated that the network size, in terms of parameters, can be drastically reduced at very little cost to accuracy, allowing for applications on hardware with limited resources or

hard real-time constraints. By combining Dice loss and the proposed scan unfolding method with cyclic padding, a fast network architecture is constructed that outperforms much slower state-of-the-art networks without these modifications. Such a combination of high performance and low run-time is required to enable autonomous driving.

4

DEEP DOMAIN ADAPTATION FOR LIDAR PERCEPTION

CONTENTS

Deep learning techniques for perception applications typically require a huge amount of annotated data matching the considered scenario to obtain reliable performances. A major assumption in these algorithms is that the training and application data share the same feature space and distribution. However, in many real-world applications, such as in the field of automated driving, this assumption does not hold, since the agents operate in an open-world setting. Furthermore, collection and annotation of large datasets for every new task and domain is extremely expensive, time-consuming, and not practical for scalable systems. For clarification, a domain is defined as the scope of application for the algorithm.

A common scenario includes solving a detection task in one domain with training data stemming from another domain. In this case, the data may differ in their feature space or follow a different data distribution. Examples for these divergent domains can be different geographical regions, weather scenarios, seasons, other

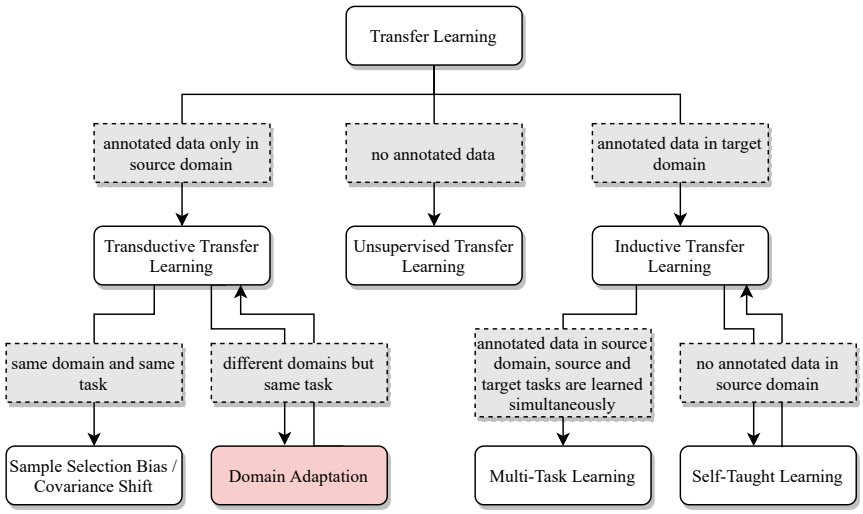


Figure 4.1: **Overview of Transfer Learning**: Domain adaptation is a type of transductive transfer learning where the same task is performed in different, but related domains with annotated data only in the source domain. The figure is based on [Triess et al. 2021a].

timely aspects, and many more. For scalable automated driving, fast development cycles, simulated data, and changing sensor setups also play a major role. Usually, when a perception system is exposed to such a domain shift, its performance drops drastically [Ganin and Lempitsky 2015]. However, it is possible to pass knowledge from a different but related source domain to the desired target domain with transfer learning. Specifically, **D**omain **A**daptation (DA) requires no manual annotations to adapt to new domains and therefore promises a cheap and fast solution to deal with domain shifts (compare fig. 4.1).

This chapter presents a comprehensive review of the recent progress in DA methods and formulates interesting research questions specifically targeted towards LiDAR perception. This chapter is based on [Triess et al. 2021a] and contains verbatim quotes of that work[1].

The chapter is organized as follows: Section 4.1 explains the terminology of DA and includes an overview on typical baselines, datasets, DA applications, and metrics. Section 4.2 categorizes common DA approaches for LiDAR. Section 4.3 discusses different aspects of the presented approaches and gives an outlook on interesting research directions that are partially addressed in this dissertation.

1 The creation of the survey paper was initially driven by Larissa Triess and was then continued as joint work with Mariella Dreissig and Christoph Rist.

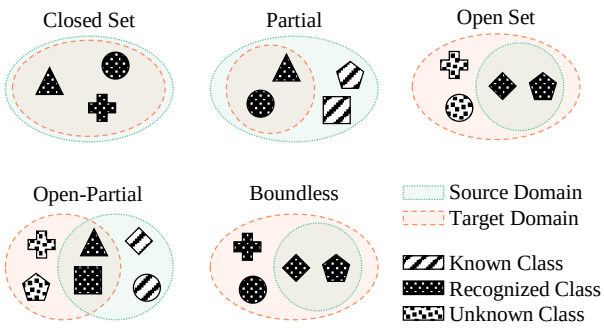


Figure 4.2: **Intersections between Source and Target Domains**: Domain adaptation can be subdivided based on the classes considered in the learning process. All classes of the source domain are known and are provided with labels. The classes that also occur in the target domain get recognized by the perception network. The ones that are not in source, but in target do not get recognized (except for boundless DA) and remain unknown. The figure is based on [Triess et al. 2021a].

4.1 BACKGROUND

Wilson and Cook [2020] provide an extensive survey on image-based DA approaches. This dissertation uses similar terminology but extends the DA methods with LiDAR-specific categories and focuses on LiDAR-related literature. The following gives an introduction to the building blocks for deep learning-based LiDAR DA research.

4.1.1 *Domain Adaptation*

Domain **A**daptation (DA) is a special type of transfer learning [Pan and Yang 2010]. Fig. 4.1 shows the localization of DA research in the field of transfer learning which is divided into three major categories: unsupervised transfer learning, where no annotated data is used; transductive transfer learning, where annotated data is only available in the source domain; inductive transfer learning, where annotations are available in the target domain.

DA is a type of transductive transfer learning where annotated source data but no annotated target data is available. It is therefore also called unsupervised domain adaptation in works that use a different terminology [Wilson and Cook 2020]. The learning process is defined by performing the same task in different but related domains. Related domains refer to domains that are placed in a similar setting, such as outdoor driving scenarios, whereas different domains refer to a specific aspect that differs, for example sunny versus rainy days.

In multi-class classification, DA can be subdivided based on the classes of the source and target domains, and on the classes considered in the learning process (fig. 4.2). Most papers deal with

Closed Set DA, where all classes appear in both the source and target domains. In Partial DA, just a subset of the classes from the source domain appears in the target domain. These two variants are easiest to handle, since no special treatment of unknown objects is required. Open Set DA is the opposite of Partial DA, such that just a subset of the classes from the target domain appears in the source domain. If both sets have both common and unique classes, it is called Open-Partial DA, which is the most common scenario for an open-world application such as autonomous driving. There also exists boundless DA which is an Open Set DA where all target classes are learned individually.

Wilson and Cook [2020] suggest a categorization of DA methods that reflects the different lines of research in that field. These include: domain-invariant feature learning, domain-mapping, normalization statistics, ensemble methods, and target discriminative methods. Section 4.2 structures the LiDAR-based DA approaches into these categories. At the time of writing, the literature did not provide works on ensemble methods or target discriminative methods for LiDAR. Yet, there exist approaches that are specific for LiDAR applications which use domain-invariant data representations. Therefore, [Triess et al. 2021a] introduces this additional category.

4.1.2 *Baselines*

Compared to DA in the camera world, the field of LiDAR DA is rather small at this time. Therefore, many LiDAR papers compare their DA approaches to image baselines to compensate for the lack of LiDAR baselines. This section gives a short overview of the baselines used in the presented papers and this dissertation.

The entropy minimization technique is one of the most often referenced baselines. Vu et al. [2019] introduce **Adv**ersarial **Ent**ropy Minimization (AdvEnt) which minimizes the distribution between the source and target based on self-information. Chen et al. [2019b] claim that the gradient of the entropy is biased towards samples that are easy to transfer in the entropy minimization approach. Therefore, they propose a maximum squares loss to balance the gradient of well-classified target samples and prevent the training to be dominated by easy-to-transfer samples. Morerio et al. [2018] show with **Min**imal-**Ent**ropy Correlation Alignment (MinEnt) that entropy minimization is induced by the optimal alignment of second order statistics between source and target domains. On this basis, they propose to use Geodesic instead of Euclidean distances, which improves alignment along non-zero curvature manifolds. MinEnt is used in chapter 7 for benchmarking purposes.

Other image methods that are used as baselines for DA are: CyCADA [Hoffman et al. 2018], an advanced CycleGAN [Zhu et al.

2017]; FeaDA [Chen et al. 2017b], a joint global and class-specific domain adversarial learning framework; and OutDA [Tsai et al. 2018], a multi-level adversarial network that performs output space DA at different feature levels.

4.1.3 *Applications and Use-Cases*

Using simulators for autonomous driving applications gained a lot of interest in the past years and also increased the research on *sim-to-real* DA. Similarly, *geography-to-geography* DA has to address changes in geographical and environmental regions that largely differ in shapes of otherwise similar objects, e.g. traffic signs. Adverse weather conditions, such as fog or rain can substantially deteriorate the detection capabilities of a LiDAR, since laser beams are being reflected and scattered by the droplets or particles in the atmosphere. Therefore, *weather-to-weather* DA considers different weather scenarios and seasons. In contrast to DA for cameras, *day-to-night* DA is not important to investigate for LiDAR, since LiDAR is an active sensor that is almost independent from external illumination.

Another important application in the field of development cycles and vehicle setup is the case of *sensor-to-sensor* DA. It tackles the differences in resolution, mounting position, or other sensor characteristics, like measurement range, noise characteristics, and reflectivity estimates. Most of the related work considers a far more general case for their research, namely *dataset-to-dataset*. It involves multiple of the above mentioned DA applications at once. Several publicly available driving datasets are used to develop and evaluate the adaptation capabilities. Here, *geography-to-geography* and *sensor-to-sensor* usually occur at once, often paired with seasonal changes, making this task especially challenging. This dissertation addresses the following use-cases: *sim-to-real* in section 6.2, *sensor-to-sensor* in section 6.1, and *dataset-to-dataset* in chapter 7.

4.2 METHODS

This section presents the state of the art on DA for LiDAR-based environment perception. The approaches are either data-driven, such as domain-invariant data representation (section 4.2.1), domain mapping (section 4.2.2), and normalization statistics (section 4.2.4), or model-driven, such as domain-invariant feature learning (section 4.2.3). Remaining methods are presented in section 4.2.5.

4.2.1 *Domain-Invariant Data Representation*

A domain-invariant representation is a hand-crafted approach to move different domains into a common representation.

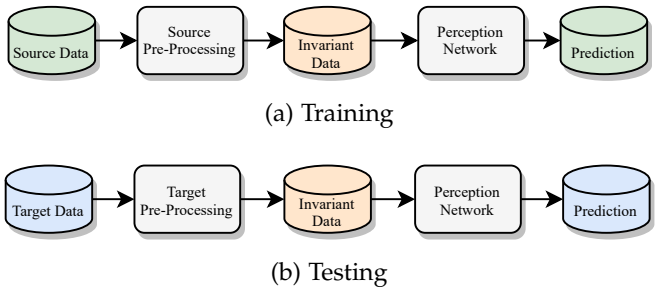


Figure 4.3: **Domain-Invariant Data Representation**: The data from the source domain at train-time (a) and the data from the target domain at test-time (b) are both converted into a hand-crafted common representation prior to being fed to the perception pipeline. The figure is based on [Triess et al. 2021a].

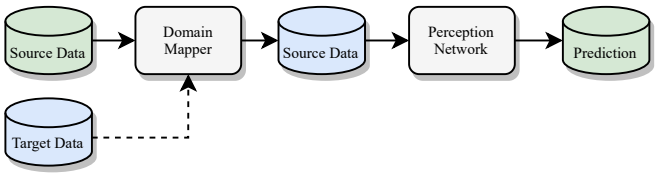
Fig. 4.3 shows that this approach is basically a data pre-processing after which a regular perception pipeline starts. It is mostly used to account for the *sensor-to-sensor* domain shift and receives special attention in LiDAR research. Available sensors vary in their resolution and sampling patterns while resulting point clouds are additionally influenced by the mounting position and the recording rate of the sensor. Consequently, the acquired data vary considerably in their statistics and distributions.

This data distribution mismatch makes it unfeasible to apply the same model to different sensors in a naive way. One method is to align the sampling in 2D space which uses LiDAR sensor-view images to either up-sample the data or drop scan lines to align the sensor resolution [Triess et al. 2019; Shan et al. 2020; Elhadidy et al. 2020; Alonso et al. 2020]. However, this is a simplification, since different sensors are usually not only characterized by the number of scan lines, but also by a multitude of other factors, such as the vertical FOV. Other methods use geometric representations in 3D space that are less prone to domain differences [Yi et al. 2021; Piewak et al. 2019]. Many methods also include a normalization of the input feature spaces with respect to different mounting positions by spatial augmentations and replacing absolute LiDAR coordinates with relative encoding schemes [Rist et al. 2019; Alonso et al. 2020].

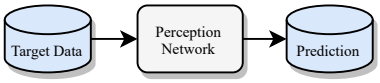
All approaches mentioned in this section can only account for small domain gaps, as they solely address the domain gap of the data representation and not the data content.

4.2.2 *Domain Mapping*

Domain mapping aims at transferring the data of one domain to another domain and is most often used in *sim-to-real* and *dataset-to-dataset* applications. Fig. 4.4 shows a typical setup for domain mapping. Annotated source data is usually transformed to appear like target data, creating a labeled pseudo-target dataset. With the



(a) Training



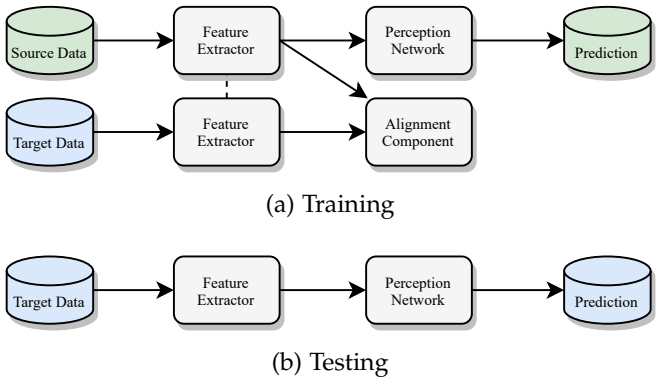
(b) Testing

Figure 4.4: **Domain Mapping**: This is the most commonly used configuration for domain mapping. During training (a) the labeled source data is conditionally (dashed line) mapped to the target domain where a perception network is trained. At test time (b), the trained perception network can directly be applied to the target data. The figure is based on [Triess et al. 2021a].

transformed data, a perception network is trained which can then be applied to target data at test time.

For images, domain mapping is usually done adversarially and at pixel-level in the form of image-to-image translation with conditional GANs [Mirza and Osindero 2014; Choi et al. 2018; Royer et al. 2020; Benaim and Wolf 2017; Taigman et al. 2017]. The generator translates a source input to the target distribution without changing the underlying semantic meaning. A perception network can then be trained on the translated data using the known source labels. The translated data shall have the same appearance as the target data. The number of papers that adversarially generate realistic LiDAR data is limited and most approaches use unmodified image GANs, such as CycleGAN [Zhu et al. 2017], and apply them to top-view projected images of the LiDAR scans [Saleh et al. 2019; Sallab et al. 2019a,b]. Caccia et al. [2019] provide an unsupervised method for both conditional and unconditional LiDAR generation in projection space and test their method on reconstruction of noisy data. The generated data does not possess any point-drops as they usually occur in real-world data. Therefore, DUSty [Nakashima and Kurazume 2021] additionally incorporates a differentiable framework that can sample binary noises to simulate these point-drops and mitigate the domain gap between the real and synthesized data. Similarly, ePointDA [Zhao et al. 2021] learns a dropout noise rendering from real data and applies it to synthetic data. A similar method is proposed and used in section 6.2.1.1 of this dissertation.

There also exists a number of methods that do not rely on adversarial training. The non-adversarial mapping techniques primarily focus on the sampling and distribution differences between LiDAR sensors. Alonso et al. [2020] use a data and class distribution alignment strategy. Langer et al. [2020] use a re-sampling technique via



(a) Training

(b) Testing

Figure 4.5: **Domain-Invariant Feature Learning**: A feature extractor network and an alignment component learn a domain-invariant feature encoding (a). At test time (b), the domain-invariant feature extractor is applied to the target data. The figure is based on [Triess et al. 2021a].

frame accumulation to address *sensor-to-sensor* domain shifts. In a second step, the semantic segmentation model has to be re-trained with geodesic correlation alignment to align second-order statistics between source and target domains to generalize to a *dataset-to-dataset* setting [Morerio et al. 2018; Wu et al. 2019].

Non-adversarial mapping techniques are usually simpler and better controllable than adversarial mapping methods (cf. section 6.3). Therefore, also the trend in domain mapping seems to move towards non-adversarial methods.

4.2.3 *Domain-Invariant Feature Learning*

State-of-the-art methods in domain-invariant feature learning employ a training procedure that encourages the model to learn a feature representation that is independent of the domain. This is done by finding or constructing a common representation space for the source and target domain. In contrast to domain-invariant data representations, these approaches are not hand-crafted but use learned features.

If the classifier model performs well on the source domain using a domain-invariant feature representation, then the classifier may generalize well to the target domain. The basic principle is depicted in fig. 4.5. Common approaches for domain-invariant feature learning can be categorized into divergence minimization and discriminator-based approaches. The former minimizes a suitable divergence measure, e.g. **E**uclidean **C**orrelation **A**lignment (ECA) between the representation of the source and the target domain within a DNN. Wu et al. [2019] adapt the work of Morerio et al. [2018] to use a loss function based on the geodesic distance between the batch statistics from both domains. Jaritz et al. [2020] use a cross-modal

loss for an information exchange between 2D images and 3D point clouds that benefits the overall performance in presence of a domain shift. Discriminator-based methods use adversarial training to force the feature encoder to learn only domain-invariant features [Jiang and Saripalli 2020; Wang et al. 2019b; Zhao et al. 2019a]. Saltori et al. [2020] present a different but interesting take on DA for LiDAR point clouds, where they exploit temporal consistency in the detection to generate pseudo-labels on the target domain. Following, they built a model which does not rely on annotations from the source domain by utilizing a variant of self-taught learning.

4.2.4 *Normalization Statistics*

The primary use of normalization techniques is to improve training convergence, speed, and performance. These advantages have initially been verified on image datasets, image related tasks, and their respective model architectures. On LiDAR data, normalization techniques are used equally for improved feature extraction for a variety of tasks [Wu et al. 2019; Zhou and Tuzel 2018; Lang et al. 2019; Qi et al. 2017a; Su et al. 2018]. On images, the properties of normalization are used for explicit DA. A set of normalization statistics per domain are expected to separate domain knowledge from task knowledge. This encourages DNNs to learn style-invariant representations of input data. Conversely, manipulating the distribution of intermediate layer activations is explicitly used for image style transfer. However, experimental studies that verify a strong similarity between style normalization on camera images and sensor or scene normalization on LiDAR are absent.

In DNNs, normalization layers improve training convergence by aligning the distributions of training data and therefore controlling internal covariate shift and the scale of the gradients. Distribution alignment is implemented by a normalization of mean and variance of activations over partitions of the batch-pixel-feature tensor. The most prominent normalization technique is **B**atch **N**ormalization (BN) [Ioffe and Szegedy 2015]. Building on the same basic idea, subsequent related normalization procedures [Wu and He 2018; Nam and Kim 2018; Ulyanov et al. 2016] address issues with BN related to implementation, network architectures, or certain data domains.

Adaptive BN is a simple and straightforward DA method that re-estimates the dataset statistics on the target domain as a natural extension of the batch norm approach [Li et al. 2017]. However, this normalization approach alone does not lead to a satisfactory object detection performance in a LiDAR *sensor-to-sensor* DA setup [Rist et al. 2019]. When training on multiple image data domains simultaneously, switching between per-domain statistics is used for DA

on image tasks as the second stage of a two-stage approach [Chang et al. 2019]. Initial pseudo-labels are iteratively refined using separate batch norm statistics for each domain. The effectiveness of per-domain statistics on LiDAR domain gaps has not been verified experimentally.

4.2.5 *Other Methods*

Recently, it is demonstrated that teacher-student knowledge distillation can increase performance on a target domain in an unsupervised DA setting on LiDAR [Caine et al. 2021]. A teacher network is trained on the source domain and creates pseudo-labels on the target domain. The smaller student network is then trained on source and target domains simultaneously. A setup to adapt to a domain with different geometries is used as experimental evaluation. The pseudo-label trained student networks show better generalization capabilities on the target domain than the teacher networks. Practitioners benefit from the simplicity of the approach and the option to adhere to inference time budgets with small student networks. However, in practice the sensor setup itself might be different instead of only the scene geometry. This yields a more difficult problem as the networks need to work on both domains simultaneously.

4.3 DISCUSSION

This section discusses the main challenges that remain open in DA for LiDAR perception after reviewing related literature They pose interesting research directions for future works.

4.3.1 *Comparability and Transfer from other Modalities*

An essential part of research is the comparability between different approaches to foster further research in promising directions. In most papers, the success of the DA process is measured by the performance of a downstream perception task. However, this has two major drawbacks. First, it is assumed that the quality of the domain adaptation process directly correlates with the performance changes in the downstream perception. However, there is no proof for this yet. Second, the methods are still not comparable, since they all use different datasets, task settings, label sets, and report different metrics. One reason might be that most of the advanced DA approaches in LiDAR only emerged recently, therefore no metric or baseline prevailed.

To mitigate the lack of a LiDAR-specific baseline, many of the presented works use image-based approaches as a baseline comparison for their own work. However, these are usually unfair evaluations,

since the baseline methods are not optimized for LiDAR data. Therefore, chapter 5 introduces a metric to judge the realism of LiDAR data, which is then applied in chapter 6 to investigate the connection between the data quality and the perception performance.

4.3.2 *Discrepancies in Domain Gap Quality*

The size of a domain gap can be measured in terms of model performance on a given task if target labels are available. Consequently, this measure of the apparent size of a domain gap is model-specific and task-specific. Nevertheless, a change of the LiDAR sensor seems to cause a greater impact on the final performance across several tasks and models than changes in weather or location. It seems that a *sensor-to-sensor* domain gap is not easily covered by learned and implicit DA methods such as normalization statistics and adversarial domain mapping that work well in the image domain. Hand-crafted DA methods based on the explicit geometric properties of LiDAR data and their representation are the only ones that yield reasonable results on *sensor-to-sensor* domain shifts so far. The results in chapter 6 and chapter 7 will support these observations.

4.3.3 *Relevance of Cross-Sensor Adaptation*

Various sensor types were developed in the past decade. Available LiDAR sensors mainly vary in their design and functionality, i.e. rotating and oscillating LiDARs [Li and Ibanez-Guzman 2020; Royo and Ballesta-Garcia 2019]. Consequently, the scan-pattern and thus the data representation differs considerably between the sensors.

Although the mechanical spinning LiDAR is mainly used for perception in the autonomous driving research [Behley et al. 2019; Caesar et al. 2020], all sensor types have their benefits. The publicly available *PandaSet* dataset [Scale AI 2020] is one of the first datasets to incorporate two different sensor types: a 360° covering spinning LiDAR and an oscillating LiDAR with a snake scan-pattern. Since it is unclear which type of sensor will prevail in the context of autonomous driving in the future, it is crucial to advance the development of sensor-invariant perception systems. The presented geometric and volumetric data representation approaches (section 4.2.1) are valuable to fully benefit from the different sensor types. Constructing a domain-invariant data representation in an intermediate step makes it possible to re-use already trained DNNs and apply them to new sensor setups. This is essential to lower the research costs and to keep up with the sensor development. However, methods that leverage the features of such an invariant data representation, without the additional overhead of constructing

such a canonical domain are better suited to be used in a real-time application. This dissertation proposes such a method in chapter 7.

4.3.4 *Adaptation in Different Weather Scenarios*

LiDAR sensors are heavily impacted by adverse weather conditions, such as rain or fog, which cause undesired measurements and leads to perception errors. There exists some work on denoising LiDAR perception [Heinzler et al. 2020], but the subject is not specifically tackled in a DA setting. With the release of new datasets containing adverse weather scenarios [Carballo et al. 2020; Bijelic et al. 2020], it is possible to foster the research for *weather-to-weather* applications. This dissertation does not cover *weather-to-weather* DA.

4.3.5 *Generative Models for Domain Translation*

The category of adversarial domain mapping techniques (section 4.2.2) includes only four approaches, none of which is capable to generate realistic LiDAR point clouds in 3D (they use top-view projections of the point clouds) [Saleh et al. 2019; Sallab et al. 2019a,b]. This is surprising, since the same strategy is thriving in the image world, where a lot of research is conducted to generate realistic images. There are two possible explanations to it: either, it is simply not necessary to use generative models to create realistic point clouds, in the sense that other approaches are far more powerful, or it is not possible to achieve the required high quality of the generated data. Either way, up until now there exists no study that either proves or contradicts any of these assumptions. Chapter 6 provides deeper insights into this discussion.

4.3.6 *Open-Partial Domain Adaptation*

The majority of the presented approaches deal with *dataset-to-dataset* applications, a manifold adaptation task, where both the sensor type and the environment change. This makes the task particularly difficult. However, another effect comes to light. Usually, datasets have unique labeling strategies and include a different set of classes. For example, *SemanticKITTI* has 28 semantic classes while *nuScenes* divides into 32 classes. Since the domains have both common and unique classes, this is called an Open-Partial DA problem [Toldo et al. 2020].

However, none of the presented approaches directly tackles the Open-Partial formulation, but rather performs a label mapping strategy that allows them to address this as a Closed Set DA problem. Some of the segmentation approaches do not focus on segmenting the entire scenery from the start, but only perform foreground

segmentation, for example cars versus background [Yi et al. 2021]. A common strategy when semantically segmenting entire scenes, is to find a common minimal class mapping between the two datasets, that discards all classes that cannot be matched [Alonso et al. 2020]. Some works even re-label one of the datasets to perfectly match the label definitions of the other dataset [Langer et al. 2020].

In the image world, there are already works that deal with the Open-Partial and Open-Set DA problems [Busto and Gall 2017; Saito et al. 2018; Luo et al. 2020]. In the LiDAR world, this field still holds a lot of potential for future research. A good strategy for these questions can help to advance in scalable systems for automated vehicles and can be regarded as part of the future work for this dissertation.

5 A METRIC TO QUANTIFY THE REALISM OF LIDAR POINT CLOUDS

CONTENTS

When working with domain mapping methods (cf. section 4.2.2), an integral part of the process is the creation of new data with the mapping process. This data is then used to train a perception model for the target domain. For this step to be successful, it is necessary that the generated data is close to the target distribution. In a GAN training, this is usually measured by the discriminator. However, the discriminator does not provide an absolute measure, but only indicates the relative realism of the generated data with respect to the current state of the generator and its own training status. Therefore, finding the optimal checkpoint during training is complicated and requires an absolute metric, as often used in the image domain. As discussed in section 4.3.1, such metrics are rare in the LiDAR domain.

To this end, this chapter proposes a novel approach to quantify the realism of LiDAR point clouds. Relevant features are learned from real-world and synthetic point clouds. The resulting metric can then assign a quality score to LiDAR samples without requiring any task specific information. Experiments confirm the soundness of the realism metric and show reliable interpolation capabilities between data with varying degree of realism.

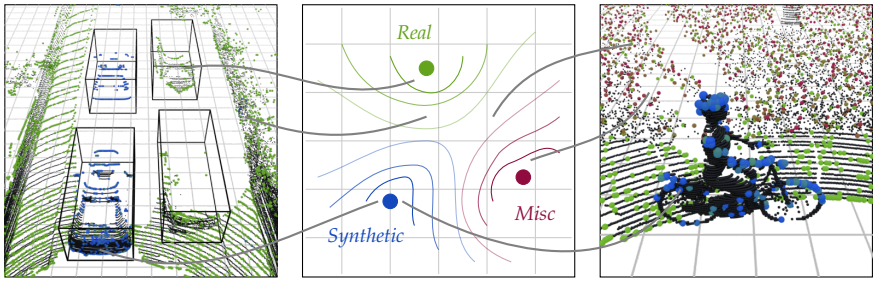


Figure 5.1: **Proposed Approach**: The realism measure has a tripartite understanding of the 3D-world (middle). The left and right image show the color-coded metric scores for query points on two example scenes. Both scenes are from the real-world dataset KITTI (*Real*) and are augmented with dynamic objects from the simulated CARLA dataset (*Synthetic*). 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*. The figure is based on [Triess et al. 2022b].

The chapter is based on [Triess et al. 2021b] and [Triess et al. 2022b] and contains verbatim quotes of these works. It starts with the introduction in section 5.1 and the presentation of related quality measures in section 5.2. Section 5.3 describes the properties and design of the proposed realism metric. Subsequently, section 5.4 presents the experimental setup to validate the proposed approach. Eventually, section 5.5 evaluates the proposed concept and section 5.6 discusses the observations. The proposed metric is used in chapter 6 to determine the quality of generated LiDAR point clouds.

5.1 OVERVIEW

Simulations and generative models, such as GANs, are often used to synthesize realistic training data samples to improve the performance of perception networks [Park et al. 2019b; Xu et al. 2021; Löhdefink and 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 and Romano 2006], little work is published about how to quantify the realism of point clouds [Shu et al. 2019; Triess et al. 2021b, 2022b]. 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 chapter closes this 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 [Saleh et al. 2019; Shan et al. 2020]. Alternatively, some works apply annotation transfer [Sallab et al. 2019a] or use the ***E**arth **M**over's **D**istance* (EMD) as an evaluation criterion [Caccia et al. 2019]. However, these methods require either annotations associated with the data or a matching target sample, 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 to reduce the domain gap between generated data and target data on which the trained perception model is applied [Triess et al. 2021a]. The correlation between realism and perception performance becomes evident in the next chapter (cf. fig. 6.3). Thus, 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, this chapter proposes a reliable metric that gives a quantitative estimate about the realism of generated LiDAR data. Fig. 5.1 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. Hierarchical feature set learning is used to confine features locally in space to avoid learning global scene context. An adversarial learning technique is used to discourage the network from encoding dataset-specific information which enables robust quantification of unseen data distributions. The resulting metric does not require any additional annotations.

The contributions of this chapter are:

- A novel way to learn a measure of realism in point clouds by learning hierarchical point set features on a proxy classification task.
- The adaptation of an adversarial technique from the fairness-in-machine-learning domain in order to eliminate dataset-specific information which allows the measure to be used on unseen datasets.

- A demonstration on how the fine-grained local realism score can be used for anomaly detection in LiDAR scans.

5.2 RELATED WORK

This section discusses GAN evaluation measures and their applicability to generated LiDAR data in section 5.2.1. Since this chapter introduces a novel learning-based LiDAR metric, section 5.2.2 gives a brief overview on metric learning.

5.2.1 *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; Borji 2019]. They can be divided into two major categories: qualitative and quantitative measures.

5.2.1.1 *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. Previous work uses 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. 2017] to provide a qualitative measure for realism in **R**ed-**G**reen-**B**lue (RGB) images. In contrast to [Ledig et al. 2017], where untrained people were asked to determine the realism, Triess et al. [2019] require 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.

5.2.1.2 *Quantitative Evaluation*

Quantitative evaluation is performed over a large collection of examples, often in an automated fashion. Table 5.1 categorizes a number of quantitative GAN measures into six categories according to their properties.

Table 5.1: **GAN Evaluation Measures**: This table categorizes GAN evaluation measures and states their most important pros (⊕) and cons (⊖) relative to the application of judging LiDAR point cloud realism. This table is based on [Triess et al. 2022b].

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. 2015b])
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 and 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 and 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 and Li 2017]	independent of data modality, per-sample score	high run-time because of nearest neighbor matching
Model Comparison	GAM [Im et al. 2016], TWRSR [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

FEATURE-BASED Feature-based metrics measure the quality 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 features from the ImageNet dataset [Deng et al. 2009]. This makes them exclusively applicable to camera image data. The ***F**réchet **P**oint Cloud **D**istance* (FPD) [Shu et al. 2019] is applicable to single-object point clouds, as it is based on features from the ShapeNet dataset [Chang et al. 2015b]. In contrast to the proposed 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 Most distribution-based measures are independent of the data domain and thus can be used to evaluate GANs operating on point clouds [Goodfellow et al. 2014; Theis et al. 2016; Tolstikhin et al. 2017; Gretton et al. 2012; Arora et al. 2018; Richardson and Weiss 2018]. 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. Some even need additional visual inspection, such as the Birthday Paradox Test [Arora et al. 2018].

CLASSIFICATION-BASED Another common approach is to use classification networks to assess the quality of GAN outputs [Arjovsky et al. 2017; Radford et al. 2016; Isola et al. 2017; Santurkar et al. 2018; Lehmann and Romano 2006; Yang et al. 2017]. **C**lassifier Two-**S**ample **T**est (C2ST), for example, assesses whether two samples are drawn from the same distribution [Lehmann and Romano 2006]. This requires freshly trained discriminators for each test on a held-out subset of the data.

OUTPUT COMPARISON Among others, computing reconstruction errors is one common method to assess generated data [Wang et al. 2016; Xiang and Li 2017]. For point clouds, ***E**arth **M**over's **D**istance* (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 There exist two types of model comparison techniques. The first includes very simple metrics to evaluate the discriminator itself, namely precision and recall. These two are helpful to be tracked in any GAN training and are capable to 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, such as the **T**ournament **W**in **R**ate and **S**kill **R**ating (TWRSR) [Olsson et al. 2018]. However, these measures are labor intensive and of high complexity as they require several network combinations and trainings.

LOW-LEVEL STATISTICS Computing low-level statistics of the underlying data is easy and fast. Examples are **S**tructural **S**imilarity **I**ndex **M**easure (SSIM), **P**eak **S**ignal-to-**N**oise **R**atio (PSNR), sharpness, contrast, mean power spectrum. These are partially specific for RGB images and are not capable to capture higher-level information.

This chapter aims at providing a practical quantitative metric to determine the realism of individual generated samples via learned features. Therefore, the proposed method is considered to be a combination of the following categories: feature-based, distribution-based, and output comparison. Revisiting this extensive collection of related literature shows that none of the existing measures can fulfill all the desired requirements presented in the next section (section 5.3.1) and are therefore not suitable as a LiDAR metric.

5.2.2 *Metric Learning*

The metric proposed in this dissertation uses an adversarial training technique to push features in a similar or dissimilar embedding. 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. In explicit metric learning setups, it is common to use siamese networks that are 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 and Ailon 2015; Dong and Shen 2018].

5.3 LIDAR REALISM METRIC

This section introduces the concept of the realism metric. Section 5.3.1 identifies the most important objectives of the metric

and derives the desired properties. Based on these properties, the architecture is designed and presented in section 5.3.2.

5.3.1 *Objective and Properties*

The aim of the metric is to provide an estimate of the level of realism for arbitrary LiDAR point clouds. The metric is designed 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 quality measures and GAN measures, a useful LiDAR point cloud metric is expected 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

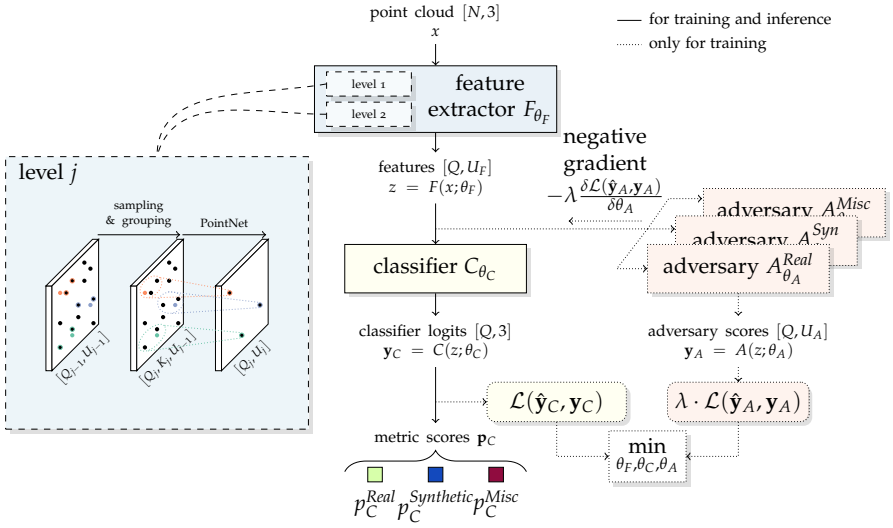


Figure 5.2: **Metric Architecture**: The feature extractor F_{θ_F} uses hierarchical feature set learning from PointNet++ [Qi et al. 2017b] to encode information about each of the Q query points and their K nearest neighbors. The neighborhood features z are then passed to the classifier C_{θ_C} which outputs probability scores $\mathbf{p}_C$ for each category (*Real*, *Synthetic*, *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; Raff and Sylvester 2018]. λ is a factor that regulates the influence of the adversarial loss, weighting the ratio of accuracy versus fairness. The figure is based on [Triess et al. 2022b].

LiDAR data generation. This enables monitoring the realism of the generated sample during the training time of the network.

The metric is implemented in such a way that the described properties are fulfilled. To differentiate the metric from a GAN discriminator, one has to note that a discriminator is not *transferable* to unseen data, since it recognizes only one specific data distribution to be realistic.

5.3.2 *Architecture*

Fig. 5.2 shows the architecture of the proposed 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*, *Synthetic*, *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 (*Synthetic*), and all the others (*Misc*), e.g. distorted or randomized data. Refer to fig. 5.1 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 (section 5.3.2.1) and are then classified into the three proposed categories (section 5.3.2.2). An adversarial learning technique is used to obtain a *transferable* metric, the concept is explained in section 5.3.2.3.

5.3.2.1 *Feature Extractor*

The blue parts of fig. 5.2 visualize the PointNet++ [Qi et al. 2017b] 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. The KNN algorithm uses the Euclidean distance between the points in 3D space. 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++, KNN search is used instead of radius search to find the neighboring points. PointNet++ was proposed for point clouds from the ShapeNet dataset [Chang et al. 2015b], 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, KNN search was found to be more practical to obtain meaningful neighborhoods in LiDAR scans compared to radius search.

Each abstraction level uses 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.

5.3.2.2 *Classification*

The yellow parts in fig. 5.2 show the classification parts of the metric model. 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, one for each of the categories *Real*, *Synthetic* and *Misc*. The component $p_{C,q}^{Real}$ 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.

5.3.2.3 *Adversarial Training*

To obtain a *transferable* metric network, the metric leverages a concept often used to design fair network architectures or domain losses [Beutel et al. 2017; Raff and 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, the feature extractor is actively discouraged from encoding information that is specific to the distribution of a single dataset. In other words – using fair networks terminology [Beutel et al. 2017] – the concrete dataset name is treated as a sensitive attribute. This procedure can improve the generalization ability towards unknown data.

To achieve this behavior, a second output path for adversarial learning is added. It consists of one adversary A_{θ_A} for each category (see orange parts in fig. 5.2). Each of the adversaries predicts classification probabilities for all the datasets in their respective category, i.e. the probability that the sample stems from a particular dataset within the category. For simplification, the following explanation assumes that 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 (in this case $U_A^{Real} = 2$, $U_A^{Synthetic} = 2$, and $U_A^{Misc} = 3$). Following the designs proposed in [Beutel et al. 2017; Raff and Sylvester 2018], all network components are trained by minimizing the losses for both heads, $\mathcal{L}_C = \mathcal{L}(\mathbf{y}_C, \hat{\mathbf{y}}_C)$ and $\mathcal{L}_A = \mathcal{L}(\mathbf{y}_A, \hat{\mathbf{y}}_A)$, but reversing the gradient in the path between the adversary input and the feature extractor. A reversed gradient is obtained by multiplying the gradient with -1. 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

$$\begin{aligned}\min_{\theta_F,\theta_C,\theta_A} \mathcal{L}\Big(C\big(F(x;\theta_F);\theta_C\big),\hat{y}_C\Big) \\ + \mathcal{L}\Big(A\big(J_\lambda[F(x;\theta_F)];\theta_A\big),\hat{y}_A\Big)\end{aligned} \tag{5.1}$$

with θ being the trainable variables and J_λ a special function [1]

$$J_\lambda[F] = F \quad \text{but} \quad \nabla J_\lambda[F] = -\lambda \cdot \nabla F \tag{5.2}$$

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

In the applications of the related literature [Beutel et al. 2017; Raff and Sylvester 2018], the sensitive attribute and the requested attribute are often correlated but have no direct coupling. In the presented case, this means that different data samples from the same dataset can 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. A mathematical reasoning is provided in appendix A.2.2. 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 $A : \{A^{Real}, A^{Synthetic}, A^{Misc}\}$ have the trainable variables $\theta_A : \{\theta_A^{Real}, \theta_A^{Synthetic}, \theta_A^{Misc}\}$. 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 $\mathcal{L}_A = \mathcal{L}_{A^{Real}} + \mathcal{L}_{A^{Synthetic}} + \mathcal{L}_{A^{Misc}}$.

5.4 EXPERIMENTAL SETUP

Table 5.2 shows the datasets used for this work. They are divided into two different groups: one that is used to train and evaluate the metric while the other group is only used for evaluation. The strict separation of training and evaluation datasets, additionally to the

1 The notation of the function is mathematically not correct, as the gradient of a function is inherently defined by the function. However this notation is commonly used in related literature [Beutel et al. 2017] and shows how the function is implemented.

Table 5.2: **Datasets**: The table lists the datasets for each category. 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. The table is based on [Triess et al. 2022b].

	Dataset	Samples	Train.	Eval.
Real	*KITTI* [Geiger et al. 2013]	18,329	✓	✓
	nuScenes [Caesar et al. 2020]	28,130	✓	✓
	PandaSet [Scale AI 2020]	-	✗	✓
Synthetic	*CARLA* [Dosovitskiy et al. 2017]	106,503	✓	✓
	GeometricSet	18,200	✓	✓
	GTA-V LiDAR [Hurl et al. 2019]	-	✗	✓
Misc	*Misc 1,2,3*	∞	✓	✓
	Misc 4	-	✗	✓

training and test splits, enables the demonstration that the proposed method is a useful measure on unknown data distributions. In both cases alike, the datasets stem from one of three categories: *Real*, *Synthetic*, *Misc*.

Within the *Real* category, publicly available real-world datasets are used for training (*KITTI*, *nuScenes*) and evaluation (*PandaSet*). For *Synthetic*, the *CARLA* simulator is used with an implementation of the sensor specifications of a Velodyne **HDL-64** (HDL-64) sensor to create ray-traced range measurements. *GeometricSet* 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, the synthetic data is augmented with little noise at training time, such that they are not trivially distinguishable from the other categories. For evaluation, the *GTA-V LiDAR* dataset is used [Hurl et al. 2019]. Its 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, a third category is added, *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 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, 1000 samples from a different split of each dataset are used to obtain the 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.

5.5 EVALUATION

This section provides extensive evaluations to demonstrate the capability and applicability of the proposed method. Section 5.5.1 sets the factor to achieve an optimal balance between accuracy and fairness. Section 5.5.2 presents the classification performance on completely unseen data distributions. Section 5.5.3 shows the feature embedding capabilities of the fair training strategy in an ablation study. Section 5.5.4 shows that the proposed metric is capable to interpolate the space between the training categories in a meaningful way. As an important application, section 5.5.5 demonstrates how the proposed method can be used for anomaly detection in point clouds. In the end, section 5.5.6 elaborates on the limitations of the proposed method. Additional visualizations and implementation details are provided in appendix A.2. The application of the metric in domain adaptation settings is shown in chapter 6.1.

5.5.1 *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.

Fig. 5.3 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 training 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 the *Real* datasets in themselves are already very diverse, especially compared to the *Synthetic* 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 *Synthetic* datasets. This makes it hard for the model to extract

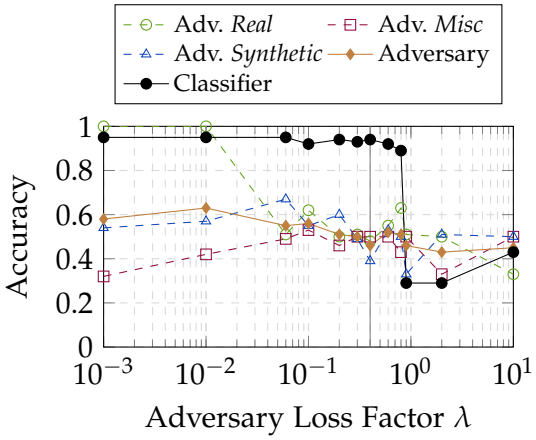


Figure 5.3: **Accuracy vs. Fairness**: Shown are the accuracy values of classifier and adversaries over the loss factor λ. At small λ, the classification accuracy is high which means good performance. However, adversary accuracy is also quite high (at least for *Real*) which means no fairness in this 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. The figure is based on [Triess et al. 2021b, 2022b].

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

All presented experiments use a factor of $\lambda = 0.3$ (indicated by the gray vertical line). Here, the classifier has a good performance (93%) while the adversary operates slightly above chance level (50%).

5.5.2 *Metric Results*

The metric network is run on the evaluation datasets, as well as on the test split of the training datasets. Fig. 5.4 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 *Synthetic* score.

The unknown datasets obtain notable results (upper part). Qualitative example frames are depicted in fig. 5.5. 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

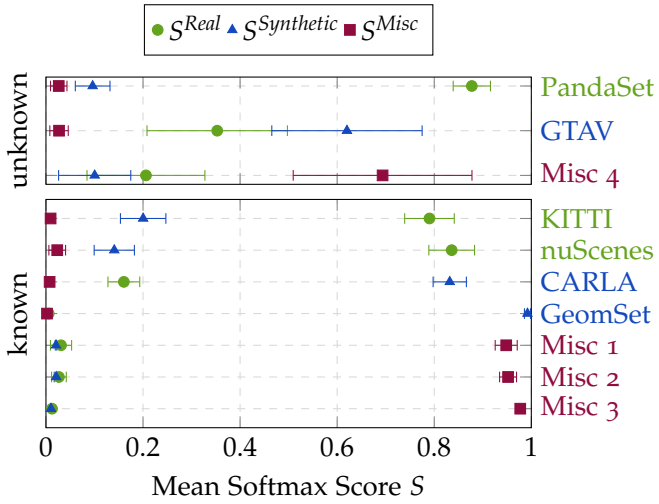


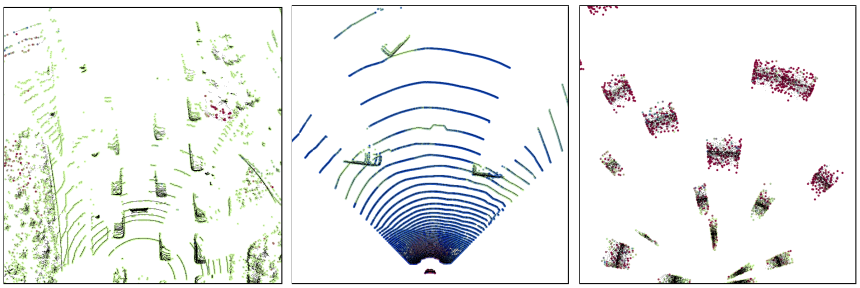
Figure 5.4: **Metric Results**: The figure shows the metric output S (mark) and the standard deviation (whisker) for *Real*, *Synthetic*, and *Misc* on different datasets. The lower part shows the results for the test split of the known datasets, while the upper part depicts one unknown dataset from each category. The color of the dataset name indicates the respective category. The figure is based on [Triess et al. 2021b, 2022b].

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 *Synthetic* scores.

The *Synthetic* dataset *GTA-V LiDAR* has a slightly different behavior. Here, $S^{Synthetic}$ 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 *Synthetic* scores is a systematic behavior of the metric caused by the data distribution. Fig. 5.5b shows that the high *Real* scores mainly stem from regions containing vehicles. *GTA-V LiDAR* 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.

5.5.3 *Adversary Ablation*

The proposed approach uses the adversarial loss to embed features for *Real*, *Synthetic*, and *Misc* while at the same time omit dataset-specific information as far as possible. To demonstrate the feature encoding behavior, additional metric networks are trained with varying adversary configurations and the learned features are visualized for the validation data.



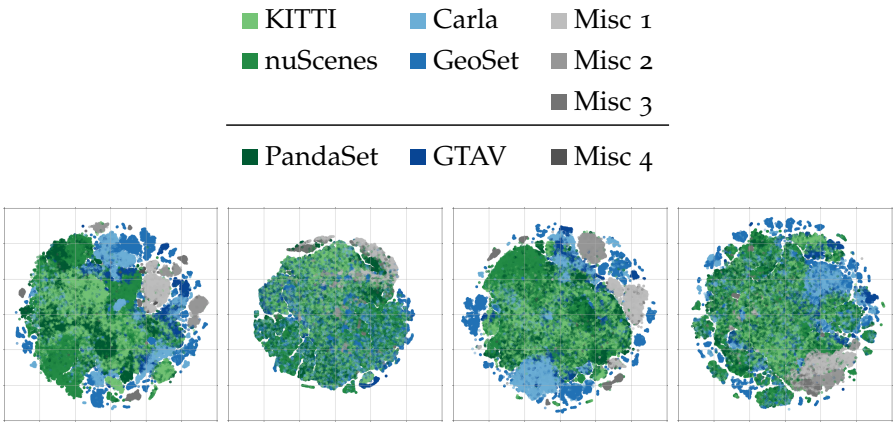
(a) PandaSet (b) GTAV (c) Misc 4

Figure 5.5: **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 *GTA-V LiDAR* dataset for *Synthetic*. The main reason for the overall high *Real* scores seem to be caused by regions that contain cars. (c) shows an example for the Misc 4 dataset. The figure is based on [Triess et al. 2022b].

Fig. 5.6 shows plots of the **t**-Distributed **S**tochastic **N**eighbor **E**mbedding (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 minimizing 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*, *GTA-V LiDAR*, *Misc 4*. They are included for demonstration purposes regarding the *transferability* to unseen data.

The two extreme cases of the configuration form fig. 5.6a and fig. 5.6b. Fig. 5.6a represents the metric as a simple classifier without an adversary, where each shade of each color forms 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. Fig. 5.6b, 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, the metric uses per-category adversaries. In [Triess et al. 2021b], the adversary was only applied for *Real*, as depicted in fig. 5.6c. In [Triess et al. 2022b], one adversary for each category is used, as represented by fig. 5.6d. 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, the per-category approach (fig. 5.6d) also



(a) No adversary (b) Common adversary (c) *Real* adversary (d) Individual adversaries

Figure 5.6: **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 the previously proposed method that only used an adversary for the *Real* category [Triess et al. 2021b]. (d) depicts the proposed approach, where one adversary per category is trained. The figure is based on [Triess et al. 2022b].

shows mixed features among the blue and gray points, whereas the previous approach shows more distinct clusters. This is especially visible for *Misc*, where fig. 5.6c has one cluster for each shade but fig. 5.6d better combines them.

Further, the feature visualization shows that the unknown dataset *PandaSet* is fully integrated into the *Real* cluster for the proposed method, as opposed to when using no adversary. The clusters of the unknown *GTA-V LiDAR* dataset mostly overlap with *Synthetic*, but also partially with *Real*. This aligns with the metric results that are shown previously for *GTA-V LiDAR*, where parts of the data containing vehicles appear quite realistic.

The adversary ablation is conducted only qualitatively, because it is not possible to compare the quantitative scores of the different versions. A metric trained as in fig. 5.6b could have a different allocation of scores in range $[0, 1]$ than a metric as in fig. 5.6d.

5.5.4 *Feature Continuity*

Another important property of such a metric is the feature continuity. This means that the metric must produce reasonable transitions in the feature space between known data points, i.e. the support sets used for training. The feature continuity is an important property when applying the metric to data that does not belong to one of

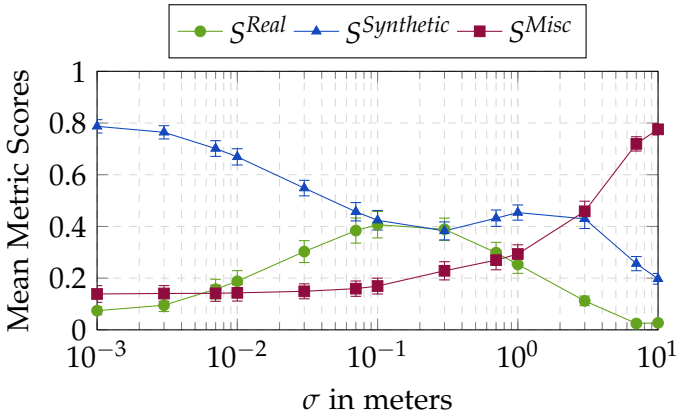


Figure 5.7: **Point Cloud Distortion**: Increasing levels of Gaussian noise are applied to the distance measurements of *CARLA* point clouds. For small additive noise levels of a few centimeters, the synthetic *CARLA* data appears more realistic. For higher σ, the noise is dominant, as indicated by decreasing *Real* and *Synthetic* scores. The figure is based on [Triess et al. 2021b].

the chosen categories, but stem from a transition in between. One example can be a point cloud that was recorded with a dirty sensor, where parts of the point cloud are distorted due to false reflections. Another example is a point cloud generated by a GAN that learns a *sim-to-real* mapping, but does not yet work perfectly.

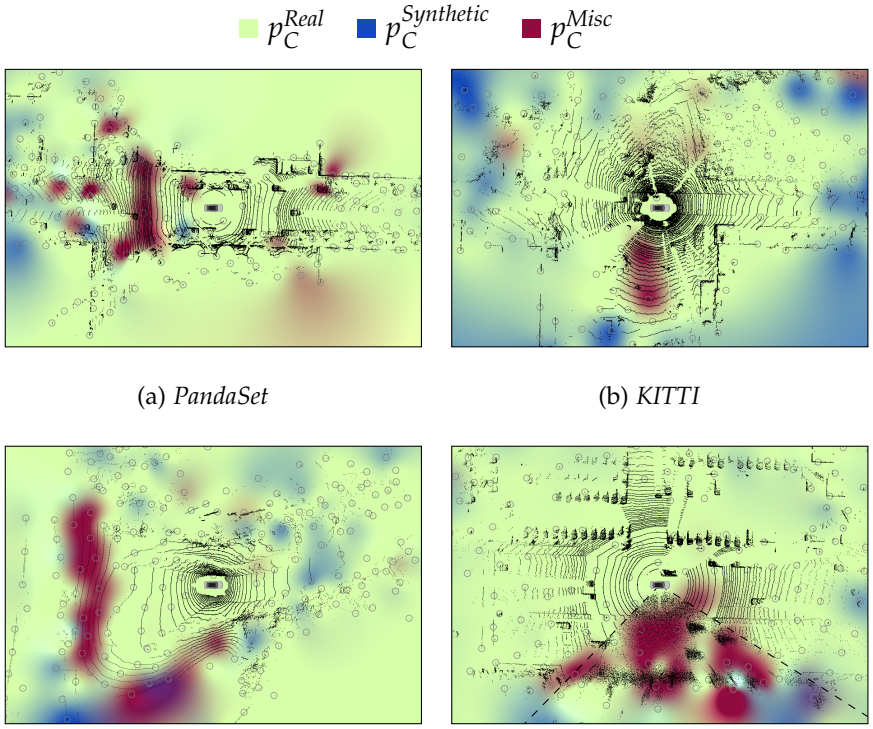
To make the experiments in this section understandable and for the sake of simplicity, the *CARLA* test split is used and augmented with different levels of noise added to the distance measurements of the point clouds. This shall approximate the simulation of different generative network states in a controllable manner. The noise is normally distributed with zero mean and varying standard deviation σ.

Fig. 5.7 shows the mean metric scores S over a wide range of additive noise levels applied to *CARLA* data. Notably, at low noise levels, the *Real* score increases. This reflects the fact that ideal synthetic data needs a certain level of range noise in order to appear more realistic. On the other hand, at high noise levels, the data barely possesses any structure, as indicated by high *Misc* and low *Real* and *Synthetic* scores.

The smooth transitions between the states can be interpreted as an indication for a disentangled latent representation within the metric network. Further, it shows the necessity for all three support sets when working with real-world and synthetic data.

5.5.5 *Anomaly Detection*

Given the locality aspect of our method, it is possible to find anomalies or regions with differing appearance within a single scan. Fig. 5.8 shows three examples where the proposed metric outputs



(a) *PandaSet* (b) *KITTI*

(c) *nuScenes* (d) *PandaSet* with noise

Figure 5.8: **Localization of Anomalies**: Example scenes with low *Real* scores. The colors represent interpolated p_C scores which are discrete values located at the query points (gray circles). In (a), the purple area marks a road section with extreme elevation changes. In the lower half of (b), the metric highlights seemingly floating branches of a huge tree, that enter the LiDAR field-of-view from above. (c) shows an unusual scene in a dead end road with steep hills surrounding the car. (d) illustrates a *PandaSet* sample where the region indicated by dashed lines has been manually distorted with additive Gaussian noise. The figure is based on [Triess et al. 2021b].

the lowest *Real* score within all test scans and one example where a part of the horizontal field of view is distorted with noise. The method can successfully identify anomalies within the support sets, i.e. *KITTI* and *nuScenes*. To a lesser extent, the method is also capable of identifying similar unusual constellations in completely unseen data (*PandaSet*). Weird sensor effects, such as the region altered with additive Gaussian noise in the *PandaSet* scan, are also detected by the metric (purple areas).

Since anomaly and out-of-distribution detection are essential components of automated driving systems, these results open a promising research direction. Extending the proposed metric to be used for anomaly detection in a wider range is therefore left for future work.

5.5.6 *Limitations*

LiDAR data is sparse and the proposed approach is dependent on measurements. Wherever there is no measurement data, the metric cannot give information about the data quality. This limits the ability for reliable prediction of very sparse data, for example at high distances ($> 100\ m$) or when information is lost in advance to metric score computation. On the other hand, this enables the processing of point clouds that are not a full 360° scan of the world.

5.6 CONCLUSION

This chapter presented a novel metric to quantify the degree of realism of local regions in LiDAR point clouds. Extensive experiments demonstrated the reliability and applicability of the proposed metric on unseen data. Through adversarial learning, a feature encoding is obtained that is able to adequately capture data realism instead of focusing on dataset specifics. The approach provides reliable interpolation capabilities between various levels of realism without requiring annotations. The predictions of the method correlate well with visual judgment, unlike reconstruction errors serving as a proxy for realism. Additional experiments demonstrated that the local realism score can be used to detect anomalies – further development is left for future work.

The introduction of the realism metric now enables detailed inspection of LiDAR data and therefore provides a practical and quantitative method to train generative models for LiDAR generation. The metric can be used to define a stopping criterion in GAN training and makes it easier to design domain mapping systems, as proposed in the next chapter.

6

DOMAIN ADAPTATION VIA DATA GENERATION FOR DOMAIN MAPPING

CONTENTS

This chapter addresses the problem of domain mapping for two applications. Section 6.1 addresses *sensor-to-sensor* adaptation by up-sampling from low resolution to high-resolution sensor data. Section 6.2 deals with *sim-to-real* adaptation, where simulated point clouds from *CARLA* shall be transferred to the *KITTI* space. An introduction to domain mapping and these domain gap applications can be found in chapter 4.

6.1 UP-SAMPLING FOR SENSOR MAPPING

A common domain adaptation scenario is to bridge gaps between different sensors. LiDAR sensors considerably vary in many properties, such as resolution, field of view, capture rate, or layer distribution. All these properties can cause performance degradation when a perception model trained on sensor A is applied to data from a sensor B. In the past years the development of LiDAR sensors showed a trend from low-resolution sensors, i.e. 16 or 32 layers, to higher-resolution sensors, i.e. 64 or 128 layers. This section investigates the effect of various LiDAR up-sampling techniques in the performance of downstream semantic segmentation to account for these development cycles.

This section is based on [Triess et al. 2019, 2022b] and contains verbatim quotes of these works[1]. The contributions of this section are:

- The comparison of five different up-sampling methods in terms of reconstruction errors and realism metric.
- Extensive evaluation of the influence of data quality and the effect on downstream semantic segmentation in the target domain.

6.1.1 *Related Work*

The aim of up-sampling is to estimate the high-resolution visual output of a corresponding low-resolution input. This section uses cylindrical two dimensional projections of structured LiDAR point clouds, therefore it is vital to also take into account analogous approaches on RGB images to solve this task. A sizable amount of literature exists on RGB image up-sampling. Yang et al. [2014] present a comprehensive evaluation of prevailing RGB up-sampling techniques prior to the adoption of CNNs. More advanced techniques, such as SRCNN [Dong et al. 2016], outperform these traditional methods. However, they cannot cope with data that features missing measurements, since dense input representations are required. The traditional methods, on the other hand, can easily be applied to cylindrical LiDAR projections. Due to their low computational complexity they can be used for real-time applications. However, the traditional re-sampling techniques are not able to restore the high-frequency information, i.e. fine details in the resized input, due to the low-pass behavior of the interpolation filters [Gavade and Sane 2014].

Over the last years a wide variety of up-sampling techniques for 3D data emerged. A number of methods considers point cloud up-sampling as a depth completion task by projecting the laser scans into sparse depth maps [Dolson et al. 2010; Liu et al. 2013; Hui et al. 2016; Uhrig et al. 2017] Here, the original structure of the input point cloud is lost and transformed into a high-resolution depth map at camera image resolution. Yu et al. [2018] proposed PU-Net which learns multilevel features per point to reconstruct an up-sampled unordered set of points, i.e. a generic point cloud. However, in some applications it is important to maintain the ordered point cloud structure provided by LiDAR sensors. First, downstream perceptual algorithms which have been designed for the structured

1 The up-sampling concept proposed in [Triess et al. 2019] is in part a contribution of the Master's thesis of Larissa Triess [Triess 2018] and is therefore no contribution of this dissertation. It is contained here for completeness. All experiments, the GAN setup, and the evaluations of this section are not contained in the Master's thesis and are therefore new contributions of this dissertation.

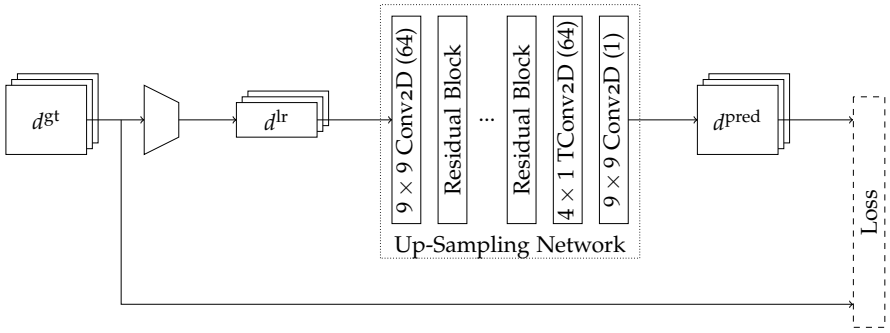


Figure 6.1: **Up-sampling Network**: The input to the network is a down-sampled spherical LiDAR projection d^{lr} of size $(L/2 \times W)$ with information about the missing measurements. The residual up-sampling network outputs an up-sampled LiDAR projection d^{pred} of size $(L \times W)$ with in-network up-scaling. The prediction and the high-resolution target d^{gt} with same size are both inputs to the loss. The figure is based on [Triess et al. 2019].

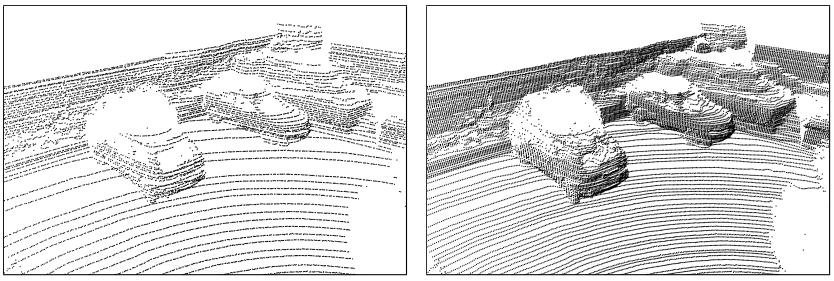
low-resolution data can still be applied. Second, it is possible to reuse valuable data recordings by up-sampling it to higher resolutions, especially when new LiDAR sensors with more layers are introduced to the market. Therefore, Triess et al. [2019] proposed a CNN-based approach to up-sample LiDAR point clouds that achieves better performance than traditional methods, but at the same time retains the desired structure of the LiDAR scan. The following section introduces the network proposed in [Triess et al. 2019] and extends the approach with an adversarial loss in section 6.1.3.2.

6.1.2 *Up-Sampling Network*

Fig. 6.1 shows the up-sampling system which transforms a low-resolution LiDAR scan into a corresponding high-resolution output. The traditional approaches, i.e. nearest neighbor and bilinear interpolation, do not require any supervision. For the learning-based approaches, the prediction is compared with the *Ground Truth* high-resolution scan via a point-wise or adversarial loss function in order to train the up-sampling network.

The model is a deep residual CNN [He et al. 2016]. It increases the resolution of a LiDAR projection by a factor $f = (f_h, f_w)$ to produce a high-resolution output. Here, only the vertical resolution is increased, such that the horizontal resolution stays the same ($f_w = 1$). The output can be understood as the equivalent of a recording from a sensor with f_h times as many layers with equivalent other settings, such as FOV.

The design of the up-sampling network is inspired by the *image transformation network* by Johnson et al. [2016]. Triess et al. [2019] introduce the following modifications to the architecture. The network consists of 16 residual blocks with no activation at the output layer



(a) Original scan with 64 layers (b) Up-sampled scan with 128 layers

Figure 6.2: **Up-sampled *KITTI* Scene**: The left image shows an original sensor recording from *KITTI*. The right image shows the same scene, up-sampled to 128 layers with the $\mathcal{L}_1$-CNN. Since the network is fully convolutional, it is able to up-sample from 64 to 128 layers, though it was trained on 32 layers. This figure is adapted from [Triess et al. 2019].

and the kernel of the fractionally strided convolution, i.e. transposed convolution, has a size of $(4, 1)$. Following Johnson et al. [2016], the convolutional layers within the residual blocks are followed by spatial **B**atch **N**ormalization and a **Re**ctified **L**inear **U**nit (ReLU) nonlinearity. The first and last layers use 9×9 kernels while all remaining convolutions have kernel sizes of 3×3.

The input to the network is a LiDAR scan with H / f_h layers, represented by a two-dimensional projection of shape $H / f_h \times W$. With up-sampling factors $(f_h, 1)$, the output is a high-resolution distance image of shape $H \times W$. Since the network is fully-convolutional, it can be applied to inputs of any resolution, cf. fig. 6.2.

6.1.3 *Losses*

Standard loss functions from the image domain need modifications to correctly process projected LiDAR scans because of missing measurements in some locations of the data (cf. section 2.1.1). The set of all valid measurements is defined as $\mathcal{V} = \{(i, j) | \text{reflection at } \theta_i, \varphi_j \text{ received}\}$, such that a two-dimensional LiDAR distance image d_{ij} can then be constructed by setting

$$d_{ij} = \begin{cases} r_{ij} & (i, j) \in \mathcal{V} \\ d^* & \text{otherwise} \end{cases} \tag{6.1}$$

with a proxy value $d^* = 0$ for the missing measurements. The losses will only be computed for the measurements contained in $\mathcal{V}$.

6.1.3.1 *Point-wise Loss*

In a supervised setting, up-sampling is a regression problem where a loss function $\mathcal{L}(\mathbf{d}^{\text{pred}}, \mathbf{d}^{\text{gt}})$ compares the generated high-resolution

distance image $\mathbf{d}^{\text{pred}} = \{d_{ij}^{\text{pred}}\}$ with its corresponding ground truth counterpart $\mathbf{d}^{\text{gt}} = \{d_{ij}^{\text{gt}}\}$. The most commonly used error functions for this application are the $\mathcal{L}_1$ and $\mathcal{L}_2$ loss functions. In the case of LiDAR distance images, these loss functions are modified to mask the missing measurements which have been replaced by d^*. Therefore, the modified point-wise loss functions are defined as

$$\mathcal{L}_\alpha = \frac{1}{\alpha \left|\mathcal{V}\right|} \sum_{(i,j)\in\mathcal{V}} \left| d_{ij}^{\text{gt}} - d_{ij}^{\text{pred}} \right|^\alpha \quad \alpha = 1, 2 \tag{6.2}$$

where $\alpha = 1$ describes the **M**ean **A**bsolute **E**rror (MAE) and $\alpha = 2$ describes the **M**ean **S**quared **E**rror (MSE).

6.1.3.2 *Adversarial Loss*

The point-wise loss encourages the network to predict high-resolution LiDAR scans where each point is close to the *Ground Truth* counterpart in a purely spatial sense. A perfect match would be ideal in theory, but this approach can fail to output realistic point clouds in practice. Note that a perfectly realistic point cloud constructed from a slightly rotated *Ground Truth* point cloud would lead to high loss values. Similarly, while an actual scan of a treetop looks like a seemingly random collection of points, an $\mathcal{L}_\alpha$-guided optimization will tend to produce smooth surfaces to decrease the overall distance error. To circumvent this problem, an adversarial loss can be used by constructing a GAN architecture.

The GAN discriminator D is adapted from [Ledig et al. 2017]. The adversarial loss is defined as

$$\min_{\theta_G} \max_{\theta_D} \left\{ \log\left[D_{\theta_D}\left(\mathbf{d}^{\text{gt}}\right)\right] + \log\left[1 - D_{\theta_D}\left(G_{\theta_G}\left(\mathbf{d}^{\text{lr}}\right)\right)\right] \right\} \tag{6.3}$$

with the up-sampling model G. Further information on the architecture are provided in appendix A.3.

6.1.4 *Metrics*

The quality of the up-sampled data is judged with three categories of metrics: 1) reconstruction errors (section 6.1.4.1), 2) realism metric (section 6.1.4.2), 3) downstream segmentation (section 6.1.4.3).

6.1.4.1 *Reconstruction Errors*

Reconstruction 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 the case of up-sampling, this target is the original high-resolution sample from which the low-resolution sample is generated as input to the up-sampling network.

Specifically, ***C**hamfer's **D**istance* (CD), MAE, and MSE are computed 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

$$\begin{aligned} d_{CD}(P^p, P^t) = &\frac{1}{|P^p|} \sum_{p^p \in P^p} \min_{p^t \in P^t} \|p^p - p^t\|_2 \\ &+ \frac{1}{|P^t|} \sum_{p^t \in P^t} \min_{p^p \in P^p} \|p^t - p^p\|_2 \end{aligned} \tag{6.4}$$

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. It can be assumed that the best match is always the original high-resolution image of the same scene, thus the metrics simplify to Cov ≈ 1.0 and MMD $\approx d_{\text{CD}}$ due to the paired translation task. Therefore, these metrics are not reported additionally to the reconstruction errors in the evaluation.

6.1.4.2 *Realism Metric*

Please refer to chapter 5 for detailed information.

6.1.4.3 *Downstream Segmentation*

The aim of domain adaptation is to improve the perception performance in the target domain. The main assumption of most metrics that judge the quality of generated data is that better data leads to better perception models. To investigate this claim, the following evaluation includes an additional semantic segmentation task in the target domain. The up-sampling methods are used to transform data from the source (low-resolution) to the target (high-resolution) domain. For each method a pseudo-dataset of different quality is generated. These pseudo-datasets are then used to train semantic segmentation models which are finally evaluated on the target domain. It is expected that if the realism metric ranks 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].

SqueezeSegV2 [Wu et al. 2019] and DarkNet21 [Milioto et al. 2019] are used as segmentation models. Both use scan unfolding as their data representation, as proposed in section 3.2.1.1. The original label mapping is altered to predict nine classes instead of

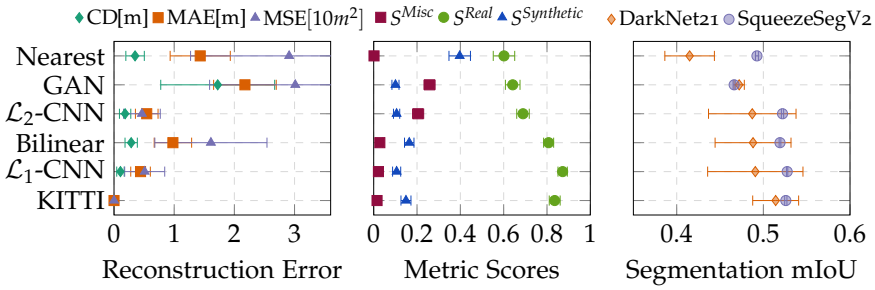


Figure 6.3: **Metric Scores for Up-Sampling Methods**: The vertical axis lists five methods to perform 4× LiDAR scan up-sampling and the high-resolution target data ("KITTI"). The left plot shows the reconstructions errors of different baseline measures. The middle plot shows the three parts of our realism measure, where S^{Real} indicates the closeness to real-world data. 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 their respective human visual judgment ratings. The figure is based on [Triess et al. 2022b].

the original 19. Details on the architecture and training can be found in appendix A.3.

6.1.5 *Evaluation*

This section presents the experimental results of the up-sampling task and discusses the relationship between the different metric results.

Fig. 6.3 is divided into three parts horizontally that represent the three categories of metrics, as introduced in section 6.1.4. The leftmost plot shows the baseline metrics, the middle shows the results of the proposed metric network, the rightmost plot shows the segmentation performance. The vertical axis on the left lists five different methods that generate the up-sampled versions of *KITTI* data, as described in section 6.1.3. For all, the up-sampling factor is set to $f = (f_h, f_w) = (4, 1)$. 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[2]. 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. The realism score, on

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

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, cf. [Johnson et al. 2016; Ledig et al. 2017].

The upper row of fig. 6.4 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. Bilinear interpolation works very well on large surfaces, but produces 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. 6.3 shows the overall results, while table 6.1 shows class-wise results. Additionally, the bottom row of fig. 6.4 visualizes segmented example point clouds produced by the models trained with the respective data. Both segmentation models, SqueezeSegV2 [Wu et al. 2019] and DarkNet21 [Milioto et al. 2019], 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, 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 6.1 show that the GAN achieves 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. 6.4 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

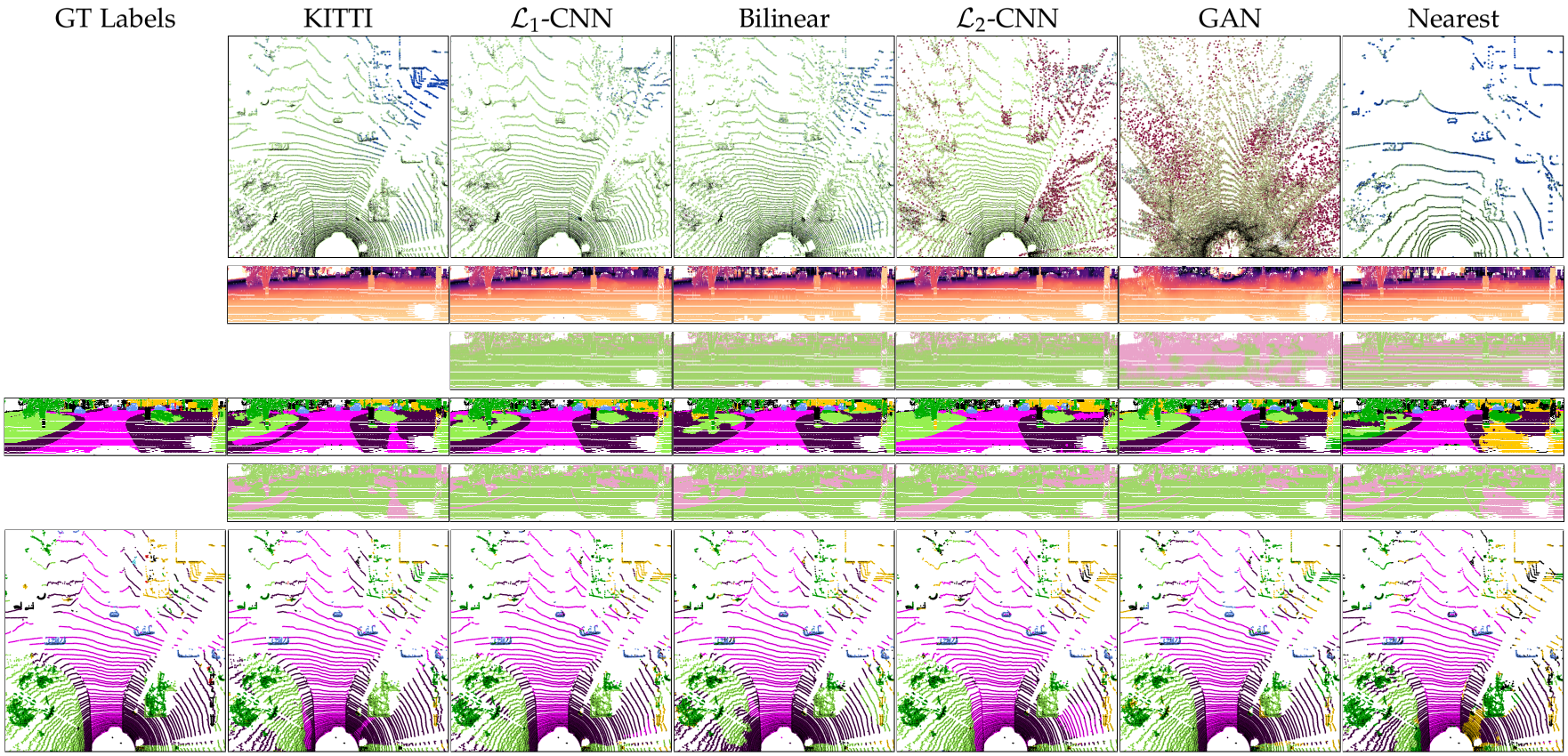


Figure 6.4: **Qualitative Up-Sampling and Segmentation Results**: The first row shows the metric results on an up-sampled *KITTI* scene. The original scan is shown in column "KITTI". The colors are soft interpolations of *Real* ■, *Synthetic* ■, and *Misc* ■. The second row shows the color-coded depth 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 6.1. The ground truth semantic labels are shown in the leftmost column. For better comparison, the fifth row 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. The figure is based on [Triess et al. 2022b].

Table 6.1: **Semantic Segmentation Performance**: The table lists the evaluation results of the DarkNet21 model for point-wise semantic segmentation. 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. The IoU is reported per class and accuracy and mIoU are reported over all classes. All numbers in the table are given in %. Best results are shown in **bold**, second best are underlined.

KITTI Version	accuracy	mean IoU	person	two-wheeler	large-vehicle	vehicle	road	sidewalk	terrain	construction	vegetation
Class frequency	train / val		0.05 / 0.16	0.06 / 0.12	0.22 / 0.10	4.4 / 6.2	21.7 / 18.9	14.5 / 12.1	07.7 / 12.9	21.3 / 14.3	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
$\mathcal{L}_2$-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
$\mathcal{L}_1$-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**

their edge outlines in the projection. The third row of fig. 6.4 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 apparently noisy $\mathcal{L}_2$-CNN version, relative errors are quite low and therefore outlines are clearly visible in the depth projection (second row). 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.

6.1.6 *Summary*

The experiments show a correlation between measured training data realism and final perception performance. Qualitatively however, the segmentation performance seems to be less affected by a reduced point cloud quality than expected by judging from the 3D images. This can be 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. These observations lead to two considerations:

First, visual judgment is highly dependent on the chosen data representation and their visualization. This is an important reason to use a quantitative metric as proposed in chapter 5 on a large amount of data. Second, the proposed metric might be more reliable to estimate the performance of downstream tasks operating on 3D space.

6.2 MAPPING FROM SIMULATION TO REAL-WORLD

Supervised learning with deep models requires huge amounts of data, which for some rare scenarios can be difficult or even impossible to obtain in real world data recordings. Simulation can help in generating samples to train deep networks and to validate existing perception and behavior algorithms before conducting field experiments. While there exist a few simulation frameworks that can offer a LiDAR model capable of providing geometrically realistic results, none of the openly available models has a concept of measurement noise. This is a huge drawback for research, as dealing with measurement noise is an integral part of any perception algorithm. As seen in section 6.1.5, the noise characteristics play an important role in the realism of the data. In addition to the missing measurement noise, simulated data has large appearance differences and simplified scene compositions compared to the target domain. While the latter can be compensated by putting in reasonably more effort into creating more realistic scenes, the first two points are more complicated to achieve. This section therefore proposes to use a GAN to transfer the data from the simulated world into the real world in order to compensate for the missing noise and varied appearance between the two domains. Semantic segmentation is used for demonstration purposes of the adaptation process.

6.2.1 *Sim-to-Real GAN*

Fig. 6.5 shows the overview of the proposed domain adaptation GAN. The structure is related to commonly known domain adaptation GANs, such as SPIGAN [Lee et al. 2019b] and CYCADA [Hoffman et al. 2018].

All data representations within the GAN are two-dimensional cylindrical depth projections of three-dimensional point clouds, further referred to as LiDAR images. The usage of these projections allows similar network structures as for RGB images. However, missing measurements that can occur in the real-world require special treatment to correctly use 2D convolutions on them. A simple fix is to apply an interpolation technique, such as morphological closing, on the images and assign pseudo measurement values where no real measurement exists in the input images before feeding it as an input

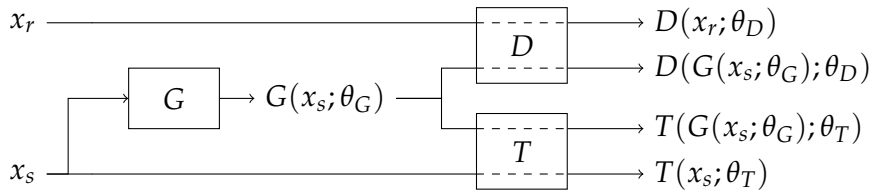


Figure 6.5: **Sim2Real Network**: The generator G transforms the synthetic input point cloud x_s into the real-world domain. The output is then fed to the discriminator D, which predicts the probability of the sample originating from the real distribution. The discriminator does the same for the real-world samples x_r. Additionally, the task predictor T receives the domain adapted $G(x_s;\theta_G)$ and the synthetic data x_s as inputs and predicts their respective semantic label maps. The GAN is then trained as defined in eq. (6.8).

to the generator. This leads to slightly better generation capabilities and more stable training of the GAN.

The GAN consists of three network components and two losses. A generator transforms a synthetic sample into the real-world. In this step it is possible to use additional information, such as normals or roughness, obtained from the simulation, as an additional input to the generator to help generating a realistic output point cloud. Subsequently, a discriminator decides to what degree a sample originates from the synthetic or the real dataset. For the adversarial loss a least-squares error is used

$$\begin{aligned}\mathcal{L}_{\text{GAN}} =& \mathbb{E}_{x_r \sim P_r}\left[\left(D\left(x_r;\theta_D\right)-1\right)^2\right] \\ &+\mathbb{E}_{x_s \sim P_s}\left[D\left(G\left(s_s;\theta_G\right);\theta_D\right)^2\right]\end{aligned} \tag{6.5}$$

where P_r denotes the real-world and P_s the synthetic data representation.

In order to preserve high-level features of the simulated scenery, a task network specialized on semantic segmentation is used. The network predicts the semantic labels for both the synthetic point cloud as well as their transformed equivalent. The predictions for both version shall be equal to the ground truth labels obtained from the simulation in order to ensure a semantics preserving generator. To measure the total task error, the cross-entropy of both versions is combined

$$\mathcal{L}_{\text{T}} = \mathcal{L}_{\text{CE}}\left(x_s, y_s\right) + \mathcal{L}_{\text{CE}}\left(G\left(x_s;\theta_G\right), y_s\right) \tag{6.6}$$

$$\mathcal{L}_{\text{CE}} = -\frac{1}{WH}\sum_{u,v}^{W,H}\sum_{c=1}^{C} \mathbb{1}_{[c=y_{u,v}]} \log\left(T\left(x;\theta_T\right)_{u,v}\right) \tag{6.7}$$

where $\mathbb{1}_{[a=b]}$ is the indicator function.

The total learning objective thus computes to

$$\min_{\theta_G \theta_T} \max_{\theta_D} \alpha \mathcal{L}_{\text{GAN}} + \beta \mathcal{L}_T \tag{6.8}$$

where α and β are the weights for adversarial loss and task prediction loss. All three networks are trained in an alternating fashion.

The following gives more detailed information on the structure of the individual components.

6.2.1.1 *Generator*

The generator represents a function that performs the domain shift between the worlds of *CARLA* and *KITTI*. The architecture uses a Darknet53 backbone and possesses two output heads [Milioto et al. 2019; Redmon and Farhadi 2018]. One of the heads generates an offset map that represents the point-wise domain shift of the sample. Combined with the original sample from the simulator it forms the new domain-adapted point cloud. The combination is represented by an addition in the sensor-view space.

The second head provides a probability map p for each point in sensor-view to represent a valid data representation. A sampling strategy is used to generate the actual valid mask m

$$m_{i,j} = \theta \left(k_{i,j} - p_{i,j} \right) \tag{6.9}$$

with $k_{i,j} \sim U(0,1)$ and the *Heaviside function* θ. This ensures a higher diversity in the resulting map in areas where the network predicts almost equal confidence for being a valid data point or not, while maintaining a consistent map in areas with high confidence, as it would using a normal threshold criteria. Nakashima and Kurazume [2021] recently and independently proposed a similar framework to drop points in synthesized LiDAR scans.

Further, the generator features a noise augmentation which gives the network the possibility to increase variance in its predictions. An additional noise channel is added after each ResNet block in the decoder part of the network. The subsequent convolution is thus able to apply more or less noise in the different feature maps.

6.2.1.2 *Discriminator*

The discriminator is a fully convolutional network following the PatchGAN approach of Isola et al. [2017]. The output scores are given for 2×32 sized patches of the 64×2000 input. Input to the network is the distance channel and the corresponding point valid mask. The network consists of five convolutional layers, each followed by batch normalization and a leaky ReLU activation function. The output layer is two-dimensional convolution with a sigmoid non-linearity.

For real data, the point valid mask has discrete values of 0 and 1. The output of the generator for the valid mask, however, contains continuous values between 0 and 1. This could make it easy for the discriminator to distinguish the two data samples Therefore, the question arises whether it is necessary to adjust the distribution of the mask values. Luc et al. [2016] investigate this issue in a small ablation experiment and conclude that the overall GAN training does not significantly depend on the chosen input mechanism. Therefore, ignoring the distribution difference between the real and generated maps and feeding them directly to the discriminator is a valid option. However, their experiments are not directly comparable to the setup shown in fig. 6.5, as they additionally use cross-entropy loss in their adversarial path. The following introduces two mechanisms for the proposed setup [Triess and Peter 2021].

SMOOTHING The values of the point valid masks for the real data are smoothed to not be discrete at 0 and 1 anymore. Implementation-wise, values are sampled from $-N$ to 0 for all missing measurements and from 0 to $+N$ for all actual measurements with N being a high number. The values are squashed in a range between 0 and 1 with a sigmoid function, with most values being either very close to 0 or 1, depending on whether they do not or do belong to a measurement. Values in between can be interpreted as label noise.

DIFFERENTIABLE DISCRETIZATION The values of the point valid masks for the generated data are discretized with a threshold. Since the discretization is not differentiable, the logistic function is used as a custom gradient, such that

$$f(x) = \begin{cases} 0 & \text{if } x < \tau \\ 1 & \text{if } x \geq \tau \end{cases} \quad \text{but} \quad \frac{\delta f(x)}{\delta x} = \frac{1}{1 + e^{-k \cdot (x - \tau))}} \tag{6.10}$$

with the threshold τ and the steepness k. Other works make use of the *Gumbel-Softmax* [Chris J. Maddison 2017; Eric Jang 2017], a continuous distribution that approximates samples from a categorical distribution.

Experiments confirmed the results of Luc et al. [2016]. However, they also showed that using the *smoothing* technique leads to slightly more variation in the outputs which is caused by additional introduced noise for the discriminator.

6.2.1.3 *Task Predictor*

Semantics are important high-level features that provide information of the scanned environment. To construct a semantics preserving generator in the domain adaptation GAN, the task predictor makes

use of the semantic labels retrieved from simulation. A network with a LiLaNet backbone is used, consisting of five blocks with $(16, 32, 64, 64, 32)$ filters [Piewak et al. 2018]. The task predictor processes cylindrical projections of the simulated and generated point clouds in two separate steps. Probabilities for each of the classes are predicted per point. These probabilities are matched against the true labels with a cross-entropy loss in both cases. Enforcing the same semantic output on both data types results in a semantics preserving generator.

6.2.2 *Experiments*

6.2.2.1 *Datasets*

The synthetic data is obtained from *CARLA* [Dosovitskiy et al. 2017]. It consists of $\sim 120k$ frames and provides semantic label maps. The real-world data is obtained from *SemanticKITTI* [Behley et al. 2019]. It provides point-wise semantic labels of consecutive point clouds for 19 different classes. All results are reported on the validation sequence of the *SemanticKITTI* dataset.

Both datasets represent data from a HDL-64 LiDAR sensor which operates at a frame rate of $10Hz$. The synthetic data is modeled as close as possible to this sensor.

6.2.2.2 *Downstream Perception*

Large amounts of synthetic data with driving scenarios are often used to improve the safety of vulnerable traffic participants. The first aim towards this goal is the increased reliability of detecting these participants. Therefore, the following investigates the performance of a perception algorithm that is trained with and without additional synthetic data. The main objective is to obtain better segmentation performances for pedestrians and riders of bicycles and motorcycles. These classes are typically underrepresented in common datasets which leads to a low accuracy for those categories, however these are the most vulnerable traffic participants.

All experiments are conducted with a RangeNet++ architecture [Milioto et al. 2019]. The network predicts point-wise semantic labels and operates on cylindrical depth projections. A label mapping is established that is compatible with both the *CARLA* and the *SemanticKITTI* dataset. For that reason it is not possible to directly compare the results to the benchmark results published with the architecture [Milioto et al. 2019]. However, a reference experiment with the reduced label set serves as the baseline. The label set contains a single ground class, combined two-wheeler and rider classes for motorcycles and bicycles, as well as their riders, respectively. The currently restricted semantic label set of *CARLA* is

Table 6.2: **Semantic Segmentation Performance**: Mean IoU and class-wise IoU given in percent [%]. The results are reported on the validation split of the *KITTI* dataset. The leftmost column indicates on which data the network is trained. Best results are shown in **bold**.

Training Data	**mIoU**	ground	vehicle	bike	pedestrian	rider	building	fence	vegetation	pole	traffic-sign
KITTI (K)	73	96	94	73	37	66	95	71	87	62	48
CARLA (C)	35	88	62	5	9	10	56	12	50	34	25
C_{Gauss}	42	92	72	4	12	26	83	10	67	26	28
$C_{BlenSor}$	38	83	67	2	11	4	82	19	56	29	33
C_{GAN}	35	89	66	4	11	7	56	10	47	29	28
K+C	73	**97**	93	70	47	65	**95**	69	86	62	49
K+C_{Gauss}	75	96	**94**	69	49	**75**	**95**	69	86	63	39
K+$C_{BlenSor}$	74	**97**	93	72	**50**	69	**95**	69	86	62	**51**
K+C_{GAN}	**76**	96	93	**76**	47	**75**	**95**	**71**	**87**	**65**	50

not the only reason for the label set reduction. Experiments showed that objects like ground, building, and car achieve IoU values close to 100% which does not leave much room or need for improvement. Therefore, the experiments rather focus on improving performance on classes that obtain quite low IoU scores when solely trained with *SemanticKITTI* data, e.g. pedestrians.

6.2.3 *Evaluation*

Table 6.2 shows the experimental results. The row with *KITTI* shows the oracle results, where the network was trained directly on the target domain. The row with *CARLA* shows the results when the network was only trained on the source domain and then applied to the target domain. The performance difference between these two rows is high and shall be reduced. Three methods are used to achieve this. First, simply apply Gaussian noise to the *CARLA* data. Second, apply BlenSor noise, which is a distance dependent noise that also drops points at the ground in far distance. This often occurs in the real-world, when the angle at which the laser hits the ground is too flat. Third, the GAN as proposed in the previous section is used to transform the appearance from the *CARLA* world to the *KITTI* world.

None of the three methods is able to increase the performance high enough in order to reach the oracle performance. The performance gap is especially large for bike, pedestrian, and rider. Interestingly,

the simplest adaptation method, applying Gaussian noise, achieves the best results. It increases the baseline performance from 35% to 42%, which is still a large gap to 73% oracle performance. These results show that the appearance gap is not the only issue here, compared to the up-sampling experiments of section 6.1. Another large component is the scene composition and included objects. When training the GAN, it also tries to incorporate these variations into its adaptation process. This leads to high distortions, similar to the ones seen in section 6.1. However, the model is not able to compensate for the scene variation, since it is designed to model an appearance shift while maintaining the semantics of the scene. As a result, the generated point clouds are less informative to the semantics model in the target domain than simply applying noise to the simulated data and is therefore not suitable as a reliable domain adaptation approach.

Another series of experiments investigates the potential of additional training data to improve the performance of the target domain. This is not an unsupervised domain adaptation setting as before, since annotated data for the target domain is used. The results are shown in the lower rows of table 6.2. The semantics model is pre-trained with four versions of the synthetic *CARLA* data and then fine-tuned with decreased learning rate on the target *KITTI* data. Here, the GAN adapted data leads to a performance increase from 73% to 76%. All classes are slightly increased. The additional data has the most effect on the pedestrian and rider classes. This is mainly caused by many persons being spawned in the simulation process of *CARLA*. There is no significant difference between the different fine-tuning versions, therefore it can be concluded that additional data augmented with some variation alone is already sufficient to increase performance in the target domain.

6.2.4 *Summary*

The experiments show that the *sim-to-real* GAN is not effective in generating a real-world pseudo dataset from *CARLA* that achieves reasonable results on *KITTI*. On the contrary, simpler adaptation mechanisms that solely rely on additive noise, seem to be more effective to solve parts of this task, while still being significantly worse than the oracle. The adaptation process is complicated, because two domain shifts have to be addressed at the same time: appearance and scene composition. However, by design the GAN is not capable to alter the scene composition.

6.3 DISCUSSION AND CONCLUSION

This section presented two different domain adaptation scenarios, low-resolution to high-resolution in section 6.1 and simulation to real-world in section 6.2. For both tasks, a GAN setup and several other adaptation methods have been introduced and investigated. In both tasks, the mapping methods have to generate realistic LiDAR scans in a target domain conditioned on a sample from a source domain.

The experiments show that generating realistic LiDAR data with GANs is complex. The proposed GAN setups are not capable to generate truly realistic LiDAR data. In the literature there are also no examples of generative models to be able to generate realistic LiDAR point clouds. There exist only a few publications that attempt to do so, however visual inspection of their data reveals similar distortion effects, as visible in this work [Saleh et al. 2019; Sallab et al. 2019a,b]. Most commonly, objects such as vehicles are stretched along the visual line into oval shapes and the entire point cloud is enclosed into a maximum distance hull.

[Triess et al. 2022b] is first to investigate the effect between the quality of the generated point clouds and the effect on the perception performance. It can be seen that there is a correlation, but the influence of the quality on the perception is not strong. So even seemingly uninformative data can still have a positive effect on the perception performance. This effect is often seen in CycleGAN architectures [Zhu et al. 2017]. Future work might investigate whether and how it is possible to make use of this property of neural networks.

In this two-stage process, where data has to be generated in order to test on the target domain, the resulting perception performance is highly dependent on the generated data, no matter how realistic it appears. Therefore, many works apply a one-stage process that learns domain-invariant features directly. To that extent, the next section proposes a novel method that uses the underlying scene geometry to learn such domain-invariant features.

7

DOMAIN ADAPTATION VIA GEOMETRY-BASED DOMAIN-INVARIANT FEATURES

CONTENTS

The previous chapter showed that the performance of the domain adaptation in two-stage processes are highly dependent on the data generation capabilities of the domain mapper and are therefore not easy to configure. To overcome this drawback, domain-invariant features can be learned directly in a one-stage process. This chapter proposes a novel method that exploits the underlying geometry of a scene to learn domain-invariant features. If a system is capable to reconstruct the scene geometry from a sparse sampling, it is possible to become independent of the sampling mechanism, i.e. the type of LiDAR sensor that is used for sampling. This is especially useful in case of *sensor-to-sensor* adaptation. However, reconstructing an entire scene to obtain semantics add additional computational overhead. Therefore, this chapter leverages the geometry implicitly by enforcing the latent representation of a semantic scene completion network to be domain-invariant. The proposed method then achieves state-of-the-art domain adaptation performance without requiring to explicitly construct a canonical domain and thus operate in an efficient one-stage process.

The idea to leverage a geometric completion task for implicit domain adaptation is published in patents [Triess and Rist 2021] and [Triess and Rist 2022] [1] .

7.1 OVERVIEW

Domain invariant feature learning is one of the most popular techniques for domain adaptation applications. Usually, a common representation space for source and target domain is learned that makes it less dependent of the domain. More details and an overview of the related work is presented in section 4.2.3.

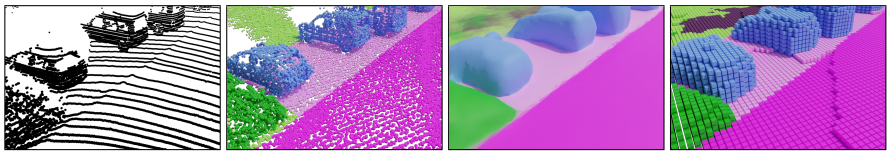
This chapter aims at leveraging the underlying scene geometry to become independent from sensor sampling patterns. The assumption is that a given scene has the same properties and geometry, independent of the sensor that samples the scene. Yi et al. [2021] model this property explicitly by running their semantic segmentation network on a canonical domain that is generated by a voxel completion network. The voxel completion network can be trained in a self-supervised fashion and is specific to each domain. The semantic labeling network is then applied to the completed scene and is therefore shared over domains. Langer et al. [2020] use a similar idea, however, they accumulate LiDAR frames from the source domain and then sub-sample new single frame point clouds with the scan pattern of the target domain. The *sensor-to-sensor* gap is closed explicitly with this approach. An alignment of second order batch statistics between synthetic and real batches of the data enables generalization for *dataset-to-dataset* cases. This is achieved by incorporating **G**eodesic **C**orrelation **A**lignment (GCA) by extending the focal loss with a geodesic loss.

In both works alike it is necessary to explicitly generate such a canonical domain which creates additional computational overhead during the inference in the target domain. This chapter therefore proposes to use the underlying geometry *implicitly*, which requires no need to explicitly construct the canonical domain. The semantic scene completion network by Rist et al. [2021] is used to learn features that encode both geometry and semantics to learn features from a source domain and apply it to the target domain.

The contributions of this chapter are:

- Use a semantic scene completion network to learn domain-invariant features based on an *implicit* aligned feature encoding for scene geometry

1 The idea was initially proposed by Larissa Triess and then refined together with Christoph Rist. Thus, concept and experimental setups are joint contributions. The implementation and experiments are part of the Master's thesis of Bjarne Johannsen [Johannsen 2022], supervised by Larissa Triess.



(a) Input points (b) Accumulated GT points (c) Mesh of continuous output (d) Voxelized output

Figure 7.1: **Semantic Scene Completion**: A semantic scene completion model usually takes sparse points (a) as input and outputs a dense representation, that can be continuous (c) or discrete (d). The accumulated Ground Truth (GT) points (b) are used as a basis for the training target. The images in this figure are taken from [Rist et al. 2021].

- Apply different domain adaptation techniques, such as early-stopping and domain losses.
- Experiments and ablations show the effectiveness and limitations of the proposed method for both high-resolution to low-resolution sensor adaptation and vice versa.

7.2 SEMANTIC SCENE COMPLETION USING LOCAL DEEP IMPLICIT FUNCTIONS ON LIDAR DATA

This section gives an introduction to semantic scene completion and introduces the approach by Rist et al. [2021], which serves as the backbone for the proposed domain adaptation approach.

The goal of scene completion is to recover a dense representation of a scene from a sparse observation. Semantic scene completion is the combination of scene completion and semantic segmentation, such that the dense representation has additional semantic information. Fig. 7.1 shows examples of such completed scenes. Fig. 7.1a shows the sparse input representation in form of a single LiDAR sweep. Many scene completion methods output the dense representation in form of voxels (fig. 7.1d). Rist et al. [2021] learn a continuous representation of the scene (fig. 7.1c). As a training target, accumulated point clouds with *Ground Truth* semantics and a point-target for free space are used (fig. 7.1b).

Fig. 7.2 shows the model architecture of the backbone network, for detailed information refer to [Rist et al. 2021]. The model takes a single LiDAR scan as an input and outputs the corresponding scene completion function. Specifically, the output is a set of densely sampled 3D coordinates with a classification probability vector. The scene completion function maps every 3D position $\mathbf{p}$ within the scene to a probability vector that defines the semantic class of the position $\mathbf{p}$. Positions that are not occupied by any object or other surface are classified as *free space*, while positions with objects are categorized into N semantic classes. Hence, the output describes the

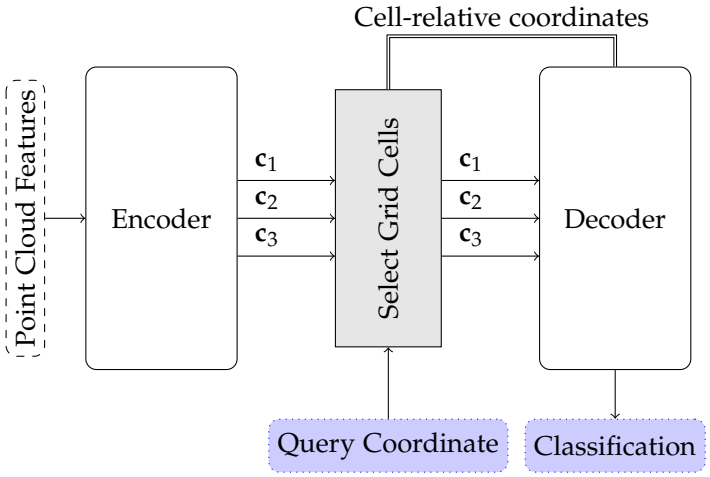


Figure 7.2: **Backbone Network Architecture**: The point cloud features are created by a feature extractor (not shown) that applies a PointNet [Qi et al. 2017a] to a top-view grid of the input point cloud. The CNN-encoder outputs feature maps at three different resolutions ($\mathbf{c}_{\{1,2,3\}}$) that make up the latent representation of the 3D scene. The decoder classifies individual coordinates within the 3D scene. Latent feature vectors and cell-relative coordinates are processed by conditional batch normalization in the decoder. The figure is inspired by [Rist et al. 2021].

geometric and semantic segmentation of space in a $N+1$ classification.

The function is built from many local functions that take two inputs: a query coordinate and a parametrization grid. Fig. 7.2 shows the three parametrization grids $\mathbf{c}_{\{1,2,3\}}$ that describe the scene at three different resolutions. A convolutional encoder is used to generate the feature maps that make up the conditioning grids. The query coordinates, which are simply sampled points in space, and the parametrization grids are then processed with the help of conditioned batch normalization in the decoder to produce the classification output [de Vries et al. 2017].

The network is trained end-to-end with time accumulated and semantically labeled point clouds as the primary training target. The training objective consists of three loss parts: the semantic loss $\mathcal{L}_S$, the geometric loss $\mathcal{L}_G$, and the consistency loss $\mathcal{L}_C$. The loss is formulated as

$$\mathcal{L} = \lambda_S \mathcal{L}_S + \lambda_G \mathcal{L}_G + \lambda_C \mathcal{L}_C \tag{7.1}$$

with the individual loss weights $\lambda_{\{S,G,C\}}$.

The semantic loss is defined as a multi-class cross-entropy loss between the accumulated semantic targets and the semantic predictions. The geometry loss is defined as the binary cross-entropy loss between the occupied and free voxels of the accumulation targets and the free-space prediction on all sampled points. The consistency loss penalizes divergence between the grid cells for a given position

without the need of *Ground Truth* data. Detailed information on the loss functions are given in [Rist et al. 2021].

7.3 METHOD

The method proposed in this chapter is built on two assumptions: First, the underlying geometric encoding of a scene is independent of the sensor sampling pattern that constructs the sparse point cloud representation of that scene [Langer et al. 2020; Yi et al. 2021]. Second, scene geometry and semantics are related and therefore learning features for one task also helps to solve the other task [Kendall et al. 2018]. The backbone model (section 7.2) already learns a joint function of geometry and semantics to solve the task of semantic scene completion. For single frame semantic segmentation in a domain adaptation setup, the proposed method leverages this concept by exploiting the implicit geometric feature representation that is assumed to be domain invariant. In order to obtain single-frame semantics in the inference, the points of the input frame are used as query coordinates in the grid cell sampling step. During training, the query coordinates are generated from a dense sampling scheme over the entire 3D space of the scene extent.

7.3.1 *Baseline Domain Transfer*

The baseline experiments measure the size of the domain gap to which the backbone network is exposed. The model is trained on the source domain and is then - without adaptation - applied to the target domain. As a training objective, the original loss formulation from eq. (7.1) is used. In the following, these experiments are indicated with the name "Backbone". The results can be found in section 7.5.1.

There are two variations for the training with *nuScenes*: One uses all frames with semantic annotations from the dataset as accumulated training targets. The other additionally uses all remaining frames that do not have semantic annotations. This causes some of the target voxels to have an occupied / free-space label, but no semantic information. These locations cannot compute the semantic loss, but still give feedback about the underlying geometry of the scene which helps to reduce the domain gap.

7.3.2 *Using Self-Supervised Target Geometry*

The baseline does not learn anything from the target domain prior to being exposed to it and therefore exhibits large domain gaps. Since this is an unsupervised domain adaptation setting, no semantic annotations are available in the target domain. However, the

geometric information of the target domain can be obtained in a self-supervised fashion.

When using the full information (i.e. geometry and semantics) from the source domain, the training objective is formulated as

$$\mathcal{L}^{\text{source}} = \lambda_S \sum_p \mathcal{L}_S^{\text{source}} + \lambda_G \sum_p \mathcal{L}_G^{\text{source}} + \lambda_C \sum_p \mathcal{L}_C^{\text{source}} \tag{7.2}$$

which is equal to the original loss, as defined in eq. (7.1). All network weights are updated. When training with the target domain information (where only the geometry is available), the training objective is formulated as

$$\mathcal{L}^{\text{target}} = \lambda_G \sum_p \mathcal{L}_G^{\text{target}} + \lambda_C \sum_p \mathcal{L}_C^{\text{target}} \tag{7.3}$$

which then only updates the weights in the encoder, since the decoder also requires semantic information to be trained properly.

This training formulation assumes - to a certain degree - that it is possible to locate the majority of the semantic information in the decoder, while the geometric information is held in the encoder. The experimental results are discussed in section 7.5.2. For a more general investigation of using the target geometry, two different training variants are tested:

FINE-TUNED The encoder of a pre-trained source domain model is fine-tuned on the target geometry, as defined in eq. (7.3), while the decoder weights are frozen.

JOINT TRAINING The model is trained simultaneously on the source and target domain. All weights are updated with eq. (7.2) when a source sample is propagated, while only the encoder weights are updated according to eq. (7.3) when a target sample is passed through.

7.3.3 *Domain Losses for Domain Invariant Features*

As discussed in section 4.2.3, many of the domain-invariant feature learning methods use domain losses to align the feature of the domains. In this section a joint learning strategy is used in combination with a domain loss. The joint learning, as discussed in the previous section, is an implicit alignment of features by the introduction of the target geometry information. The domain loss instead is an explicit alignment of the latent feature grid by aligning second order statistics [Morerio et al. 2018; Wu et al. 2019].

In two separate experiments, **E**uclidean **C**orrelation **A**lignment (ECA) and **G**eodesic **C**orrelation **A**lignment (GCA) are applied to the latent feature grids $c_{\{1,2,3\}}$ [Morerio et al. 2018]. Samples from

each domain are fed to the model and the geodesic loss is then computed with the covariance matrices C^{source} and C^{target} of each latent feature grid. These are obtained by reducing the spatial dimension of the latent feature grids to a single axis. The covariance matrices are diagonalized via single-value decomposition to obtain the eigenvalues D^{source} and D^{target} and the respective eigenvectors U and V. The loss for the feature grid c_i is then defined as the geodesic log-Euclidean distance

$$\mathcal{L}_{\text{GCA},c_i} = \frac{1}{4d^2} \left\| U \log\left(D^{\text{source}}\right) U^T - V \log\left(D^{\text{target}}\right) V^T \right\|_F^2 \tag{7.4}$$

in terms of the (squared) Frobenius norm $\|\cdot\|_F$ with the number of dimensions $d = 3$ representing the x, y, z coordinates.

The combined training objective is then defined as

$$\begin{aligned} \mathcal{L} = {} & \lambda_S \sum_p \mathcal{L}_S^{\text{source}} + \lambda_G \sum_p \mathcal{L}_G^{\text{source}} + \lambda_C \sum_p \mathcal{L}_C^{\text{source}} \\ & + \lambda_G \sum_p \mathcal{L}_G^{\text{target}} + \lambda_C \sum_p \mathcal{L}_C^{\text{target}} \\ & + \lambda_{\text{GCA},c_1} \mathcal{L}_{\text{GCA},c_1} + \lambda_{\text{GCA},c_2} \mathcal{L}_{\text{GCA},c_2} + \lambda_{\text{GCA},c_3} \mathcal{L}_{\text{GCA},c_3} \end{aligned} \tag{7.5}$$

with the loss weights λ_{GCA,c_i} for the domain losses. The results are presented in section 7.5.3.

7.4 DATASET CURATION

This chapter uses the *SemanticKITTI* [Geiger et al. 2013; Behley et al. 2019] and the *nuScenes* [Caesar et al. 2020] datasets. Besides the different recording locations (Karlsruhe vs. Singapore/Boston), the main difference between the two datasets consists in the sensor setup. The *KITTI* dataset uses a HDL-64 with 64 vertical layers and a vertical FOV of 26.33° that spins at $10Hz$, which results in approximately 64×2000 points per 360° scan. The *nuScenes* dataset uses a Velodyne **VLP-32** (VLP-32) with 32 vertical layers and a vertical FOV of 41.34° that spins at $20Hz$, which results in approximately 32×1000 points per 360° scan. This means a single *nuScenes* point cloud only has one fourth of the points compared to a *KITTI* point cloud. Furthermore, the *nuScenes* dataset provides semantic labels only for key frames (every tenth frame), therefore only 10% of the listed frames posses semantic *Ground Truth* annotations.

Another major difference between the datasets are the annotation rules. Since this chapter focuses on closed set domain adaptation, a modified label mapping that matches both datasets is proposed. Table 7.1 lists the classes predicted in this work and their mapping to the original classes of the two datasets. Some classes are merged if a semantically meaningful mapping can be established. Very seldom

Table 7.1: **Label Mapping**: The proposed mapping of the classes from the *SemanticKITTI* and the *nuScenes* dataset. All remaining classes are treated as unlabeled, e.g. wheelchair, stroller.

SemanticKITTI		**mapped**		***nuScenes***
person, bicyclist, motorcyclist	»	person	«	adult, child, construction worker, police officer
road, lane-marking, parking	»	road	«	drivable surface
sidewalk	»	sidewalk	«	sidewalk
terrain	»	terrain	«	terrain
other-ground	»	other ground	«	other flat
car	»	car	«	car
bicycle	»	bicycle	«	bicycle
motorcycle	»	motorcycle	«	motorcycle
truck, on-rails, other vehicle, bus	»	other vehicle	«	truck, trailer, construction vehicle, bus
vegetation, trunk	»	vegetation	«	vegetation
building, fence, traffic-sign, pole	»	infrastructure	«	man-made, barrier, traffic-cone

or not well defined classes, such as wheelchairs, are removed from the mapping and are treated as unlabeled.

Both datasets alike require a few pre-processing steps to generate free-space voxels as training targets. To do so, the point clouds are accumulated over time and then voxelized. Dynamic objects are filtered out in this step to ensure clean accumulation. For generating the target for a specific frame, the dynamic objects of that particular frame are re-inserted into the accumulated scene. More information can be found in [Rist et al. 2021]. The handling of the *nuScenes* dataset requires additional pre-processing steps, to compensate for the partial annotations and missing dynamic flags. Details on the pre-processing of *nuScenes* are given in appendix A.4.

7.5 EXPERIMENTS

The following experiments compare the performances of different versions of the proposed method. The results show an increased

target performance compared to existing methods. First, section 7.5.1 presents the baseline performances. Section 7.5.2 and section 7.5.3 present the results of two versions of the proposed method and the results are summarized in section 7.5.4. Finally, section 7.5.5 provides comparisons to state of the art and shows the advantages of the proposed approach.

7.5.1 *Baseline and Domain Gap*

Table 7.2 shows the single-frame semantic segmentation performance of the backbone model. It achieves 54.4% mIoU for *KITTI* and 58.7% mIoU for *nuScenes* when trained and evaluated on the same domain. When the model is applied to the other domain at test time, the performance drops by 27.8 percent-points and 42.5 percent-points, respectively. These numbers can be interpreted as a measure for the domain gap. The domain gap of training on *KITTI* and testing on *nuScenes* (K $\rightarrow$ N) is significantly lower than the other way around (N $\rightarrow$ K). Reasons for this behavior could be the reduced amount of labeled *nuScenes* data, since only every tenth key frame is labeled, or the lower resolution of the *nuScenes* data caused by the sensor specifications.

Therefore, an additional experiment, shown in the two right columns of table 7.2, is conducted that also uses the unlabeled frames of *nuScenes* for the geometry target during training. It can be observed that the in-domain segmentation performance decreases from 58.7% to 47.4%, however, the performance on the target domain increases from 16.2% to 19.8%. The in-domain performance decrease for segmentation is caused by the higher relative weighting of the geometry loss over the complete training dataset. In order to prevent even stronger tendencies in this direction, the factor that weights the semantic and geometry losses is modified from 1.0 to 0.1 to address the effect caused by the ten-fold amount of the geometry information. However, the additional geometry information helps to understand semantic features of the target domain and can therefore be used as an essential component in a domain adaptation setting.

The class-wise results of table 7.2 show that the *nuScenes*-trained model performs poorly for the ground classes on the *KITTI* data. Usually, the ground classes are easy to learn due to its structure and the amount of available datapoints, therefore it has such a great impact on the domain gap here. One reason is the limited range of the ground measurements for *nuScenes*. For the class "car", both directions perform quite similar, while additional geometry information almost completely removes the domain gap for "car" in the N $\rightarrow$ K case. Classes with smaller and fewer objects, such as "person", "bicycle", and "motorcycle", already exhibit decreased

Table 7.2: **Semantic Segmentation Baseline**: Shown are the single-frame semantic segmentation results of the backbone model [Rist et al. 2021] with the proposed label mapping. Additionally, the domain gap is evaluated by naively testing the source-trained model on the target domain. "K" stands for *KITTI* and "N" stands for *nuScenes*.

Approach	Backbone		Backbone		Backbone*	
source→target	K→K	K→N	N→N	N→K	N→N	N→K
mean IoU [%]	54.4	26.6	58.7	16.2	47.4	19.8
size of domain gap	⌞ -27.8 ⌟		⌞ -42.5 ⌟		⌞ -27.6 ⌟	
person	52.1	1.8	44.1	4.0	10.5	5.0
car	91.6	53.3	74.1	48.5	70.0	71.1
bicycle	10.6	0.0	0.0	0.0	0.0	0.0
motorcycle	10.8	1.5	42.5	0.2	3.0	0.0
other-vehicle	12.8	10.0	57.8	3.2	22.2	2.5
infrastructure	86.8	45.2	76.0	45.5	68.2	50.0
road	92.9	77.6	94.2	3.6	90.3	3.0
sidewalk	80.6	28.9	64.0	0.2	61.8	0.4
other-ground	0.0	4.7	53.0	0.0	57.4	0.0
terrain	75.6	7.9	67.0	25.8	61.7	28.4
vegetation	84.3	31.3	72.9	47.1	59.0	57.4

*The unlabeled *nuScenes* data is used in addition to the annotated data to compute the geometry loss.

in-domain performance and therefore also suffer greatly under the domain switch.

One factor to take into account when using models across domains, is the training state of the model. Additional evaluations show that early stopping decreases the in-domain performance, but at the same time improves the inter-domain performance. Early-stopping is a strategy to not train a model until convergence. Specifically, the training is stopped as soon as it starts to converge, therefore the model is not fitted perfectly to the domain during the long tail of the training convergence. This means a simple early stopping strategy that prevents in-domain over-fitting can be used as a domain regularization strategy and is therefore a simple viable step towards domain generalization.

7.5.2 *Using Self-Supervised Target Geometry*

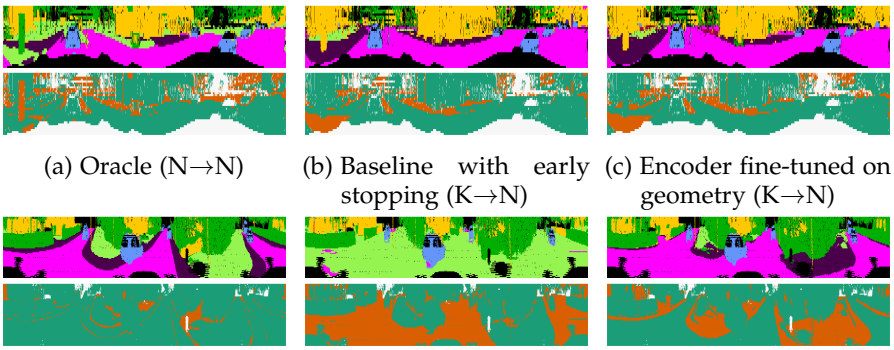
The approach presented in this section uses self-supervised target geometry in addition to the complete source information to train the model encoder. Two different training mechanisms are investigated: The first one uses as a pre-trained source domain model from section 7.5.1 and then fine-tunes the encoder on the target geometry, while the decoder weights are frozen. The second mechanism is employed as a joint training strategy, where one training step consists of two parts: firstly, the complete network is trained on source semantics and geometry and secondly the encoder is trained only on the target geometry. The detailed description of the setup can be found in section 7.3.2.

Table 7.3 shows the quantitative results of the experiments. Both the fine-tuning of the encoder and the joint training with self-supervised target geometry results in a significant performance increase for N $\rightarrow$ K compared to the baseline. The highest increase can be observed for the "road" class. However, for the opposite direction, i.e. K $\rightarrow$ N, the performance decreases compared to the baseline. This effect is stronger for the joint training than for the fine-tuning. This means that using self-supervised geometry is only beneficial if the geometry information has a higher resolution than the source geometry information. Since *KITTI* has a higher resolution than *nuScenes*, fine-tuning or joint training with *KITTI* is beneficial, while doing the same with *nuScenes* even impairs the performance. Overall, the fine-tuning achieves better results than the joint training. It has the additional advantage, that pre-trained source domain models can be used and don't need to be trained from scratch, which is needed for the joint training

Fig. 7.3 shows qualitative results of the oracle (in-domain performance), baseline, and the encoder fine-tuning experiments. The error image shows that the baseline fails to segment the road sur-

Table 7.3: **Influence of Target Geometry Information on Segmentation Performance**: Shown are the segmentation performances when the encoder of the model is trained with additional geometry information of the target domain. The column "Fine-Tuned" leverages fine-tuning, while "Joint Train." shows the results for the joint training of target geometry and source semantics. The oracle and the baseline are listed as reference to judge the domain gap and the improvement of the proposed method. "K" stands for *KITTI* and "N" stands for *nuScenes*.

Approach	Oracle	Baseline	Fine-Tuned	Joint Train.	Oracle	Baseline	Fine-Tuned	Joint Train.
	N	K→N			K	N→K		
mean IoU [%]	58.7	**32.2**	28.9	26.7	54.4	24.2	**29.2**	28.4
person	44.1	**5.1**	2.7	0.0	52.1	**1.3**	0.0	0.0
car	74.1	**67.2**	61.3	55.3	91.6	72.2	**82.4**	50.1
bicycle	0.0	0.0	0.0	0.0	10.6	0.0	0.0	0.0
motorcycle	42.5	0.5	**1.7**	0.0	10.8	0.0	0.0	0.0
other-vehicle	57.8	**36.9**	19.1	23.8	12.8	3.1	**3.9**	2.8
infrastructure	76.0	**64.7**	60.2	62.7	86.8	**71.9**	71.2	65.8
road	94.2	**80.7**	80.0	70.1	92.9	15.1	**56.5**	54.1
sidewalk	64.0	**33.2**	29.1	5.6	80.6	0.1	9.3	**25.6**
other-ground	53.0	3.7	**5.2**	0.0	0.0	0.0	0.0	0.0
terrain	67.0	4.8	9.2	**18.7**	75.6	31.5	20.3	**48.3**
vegetation	72.9	57.0	49.5	**57.4**	84.3	71.5	**77.1**	66.3



(a) Oracle (N→N) (b) Baseline with early stopping (K→N) (c) Encoder fine-tuned on geometry (K→N)

(d) Oracle (K→K) (e) Baseline with early stopping (N→K) (f) Encoder fine-tuned on geometry (N→K)

Figure 7.3: **Qualitative Segmentation Results**: Shown are the semantic segmentation results (top) and the prediction errors (bottom). Green means, the correct class was predicted and red means, the wrong class was predicted for that measurement. White and black are missing or unlabeled positions. The colors of the semantic classes are listed in the result tables throughout the chapter.

face correctly in the N → K case, while performing reasonably for K → N. When fine-tuning on the target geometry, the segmentation of the road surface significantly improves for N → K, but still shows more incorrectly classified regions compared to K → N and the oracle. However, the segmentation quality on geometrically prominent scene content is now classified in a semantically meaningful way by correctly separating geometric edges, such as curb stones. For K → N, no significant performance change can be observed visually.

7.5.3 *Domain Losses for Domain Invariant Features*

This section investigates the effect of using additional domain losses to align the second order statistics of the latent features of both source and target domain. Detailed information on the setup can be found in section 7.3.3.

Table 7.4 gives an overview on the domain loss results. Overall, it can be observed that using additional domain losses does not improve the performance compared to using the target geometry information. On the contrary, introducing additional domain losses even impairs the segmentation performance on the target domain. This means that the pure geometric information is better suited for the model to learn a domain invariant feature representation for semantic segmentation compared to GCA and ECA.

Table 7.4: **Influence of Domain Losses on Segmentation Performance**: Shown are the segmentation performances for the model with different combinations of the source and target geometry losses in combination with **G**eodesic **C**orrelation **A**lignment (GCA) and **E**uclidean **C**orrelation **A**lignment (ECA). The oracle and the baseline are listed as reference to judge the domain gap and the improvement of the proposed method. "K" stands for *KITTI* and "N" stands for *nuScenes*.

Domain	N	K→N		K	N→K				
Early Stopping		✓	✓		✓	✓	✓	✓	✓
Target geometry			✓				✓	✓	✓
GCA			✓			✓		✓	
ECA									✓
mean IoU [%]	58.7	32.2	32.2	54.4	24.2	26.0	**33.1**	28.5	30.2
person	44.1	**5.1**	0.4	52.1	1.3	0.0	0.1	0.5	**8.7**
car	74.1	**67.2**	61.7	91.6	72.2	56.7	65.8	51.8	**77.4**
bicycle	0.0	0.0	0.0	10.6	0.0	0.0	0.0	0.0	0.0
motorcycle	42.5	0.5	**2.6**	10.8	0.0	0.0	0.0	0.0	0.0
other-vehicle	57.8	**36.9**	15.6	12.8	3.1	0.0	**42.9**	0.1	0.0
infrastructure	76.0	64.7	**65.1**	86.8	**71.9**	37.3	61.8	58.3	43.3
road	94.2	80.7	**83.8**	92.9	15.1	50.6	**77.8**	65.3	55.5
sidewalk	64.0	33.2	**36.2**	80.6	0.1	8.0	**20.6**	7.7	2.1
other-ground	53.0	**3.7**	0.0	0.0	0.0	0.0	**2.1**	0.0	0.0
terrain	67.0	4.8	**26.7**	75.6	31.5	32.9	32.0	**57.4**	38.0
vegetation	72.9	57.0	**62.2**	84.3	71.5	74.0	61.2	72.6	**76.4**

7.5.4 *Summary*

The experiments show, that early stopping is a simple yet effective method for regularizing cross-domain segmentation. Leveraging additional geometric information from the target domain is especially useful when the target domain has a higher resolution than the source domain. Such, additional information and more details on the geometry of the target domain can be obtained. If the source domain has a higher resolution than the target domain, no significant improvement can be observed if the lower resolution target geometry information is used for training. However, adapting from low to high resolution is harder to solve, as points cannot simply be dropped during inference to obtain the same resolution, and is therefore often not addressed by related work. The proposed method has a big advantage in this field. Comparisons to related work are provided in the next section.

7.5.5 *Comparison against State of the Art*

This section compares the proposed method against works proposed by Langer et al. [2020] and Yi et al. [2021], as introduced in section 7.1. Both works use a different label mapping than the one proposed in table 7.1. In order to make the results comparable, additional experiments with the modified label mapping are conducted. Therefore, numbers are not comparable to the ones presented in the sections above or between the two methods. The label mappings for both experiments are listed in appendix A.4.

7.5.5.1 *Comparison against Langer et al. [2020]*

The domain adaptation method proposed by Langer et al. [2020] accumulates LiDAR frames from the source domain and then subsamples new single frame point clouds with the scan pattern of the target domain. Afterwards, they use this data to train their segmentation model in conjunction with a GCA to align the second order statistics of the features. Because of this sub-sampling approach in combination with how the remission information is generated, Langer et al. [2020] state that their method is only applicable to high-to-low resolution domain adaptation, therefore this section only reports results for *KITTI* $\rightarrow$ *nuScenes* adaptation. The validation set used by Langer et al. [2020] significantly deviates from the one used in the previous experiments. The authors re-labeled an entire *nuScenes* sequence with the same label mapping as used in the *SemanticKITTI* dataset and dropped all unknown classes. This reduces the domain gap introduced by the label mapping and forms a closed set domain adaptation problem.

Table 7.5: **Segmentation Comparison to Langer et al. [2020]**: Shown are the results for semantic segmentation on the *nuScenes* dataset in a *KITTI* to *nuScenes* domain adaptation setting. The left columns show three versions of [Langer et al. 2020], while the right columns show three versions of the proposed approach.

	[Langer et al. 2020]			Proposed Approach		
	Baseline	MB	CP + GCA	Early Stopping	Target Geometry	Geometry + GCA
mean IoU [%]	12.3	30.0	35.9	**36.4**	30.7	34.7
person	0.1	14.1	**18.6**	0.0	0.0	0.0
car	26.4	56.7	70.8	**79.6**	68.3	73.5
bicycle	0.0	0.1	**8.8**	0.0	0.0	0.0
other-vehicle	2.6	**10.6**	6.4	0.0	0.0	0.0
building	42.2	77.2	80.9	88.7	77.7	**89.9**
fence	2.8	35.8	43.0	24.0	34.0	**45.4**
pole	8.0	33.7	35.3	**50.0**	35.6	48.2
traffic-sign	4.1	11.6	16.3	**23.8**	6.3	13.6
trunk	0.5	2.7	3.4	**7.3**	0.0	0.2
road	68.1	85.8	88.5	**89.9**	87.5	88.5
sidewalk	0.1	26.4	53.0	**55.9**	47.6	47.8
terrain	0.1	2.1	0.8	**8.8**	6.1	4.4
vegetation	4.9	32.8	41.6	**45.0**	36.6	40.0

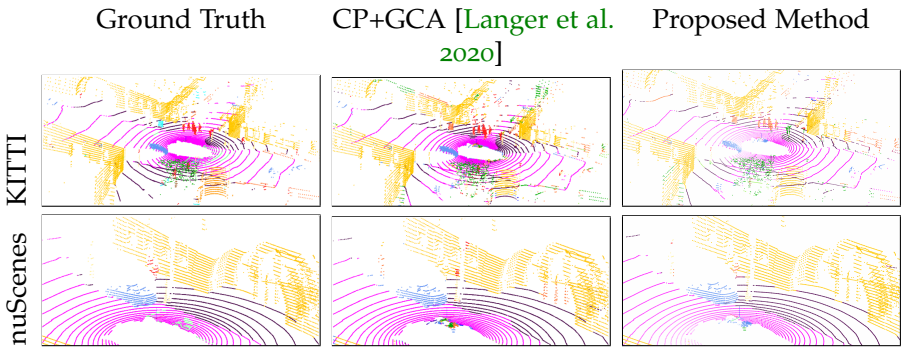


Figure 7.4: **Segmentation Comparison to Langer et al. [2020]**: Shown are the qualitative semantic segmentation results on *KITTI* (top) and *nuScenes* (bottom). The left column shows the *Ground Truth* annotations, the middle column the results with the domain adaptation method by Langer et al. [2020], and the right column shows the results of the proposed method with geometry information from the target domain.

Table 7.5 shows that the variants of the proposed approach performs on par with the variants of Langer et al. [2020]. Their approach performs significantly better on geometrically inconsistent classes, such as bicycles or persons. The proposed approach, on the other hand, achieves significantly better results for geometrically distinctive classes, such as poles and traffic-signs.

Fig. 7.4 shows qualitative results on both *nuScenes* and *KITTI*. The proposed method and the method by Langer et al. [2020] show different failure cases in the segmentation. Langer et al. [2020] method mostly fails for single points and therefore appears noisy or random. When the proposed approach fails, it usually fails to estimate the right semantic class for the entire object while producing sharp and correct object or instance boundaries. This is caused by the network learning geometric segmentation in addition to the semantics, which is a key attribute of the backbone model [Rist et al. 2021].

7.5.5.2 *Comparison against Yi et al. [2021]*

Complete & Label by Yi et al. [2021] creates a canonical domain that is generated by a voxel completion network. A semantic segmentation network is then applied to the target domain voxel samples generated by the surface completion. As a backbone, they use MinkowskiNet [Choy et al. 2019]. Table 7.6 shows the performance of the backbone model MinkowskiNet compared to the performance of the backbone model of the proposed approach [Rist et al. 2021]. For both datasets, MinkowskiNet performs significantly better. The reasons is, that [Rist et al. 2021] is a model for semantic scene completion, solving two tasks at once, while [Choy et al. 2019] can completely focus on the semantic segmentation problem.

Table 7.6: **Segmentation Comparison to Yi et al. [2021] and other Baselines**: Shown is the mean semantic segmentation IoU. Considered are all the 10 overlapping categories between the *KITTI* (K) and the *nuScenes* (N) dataset: car, bicycle, motorcycle, truck, other vehicle, pedestrian, drivable surface, sidewalk, terrain, and vegetation. Best results are shown in **bold**, second best are underlined.

Base Model	Description	K	N
MinkowskiNet	*baseline by Yi et al. [2021]*	50.2	54.4
SCSSNet	*backbone by Rist et al. [2021]*	41.9	45.3
Base Model	**DA Approach**	**K→N**	**N→K**
MinkowskiNet	FeaDA [Chen et al. 2017b]	27.2	21.4
MinkowskiNet	OutDA [Tsai et al. 2018]	26.5	22.7
MinkowskiNet	SWD [Lee et al. 2019a]	27.7	24.5
MinkowskiNet	SQSGV2 [Wu et al. 2019]	10.1	13.4
MinkowskiNet	3DGCA [Yi et al. 2021]	27.4	23.9
MinkowskiNet	Complete & Label [Yi et al. 2021]	**31.6**	**33.7**
SCSSNet	Early Stopping (Baseline)	30.0	14.1
SCSSNet	Target Geometry + GCA	30.8	27.1

Nevertheless, table 7.6 shows that the proposed DA method still achieves comparable results on the target domain, though the backbone network itself seems to be weaker. For K → N, the proposed method performs better than all previous approaches and performs only slightly worse than [Yi et al. 2021]. For N → K, it is necessary to use target geometry and GCA to achieve reasonable performance, while for K → N even the early stopping is enough to achieve comparable performance. This shows that the method holds great potential and improving the backbone model itself may already help the domain adaptation capabilities. Further, it has to be noted, that the proposed method does not introduce any overhead during inference and runs at real-time compared to the computationally heavy two-step approach of Yi et al. [2021].

7.6 DISCUSSION

The experiments show that the domain adaptation from low-to-high resolution is more challenging than the other way around. This might be the reason, why most related work, except for [Yi et al. 2021], only deal with the high-to-low resolution case. The most significant failure case of the proposed method in the low-to-high resolution case, seems to be the segmentation of the ground plane. Other objects, such as cars show high quality semantics.

Another interesting point is to discuss the assumption made at the beginning of this chapter. The proposed method is built around the assumption that geometric information, which can be obtained in a self-supervised manner, can help to learn semantic information. The qualitative results showed that the semantic and scene geometry are not closely related but share common features that benefit the other task. In contrast to other methods, the proposed approach suffered less from random segmentation errors, but usually segmented the scene perfectly on the object outlines, while sometimes deciding for the wrong semantic class.

Future work might involve further improvements on using additional domain losses. In the presented experiments, the domain losses did not improve the performance significantly. However, one method to improve the feature alignment is a different sampling strategy in training. It could be beneficial to sample exactly the same spatial points across domains to create domain query points which are then used for the domain losses. Further improvements might involve a transformer-based decoder architecture in the backbone to leverage attention mechanisms for improved segmentation.

7.7 CONCLUSION

This chapter proposed a novel method of unsupervised domain adaptation for LiDAR semantic segmentation. The proposed approach uses the underlying scene geometry of both the source and the target domain to transfer semantic cues from the source to the target domain. A variety of experiments show that the joint learning of source and target scene geometry creates a domain-invariant feature representation. In most cases, feature extraction with local functions over the parameterization grids have shown to be more effective in learning domain-invariant features than explicit feature alignment. Using the learned representation greatly decreases the domain gap in both high-to-low and low-to-high resolution cases. While the method is more effective for high-to-low adaptation, the approach greatly profits from the higher resolution target geometry in the low-to-high case.

Comparison to state of the art shows, that the proposed approach achieves comparable performance to Complete & Label by Yi et al. [2021], without introducing any additional overhead in the inference. This is achieved by encoding implicit geometry in feature space instead of constructing an explicit canonical domain. Further, the proposed approach outperforms the work by Langer et al. [2020] in terms of semantic segmentation performance measured in mean IoU. In contrast to that work, the proposed approach cannot only cover the high-to-low resolution case, but also the more difficult low-to-high resolution adaption. These features make the proposed

approach an important asset for domain adaptation applications that need to handle evolving sensor specifications and real-time constraints.

8

POINT CLOUD GENERATION WITH CONTINUOUS CONDITIONING

CONTENTS

A general aim when developing perception algorithms is to improve their performance. In practice it is more desirable to have good and robust performance on certain important objects instead of the entire scene. These usually include vulnerable road users, such as pedestrians or cyclists, other vehicles, and objects that are located on the road. Unfortunately, measurements on these objects are often the least represented in the data. LiDAR points that belong to cars only make up 4.08% of the *KITTI* dataset compared to 19.87% road points (sidewalk and other surfaces not included), while pedestrians are only present in 0.01% of the measured points. This causes most

of the known perception models to perform worse on the lesser represented classes, which can also be observed throughout the experiments in this dissertation.

One well known method to counteract this issue in the field of LiDAR perception is to use object augmentation [Yan et al. 2018; Lang et al. 2019; Baur et al. 2019]. Specifically, additional objects are placed into the scene at training time to compensate for the class imbalance. Additionally, specific underrepresented scenarios can be created with object augmentation, for example critical driving scenarios. This can greatly improve the performance for the underrepresented classes and scenarios.

Typical augmentation approaches select existing objects from the database and re-insert them into the scenes. This procedure limits the selection to the size and diversity of the provided object database. However, it is important that these inserted objects are of high quality and diversity and therefore it is favorable to have an unlimited number of objects at ones disposal. Generative models are already used to synthesize numerous 3D objects of high quality and diversity [Shu et al. 2019]. However, there is typically no control over properties of generated objects and therefore it is not possible to parameterize scene compositions.

This chapter proposes a novel GAN setup that generates 3D point cloud shapes conditioned on a continuous parameter. This parameter guides the generative process to create a custom-fit object with a desired 3D shape. Having control over the properties of the object enables customizable scene configurations. At the time of writing, no other work used continuous interpretable descriptions to generate 3D objects. This chapter is based on [Triess et al. 2022a] and contains verbatim quotes of that work[1].

8.1 OVERVIEW

Generative models, such as GANs [Goodfellow et al. 2014] or VAEs [Kingma and Welling 2014], are often used to generate completely new samples with high quality and diversity. These approaches have been initially introduced for image generation, but lately a number of approaches for 3D generation in general [Sun et al. 2020; Yang et al. 2019; Luo and Hu 2021] and point cloud generation [Achlioptas et al. 2018; Valsesia et al. 2019; Shu et al. 2019] in particular have emerged. However, none of these methods is capable of actively influencing specific properties of the generated

1 The work [Triess et al. 2022a] partially evolved from the Master's thesis of Andre Bühler [Bühler 2021], which was supervised by Larissa Triess. The problem formulation, initial concept, overall GAN setup design, evaluation concept, and application are contributions by Larissa Triess. The choice of the backbone model, the implementation details, and the metric selection are contributions by Andre Bühler.

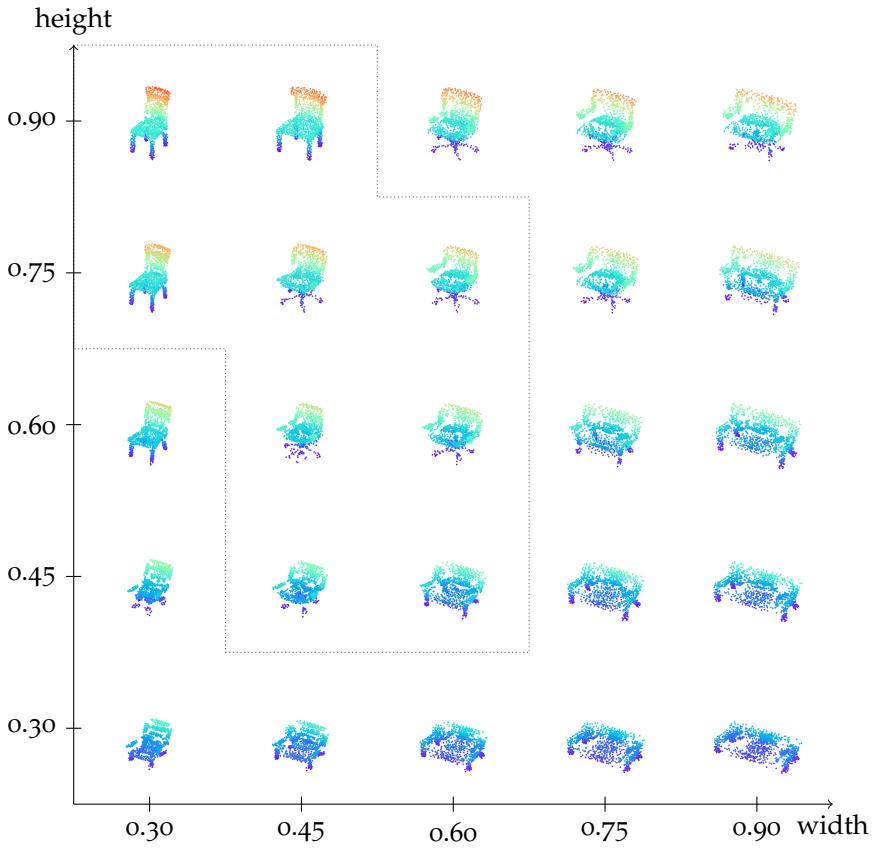


Figure 8.1: **Generated Objects conditioned on different Dimensions**: The proposed method can generate a diverse set of object shapes conditioned on object dimensions (values on the axes). The figure shows objects generated from the same latent vector $\mathbf{z}$, but with different continuous conditioning parameters $\mathbf{y}$. The generated objects are realistic and semantically meaningful. Our method generalizes to out-of-distribution dimensions (outside of dotted shape). This figure is based on [Triess et al. 2022a].

object, such as the height and width (see fig. 8.1). For point clouds it would be easy to generate shapes with the aforementioned methods and then simply re-scale their dimensions to obtain parameterizable object generation. However, there is a correlation between the shape of an object and its dimensions. For example, simply scaling down an SUV looks unrealistic since vehicles with smaller dimensions are usually small passenger cars. Therefore, the aim is to design a conditioned generative process that creates custom-fit shapes with a correct semantic meaning of the object.

This chapter proposes a method to use descriptions in form of continuous conditional parameters within a GAN to generate objects with desired properties. These properties can, for example, describe the dimensions of the object, such that the GAN generates a custom-fit shape while maintaining generation quality and diversity. Using continuous conditions requires new training setups as most of the

class-conditioning solutions are not transferable to the domain of continuous labels [Ding et al. 2021]. The challenges arise from an infinitely large set of parameters and conditioning regions where no training data samples exist.

The contributions of this chapter are as follows:

- The proposed method is first to use continuous parameters for point cloud generation. The continuous conditioning is formulated as a multi-task problem by adapting the concept of auxiliary classifier GANs [Odena et al. 2017].
- A **K**ernel **D**ensity **E**stimation (KDE) is used to sample the generator conditioning input from the parameter distribution for training. A specifically designed experimental setup shows that this sampling strategy leads to a significant performance increase in regions with few samples and improves the generation quality.
- Qualitative evaluations show that explicit control over the object dimensions is gained while maintaining generation quality and diversity.

8.2 RELATED WORK

Generative models and condition methods are the key components of this chapter. The following sections give an overview on related works in this area.

8.2.1 *3D Generative Models*

In recent years, a variety of methods emerged to synthesize realistic data. These generative methods are based on two major concepts: VAEs [Kingma and Welling 2014] and GANs [Goodfellow et al. 2014]. Initially proposed to generate realistic 2D images, these concepts have widely been adapted to fulfill a variety of tasks [Song et al. 2018; Chen et al. 2019a; Karras et al. 2019] on different modalities. Wu et al. [2016] were first to propose an unsupervised deep generative approach for 3D data generation from probabilistic input. Their voxel-based GAN allows to directly adapt concepts from image-based setups, but is limited in resolution due to computational inefficiency. Therefore, subsequent works focus on surface [Mescheder et al. 2019; Chen and Zhang 2019; Michalkiewicz et al. 2019; Park et al. 2019a] or point cloud [Qi et al. 2017a,b; Fan et al. 2017] representations instead. Achlioptas et al. [2018] propose architectures for both VAEs and GANs to generate point cloud objects. The use of relative simple models based on fully connected layers combined with PointNet layers limits the ability to produce more realistic objects. The method in [Valsesia et al. 2019] exploits local topology

by using a computationally heavy *k*-nearest neighbor technique to produce geometrically accurate point clouds. TreeGAN [Shu et al. 2019] aims to improve the expressiveness of the generative models by introducing convolution-like layers and up-sampling techniques to the generator. Wang et al. [2020] deal with the choice of suitable discriminator architectures for 3D generation and show that models perform better when focusing on overall object shapes as well as sampling quality.

8.2.2 *Conditional Generation*

In many cases additional conditioning parameters are desired to generate specific object categories or styles. The most prominent example, conditional GAN [Mirza and Osindero 2014], uses explicit conditioning and forms the basis of many other approaches [Zhu et al. 2017; Taigman et al. 2017; Hoffman et al. 2018]. AcGAN [Odena et al. 2017] refines this concept of class conditioning by using an additional auxiliary classifier in the discriminator to ensure class specific content. In order to condition the output on arbitrary combinations of discrete attributes, some works use annotated images as input to the GAN [He et al. 2019; Perarnau et al. 2016]. StyleGAN and others investigate how to enhance desirable properties in GAN latent spaces to influence the characteristics of generated images selectively [Karras et al. 2019, 2020; Härkönen et al. 2020].

8.2.3 *Continuous Conditioning*

Many of the aforementioned concepts use discrete label conditioning. However, attributes like rotation in angles or age in years are by definition continuous. Using continuous conditions is a mathematically different problem than solving categorical conditioning problems, such as classification [Ding et al. 2021]. First, there may be few or no real samples for some regression labels and second, conventional label input methods, i.e. one-hot encoding, are not possible for an infinite number of regression labels. The **C**ontinuous **c**onditional **G**enerative **A**dversarial **N**etwork (CcGAN) [Ding et al. 2021] is first to solve the aforementioned problems by introducing a new GAN loss and a novel way to input the labels based on label projection [Miyato and Koyama 2018]. Shoshan et al. [2021] propose to use attribute specific pre-trained classifiers to enhance desired properties on the generative behavior. A subsequent training of mapping networks allows to generate noise vectors which produce explicit continuous attributes. Their method produces convincing results but requires an extensive amount of labeling and well pre-trained classifiers for each attribute.

8.2.4 *3D Conditional Generation*

The aforementioned conditioning strategies have all been proposed for image synthesis. There are some works that use text-based conditioning to generate 3D scenes [Chang et al. 2015a, 2014; Chen et al. 2018] with focus on database composition. Other approaches use symbolic part-tree descriptions to generate 3D objects with predefined compositions [Mo et al. 2020] or use occupancy networks for image based generation and coloring of 3D objects [Mescheder et al. 2019]. Although being related, our method focuses on conditioning point cloud generation using continuous physical parameters.

8.3 USING TREEGAN AS THE BACKBONE MODEL

This section gives a short introduction to the backbone network. The proposed method builds upon TreeGAN [Shu et al. 2019], a state-of-the-art GAN architecture that can generate point cloud objects with high quality and diversity. The generator consists of stacked tree graph convolution layers (TreeGCN). It receives a random noise vector $\mathbf{z} \in \mathbb{R}^{96}$ as input and outputs a point cloud $\mathbf{x}_{\text{gen}} = G(\mathbf{z}) \in \mathbb{R}^{2048\times 3}$. The loss setup follows that of a Wasserstein-GAN [Gulrajani et al. 2017]. Thus, the loss function of the generator G is defined as

$$\mathcal{L}_{G,\text{adv}} = -\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}\left[D\left(\mathbf{x}_{\text{gen}}\right)\right], \tag{8.1}$$

where D denotes the discriminator. The latent code distribution $\mathcal{Z}$ is sampled from a Normal distribution $\mathbf{z} \in \mathcal{N}(\mathbf{0}, I)$.

The discriminator follows a PointNet architecture [Qi et al. 2017a]. It either receives a real ($\mathbf{x}_{\text{real}}$) or a generated ($\mathbf{x}_{\text{gen}}$) point cloud $\mathbf{x} \in \mathbb{R}^{2048\times 3}$ as input and outputs a single scalar $D(\mathbf{x})$. The output estimates whether the sample originates from the distribution of the real or generated samples. The loss function of the discriminator D is defined as

$$\begin{aligned}\mathcal{L}_{D,\text{adv}} =& \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}\left[D\left(\mathbf{x}_{\text{gen}}\right)\right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{R}}\left[D(\mathbf{x}_{\text{real}})\right] \\ &+ \lambda_{\text{gp}} \cdot \mathbb{E}_{\hat{\mathbf{x}}}\left[\left(\|\Delta_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1\right)^2\right],\end{aligned} \tag{8.2}$$

where $\hat{\mathbf{x}}$ is sampled from line segments between real and fake point clouds and λ_{gp} is the weighting parameter for the gradient penalty term [Gulrajani et al. 2017].

8.4 METHOD

Fig. 8.2 presents the GAN setup of the proposed continuous conditioning architecture. The concept of the continuous parameters is introduced in section 8.4.1 while section 8.4.2 explains the required

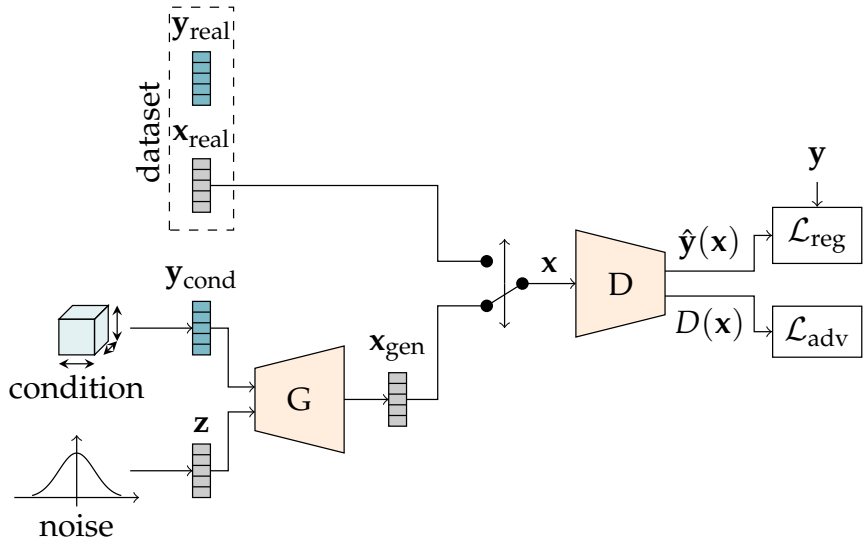


Figure 8.2: **Architecture**: The generator G generates a point cloud $\mathbf{x}_{\text{gen}}$ from a random vector $\mathbf{z}$ and a continuous parameter $\mathbf{y}_{\text{cond}}$. In alternation with a real point cloud $\mathbf{x}_{\text{real}}$, $\mathbf{x}_{\text{gen}}$ is fed to the discriminator, which predicts the probability of the sample originating from the real distribution $D(\mathbf{x})$ and an estimate of the continuous parameter $\hat{\mathbf{y}}(\mathbf{x})$. With these two outputs, the adversarial $\mathcal{L}_{\text{adv}}$ and regression $\mathcal{L}_{\text{reg}}$ losses are computed. This figure is based on [Triess et al. 2022a].

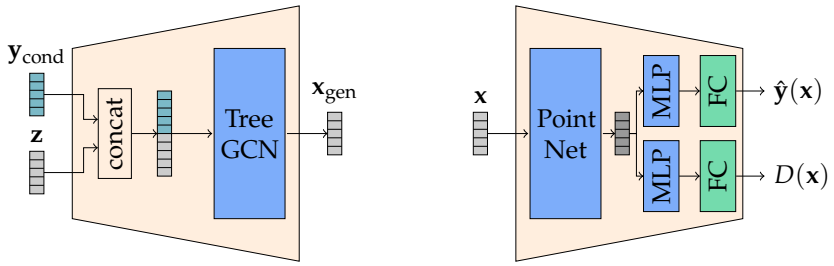
sampling strategy. Section 8.4.3 describes the model architecture by introducing the changes made to the backbone network to incorporate the continuous conditioning. The adapted training losses are introduced in section 8.4.4.

8.4.1 *Continuous Parameters*

With the help of the continuous parameters, the generation process is extended from a purely stochastic to a guided process. The aim is to control the outer dimensions of the generated object while maintaining diversity regarding the type and shape of the object. The object is defined as a vector of points $\mathbf{x} \in \mathbb{R}^{N\times 3}$ with $N = 2048$ number of points and the dimensions $[x, y, z]$. The parameter $\mathbf{y} \in \mathbb{R}^3$ defines the extent of the object in each dimension $\mathbf{y} = (\Delta x, \Delta y, \Delta z) \in [0, 1]^3$. Therefore, the parameters $\mathbf{y}_{\text{real}}$ for the training data $\mathbf{x}_{\text{real}}$ can easily be computed from the data itself with $\mathbf{y}_{\text{real}} = \|\max(\mathbf{x}_{\text{real}}) - \min(\mathbf{x}_{\text{real}})\|$.

8.4.2 *Label Sampling for Training*

At training time, parameters $\mathbf{y}_{\text{cond}}$ have to be sampled as a second input to the generator G. The easiest method is to sample randomly in $[0, 1]$. However, this can lead to a description of unsuitable object



(a) Generator (b) Discriminator

Figure 8.3: **Model Details**: (a) shows the label input mechanism of the generator. A simple concatenation is used and the result is then fed to the tree graph convolution network (TreeGCN). The discriminator model is depicted in (b). After a common feature extractor (PointNet), the model splits in two identical parts, where $D(\mathbf{x})$ is the adversarial feedback and $\hat{\mathbf{y}}(\mathbf{x})$ is the prediction of the continuous object description. This figure is based on [Triess et al. 2022a].

dimensions. To circumvent this issue, it is possible to sample the dimensions from the training dataset. This ensures that only actually possible conditioning parameters are sampled. However, this limits the generator training to a fixed number of input conditions. Therefore, this work proposes to sample the dimensions from the distribution of the dimensions within the training dataset. A KDE is computed over the whole dataset prior to training. This again assures that only suitable dimensions are drawn, but it does not limit them to be present in the dataset. In all cases, the aim of the generator is to generate point clouds that have dimensions which are close to the conditioned dimensions $\mathbf{y}_{\text{cond}}$.

8.4.3 *Model*

This section only describes the changes made to the TreeGAN backbone architecture (section 8.3). Fig. 8.3 shows how the label conditioning is incorporated into the generator and discriminator models. For the generator (fig. 8.3a), the architecture is extended to receive an additional input, the conditioning vector $\mathbf{y}_{\text{cond}}$. This vector is simply concatenated to the noise vector $\mathbf{z}$, extending the input to $\mathbb{R}^{96+3}$. The result is then fed to the otherwise unmodified TreeGCN [Shu et al. 2019] which outputs the point cloud $\mathbf{x}_{\text{gen}} = G(\mathbf{y}_{\text{cond}}, \mathbf{z})$. The discriminator input $\mathbf{x}$ is either a point cloud from the dataset $\mathbf{x}_{\text{real}}$ or a generated point cloud from the generator $\mathbf{x}_{\text{gen}}$ (fig. 8.3b). The network has two outputs, one for the standard adversarial feedback $D(\mathbf{x})$ and one to estimate the continuous parameter conditioning of the presented data $\hat{\mathbf{y}} = \hat{\mathbf{y}}(\mathbf{x})$. This concept is adapted from the **A**uxiliary **c**lassifier **G**enerative **A**dversarial **N**etwork (AcGAN) [Odena et al. 2017; Atienza 2019]. The idea is to leverage potential synergies be-

tween the two tasks within the shared discriminator layers. Previous work showed that such multi-task synergies can improve the performance of the sub tasks [Standley et al. 2020].

8.4.4 *Losses*

The training objective is composed of two parts, the adversarial loss $\mathcal{L}_{\text{adv}}$ and the regression loss $\mathcal{L}_{\text{reg}}$ for the continuous parameter. Just like TreeGAN, the proposed model utilizes the Wasserstein objective function with gradient penalty for the adversarial loss [Arjovsky et al. 2017; Gulrajani et al. 2017]. For the parameter regression, an L2-norm is used

$$\mathcal{L}_{\text{reg}}\left(\mathbf{y}, \hat{\mathbf{y}}\right) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2 \tag{8.3}$$

with the predicted label $\hat{\mathbf{y}}$ and its target label $\mathbf{y}$.

The adversarial part of the generator loss $\mathcal{L}_{G,\text{adv}}$ is defined in eq. (8.1) with a slight change to the definition of the generated point cloud $\mathbf{x}_{\text{gen}}$. It is now also dependent on the input conditioning and is therefore defined as $G(\mathbf{y}_{\text{cond}}, \mathbf{z})$ (instead of the unconditioned version $G(\mathbf{z})$). For the generator regression, the $\mathcal{L}_{\text{reg}}$ is computed between the discriminator prediction for the condition parameter of the generated sample $\hat{\mathbf{y}}_{\text{gen}}$ and the actually requested parameter $\mathbf{y}_{\text{cond}}$. This results in the overall generator loss

$$\mathcal{L}_G = \lambda_{\text{adv}} \cdot \mathcal{L}_{G,\text{adv}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{cond}}, \hat{\mathbf{y}}_{\text{gen}}\right) \tag{8.4}$$

with the loss weighting factors λ_{adv} and λ_{reg}.

Eq. (8.2) defines the adversarial loss of the discriminator. The regression loss is defined as the error between the parameter of the real data $\mathbf{y}_{\text{real}}$ and its corresponding prediction $\hat{\mathbf{y}}_{\text{real}}$. Analogously to the generator, this leads to the discriminator loss

$$\mathcal{L}_D = \lambda_{\text{adv}} \cdot \mathcal{L}_{D,\text{adv}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{real}}, \hat{\mathbf{y}}_{\text{real}}\right) \tag{8.5}$$

with the loss weights λ_{adv} and λ_{reg} as in eq. (8.4). The parameters of the generated samples are not considered in the regression loss, because for many use cases the actual parameter $\mathbf{y}_{\text{gen}}$ of the generated point cloud $\mathbf{x}_{\text{gen}}$ is unknown, i.e. cannot be trivially retrieved from $\mathbf{x}_{\text{gen}}$. Details and additional loss variations can be found in appendix A.5.2.

8.5 EXPERIMENTS

This section gives an overview on the experimental setups and resources. Further details are provided in appendix A.5. The results of the experiments are presented in section 8.6.

8.5.1 *Dataset and Metrics*

The ShapeNetPart [Yi et al. 2016] dataset is used for the experiments. It is a dataset with part annotations of more than 30,000 3D shapes in 16 object categories from ShapeNetCore [Chang et al. 2015b]. To compare the results in terms of generation quality, a pre-trained version of the original FPD [Shu et al. 2019] is used. The adherence to the conditioning properties is evaluated by calculating the MSE for each dimension extent (Δx, Δy, Δz) of the generated object $\mathbf{x}_{\text{gen}}$ versus the desired input parametrization $\mathbf{y}_{\text{cond}}$. These are the two most important metrics, but we also report **Cov**erage (COV), MMD, and ***J**ensen-**S**hannon **D**ivergence* (JSD) [Achlioptas et al. 2018] to make our work comparable to existing methods.

JSD is a coarse metric that measures the degree to which axis aligned real point clouds tend to occupy similar locations as those of the generated point clouds. The diversity of the samples is measured with COV which estimates how well the generator can cover the real data distribution. Orthogonally, MMD measures how similar the generated samples are to the real ones. This means that a generator that simply repeats samples from the dataset receives a better score than a generator that synthesizes more diverse objects. Therefore, the main focus of this evaluation is on the qualitative results which show the advantages of the proposed method.

8.5.2 *Implementation Details*

The model is built upon the existing implementation of Tree-GAN [Shu et al. 2019] that is further referred to as the backbone model. The training parameters are also identical. Only the changes introduced in section 8.4 are applied to keep the method directly comparable to the backbone and no further mechanisms to enhance generation quality are incorporated.

The loss weights λ_{adv} and λ_{reg} from eq. (8.4) and eq. (8.5) are variable and learned together with the rest of the model parameters, as proposed by Kendall et al. [2018].

All networks are trained for 3000 epochs and then the checkpoint with the lowest combined metric score is selected. The combined metric is defined as the product of FPD and MSE. This ensures fidelity as well as correctness and considers differing value ranges.

8.5.3 *Baselines*

The proposed method is compared to two baselines. These baselines represent possible design choices to achieve the desired aim of generating point clouds of specific dimensions, as no other methods for comparison exists.

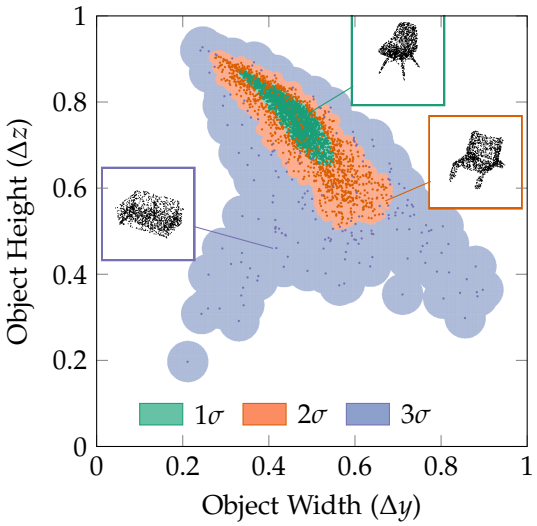


Figure 8.4: **Region-classified Dimension Distribution**: The plot shows all samples (marks) of the chair class for the entire training dataset in terms of their dimension extent in height Δz and width Δy. The three colors represent the resulting regions from a KNN classifier with k=20 based on a KDE. The regions correspond to $1\sigma \approx 68\%$ (green), $2\sigma \approx 27\%$ (orange), and $3\sigma = 5\%$ (blue) of the entire data distribution. This figure is based on [Triess et al. 2022a].

B1 – BACKBONE WITH RE-SAMPLING A simple method to generate shapes of desired dimensions with an existing model, such as the backbone, is to generate point clouds from multiple sampled z vectors and then choose the one that has the dimensions that are closest to the ones requested. The advantage is that it is expected that the fidelity of the samples with this variation is similar to the one of the backbone network itself. However, the sampling process is time consuming and the dimensions might not fit exactly, especially if these dimensions are not well reflected in the training dataset. For the experiments, 10 versions per object are sampled and then the one with the smallest MSE is selected.

B2 – BACKBONE WITH SCALING Another variant is to add a subsequent scaling block that squashes the generated point cloud into the desired dimensions. In contrast to B1, it is expected that the regression error on the dimensions is zero. However, the generation fidelity can be severely impacted. For example, objects are no longer semantically meaningful if their shapes are stretched along an axis, such that the shapes no longer represent objects as they could appear in the real world.

8.5.4 *Distribution Sampling*

A key aspect of the proposed method is that it is possible to actively sample from different regions of the conditioning vector distribution. Therefore, the evaluation section specifically investigates the generation capabilities within different regions of the data distribution. Fig. 8.4 shows the distribution of the object dimensions of the training dataset for the chair class. The distribution is divided into three sections, indicated by the coloring, that represent the quantiles of a Normal distribution. The sections are classified into regions where $1\sigma \approx 68\%$, $2\sigma \approx 27\%$, and $3\sigma \approx 5\%$, of the data lies, computed with a KDE. In additional experiments, 1000 samples are generated for each region and then FPD and MSE are compared to show the effectiveness of the proposed method.

8.6 RESULTS

This section presents and discusses the results of the experiments explained in section 8.5.

8.6.1 *Quantitative Results*

Table 8.1 contains quantitative comparisons in terms of typical metrics used to evaluate the quality of generated point cloud objects. The rows for the backbone are included for reference but cannot be used for comparison, as they do not have the conditioning ability. B1 and B2 each symbolize the two corner cases. On the one hand, B1 obtains the lowest FPD but the highest MSE. This is the result of exploiting the good generation quality of the backbone and combining it with a sampling mechanism to obtain objects of desired dimensions. However, it becomes clear that it is hard to obtain the desired shape configuration in an acceptable inference time, while the inference time depends on how many samples have to be drawn for one inference step. On the other hand, there is B2 which obtains an MSE of zero by construction. It simply scales the object to the desired size. However, this comes at the cost of realism, as evident by the increase in FPD. The proposed approach lies in between B1 and B2 regarding FPD and obtains a very low MSE of only 0.28% for "Chair". The other metrics show relatively similar performance for all methods.

The experiments indicate that the proposed method is capable of ensuring desired dimensions while maintaining high quality. However, it has to be noted that the full potential of the method is better visible in the following qualitative results.

It is important to note that these baselines are only applicable since it is possible to easily compute the dimension parameter from

the data itself. For many other applications this is not the case and the proposed method offers a suitable solution (see also section 8.7).

Additional evaluations of the label incorporation of the traditional cGAN [Mirza and Osindero 2014] and CcGAN [Ding et al. 2021] combined with the backbone network did not show satisfying results (appendix A.5.2). In the case of cGAN, the conditioning parameter was simply ignored by generator, while training of CcGAN was very unstable and did not converge.

8.6.2 *Label and Region Sampling Ablations*

Fig. 8.5 shows the results of our distribution sampling experiments on the chair class (explained in section 8.5.4). The results are reported for three differently trained versions of the proposed model. They differ in the way the conditioning parameter $\mathbf{y}_{\text{cond}}$ is sampled at training time (see section 8.4.2). The green dots correspond to random sampling in $[0, 1]$, while orange means that only parameters existing in the training dataset are being used, i.e. the marks of fig. 8.4. The proposed version (purple) uses all possible real numbers sampled from a KDE of the training distribution, i.e. the entire region in fig. 8.4.

It can be observed that naturally MSE and FPD increase when moving away from the distribution center of gravity, i.e. increase 1σ to 2σ and 3σ. For random sampling, significantly worse performance can be observed in all three sampling categories compared to the other two methods. For 1σ and 2σ the two other sampling methods perform almost equally well. However, the proposed sampling strategy introduces a significant performance improvement in the most sparsely populated region where only 5% of all training data lies. This shows the advantage of the distribution-based training over a data-based training, especially for higher σ. Further results can be found in appendix A.5.3.

8.6.3 *Continuous Parameter Interpolation*

A decisive advantage of the proposed method is that the object dimensions can be influenced actively and directly. Fig. 8.6 shows several examples of generated chairs where the continuous parameter for the object height is changed. For different latent vectors (rows), the network generates different shapes of good quality and diversity. When moving away from the main distribution (outermost samples) quality declines but still results in semantically meaningful objects. Another example is depicted in fig. 8.7 where tables with different widths are being generated. Again, while the baseline simply stretches the object, the model learns that wider tables have four legs at each corner instead of only one in the middle.

Table 8.1: **Quantitative Comparison**: Reported are results for the five largest classes of the ShapeNetPart dataset in terms of the metrics used by Shu et al. [2019] and Achlioptas et al. [2018]. Additionally, the regression error (MSE) is reported for the introduced task. A freshly trained TreeGAN [Shu et al. 2019] serves as the backbone network for which the results are reported for reference. The baselines B1 and B2 are described in section 8.5.3. Baseline 1 is the backbone network where ten versions per object are sampled and the one with the best matching dimensions is chosen. Baseline 2 uses the backbone and then scales the object to the desired dimensions. All evaluations are conducted on a hold-out validation split. Shu et al. [2019] use the entire dataset for training, therefore values might vary slightly. This table is based on [Triess et al. 2022a].

Shape	Model	FPD (↓)	MSE [%] (↓)	MMD (↓)		COV (↑)		JSD (↓)
				CD	EMD	CD	EMD	
Table	Backbone	4.5245	–	0.0023	0.0863	0.4750	0.3750	0.1671
	Baseline 1	3.3009	52.88	0.0026	0.0912	0.4875	0.3500	0.1423
	Baseline 2	4.2851	0.00	0.0028	0.0920	0.5375	0.3125	0.1661
	Proposed	3.0692	0.16	0.0019	0.1073	0.4875	0.3000	0.1313
Chair	Backbone	0.9525	–	0.0020	0.1027	0.4875	0.2500	0.1082
	Baseline 1	1.3674	26.07	0.0023	0.1013	0.4750	0.2625	0.1123
	Baseline 2	1.9259	0.00	0.0021	0.1003	0.4875	0.2625	0.1068
	Proposed	1.5290	0.28	0.0022	0.1059	0.4625	0.3125	0.1434
Airplane	Backbone	1.2947	–	0.0002	0.0805	0.4375	0.1375	0.1887
	Baseline 1	1.0209	15.08	0.0003	0.0812	0.4500	0.1125	0.1819
	Baseline 2	1.6613	0.00	0.0003	0.0783	0.5250	0.1375	0.1834
	Proposed	0.8691	0.30	0.0003	0.0724	0.5000	0.1250	0.1291
Car	Backbone	1.0816	–	0.0009	0.0656	0.4250	0.2375	0.0692
	Baseline 1	2.6293	5.15	0.0009	0.0651	0.4500	0.2000	0.0743
	Baseline 2	2.2045	0.00	0.0009	0.0634	0.4375	0.2750	0.0670
	Proposed	1.7129	0.69	0.0010	0.0708	0.4125	0.1250	0.0714
Lamp	Backbone	2.9954	–	0.0037	0.1630	0.4500	0.2750	0.2642
	Baseline 1	3.4737	79.65	0.0029	0.1614	0.4500	0.2125	0.2569
	Baseline 2	36.5025	0.00	0.0041	0.1606	0.4500	0.2250	0.2653
	Proposed	7.5012	0.77	0.0038	0.1917	0.4375	0.1750	0.2576
Total	Backbone	2.1697	–	0.0018	0.0996	0.4550	0.2550	0.1595
	Baseline 1	**2.3584**	35.77	**0.0018**	0.1000	0.4625	0.2275	0.1535
	Baseline 2	9.3159	**0.00**	0.0020	**0.0989**	**0.4875**	**0.2425**	0.1577
	Proposed	2.9363	0.44	**0.0018**	0.1096	0.4200	0.2075	**0.1466**

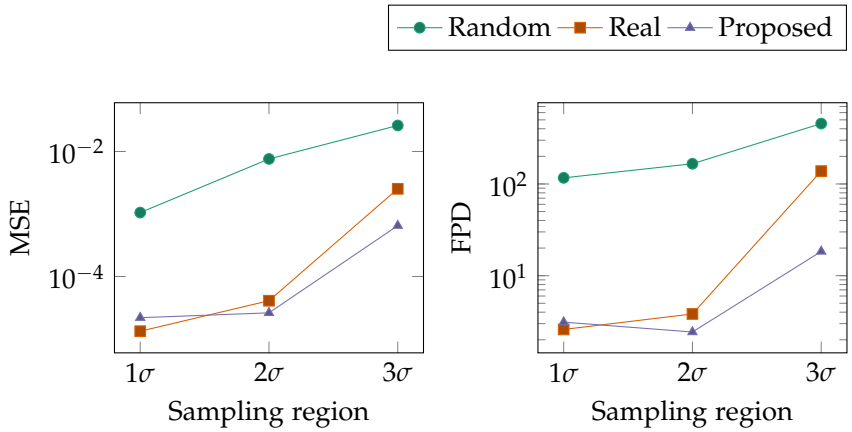


Figure 8.5: **Distribution Sampling Performance**: The figure shows performances for three sampling regions of the data distribution for the proposed architecture trained with three different label sampling strategies. The regions correspond to the labels on the x-axis, as defined in fig. 8.4. For each region, 1000 samples are generated. The left plot shows the MSE of the dimension regression. On the right, the FPD of the generated samples is presented. This figure is based on [Triess et al. 2022a].

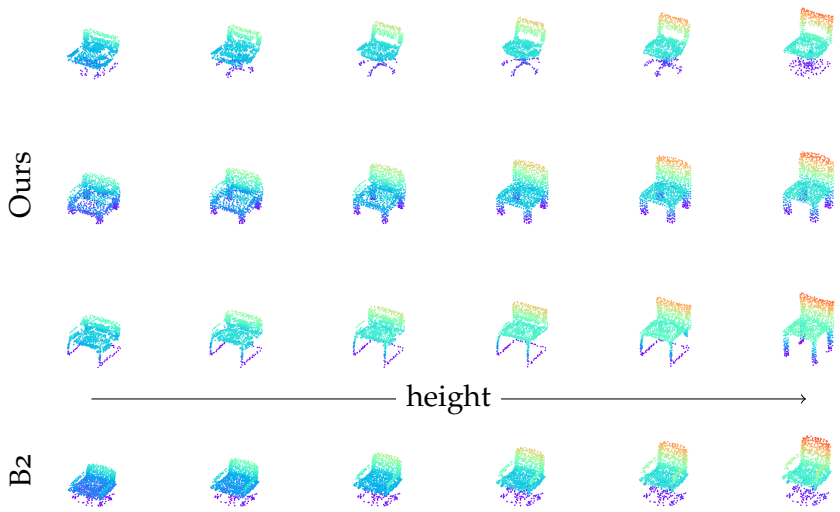


Figure 8.6: **Manipulation of Chair Height**: Each row of objects is generated from a different latent vector while the length and width of the object is kept the same. From left to right, we increase the object height from 0.4 to 0.9. The upper three rows show chairs generated with our method, while the bottom row shows the backbone network with scaling to the requested height (B2). This figure is based on [Triess et al. 2022a].

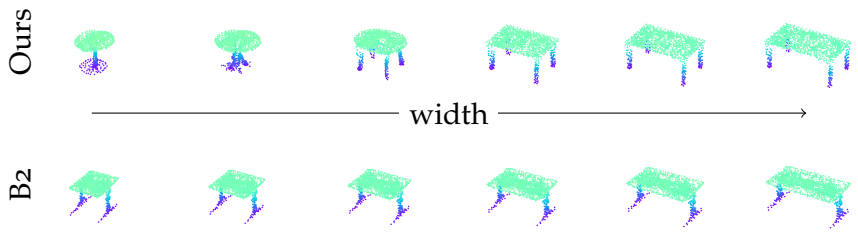


Figure 8.7: **Manipulation of Table Width**: The object width is increased from left (0.4) to right (0.9) while keeping length and height constant. The upper row shows tables generated with the proposed method, while the bottom row shows the backbone network with scaling to the requested width (B2). The baseline simply stretches the table, while the proposed method generates semantically meaningful object shapes. This figure is based on [Triess et al. 2022a].

8.6.4 *Out-of-Distribution Generation*

Fig. 8.1 shows a larger span of width and height conditioning, where only the samples enclosed in the dotted shape are conditioned on parameters sampled from within the distribution. The dataset does not contain any chairs with heights or widths that are larger or smaller than the gray dots indicate (refer to fig. 8.4). The generated shapes for extreme low or high height or width do not necessarily resemble realistic objects, but the generator still maintains the approximate object configuration and even adjusts the shape to this unusual configuration. All generated objects show smooth transitions between the configurations, even when out-of-distribution, and do not collapse in shape. Fig. 8.1 also shows that the model learns a semantic meaning associated with the dimensions. For example, an office chair is usually not narrow and high, as these are usually dinner table chairs, but office chairs are also not very wide and low, as these are usually arm chairs or sofas.

8.6.5 *Diversity and Novelty*

Fig. 8.8 shows five generated examples and five examples from the dataset that have the same dimensions. The generated shapes are diverse and considerably distinct from the dataset samples. This shows two things: First, the proposed method does not simply generate the same object for a given set of dimensions when changing the latent vector. Second, it also does not learn a simple lookup by reproducing the samples from the dataset that lie close to the desired dimensions.

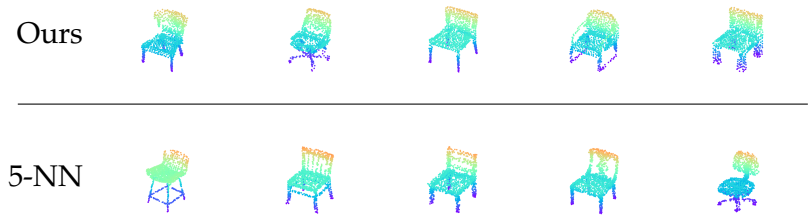


Figure 8.8: **Generation Diversity**: The upper row shows examples generated from five different latent vectors at a fixed dimension $\mathbf{y} = [0.45, 0.42, 0.70]$. The lower row shows the $k = 5$ nearest neighbors from the training dataset in terms of being closest to $\mathbf{y}$. This shows that our method can generate a variety of shapes for the same dimension and does not perform a simple lookup from the dataset. This figure is based on [Triess et al. 2022a].

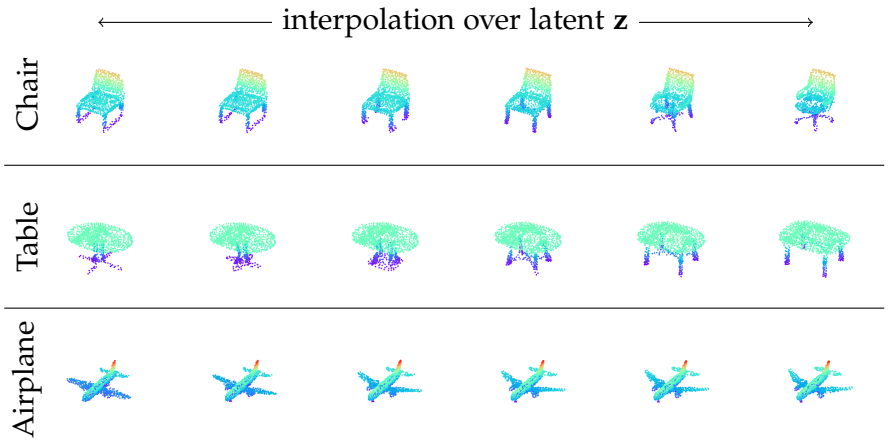


Figure 8.9: **Latent Interpolation**: Interpolation between two random latent vectors $\mathbf{z}$ at constant dimension vector $\mathbf{y}_{\text{cond}}$. This figure is based on [Triess et al. 2022a].

8.6.6 *Latent Interpolation*

Another important aspect of generative processes are disentangled latent representations and smooth representations within the network parameters. Similar to related work, this property is demonstrated by interpolating between two latent vectors. Fig. 8.9 shows that the proposed model generates smooth transitions from left to right for different object shapes. This property was already included in the backbone network, but this experiment shows that this property was not compromised by the introduction of the continuous conditioning.

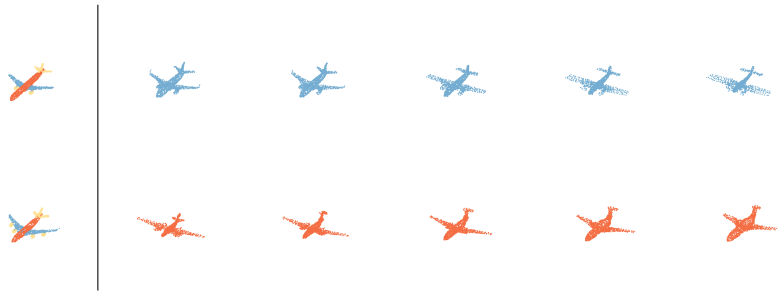


Figure 8.10: **Setting Percentage of Object Parts**: The left column shows two part-annotated examples from the dataset with the airplane body in red, wings in blue, and the remaining parts in yellow. In the upper row we use the proposed method to increase the ratio of wing points within the object, while the lower row increases the percentage of the body part from left to right. Our method constructs semantically meaningful shapes according to the continuous parameter for a fixed latent vector. This figure is based on [Triess et al. 2022a].

8.7 DISCUSSION

Influencing object dimensions is only one application out of many for explicit continuous conditioning. Fig. 8.10 demonstrates how the proposed method can be used to influence the percentage of points in certain object parts compared to the entire shape. In contrast to the object dimensions it is not trivially possible to retrieve this parameter from the generated shape (it is possible to use an extra part segmentation model). Therefore, the use of the discriminator regression is vital, since a naive regression loss by computing the parameter from the generated shape is not applicable.

Next steps for this work include the adaptation to real-world data and the automotive domain. Objects scanned with a laser, e.g. LiDAR, are often only seen from a certain viewpoint in contrast to the objects used in this work. This viewpoint can be modeled by using an additional continuous parameter that defines the angle from which the object was observed. Regarding the proposed application for augmentation of autonomous driving LiDAR scenes, it is then possible to define the size and position of a 3D bounding box for which the GAN then generates a custom-fit object. A challenge will be to adapt the backbone architecture to generate a variable number of points, as common for real-world data.

8.8 CONCLUSION

This chapter presented a novel GAN setup for 3D shape generation that uses continuous conditional parameters to actively influence the dimensions of the generated shapes. Extensive experiments showed that the proposed method is capable of generating custom-fit objects

that adhere to the desired configuration while maintaining good generation quality and diversity. The distribution label sampling proofed to be superior to sampling existing parameters from the training dataset. The chapter also demonstrated the capability of the network to generate samples from outside the distribution and gave a preview on potential other applications. Future work involves the adaptation to real world data, such as LiDAR scans.

9 CONCLUSION

CONTENTS

This dissertation presented a framework for domain adaptation and semantic segmentation of LiDAR point clouds in the context of autonomous driving. Among others, the framework includes data generation with generative models, monitored with a reliable realism metric which shows the correlation between data realism and semantic segmentation performance. Chapter 3 introduced a number of improvements to existing architectures and data representations that increase the final segmentation performance. The proposed scan unfolding combined with cyclic padding increases the segmentation performance and is used in all experiments throughout this dissertation. Next, chapter 4 discussed existing approaches and open research questions in the field of domain adaptation for LiDAR perception. The questions raised in the discussion are answered in the domain adaptation chapters of this dissertation. As an important component, chapter 5 proposed a novel quantitative metric to estimate the realism of LiDAR point clouds based on learned features. This metric also allows exciting applications in the field of anomaly detection. Additionally, the metric is used in chapter 6 to evaluate the quality of generated data in a domain adaptation application. The experiments showed that there is a correlation between the quality of the data and the final perception performance. However, the proposed GAN setups struggle to generate truly realistic LiDAR data while seemingly uninformative data can still have a positive effect on the perception performance. In order to be less dependent on the generated data, chapter 7 proposed a novel method that uses the underlying scene geometry to learn domain-invariant features directly. The geometric information greatly improves the domain adaptation capabilities of the segmentation model and outperforms existing state of the art methods. As an additional application, chapter 8 presented a novel method to generate semantically meaningful single object point cloud shapes from continuous descriptions.

The following section explicitly addresses the research questions initially posed in section 1.2 and summarizes the lessons learned in section 9.1. Section 9.2 concludes this dissertation with an outlook on possible future work.

9.1 DISCUSSION

HUMAN INTERPRETABLE TRAINING DATA A significant fraction of this dissertation dealt with generating realistic LiDAR data, determining the realism of the data, and investigating whether this has an influence on the perception performance. In chapter 6, human inspection and the proposed realism metric showed that existing and proposed LiDAR point cloud generators struggle to generate realistic LiDAR point clouds. Investigations on the relationship between the estimated realism and the resulting perception performance showed a clear correlation between the two. However, from a human perspective this is hard to relate to, since seemingly uninformative data can also have a positive effect on the performance. Therefore, it is vital to have a data-driven realism estimation, as proposed in chapter 5, that extracts and interprets features that are important for the downstream perception module. Furthermore, the lack of human interpretability of the data and its impacts makes it hard to model a mapping-based domain adaptation setup for safety critical applications, as automated driving. In current regulations, interpretability and determinism are of high importance. However, the shown results raise the question whether it is even desirable to enforce interpretable intermediate data. Instead of trying to create a human interpretable, seemingly realistic dataset, the domain mapping pipeline could be treated as a black-box system, such that only the final performance is relevant. This is similar to the concept in domain-invariant feature learning (chapter 7) which does not rely on interpretable intermediate data representations and has attracted more attention in the research community recently. More generally speaking, the trend in many works on perception and prediction for automated driving also goes towards end-to-end like structures, such as early sensor fusion or agent prediction directly from sensor inputs.

GEOMETRY AS A DOMAIN GENERALIZER At the beginning of this dissertation, the question was raised whether geometric information can help to learn semantics for other domains. The experiments in chapter 7 show promising results. To use geometric information from the target domain is especially useful if the target domain has a higher resolution than the source domain. This property of the proposed approach is particularly welcome, since most of the literature deals with high-to-low resolution adaptation, as the other way around is more challenging and often not feasible with their proposed methods. The failure cases of the proposed method show geometrically correct edges in the segmentation maps, even if the predicted class is not correct, in contrast to other methods which have random segmentation errors. This behavior can be beneficial if combined for example with a hierarchical label mapping. When-

ever the network is unsure about the actual class of the correctly segmented object, it could select a parent class with higher certainty. For example, instead of distinguishing between passenger car and van, it could select vehicle. This combination can model uncertainties within the network and detect out-of-distribution objects while improving segmentation performance, since geometric outlines are correct with high probability.

FACTORS FOR SUCCESSFUL LIDAR PERCEPTION Throughout this dissertation, the goal was to perform a semantic segmentation on a LiDAR point cloud. By reviewing this task from multiple angles, this dissertation presented a number of factors that influence the final segmentation performance. Chapter 6 showed that data does not need to be human-interpretable, rather it is important what kind of features can be extracted in the encoding phase, as shown in chapter 7. However, the data representation and the quality of the labels have a significant impact on the performance, as it can be observed with the scan unfolding and cyclic padding proposed in chapter 3. For example, a segmentation model with 50% runtime and 25% capacity of the baseline can be combined with these measures and can easily obtain the same segmentation performance. In general it is important to define what a successful perception means for the required application. In academic benchmarks, the performance on the test set is often the most important factor. Therefore, aspects like runtime or over-fitting are often neglected in the publications, though these are important values for real-time applications, especially when they operate in an open world setting, such as automated vehicles. Another important factor that gained more importance over the last years, is the successful combination of data from multiple sensors. To obtain a detailed vision around the vehicle, often multiple LiDAR and other sensors are used. A recent trend goes towards using **B**ird's-**E**ye **V**iew (BEV) transformers to fuse multiple sensor inputs in a BEV feature space.

DOMAIN ADAPTATION FOR OPEN-WORLD APPLICATIONS Having an open-world setting is the main motivation to address the problem of domain adaptation in the context of automated driving. The driving systems need to be robust and reliable in a variety of situations that might not be foreseeable. If a human driver is confronted with such a situation and has enough time to react, they can easily assess what would be an adequate way to behave. This is because humans are capable to transfer knowledge from other domains. For example, if there is an elephant on the road, the human might never have seen an elephant on the road, but since they know that this is an elephant, they will slow down the vehicle. A neural network might not even recognize the elephant as an object on the road and even if it does, it might not respond properly. Chapter 4

introduced domain adaptation as a kind of transfer learning with many different aspects as they have been discussed throughout this dissertation. From a research perspective all these areas are interesting in terms of solving highly complex and relevant problems. However, from an application point of view, the methods need to be practical and efficient. From what was presented in this dissertation, transferring data from one domain to another is a highly complex task which requires multiple steps that might not even result in the desired behavior. Therefore this technique is probably better suited for non-safety-critical applications, such as style transfer for art, deep fakes, and virtual avatars. For safety-critical applications, techniques with more general transfer learning capabilities are necessary. One example is proposed in chapter 7, where knowledge from both the source and the target domain is used to solve a task for both domains, while annotations are only available for the source domain.

9.2 FUTURE WORK

This section goes beyond the scope of this dissertation and discusses potential future work that could be conducted based on the findings in this dissertation. Potential shortcomings of this dissertation are explicitly addressed and open questions are raised that should be answered through future research.

OBJECT GENERATION FOR REAL-WORLD DATA Object augmentation at training time is a currently used and quite simple method to improve detection of desired objects at inference time. Chapter 8 proposed a novel concept to adversarially generate such objects. Future work could involve the adaptation to real-world automotive LiDAR data. To this extend, the viewpoint and distance to the object must be modeled additionally to the proposed properties of object size and aspect ratios. Sequences of driving scenes can be used to obtain training data for multiple viewpoints and distances of the same object, which can then also be modeled with the continuous parameter. A challenge will be to design an architecture that can generate a variable amount of points, as common for LiDAR point clouds. To generate new viewpoints of the same object, methods like neural radiance fields could be adapted for LiDAR data [Mildenhall et al. 2020]. In the end, a configurable and diverse augmentation technique is highly beneficial to model various driving situations, corner cases, and safety critical scenarios for training and testing.

MULTI-MODALITY AND TIME CONSISTENCY This dissertation focused on single-frame LiDAR perception as an integral part of automated driving systems and filled research gaps in some areas, for

example LiDAR domain mapping in chapter 6 and the introduction of a proper quantitative realism metric for LiDAR in chapter 5. A driving force in human perception is the combination of sensory inputs from multiple sources over a longer period of time to establish a detailed understanding of the current and possible upcoming situations. Such multi-cue information also renders beneficial in LiDAR perception, such as object detection and tracking [Emmerichs et al. 2021]. However, in the research area of transfer learning, the usage of multi-modal time-series information should be extended. Often multi-modal information is only used at the training phase to obtain depth *Ground Truth* from the LiDAR sensor for many detection and classification tasks in the camera domain. However, using multi-modal information also at inference time is beneficial, especially in domain transfer applications. Jaritz et al. [2020] design a specific cross-modal loss to combine camera and LiDAR to facilitate an information exchange between the two modalities which benefits the overall performance in presence of a domain gap. Combining this with a timely aspect can exploit temporal consistencies which can greatly help in adverse weather or other scenarios that exhibit domain gaps. Up until now, only Saltori et al. [2020] exploit temporal information for domain adaptation applications. However they use it in an offline fashion to generate pseudo-labels on the target domain for self-taught learning. Future work should include using multi-modal and time-series information in an online-fashion to mitigate a large amount of domain gaps. These mainly include the following cases: *day-to-night*, *sensor-to-sensor*, *weather-to-weather*. For these domain gaps, often one sensor type is more affected than another or disturbances can be mitigated with temporal consistency [Jaritz et al. 2020; Saltori et al. 2020].

KNOWLEDGE TRANSFER FROM OTHER DOMAINS All measures discussed before aim at providing methods to make perception systems more robust to domain shifts. However, the task of domain adaptation is currently only an intermediate step towards robust and reliable autonomous systems. The actual aim is so train neural networks in such a way that automated vehicles can always be deployed anywhere and anytime on earth (cf. SAE level 5 in fig. 1.1). This requires good transfer learning capabilities and includes solving open set problems (cf. fig. 4.2). Most camera perception systems initialize their models with pre-trained weights from the ImageNet dataset. This leads to better performance and generalization capabilities. Another area of research covers the field of natural language image captioning, where a model needs to describe the contents of a scene with words. This requires the model to learn hierarchical dependencies over large areas in an image. Both methods are commonly not applied to LiDAR-based perception systems which can mainly be attributed to the lack of openly available datasets for

these tasks. Independent of the sensor modality however, this combination of transfer learning and context understanding could be a solution towards more general perception for open-world problems, such as automated driving. Specifically, future work should consider leveraging language models that are trained on tremendous amount of data and learned highly complex object dependencies to initialize the neural networks in the automated driving system. Thus, extensive knowledge from other domains could be incorporated into the system and can then be fit onto the context of automated driving. This is similar to how humans would approach unusual encounters while driving. Often similar objects or scenarios have been observed in another context and therefore a human can adequately respond to the situation. The strategy is to move closer to general artificial intelligence and thus enable autonomous driving at human level and above.

A

APPENDIX

CONTENTS

A.1 SCAN-BASED SEMANTIC SEGMENTATION

This section holds additional information about the scan unfolding algorithm proposed in chapter 3.

The scan unfolding method is designed to be a proxy representation of the original raw sensor data with reduced mutual point occlusions. The conducted back-projection is only necessary since the dataset does not provide the direct sensor output or an index-map for simple back-projection. When working in an actual autonomous driving stack, the preprocessing needed for the scan unfolding can be omitted, as the LiDAR scanner directly provides the depth-image format.

The following provides algorithm 1 that exploits the distinct data representation of the *KITTI* dataset to generate the desired scan pattern. The algorithm is applied to the uncorrected scan data (without ego-motion compensation), which is accessible via the raw data of *KITTI*. The *KITTI* raw format lists LiDAR points of an accumulated

360 degree scan in order of their vertical index of the associated sensor scan line. However, the crossovers between two consecutive scan lines happen at the cut to the rear of the vehicle and are not indicated in the provided data. Thus the task of detecting these positions to assign each point to its vertical index remains and is addressed by algorithm 1.

Algorithm 1: Scan Unfolding on *KITTI*: *threshold* is chosen to be larger than the horizontal resolution (*KITTI*: $threshold = 0.3°$). The algorithm is published in [Triess et al. 2020].

Data: An array *points* of size $N \times 3$, a tuple (H, W)
Result: *projection* of *points* with shape $H \times W$

$depth \longleftarrow \sqrt{points_x^2 + points_y^2 + points_z^2}$
$rows \longleftarrow$ `GetRows(`*points*`)`
$columns \longleftarrow$ `GetColumns(`*points*`)`
sort *columns*, *rows* and *depth* by decreasing *depth*
projection $\longleftarrow$ array of shape $H \times W$
projection[*columns*, *rows*] = *depth*

Function `GetRows(`*points*`)`**:**
  $\phi \longleftarrow \text{atan2}(points_y, points_x)$
  $jump \longleftarrow |\phi[1:] - \phi[:-1]| > threshold$
  $jump \longleftarrow [0] + jump$
  rows $\longleftarrow$ cumulative sum over *jump*
  return *rows*

Function `GetColumns(`*points*`)`**:**
  $\phi \longleftarrow \text{atan2}(points_y, points_x)$
  $columns \longleftarrow W \cdot (\pi - \phi) / (2\pi)$
  return *columns*

A.2 A METRIC TO QUANTIFY THE REALISM OF LIDAR POINT CLOUDS

This section contains supplementary material for chapter 5 and covers details of the DNN architectures, hyperparameters, evaluation, and additional qualitative results. Section A.2.1 contains detailed listings of the hyperparameters of the metric network and an analysis of the training as additional information to the architecture in section 5.3.2. Section A.2.3 provides additional qualitative results.

A.2.1 *Implementation Details and Hyperparameters*

Table A.1 lists all layers, inputs, and operations of the proposed DNN architecture. TensorFlow is used to implement online data processing, neural network weight optimization, and network inference. The implementation is oriented on the original PointNet++ implementation [Qi et al. 2017b][1]. The Adam optimizer is used for optimization. The learning rate is initially set to $1e^{-3}$ with exponential warm-up and decay.

Table A.1: **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_i are the number of query points at abstraction level i. K_i are the number of nearest neighbors to search at abstraction level i. U_C and U_A are the number of output units of the classifier and adversary, respectively. The table is based on [Triess et al. 2021b].

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	$[2048 \times 10]$	Indices of the K_1 nearest neigbors per query
5	1, 2, ↑	Group	$[2048 \times 10 \times 3]$	Grouped neighborhoods
6	↑	Neighborhood normalization	$[2048 \times 10 \times 3]$	Translation normalization towards query point
7	↑	(Conv+**L**eaky **Re**ctified **L**inear **U**nit (LReLU)) $\times 2$	$[2048 \times 10 \times 64]$	Kernel 1×1, stride 1
8	↑	Conv+LReLU	$[2048 \times 10 \times 128]$	Kernel 1×1, stride 1

1 PointNet++ code https://github.com/charlesq34/pointnet2

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+LReLU) $\times 2$	$[256 \times 10 \times 128]$	Kernel 1×1, stride 1
17	↑	Conv+LReLU	$[256 \times 10 \times 256]$	Kernel 1×1, stride 1
18	↑	ReduceMax	$[256 \times 256]$	Maximum over neighborhood features $\rightarrow$ latent representation z
Classifier / Adversary				
19	↑	Dense+LReLU	$[256 \times 128]$	
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 $p_{C,A}$

In contrast to PointNet++, a KNN search is used instead of radius search. PointNet++ operates on point clouds from the ShapeNet dataset, which contains 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. This work found KNN search more practical to obtain meaningful neighborhoods in LiDAR point clouds compared to radius search.

The outputs of the classifier C and adversary A have U_C and U_A channels, respectively. The classifier outputs the scores for each of the $U_C = 3$ categories, namely *Real*, *Synthetic*, *Misc*. In the adversary, the output has $U_A = 7$ channels, one for each of the support sets. However, fig. 5.2 only shows outputs for the two support sets from the *Real* category. For simplicity, the details of the implementation are not visualized in the respective figure and are indicated by the filter triangle in the path between the feature extractor and the adversary. The filter is implemented as a class weighting when computing the loss from the adversary output. The class weights w for each dataset d are set to

$$w_d = \begin{cases} 1, & \text{if } d \in \mathit{Real} \\ 0, & \text{otherwise} \end{cases} . \tag{A.1}$$

This was found to be the easiest and most stable way to implement the desired behavior in TensorFlow graph mode.

A.2.2 *Theoretical Lower Bound*

Generally, classification accuracy ACC is defined in range $[0, 1]$. However, the lower bound is actually $\frac{1}{U}$ with U being the number of classes. The classes in the presented fair learning setup are set to $U_C = 3$ and $U_A = U_A^{Real} + U_A^{Synthetic} + U_A^{Misc} = 2 + 2 + 3 = 7$. Therefore, one would assume that if the network has the best possible performance and the best possible fairness, the accuracy result to $\mathrm{ACC}_C \approx 1$ and $\mathrm{ACC}_A \approx \frac{1}{U_A} = \frac{1}{7}$. Due to the direct correspondence between categories and datasets this is not the case for the adversary. The resulting confusion matrices are schematically illustrated in fig. A.1. The pseudo-diagonal for the adversary is caused by the perfect classification capabilities of the classifier, which prevents a perfect confusion in the adversary. The new lower bound of the adversary accuracy in this state can now be formulated as

$$\mathrm{ACC}_A \approx \frac{U_C}{U_A} = \frac{3}{7} \approx 42.9\% \tag{A.2}$$

which is considerably higher than $\mathrm{ACC}_A \approx \frac{1}{U_A} = \frac{1}{7} \approx 14.3\%$.

A.2.3 *Qualitative Results*

This section provides additional visualizations. Fig. A.2 shows the qualitative results of the realism metric when being applied to one *CARLA* sample with varying additive noise. The quantitative results

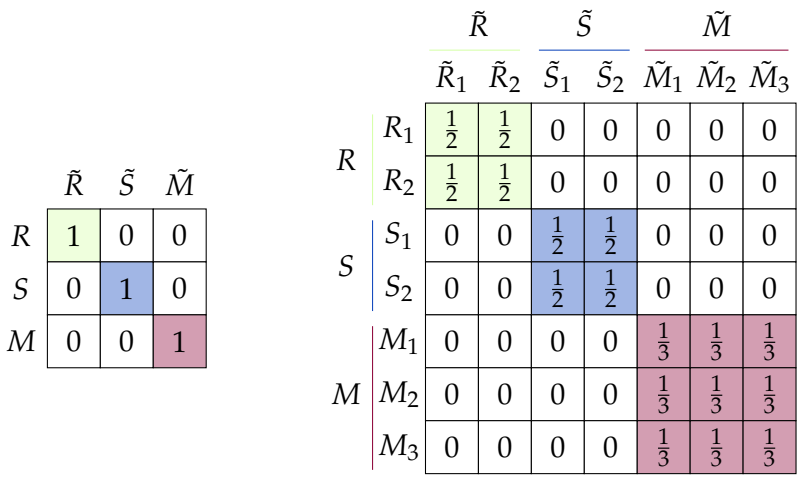


Figure A.1: **Lower Bound Confusion Matrix**: The figure shows schematic confusion matrices for the classifier (left) and the adversary (right). Here, the classifier has $U_C = 3$ categories (R, S, M) and the adversary has $U_A = 7$ output channels for the respective datasets of the categories. If the network is trained with maximum accuracy (100%), the classifier confusion matrix is a diagonal matrix, as shown on the left. Assuming the adversary is maximally confused in this state, a pseudo-diagonal confusion emerges, increasing the theoretical lower bound of adversary accuracy from $\frac{1}{U_A}$ to $\frac{U_C}{U_A}$. The figure is based on [Triess et al. 2021b].

over the whole test split are presented in section 5.5.4. Fig. A.3 illustrates the example scenes from section 5.5.5 in 3D for better visualization of the detected anomalies.

A.3 DOMAIN ADAPTATION VIA DATA GENERATION FOR DOMAIN MAPPING

In this section additional information for the domain mapping experiments in chapter 6 are provided.

A.3.1 *Up-sampling Details*

This section gives additional details on the up-sampling experiments of section 6.1.5. 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 the resize method from TensorFlow (`tf.image.resize(images, size, method=ResizeMethod.BILINEAR)`) is used. For all other experiments, the generator from the SRGAN architecture Ledig et al. [2017] is used and for the GAN experiments, also the discriminator architecture.

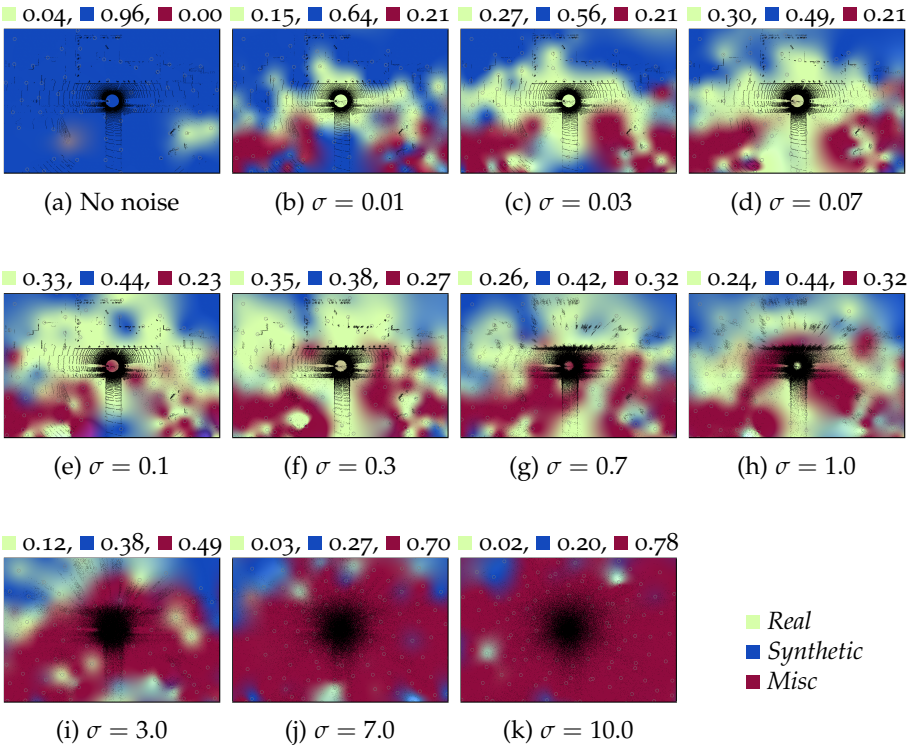
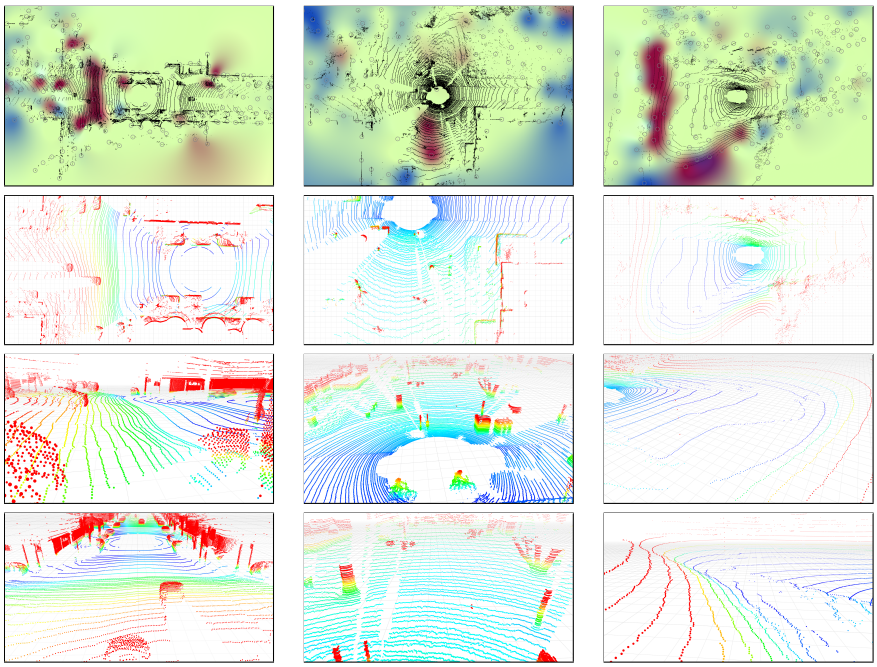


Figure A.2: ***CARLA* Sample with Gaussian Noise**: Example image series of a single *CARLA* sample with additive Gaussian noise of varying standard deviation σ. The colors show the probabilities for each category at each local region. The numbers above each image show the mean score per category for the entire scene. When adding noise with small σ, the sample appears more realistic but when the noise gets too strong, the data does not contain any structuring anymore. The figure is based on [Triess et al. 2021b].



(a) *PandaSet* (b) *KITTI* (c) *nuScenes*

Figure A.3: **Localization of Anomalies**: Shown are the same example scenes from the main paper with low *Real* scores. The upper images show the color-coded metric results, the lower images are 3D visualization of the same scene for better understanding of the scene contents. In (a), the metric marks a road section with extreme elevation changes. The street is covered with bumps. Additionally, there is a steep elevation gain in the rear of the ego-vehicle (left side). In the lower half of (b), the metric highlights seemingly floating branches of two trees that enter the LiDAR field-of-view from above. The height-encoded color shows the red branch clusters floating above the cyan road surface. (c) shows an unusual scene in a dead end road with steep hills surrounding the car. The figure is based on [Triess et al. 2021b].

A.3.1.1 *Implementation Details*

Table A.2 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

$$\min_{\theta_G} \mathcal{L}_\alpha = \min_{\theta_G} \frac{1}{\alpha|\gamma|} \sum_{(i,j)\in\gamma} \left| r_{i,j}^{\text{gt}} - r_{i,j}^{\text{hr}} \right| \tag{A.3}$$

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

$$r^{\text{hr}} = G_{\theta_G}\left(r^{\text{lr}}\right) \tag{A.4}$$

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

Table A.3 lists all layers, inputs, and operations of the SRGAN discriminator architecture. Here, an adversarial loss, defined as

$$\min_{\theta_G} \max_{\theta_D} \left\{ \log\left[D_{\theta_D}\left(r^{\text{gt}}\right)\right] + \log\left[1 - D_{\theta_D}\left(G_{\theta_G}\left(r^{\text{lr}}\right)\right)\right] \right\} \tag{A.5}$$

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

A.3.1.2 *Additional Experimental Results*

Section 6.1.5 shows the realism metric results on up-sampled *KITTI* data, while the original *KITTI* was also used to train the metric model. Fig. A.4 shows three additional scenes that are up-sampled with the same methods, but the original data is taken from the *PandaSet* dataset, which a completely unknown dataset. A similar behavior over the different up-sampling methods as to the *KITTI* dataset can be observed.

A.4 DOMAIN INVARIANT FEATURE LEARNING

This section is the appendix for chapter 7. Section A.4.1 explains the pre-processing of the *nuScenes* dataset and section A.4.2 lists the label mappings for the two state of the art comparision experiments.

A.4.1 *Pre-processing of the* nuScenes *dataset*

The *nuScenes* dataset requires additional pre-processing compared to *KITTI* to be used in this work. Since moving objects are captured multiple times at different positions when accumulating point clouds, it is essential to filter them out. *nuScenes* does not provide such a dynamic flag, therefore the *Ground Truth* bounding boxes

Table A.2: **SRGAN Generator Architecture**: Detailed network architecture and input format definition of the SRGAN generator [Ledig et al. 2017]. 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. The table is based on [Triess et al. 2021b].

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+PReLU	$[H, W, 64]$	Kernel 9×9, stride 1
4	↑	Conv+BN+PReLU	$[H, W, 64]$	Kernel 3×3, stride 1
5	↑	Conv+BN	$[H, W, 64]$	Kernel 3×3, stride 1
6	↑, 3	Add	$[H, W, 64]$	Element-wise addition
7	↑	Repeat (4-6)	$[H, W, 64]$	$\times 16$ repetition of residual blocks
8	↑	Conv+BN	$[H, W, 64]$	Kernel 3×3, stride 1
9	↑, 3	Add	$[H, W, 64]$	Element-wise addition
		Super-resolution blocks		
10	↑	Conv	$[H, W, 256]$	Kernel 3×3, stride 1
11	↑	SubpixelShuffle	$[2 \cdot H, W, 128]$	Reshape by moving values from the channel to the spatial dimension
12	↑	PReLU	$[2 \cdot H, W, 128]$	
13	↑	Repeat (10-12)	$[f_{\text{up}} \cdot H, W, 128]$	$\times \log_2 f_{\text{up}}$ repetition with f_{up} as the desired upsampling factor
14	↑	Conv	$[f_{\text{up}} \cdot H, W, 1]$	Kernel 9×9, stride 1

Table A.3: **SRGAN Discriminator Architecture**: Detailed network architecture and input format definition of the SRGAN discriminator [Ledig et al. 2017]. The input to the network is either the ground truth r^{gt} or the prediction from the generator r^{hr}. The table is based on [Triess et al. 2021b].

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

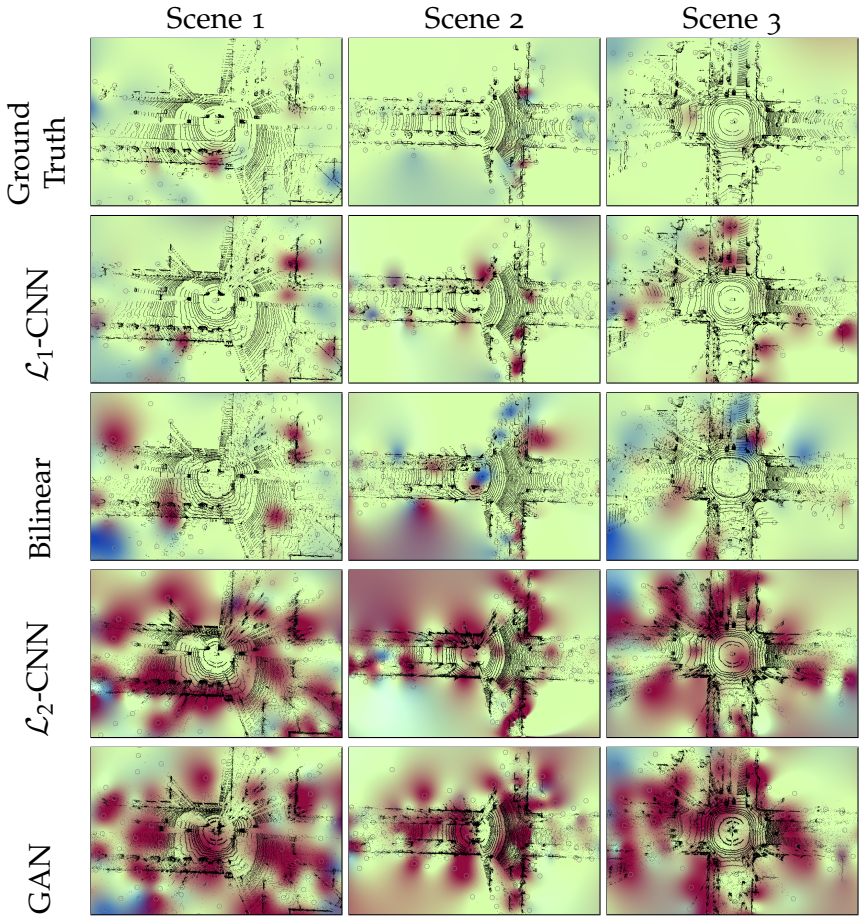
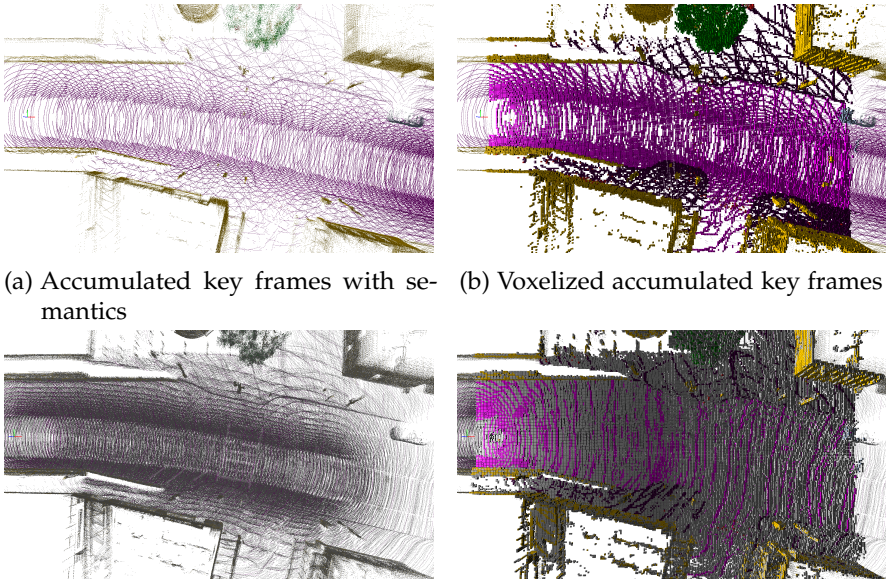


Figure A.4: ***PandaSet* Up-Sampling**: Shown are three example scenes from the *PandaSet* test split. The first row shows the high-resolution Ground Truth, i.e. target. The four rows below show the corresponding reconstructions of different techniques for 4× upsampling. Note that the query points are sampled at different locations for each image, leading to varying score computation for similar regions, and that the height above ground of the query points is not encoded in this visualization. The figure is based on [Triess et al. 2021b]. (■ *Real*, ■ *Synthetic*, ■ *Misc*)



(a) Accumulated key frames with semantics

(b) Voxelized accumulated key frames

(c) Accumulated unlabeled frames

(d) Voxelized combined data

Figure A.5: **Visualization of *nuScenes* pre-processing**: The accumulated semantic data from *nuScenes* (a) is too sparse to serve as a proper training target, since many voxels would be marked as free-space, though they actually contain objects (b). Therefore, the additional unlabeled data (c) is used to create dense geometry with sparse semantic information as a training target (d).

with dynamic properties of the object detection benchmark are used to generate the missing dynamic labels.

Since *nuScenes* is quite sparse, even when accumulated, the additional 90% unlabeled frames are also used for the geometry target. For these frames, no *Ground Truth* bounding boxes exist to recover the dynamic flag from. Therefore, the kinematic properties of all dynamic objects are estimated for the unlabeled frames and used to automatically label the dynamic objects.

One other drawback of the *nuScenes* dataset is the simplification of the ego-pose, which re-sets the vertical position to zero for each frame. This results in surface errors in distance when the vehicle drives on hilly ground. Using **I**terative **C**losest **P**oint (ICP) algorithm could improve the target data generation and potentially further improve the achieved results.

A.4.2 *Label Mapping for State of the Art Comparisons*

Section 7.5.5.2 compares the performance of the proposed method to [Yi et al. 2021]. Fig. A.4 lists the class mapping, proposed by Yi et al. [2021], as used for the experiments in that section.

Section 7.5.5.1 compares the performance of the proposed method to [Langer et al. 2020]. In their work, the *nuScenes* dataset is re-labeled with the same classes as *SemanticKITTI*. Therefore, no extra

label mapping is needed. The labels are published together with the paper.

A.5 POINT CLOUD GENERATION WITH CONTINUOUS CONDITIONING

This section contains supplementary material for chapter 8 and covers details of the DNN architectures, hyperparameters, evaluation, and additional results. Section A.5.1 gives all the training details of our proposed approach. For the sake of completeness, section A.5.2 discusses further experiments that are indicated in the main paper. Section A.5.3 presents further qualitative and quantitative results to complement the results section of the main paper.

A.5.1 *Implementation Details*

A.5.1.1 *Architecture*

Table A.5 lists all layers, inputs, and operations of the DNN architecture for the generator model. The code[2] from the original *PyTorch* implementation of TreeGAN [Shu et al. 2019] is used. Except for the input layers, the configuration is equal to the one of TreeGAN. Table A.6 lists all layers, inputs, and operations of the discriminator DNN architecture. Here, the PointNet [Qi et al. 2017a] contained in the TreeGAN code was used as a basis. The split of the network for the auxiliary classifier mode is located directly after the PointNet feature extractor layers. It is followed by two identical sequences of linear operations, where the adversarial head outputs a vector of size 1, while the regression head outputs a vector of size d as the continuous conditioning parameter.

A.5.1.2 *Training*

Two Adam optimizers are used for optimization, one for the parameters of the generator and one for the discriminator. For both, the learning rate is set to 10^{-4}. Additionally, the two weighting factors λ_{adv} and λ_{reg} for the losses are optimized. The losses for the generator and discriminator are formulated as

$$\mathcal{L} = \lambda_{\text{adv}} \cdot \mathcal{L}_{\text{adv}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}$$

with the adversarial loss $\mathcal{L}_{\text{adv}}$ and the regression loss $\mathcal{L}_{\text{reg}}$. In order to avoid simply learning weighting factors of zero and to ensure stable training convergence at the same time, the loss is implemented as

$$\mathcal{L} = \mathcal{L}_{\text{adv}} \cdot e^{v_{\text{adv}}} + v_{\text{adv}} + \mathcal{L}_{\text{reg}} \cdot e^{v_{\text{reg}}} + v_{\text{reg}}$$

2 TreeGAN code: https://github.com/seowok/TreeGAN

Table A.4: **Label Mapping for [Yi et al. 2021]**: Mapping of the classes from the *SemanticKITTI* and the *nuScenes* dataset, as used by Yi et al. [2021].

SemanticKITTI		**mapped**		***nuScenes***
person	»	person	«	adult, child, construction worker, police officer, personal mobility
road, lane-marking, parking	»	road	«	drivable surface
sidewalk	»	sidewalk	«	sidewalk
terrain	»	terrain	«	terrain
car	»	car	«	car
bicycle	»	bicycle	«	bicycle
motorcycle	»	motorcycle	«	motorcycle
bus, other vehicle	»	other vehicle	«	trailer, construction vehicle, emergency vehicle, bus
truck	»	truck	«	truck
vegetation, trunk	»	vegetation	«	vegetation
unlabeled, outlier, on-rails, bicyclist, motorcyclist, other-ground, building, fence, other-structure, pole, traffic-sign, other-object	»	noise	«	noise, animal, stroller, wheelchair, barrier, debris, pushable/pullable, traffic-cone, bicycle rack, manmade, ego vehicle

Table A.5: **Generator 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. d is the number of dimensions of the conditioning parameter. In case of the dimensions extent $d = 3$, while for the influence of the object part percentage $d = 1$. This table is based on [Triess et al. 2022a].

ID	Inputs	Operation	Output Shape
1	$\mathbf{z}$	Sample latent vector $\mathbf{z} \sim \mathcal{Z} = \mathcal{N}(\mathbf{0}, I)$	$[96]$
2	$\mathbf{y}_{\text{cond}}$	Sample continuous parameter $\mathbf{y}_{\text{cond}} \sim \text{KDE}(\mathbf{y}_{\text{real}})$	$[d]$
		Label Handling	
3	1	Linear Layer	$[64]$
4	2	Linear Layer	$[32]$
5	3, 4	Concatenate	$[1 \times 96]$
		Tree Graph Convolution (TreeGC) Network	
6	↑	Tree Graph Convolution + LReLU	$[1 \times 256]$
7	↑	Branching	$[2 \times 256]$
8	↑	Tree Graph Convolution + LReLU	$[2 \times 256]$
9	↑	Branching	$[4 \times 256]$
10	↑	Tree Graph Convolution + LReLU	$[4 \times 256]$
11	↑	Branching	$[8 \times 256]$
12	↑	Tree Graph Convolution + LReLU	$[8 \times 128]$
13	↑	Branching	$[16 \times 128]$
14	↑	Tree Graph Convolution + LReLU	$[16 \times 128]$
15	↑	Branching	$[32 \times 128]$
16	↑	Tree Graph Convolution + LReLU	$[32 \times 128]$
17	↑	Branching	$[2048 \times 128]$
18	↑	Tree Graph Convolution	$[2048 \times 3]$

Table A.6: **Discriminator 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. d is the number of dimensions of the conditioning parameter. In case of the dimensions extent $d = 3$, while for the influence of the object part percentage $d = 1$. This table is based on [Triess et al. 2022a].

ID	Inputs	Operation	Output Shape	Description
1	point cloud $\mathbf{x}$	x, y, z	$[2048 \times 3]$	Input point cloud $\mathbf{x} = \mathbf{x}_{\text{real}}$ for real data and $\mathbf{x} = \mathbf{x}_{\text{gen}}$ for generated data.
		PointNet Feature Extractor		
2	↑	Conv1D + LReLU	$[2048 \times 64]$	Kernel size 1×1, stride 1
3	↑	Conv1D + LReLU	$[2048 \times 128]$	Kernel size 1×1, stride 1
4	↑	Conv1D + LReLU	$[2048 \times 256]$	Kernel size 1×1, stride 1
5	↑	Conv1D + LReLU	$[2048 \times 512]$	Kernel size 1×1, stride 1
6	↑	Conv1D + LReLU	$[2048 \times 1024]$	Kernel size 1×1, stride 1
7	↑	MaxPool	$[1024]$	Global features
		Adversarial Output Head		
8	↑	Linear Layer + LReLU	$[1024]$	
9	↑	Linear Layer + LReLU	$[512]$	
10	↑	Linear Layer + LReLU	$[512]$	
11	↑	Linear Layer	$[1]$	Output vector $D(\mathbf{x})$
		Regression Output Head		
12	7	Linear Layer + LReLU	$[1024]$	
13	↑	Linear Layer + LReLU	$[512]$	
14	↑	Linear Layer + LReLU	$[512]$	
15	↑	Linear Layer	$[d]$	Output vector $\hat{\mathbf{y}}(\mathbf{x})$

with v_{adv} and v_{reg} being the trainable variables. Both variables are initialized to $v = 0$ at the beginning of the training, such that both loss parts are equally weighted.

A.5.2 *Additional Analysis*

A.5.2.1 *Loss Variations*

Both the generator and the discriminator loss consist of an adversarial part and a regression part. The generator regression loss computes the error between the requested parameter $\mathbf{y}_{\text{cond}}$ and the corresponding discriminator prediction, while the discriminator regression is defined as the error between the parameter of the real data $\mathbf{y}_{\text{real}}$ and its corresponding prediction $\hat{\mathbf{y}}_{\text{real}}$, such that

$$\begin{aligned} \mathcal{L}_{G,\text{reg}} &= \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{cond}}, \hat{\mathbf{y}}_{\text{gen}}\right) \quad \text{and} \\ \mathcal{L}_{D,\text{reg}} &= \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{real}}, \hat{\mathbf{y}}_{\text{real}}\right). \end{aligned}$$

It is notable that $\mathcal{L}_{D,\text{reg}}$ is only computed for the real samples and not for the generated samples and that $\mathcal{L}_{G,\text{reg}}$ uses the discriminator prediction $\hat{\mathbf{y}}_{\text{gen}}$ instead of the configuration of the actually generated object $\mathbf{y}_{\text{gen}}$. These design choices can be attributed to the fact that in many cases the actual parameter $\mathbf{y}_{\text{gen}}$ of the generated point cloud $\mathbf{x}_{\text{gen}}$ is unknown, i.e. cannot be trivially retrieved from $\mathbf{x}_{\text{gen}}$. Since the application of influencing object dimensions offers the possibility to simply compute $\mathbf{y}_{\text{gen}}$ from $\mathbf{x}_{\text{gen}}$ with $\mathbf{y}_{\text{gen}} = \|\max(\mathbf{x}_{\text{gen}}) - \min(\mathbf{x}_{\text{gen}})\|$, additional loss variations that exploit this property are investigated.

First, the generator and discriminator regression losses are defined as

$$\begin{aligned} \mathcal{L}_{G,\text{reg}} &= \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{cond}}, \mathbf{y}_{\text{gen}}\right) \quad \text{and} \\ \mathcal{L}_{D,\text{reg}} &= \frac{1}{2}\Big[\mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{real}}, \hat{\mathbf{y}}_{\text{real}}\right) + \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{gen}}, \hat{\mathbf{y}}_{\text{gen}}\right)\Big], \end{aligned}$$

respectively. The experiments showed that this leads to slightly better training convergence, but not to significant performance gains in the final model in terms of FPD or MSE. Therefore, it can be concluded that the proposed losses of chapter 8 are a good mechanism to train the model when $\mathbf{y}_{\text{gen}}$ is otherwise unknown, as for most applications.

Second, another loss configuration is investigated. As for the discriminator, the proposed method skips the generated part of the

loss entirely. However, it is also possible to formulate the losses as follows

$$\mathcal{L}_{G,\text{reg}} = \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{cond}}, \hat{\mathbf{y}}_{\text{gen}}\right) \quad \text{and}$$
$$\mathcal{L}_{D,\text{reg}} = \frac{1}{2}\Big[\mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{real}}, \hat{\mathbf{y}}_{\text{real}}\right) + \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{cond}}, \hat{\mathbf{y}}_{\text{gen}}\right)\Big],$$

where $\mathbf{y}_{\text{gen}}$ from above is replaced with $\hat{\mathbf{y}}_{\text{real}}$ in the generator and $\mathbf{y}_{\text{cond}}$ in the discriminator. However, experiments showed that this leads to unstable training and results in a significantly worse model performance. This can be attributed to false feedback for the discriminator, especially in the beginning of the training. At that time, the generator is not yet well enough trained to output shapes that are close to the requested parameters ($\mathbf{y}_{\text{gen}} \neq \hat{\mathbf{y}}_{\text{gen}}$). Therefore, it is best to not include the generated path for the discriminator regression at all.

A.5.2.2 *Other Conditioning Concepts*

Additionally to the proposed method, other configurations for the continuous conditioning of point cloud generation are investigated. These methods either do not yield promising results or are limited in applicability. Therefore, they are not included in chapter 8. For the sake of completeness and reproducibility, all relevant implementation details are included here and the overall results of the experiments are stated.

CGAN WITH CONTINUOUS PARAMETERS The concepts of cGAN (refer to fig. A.6) and CcGAN [Ding et al. 2021] are adapted to work with TreeGAN [Shu et al. 2019] as the backbone network. In contrast to the proposed approach, this is referred to as an implicit conditioning scheme, since there is no explicit excitation that forces the model to use the conditioning input explicitly. The discriminator receives both the point cloud and the conditioning parameter as an input. The loss function of the generator G is defined as

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}\left[D\left(\mathbf{y}_{\text{cond}}, G(\mathbf{y}_{\text{cond}}, \mathbf{z})\right)\right]$$

where $\mathcal{Z}$ represents the latent code distribution which follows a Normal distribution, such that $z \in \mathcal{N}(0,1)$. The loss function of the discriminator is defined as

$$\begin{aligned}\mathcal{L}_D =& \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}\left[D\left(\mathbf{y}_{\text{cond}}, G(\mathbf{y}_{\text{cond}}, \mathbf{z})\right)\right] \\ &- \mathbb{E}_{\mathbf{x}\sim\mathcal{R}}\left[D\left(\mathbf{y}_{\text{real}}, \mathbf{x}\right)\right] + \mathcal{L}_{\text{gp}}\end{aligned}$$

with the gradient penalty $\mathcal{L}_{\text{gp}}$ as defined in chapter 8.

As a first variant, the traditional label incorporation introduced by cGAN is used [Mirza and Osindero 2014]. This is referred to as the

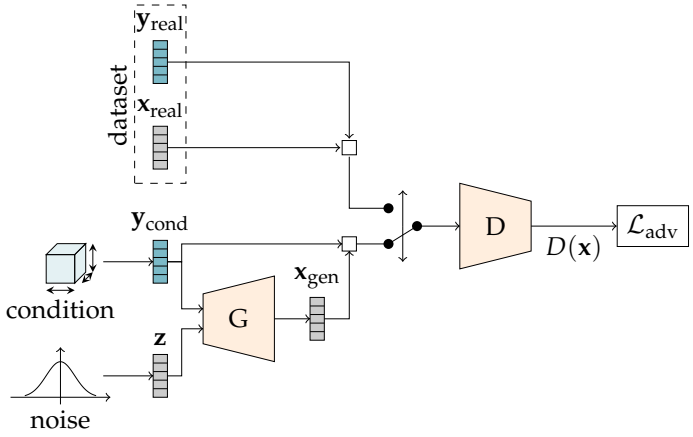
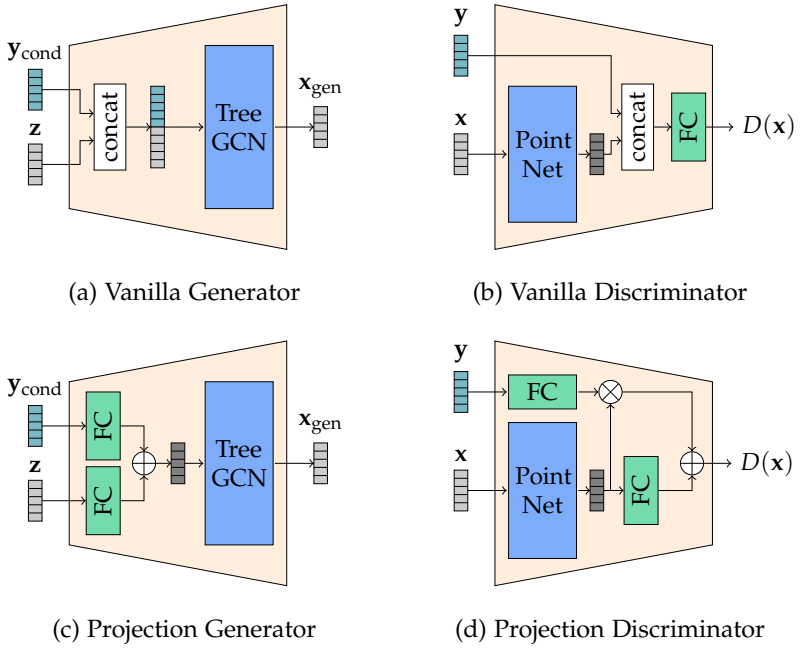


Figure A.6: **cGAN with Continuous Parameters**: The generator G generates a point cloud $\mathbf{x}_{\text{gen}}$ from a random vector $\mathbf{z}$ and a continuous parameter $\mathbf{y}$. The discriminator receives a set of a point cloud and a parameter either from the real $\{\mathbf{y}_{\text{real}}, \mathbf{x}_{\text{real}}\}$ or the generated distribution $\{\mathbf{y}_{\text{cond}}, \mathbf{x}_{\text{gen}}\}$. It then outputs an estimate whether the set is real or generated with which the adversarial loss $\mathcal{L}_{\text{adv}}$ is computed. This figure is based on [Triess et al. 2022a].



(a) Vanilla Generator

(b) Vanilla Discriminator

(c) Projection Generator

(d) Projection Discriminator

Figure A.7: **Parameter Handling Details**: The figure shows the generator and discriminator input handling for Vanilla and Projection cGAN. This figure is based on [Triess et al. 2022a].

Vanilla cGAN. The generator details are depicted in fig. A.7a and the discriminator details are shown in fig. A.7b. The generator label incorporation simply concatenates the latent and the label vector prior to feeding it to the **G**raph **C**onvolution **N**etwork (GCN). In the discriminator, the point cloud is processed by PointNet which outputs a feature vector to which the label vector is concatenated before being processed by a final set of linear layers.

For the second variant, the label input configuration proposed by CcGAN [Ding et al. 2021] is used to handle continuous parameters. Their approach is inspired by the method of label projection [Miyato and Koyama 2018], which is why this chapter refers to this variant as Projection cGAN. Details for the generator and discriminator are shown in fig. A.7c and fig. A.7d, respectively. In contrast to Vanilla cGAN, both the latent and the label vectors are passed through a linear layer first, after which both are added together element-wise. For the discriminator, the label vector is propagated through a linear layer after which the inner product with the features from PointNet is calculated. The features are passed through the final set of linear layers after which the result is added to the result of the inner product.

As mentioned in chapter 8, both variants did not achieve satisfying results. The Vanilla variant ignored the conditioning entirely which led to very high MSE values (about four magnitudes higher than our proposed method). The performance in terms of FPD is close to the backbone. For the Projection variant, very unstable training courses were observed that often led to a collapse of the training with the model performing significantly worse in all metrics compared to all other models.

CGAN WITH REGRESSION The ability of influencing the dimensions of an object has the major advantage that it is possible to directly compute the dimensions from the generated object to check whether the generator worked properly. This can also be used as a training signal. Fig. A.8 shows an alternative approach for the proposed architecture, where a standard unconditioned discriminator is combined with an additional regression component. The generation of the desired dimensions is explicitly enforced with the regression loss, therefore this variant is referred to as the Regression cGAN. The generator loss is defined as

$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}\left[D\left(G(\mathbf{y}_{\text{cond}}, \mathbf{z})\right)\right] + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}\left(\mathbf{y}_{\text{cond}}, \mathbf{y}_{\text{gen}}\right)$$

where $\mathbf{y}_{\text{gen}}$ are the actual dimensions calculated from the generated point cloud, and λ_{reg} is the weighting factor for the regression loss. The generator loss is solely responsible to enforce the adherence to the dimension conditioning since the discriminator is not condi-

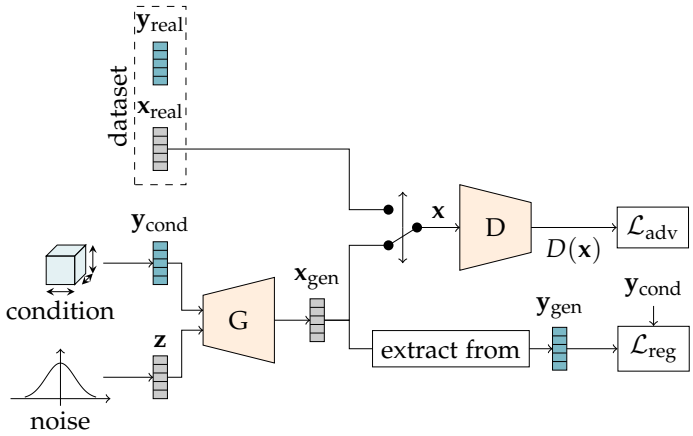


Figure A.8: **cGAN with Additional Regression**: The generator G generates a point cloud $\mathbf{x}_{\text{gen}}$ from a random vector z and a regression label y. The discriminator either receives a real or a generated point cloud and predicts the probability of the sample originating from the real distribution. Additionally, the dimensions $\mathbf{y}_{\text{gen}}$ of the generated point cloud are extracted which are then used to compute the regression error $\mathcal{L}_{\text{reg}}$. This figure is based on [Triess et al. 2022a].

Table A.7: **Quantitative Comparison to Regression cGAN**: Results are reported for the classes "Chair" and "Airplane". All evaluations are conducted on a hold-out validation split. For both FPD and MSE smaller values indicate a better performance. MSE is given in % relative to the size of a unit cube. This table is based on [Triess et al. 2022a].

Model	Chair		Airplane	
	FPD	MSE	FPD	MSE
Reg. cGAN	**1.4420**	1.85	0.8802	20.19
Ours	1.5290	**0.28**	**0.8691**	**0.30**

tioned on the label input. The discriminator only judges from which distribution a sample originates from. Its loss function is defined as

$$\mathcal{L}_D = \mathbb{E}_{\mathbf{z}\sim\mathcal{Z}}\left[D\left(G(\mathbf{y}_{\text{cond}}, \mathbf{z})\right)\right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{R}}\left[D(\mathbf{x}_{\text{real}})\right] + \mathcal{L}_{\text{gp}}.$$

The training behavior of the Regression cGAN is fundamentally different to the one of the proposed method. It can be observed that FPD is optimized first, hitting a minimum value at a quite early point during training where MSE is still quite high. The results for this checkpoint are listed in table A.7. From this point onward MSE is further minimized at the expense of FPD performance. In contrast to the proposed method, Regression cGAN aims at minimizing MSE down to zero while accepting FPD values that are magnitudes higher

than for the proposed method. This means no realistic object shapes are being generated.

Considering these results and the fact that Regression cGAN can only be used for specific applications where $\mathbf{y}_{\text{gen}}$ can easily be retrieved from the generated data, it can be concluded that the proposed method is superior to Regression cGAN. The proposed method offers a much easier and more stable handling of training while resulting in a very good performing model that is applicable to many scenarios.

A.5.3 *Additional Results*

Table A.8 gives details on the region-based performance of the five largest classes of the ShapeNetPart dataset. The class distribution of the dataset is shown in fig. A.9. For the experiments, classes with less than 1,000 object shapes are not considered. The quantitative results, especially when comparing the proposed method to the chosen baselines, also depend on the distribution of the influencing parameter. For comparison, fig. A.10 shows the distribution of the five largest classes in terms of their extent in object height and width. While some classes, like "Table", have a wide distribution, others are more densely packed, like "Car". Especially for the "Lamp" class, the stretching baseline (B2) achieves bad performance in terms of FPD, which can be attributed to its unique distribution.

Fig. A.11 gives an overview on the quality and the diversity of the generated samples for the five different shapes.

Table A.8: **Region-based Performance**: The table shows performances for three sampling regions (σ_1, σ_2, σ_3) of the data distribution for the proposed architecture. The model is trained with two different sampling strategies each: the proposed version where labels are sampled from the KDE of the distributions (areas of fig. 8.4), and the default version where labels are sampled from the list of labels contained in the dataset (marks of fig. 8.4). For each σ-region, 1000 samples are generated. Reported are the MSE of the dimension regression and the FPD of the generated samples. Especially for less densely populated regions, i.e.σ_3, the proposed sampling strategy achieves better results. This table is based on [Triess et al. 2022a].

Shape	Label Sampling	FPD (↓)			MSE [%] (↓)		
		σ_1	σ_2	σ_3	σ_1	σ_2	σ_3
Table	proposed distr. sampling	**4.7881**	**11.1556**	**83.1944**	**0.14**	**0.27**	**17.49**
	from dataset samples	5.5063	23.1117	153.7859	0.25	1.15	52.01
Chair	proposed distr. sampling	3.1137	**2.4310**	**18.3729**	0.22	**0.26**	**6.51**
	from dataset samples	**2.5915**	3.8257	138.0703	**0.13**	0.41	25.30
Airplane	proposed distr. sampling	2.8487	**1.6660**	**22.4949**	0.20	**0.36**	7.62
	from dataset samples	**2.5138**	2.0912	42.4436	**0.17**	0.39	**5.17**
Car	proposed distr. sampling	4.7105	11.8133	**32.1120**	**0.61**	**0.89**	**3.74**
	from dataset samples	**3.7981**	**8.1593**	60.2905	0.77	1.13	4.13
Lamp	proposed distr. sampling	5.7873	**20.2216**	**111.7557**	**0.44**	**1.84**	**11.50**
	from dataset samples	**4.4660**	26.0117	290.8956	1.36	8.16	69.97

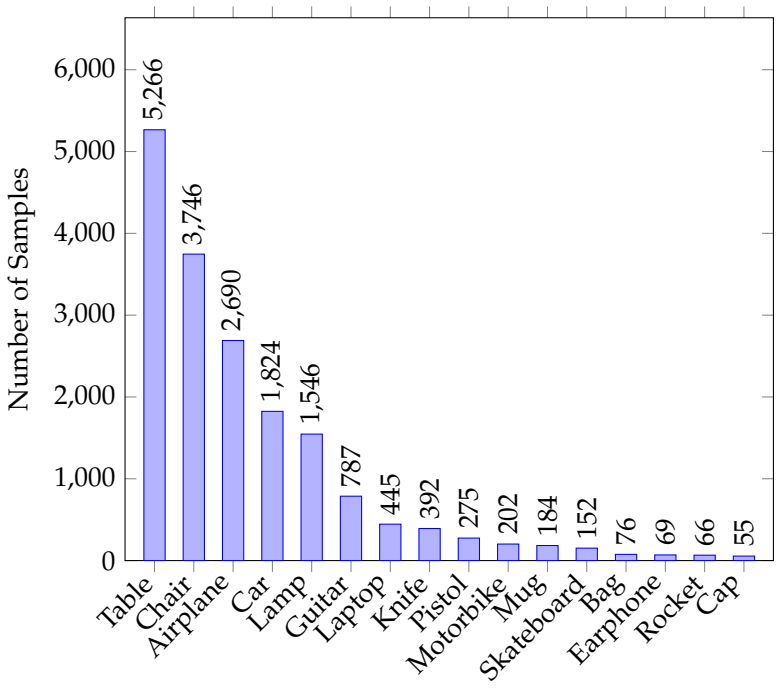
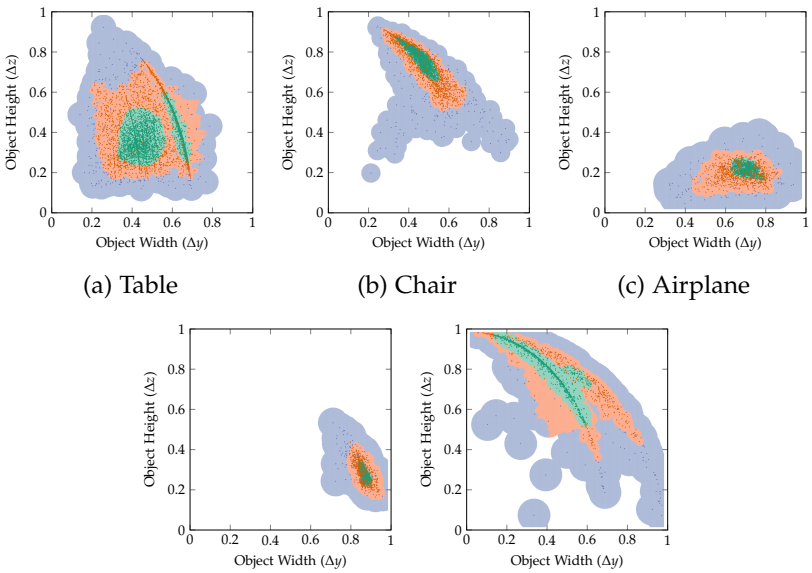


Figure A.9: **Dataset Class Distribution**: Shown is the class distribution of the ShapeNetPart dataset [Yi et al. 2016]. The five largest classes are used for the experiments: "Table", "Chair", "Airplane", "Car", and "Lamp". This figure is based on [Triess et al. 2022a].



(a) Table (b) Chair (c) Airplane

(d) Car (e) Lamp

Figure A.10: **Region-classified Dimension Distribution**: Shown are the sample distributions of five different object classes according to their extent in height Δz and width Δy. The length Δx of the object is not considered in this visualization. Each mark corresponds to one shape in the dataset. The three colors represent the resulting regions from KNN classifier with k=20 based on a KDE. The regions correspond to $1\sigma \approx$ 68% (green), $2\sigma \approx$ 27% (orange), and $3\sigma = 5\%$ (blue) of the entire data distribution. (Best viewed in color.) This figure is based on [Triess et al. 2022a].

Figure A.11: **Example Showcase**: The figure shows shapes generated with the proposed method from randomly sampled latent and conditioning parameters. The conditioning parameters are sampled from the KDE of the real data distribution, therefore only shapes for realistic dimensions are shown here. This figure is based on [Triess et al. 2022a].

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning Representations and Generative Models for 3D Point Clouds. In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. (Cited on pages 49, 50, 106, 108, 114, and 118.)

C. Agia, R. Cheng, Y. Ren, and B. Liu. S3CNet: A Sparse Semantic Scene Completion Network for LiDAR Point Clouds. In *Proc. Conf. on Robot Learning (CoRL)*, pages 2148–2161, 2020. (Cited on page 13.)

I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo. Domain Adaptation in LiDAR Semantic Segmentation. *arXiv.org*, 2020. (Cited on pages 36, 37, and 43.)

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv.org*, 2017. (Cited on pages 49, 50, and 113.)

S. Arora, A. Risteski, and Y. Zhang. Do GANs learn the distribution? Some Theory and Empirics. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. (Cited on pages 49 and 50.)

R. Atienza. A Conditional Generative Adversarial Network for Rendering Point Clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019. (Cited on page 112.)

V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 2481–2495, 2017. (Cited on page 14.)

S. A. Baur, F. Moosmann, S. Wirges, and C. B. Rist. Real-time 3D LiDAR Flow for Autonomous Vehicles. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1288–1295, 2019. (Cited on page 106.)

J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 9296–9306, 2019. (Cited on pages 11, 14, 17, 18, 26, 41, 81, and 91.)

S. Benaim and L. Wolf. One-Sided Unsupervised Domain Mapping. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (Cited on page 37.)

C. F. Benz. Fahrzeug mit Gasmotorenbetrieb, 1886. Publication Date: 1886/01/29. DE. Patent 37435. (Cited on page 1.)

A. Beutel, J. Chen, Z. Zhao, and E. H. Chi. Data Decisions and Theoretical Implications when Adversarially Learning Fair Representations. In *Workshop on Fairness, Accountability, and Transparency in Machine Learning*, 2017. (Cited on pages 53, 55, and 56.)

M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. Dietmayer, and F. Heide. Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 11679–11689, 2020. (Cited on page 42.)

A. Borji. Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding (CVIU)*, pages 41–65, 2019. (Cited on pages 47 and 48.)

A. Boulch, B. L. Saux, and N. Audebert. Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks. In *Proceedings of the Eurographics Workshop on 3D Object Retrieval*, 2017. (Cited on page 13.)

A. Bühler. GAN-based Synthesis of 3D Point Clouds conditioned on Continuous Object Dimensions. Master's thesis, University of Stuttgart, 2021. (Cited on page 106.)

P. P. Busto and J. Gall. Open Set Domain Adaptation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 754–763, 2017. (Cited on page 43.)

L. Caccia, H. van Hoof, A. Courville, and J. Pineau. Deep Generative Modeling of LiDAR Data. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 5034–5040, 2019. (Cited on pages 37, 47, and 50.)

H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020. (Cited on pages 3, 11, 20, 41, 57, and 91.)

B. Caine, R. Roelofs, V. Vasudevan, J. Ngiam, Y. Chai, Z. Chen, and J. Shlens. Pseudo-labeling for Scalable 3D Object Detection. *arXiv.org*, 2021. (Cited on page 40.)

A. Carballo, J. Lambert, A. Monrroy, D. Wong, P. Narksri, Y. Kitsukawa, E. Takeuchi, S. Kato, and K. Takeda. LIBRE: The Multiple 3D LiDAR Dataset. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1101, 2020. (Cited on page 42.)

A. Chang, M. Savva, and C. D. Manning. Learning Spatial Knowledge for Text to 3D Scene Generation. In *Proc. of the Conf. on*

Empirical Methods in Natural Language Processing (EMNLP), pages 2028–2038, 2014. (Cited on page 110.)

A. Chang, W. Monroe, M. Savva, C. Potts, and C. D. Manning. Text to 3D Scene Generation with Rich Lexical Grounding. In *Proc. of the International Joint Conf. on Natural Language Processing (IJCNLP)*, pages 53–62, 2015a. (Cited on page 110.)

A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv.org*, 2015b. (Cited on pages 49, 50, 54, and 114.)

W.-G. Chang, T. You, S. Seo, S. Kwak, and B. Han. Domain-Specific Batch Normalization for Unsupervised Domain Adaptation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7346–7354, 2019. (Cited on page 40.)

T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li. Mode Regularized Generative Adversarial Networks. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. (Cited on page 49.)

K. Chen, C. B. Choy, M. Savva, A. X. Chang, T. Funkhouser, and S. Savarese. Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings. In *Proc. of the Asian Conf. on Computer Vision (ACCV)*, pages 100–116, 2018. (Cited on page 110.)

L. Chen, L. Wu, Z. Hu, and M. Wang. Quality-Aware Unpaired Image-to-Image Translation. In *IEEE Trans. on Multimedia (TMM)*, pages 2664–2674, 2019a. (Cited on page 108.)

L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv.org*, 2017a. (Cited on page 14.)

M. Chen, H. Xue, and D. Cai. Domain Adaptation for Semantic Segmentation With Maximum Squares Loss. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 2090–2099, 2019b. (Cited on page 34.)

X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2180–2188, 2016. (Cited on page 48.)

Y.-H. Chen, W.-Y. Chen, Y.-T. Chen, B.-C. Tsai, Y.-C. F. Wang, and M. Sun. No More Discrimination: Cross City Adaptation of Road Scene Segmenters. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 2011–2020, 2017b. (Cited on pages 35 and 102.)

Z. Chen and H. Zhang. Learning Implicit Fields for Generative Shape Modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5932–5941, 2019. (Cited on page 108.)

D. Chicco. Siamese Neural Networks: An Overview. *Artificial Neural Networks*, pages 73–94, 2021. (Cited on page 51.)

Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. StarGAN: Unified Generative Adversarial Networks for Multi-domain Image-to-Image Translation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8789–8797, 2018. (Cited on page 37.)

C. Choy, J. Gwak, and S. Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019. (Cited on page 101.)

Y. W. T. Chris J. Maddison, Andriy Mnih. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. (Cited on page 80.)

H. de Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. (Cited on page 88.)

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. (Cited on pages 49 and 50.)

P. Dhariwal and A. Q. Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. (Cited on page 14.)

L. R. Dice. Measures of the Amount of Ecologic Association Between Species. *Ecology*, pages 297–302, 1945. (Cited on page 21.)

X. Ding, Y. Wang, Z. Xu, W. J. Welch, and Z. J. Wang. CcGAN: Continuous Conditional Generative Adversarial Networks for Image Generation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. (Cited on pages 108, 109, 117, 149, and 151.)

J. Dolson, J. Baek, C. Plagemann, and S. Thrun. Upsampling range data in dynamic environments. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1141–1148, 2010. (Cited on page 68.)

C. Dong, C. C. Loy, K. He, and X. Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 295–307, 2016. (Cited on page 68.)

X. Dong and J. Shen. Triplet Loss in Siamese Network for Object Tracking. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 472–488, 2018. (Cited on page 51.)

A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An Open Urban Driving Simulator. In *Proc. Conf. on Robot Learning (CoRL)*, 2017. (Cited on pages 57 and 81.)

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. (Cited on page 18.)

A. Elhadidy, M. Afifi, M. Hassoubah, Y. Ali, and M. ElHelw. Improved Semantic Segmentation of Low-Resolution 3D Point Clouds Using Supervised Domain Adaptation. In *Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 588–593, 2020. (Cited on page 36.)

D. Emmerichs, P. Pinggera, and B. Ommer. VelocityNet: Motion-Driven Feature Aggregation for 3D Object Detection in Point Cloud Sequences. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 13279–13285, 2021. (Cited on page 129.)

B. P. Eric Jang, Shixiang Gu. Categorical Reparameterization with Gumbel-Softmax. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. (Cited on page 80.)

H. Fan, H. Su, and L. Guibas. A Point Set Generation Network for 3D Object Reconstruction from a Single Image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2463–2471, 2017. (Cited on page 108.)

Y. Ganin and V. Lempitsky. Unsupervised Domain Adaptation by Backpropagation. In *Proceedings of the International Conference on Machine Learning*, pages 1180–1189, 2015. (Cited on page 32.)

A. Gavade and P. Sane. Super Resolution Image Reconstruction By Using Bicubic Interpolation. In *National Conference on Advanced Technologies in Electrical and Electronic Systems*, 2014. (Cited on page 68.)

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012. (Cited on pages 10 and 17.)

A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, pages 1231–1237, 2013. (Cited on pages 20, 57, and 91.)

I. J. Good. Some terminology and notation in information theory. *Proceedings of the IEE - Part C: Monographs*, pages 200–204, 1956. (Cited on page 20.)

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. (Cited on pages 14, 15, 46, 48, 49, 50, 106, and 108.)

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. (Cited on pages 9 and 13.)

B. Graham, M. Engelcke, and L. van der Maaten. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9224–9232, 2018. (Cited on page 13.)

K. Gregor and Y. LeCun. Emergence of Complex-Like Cells in a Temporal Product Network with Local Receptive Fields. *arXiv.org*, 2010. (Cited on page 25.)

A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research (JMLR)*, pages 1–51, 2012. (Cited on pages 49, 50, and 72.)

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5769–5779, 2017. (Cited on pages 110 and 113.)

Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 4338–4364, 2021. (Cited on page 13.)

S. Gurumurthy, R. K. Sarvadevabhatla, and V. B. Radhakrishnan. DeLiGAN: Generative Adversarial Networks for Diverse and Limited Data. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4941–4949, 2017. (Cited on page 49.)

E. Härkönen, A. Hertzmann, J. Lehtinen, and S. Paris. GANSpace: Discovering Interpretable GAN Controls. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9841–9850, 2020. (Cited on page 109.)

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. (Cited on page 69.)

Z. He, W. Zuo, M. Kan, S. Shan, and X. Chen. AttGAN: Facial Attribute Editing by Only Changing What You Want. In *IEEE Trans. on Image Processing (TIP)*, pages 5464–5478, 2019. (Cited on page 109.)

R. Heinzler, F. Piewak, P. Schindler, and W. Stork. CNN-Based Lidar Point Cloud De-Noising in Adverse Weather. In *IEEE Robotics and Automation Letters (RA-L)*, pages 2514–2521, 2020. (Cited on page 42.)

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6629–6640, 2017. (Cited on pages 46, 49, and 50.)

E. Hoffer and N. Ailon. Deep Metric Learning using Triplet Network. In *Similarity-Based Pattern Recognition (SIMBAD)*, pages 84–92, 2015. (Cited on page 51.)

J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. CyCADA: Cycle Consistent Adversarial Domain Adaptation. In *Proc. of the International Conf. on Machine learning (ICML)*, pages 1989–1998, 2018. (Cited on pages 16, 34, 77, and 109.)

B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise Convolutional Neural Networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 984–993, 2018. (Cited on page 13.)

G. B. Huang, H. Lee, and E. Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2518–2525, 2012. (Cited on page 25.)

X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie. Stacked Generative Adversarial Networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1866–1875, 2017. (Cited on page 48.)

T.-W. Hui, C. C. Loy, and X. Tang. Depth Map Super-Resolution by Deep Multi-Scale Guidance. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 353–369, 2016. (Cited on page 68.)

B. Hurl, K. Czarnecki, and S. L. Waslander. Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception. *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529, 2019. (Cited on page 57.)

D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic. Generating images with recurrent adversarial networks. *arXiv.org*, 2016. (Cited on page 49.)

S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. of the International Conf. on Machine learning (ICML)*, pages 448–456, 2015. (Cited on page 39.)

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017. (Cited on pages 16, 49, 50, and 79.)

P. Jaccard. Etude de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, pages 547–579, 1901. (Cited on page 20.)

M. Jaritz, T.-H. Vu, R. de Charette, E. Wirbel, and P. Pérez. xMUDA: Cross-Modal Unsupervised Domain Adaptation for 3D Semantic Segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12602–12611, 2020. (Cited on pages 38 and 129.)

P. Jiang and S. Saripalli. LiDARNet: A Boundary-Aware Domain Adaptation Model for Lidar Point Cloud Semantic. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 2457–2464, 2020. (Cited on page 39.)

B. Johannsen. Domain Invariant Feature Learning for Cross-Sensor Semantic Segmentation of LiDAR Point Clouds by Leveraging the underlying Scene Geometry. Master's thesis, University of Stuttgart, 2022. (Cited on page 86.)

J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 694–711, 2016. (Cited on pages 69, 70, and 74.)

T. Karras, S. Laine, and T. Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proc. IEEE Conf. on*

Computer Vision and Pattern Recognition (CVPR), pages 4217–4228, 2019. (Cited on pages 108 and 109.)

T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and Improving the Image Quality of StyleGAN. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020. (Cited on page 109.)

O. S. Kayhan and J. C. van Gemert. On Translation Invariance in CNNs: Convolutional Layers Can Exploit Absolute Spatial Location. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14262–14273, 2020. (Cited on page 28.)

A. Kendall, Y. Gal, and R. Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7482–7491, 2018. (Cited on pages 89 and 114.)

D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2014. (Cited on pages 14, 106, and 108.)

P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011. (Cited on page 14.)

L. Landrieu and M. Simonovsky. Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4558–4567, 2018. (Cited on page 13.)

A. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, 2019. (Cited on pages 3, 13, 39, and 106.)

F. Langer, A. Milioto, A. Haag, J. Behley, and C. Stachniss. Domain Transfer for Semantic Segmentation of LiDAR Data using Deep Neural Networks. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 8263–8270, 2020. (Cited on pages 37, 43, 86, 89, 99, 100, 101, 103, 143, 159, and 160.)

F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg. Deep Projective 3D Semantic Segmentation. In *Proc. of the International Conf. on Computer Analysis of Images and Patterns (CAIP)*, pages 95–107, 2017. (Cited on page 13.)

C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial

Network. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114, 2017. (Cited on pages 48, 71, 74, 136, 140, and 141.)

C.-Y. Lee, T. Batra, M. H. Baig, and D. Ulbricht. Sliced Wasserstein Discrepancy for Unsupervised Domain Adaptation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10277–10287, 2019a. (Cited on page 102.)

K.-H. Lee, G. Ros, J. Li, and A. Gaidon. SPIGAN: Privileged Adversarial Learning from Simulation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019b. (Cited on page 77.)

E. L. Lehmann and J. P. Romano. *Testing Statistical Hypotheses*. Springer Science & Business Media, 2006. (Cited on pages 46, 49, and 50.)

D. Li, H. Ling, S. W. Kim, K. Kreis, A. Barriuso, S. Fidler, and A. Torralba. BigDatasetGAN: Synthesizing ImageNet with Pixel-wise Annotations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. (Cited on page 46.)

Y. Li and J. Ibanez-Guzman. LiDAR for Autonomous Driving: The Principles, Challenges and Trends for Automotive LiDAR and Perception Systems. In *Signal Processing Magazine*, pages 50–61, 2020. (Cited on page 41.)

Y. Li, N. Wang, J. Shi, J. Liu, and X. Hou. Revisiting Batch Normalization for Practical Domain Adaptation. In *Proc. of the International Conf. on Learning Representations (ICLR) Workshops*, 2017. (Cited on page 39.)

G. Lin, F. Liu, A. Milan, C. Shen, and I. Reid. RefineNet: Multi-Path Refinement Networks for Dense Prediction. In *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1228–1242, 2020. (Cited on page 14.)

Z. Lin, A. Khetan, G. Fanti, and S. Oh. PacGAN: The power of two samples in generative adversarial networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1505–1514, 2018. (Cited on page 48.)

M.-Y. Liu, O. Tuzel, and Y. Taguchi. Joint Geodesic Upsampling of Depth Images. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 169–176, 2013. (Cited on page 68.)

X. Liu, C. R. Qi, and L. J. Guibas. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019. (Cited on page 13.)

J. Löhdefink and T. Fingscheidt. Improving Performance of Semantic Segmentation CycleGANs by Noise Injection into the Latent Segmentation Space. *arXiv.org*, 2022. (Cited on page 46.)

P. Luc, C. Couprie, S. Chintala, and J. Verbeek. Semantic Segmentation using Adversarial Networks. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2016. (Cited on page 80.)

M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs Created Equal? A Large-Scale Study. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 698–707, 2018. (Cited on page 48.)

S. Luo and W. Hu. Diffusion Probabilistic Models for 3D Point Cloud Generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2836–2844, 2021. (Cited on page 106.)

Y. Luo, Z. Wang, Z. Huang, and M. Baktashmotlagh. Progressive Graph Learning for Open-Set Domain Adaptation. In *Proc. of the International Conf. on Machine learning (ICML)*, pages 6468–6478, 2020. (Cited on page 43.)

M. F. Mathieu, J. J. Zhao, J. Zhao, A. Ramesh, P. Sprechmann, and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5047–5055, 2016. (Cited on page 48.)

H.-Y. Meng, L. Gao, Y. Lai, and D. Manocha. VV-Net: Voxel VAE Net with Group Convolutions for Point Cloud Segmentation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 8499–8507, 2019. (Cited on page 13.)

Mercedes-Benz Group AG. First internationally valid system approval for conditionally automated driving, 2021. URL https://group.mercedes-benz.com/innovation/product-innovation/autonomous-driving/system-approval-for-conditionally-automated-driving.html. Accessed 2022/07/07. (Cited on page 2.)

L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4455–4465, 2019. (Cited on pages 108 and 110.)

M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson. Deep Level Sets: Implicit Surface Representations for 3D Shape Inference. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. (Cited on page 108.)

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 405–421, 2020. (Cited on page 128.)

A. Milioto, I. Vizzo, J. Behley, and C. Stachniss. RangeNet++: Fast and Accurate LiDAR Semantic Segmentation. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 4213–4220, 2019. (Cited on pages 3, 13, 19, 20, 21, 22, 26, 27, 28, 72, 74, 79, and 81.)

M. Mirza and S. Osindero. Conditional Generative Adversarial Nets. *arXiv.org*, 2014. (Cited on pages 15, 16, 37, 109, 117, and 149.)

T. Miyato and M. Koyama. cGANs with Projection Discriminator. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. (Cited on pages 109 and 151.)

K. Mo, H. Wang, X. Yan, and L. Guibas. PT2PC: Learning to Generate 3D Point Cloud Shapes from Part Tree Conditions. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 683–701, 2020. (Cited on page 110.)

P. Morerio, J. Cavazza, and V. Murino. Minimal-Entropy Correlation Alignment for Unsupervised Deep Domain Adaptation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. (Cited on pages 34, 38, and 90.)

K. Nakashima and R. Kurazume. Learning to Drop Points for LiDAR Scan Synthesis. In *Proc. IEEE International Conf. on Intelligent Robots and Systems (IROS)*, pages 222–229, 2021. (Cited on pages 37 and 79.)

H. Nam and H.-E. Kim. Batch-Instance Normalization for Adaptively Style-Invariant Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2563–2572, 2018. (Cited on page 39.)

A. Ng and M. Jordan. On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2001. (Cited on page 14.)

A. Odena, C. Olah, and J. Shlens. Conditional Image Synthesis with Auxiliary Classifier GANs. In *Proceedings of the International Conference on Machine Learning*, pages 2642–2651, 2017. (Cited on pages 108, 109, and 112.)

C. Olsson, S. Bhupatiraju, T. Brown, A. Odena, and I. Goodfellow. Skill Rating for Generative Models. *arXiv.org*, 2018. (Cited on pages 49 and 51.)

S. J. Pan and Q. Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 1345–1359, 2010. (Cited on page 33.)

J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019a. (Cited on page 108.)

T. Park, M. Liu, T. Wang, and J. Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2332–2341, 2019b. (Cited on page 46.)

G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez. Invertible Conditional GANs for image editing. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2016. (Cited on page 109.)

F. Piewak, P. Pinggera, M. Schäfer, D. Peter, B. Schwarz, N. Schneider, D. Pfeiffer, M. Enzweiler, and M. Zöllner. Boosting LiDAR-based Semantic Labeling by Cross-Modal Training Data Generation. In *Proc. of the European Conf. on Computer Vision (ECCV) Workshops*, pages 497–513, 2018. (Cited on pages 14, 17, and 81.)

F. Piewak, P. Pinggera, and M. Zöllner. Analyzing the Cross-Sensor Portability of Neural Network Architectures for LiDAR-based Semantic Labeling. In *Proc. IEEE Conf. on Intelligent Transportation Systems (ITSC)*, pages 3419–3426, 2019. (Cited on page 36.)

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017a. (Cited on pages 13, 39, 88, 108, 110, and 144.)

C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5105–5114, 2017b. (Cited on pages 13, 53, 54, 108, and 133.)

A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv.org*, 2016. (Cited on pages 49 and 50.)

E. Raff and J. Sylvester. Gradient Reversal against Discrimination: A Fair Neural Network Learning Approach. In *Proc. IEEE International Conf. on Data Science and Advanced Analytics (DSAA)*, pages 189–198, 2018. (Cited on pages 53, 55, and 56.)

J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv.org*, 2018. (Cited on page 79.)

D. Rethage, J. Wald, J. Sturm, N. Navab, and F. Tombari. Fully-Convolutional Point Networks for Large-Scale Point Clouds. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 625–640, 2018. (Cited on page 13.)

E. Richardson and Y. Weiss. On GANs and GMMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5852–5863, 2018. (Cited on pages 49 and 50.)

C. B. Rist, M. Enzweiler, and D. M. Gavrila. Cross-Sensor Deep Domain Adaptation for LiDAR Detection and Segmentation. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1535–1542, 2019. (Cited on pages 36 and 39.)

C. B. Rist, D. Schmidt, M. Enzweiler, and D. M. Gavrila. SCSSnet: Learning Spatially-Conditioned Scene Segmentation on LiDAR Point Clouds. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1086–1093, 2020. (Cited on page 13.)

C. B. Rist, D. Emmerichs, M. Enzweiler, and D. M. Gavrila. Semantic Scene Completion using Local Deep Implicit Functions on LiDAR Data. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1–1, 2021. (Cited on pages 86, 87, 88, 89, 92, 94, 101, and 102.)

A. Royer, K. Bousmalis, S. Gouws, F. Bertsch, I. Mosseri, F. Cole, and K. Murphy. XGAN: Unsupervised Image-to-Image Translation for Many-to-Many Mappings. *Domain Adaptation for Visual Understanding*, pages 33–49, 2020. (Cited on page 37.)

S. Royo and M. Ballesta-Garcia. An Overview of Lidar Imaging Systems for Autonomous Vehicles. *Applied Sciences*, 2019. (Cited on page 41.)

SAE International. SAE Levels of Driving Automation Refined for Clarity and International Audience, 2021. URL https://www.sae.org/blog/sae-j3016-update. Accessed 2022/06/13. (Cited on page 2.)

K. Saito, S. Yamamoto, Y. Ushiku, and T. Harada. Open Set Domain Adaptation by Backpropagation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 156–171, 2018. (Cited on page 43.)

K. Saleh, A. Abobakr, M. Attia, J. Iskander, D. Nahavandi, and M. Hossny. Domain Adaptation for Vehicle Detection from Bird's Eye View LiDAR Point Cloud Data. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops*, 2019. (Cited on pages 37, 42, 47, and 84.)

T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2234–2242, 2016. (Cited on pages 46, 49, and 50.)

A. E. Sallab, I. Sobh, M. Zahran, and N. Essam. LiDAR Sensor modeling and Data augmentation with GANs for Autonomous driving. In *Proc. of the International Conf. on Machine learning (ICML) Workshops*, 2019a. (Cited on pages 37, 42, 47, and 84.)

A. E. Sallab, I. Sobh, M. Zahran, and M. Shawky. Unsupervised Neural Sensor Models for Synthetic LiDAR Data Augmentation. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2019b. (Cited on pages 37, 42, and 84.)

C. Saltori, S. Lathuiliére, N. Sebe, E. Ricci, and F. Galasso. SF-UDA3D: Source-Free Unsupervised Domain Adaptation for LiDAR-Based 3D Object Detection. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2020. (Cited on pages 39 and 129.)

S. Santurkar, L. Schmidt, and A. Madry. A Classification-Based Study of Covariate Shift in GAN Distributions. In *Proc. of the International Conf. on Machine learning (ICML)*, pages 4480–4489, 2018. (Cited on pages 49 and 50.)

Scale AI. PandaSet, 2020. https://pandaset.org. (Cited on pages 41 and 57.)

T. Shan, J. Wang, F. Chen, P. Szenher, and B. Englot. Simulation-based Lidar Super-resolution for Ground Vehicles. *Robotics and Autonomous Systems (RAS)*, page 103647, 2020. (Cited on pages 36 and 47.)

E. Shelhamer, J. Long, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, pages 640–651, 2017. (Cited on page 13.)

A. Shoshan, N. Bhonker, I. Kviatkovsky, and G. Medioni. GAN-Control: Explicitly Controllable GANs. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 14083–14093, 2021. (Cited on page 109.)

D. W. Shu, S. W. Park, and J. Kwon. 3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 3858–3867, 2019. (Cited on pages 46, 49, 50, 106, 109, 110, 112, 114, 118, 144, and 149.)

K. Sirohi, R. Mohan, D. Büscher, W. Burgard, and A. Valada. EfficientLPS: Efficient LiDAR Panoptic Segmentation. *IEEE Trans. on Robotics*, pages 1894–1914, 2022. (Cited on pages 3 and 13.)

Y. Song, C. Yang, Z. Lin, X. Liu, Q. Huang, H. Li, and C.-C. J. Kuo. Contextual-Based Image Inpainting: Infer, Match, and Translate. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 3–18, 2018. (Cited on page 108.)

T. Sorensen. A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species and Its Application to Analyses of the Vegetation on Danish Commons. *Biologiske Skrifter*, 1948. (Cited on page 21.)

A. Srivastava, L. Valkov, C. Russell, M. U. Gutmann, and C. Sutton. VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3310–3320, 2017. (Cited on page 48.)

T. Standley, A. R. Zamir, D. Chen, L. Guibas, J. Malik, and S. Savarese. Which Tasks Should Be Learned Together in Multi-task Learning? In *Proc. of the International Conf. on Machine learning (ICML)*, pages 9120–9132, 2020. (Cited on page 113.)

Statistisches Bundesamt. Trend in the number of persons killed in road traffic accidents, 2020. URL https://www.destatis.de/EN/Press/2020/02/PE20_061_46241.html. Accessed 2022/06/13. (Cited on page 1.)

H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539, 2018. (Cited on page 39.)

Y. Sun, Y. Wang, Z. Liu, J. E. Siegel, and S. E. Sarma. PointGrow: Autoregressively Learned Point Cloud Generation with Self-Attention. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 61–70, 2020. (Cited on page 106.)

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1701–1708, 2014. (Cited on pages 18 and 25.)

Y. Taigman, A. Polyak, and L. Wolf. Unsupervised Cross-Domain Image Generation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. (Cited on pages 37 and 109.)

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *Proc. of the International Conf. on Learning*

Representations (ICLR), pages 1–10, 2016. (Cited on pages 46, 49, and 50.)

H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6410–6419, 2019. (Cited on page 13.)

M. Toldo, A. Maracani, U. Michieli, and P. Zanuttigh. Unsupervised Domain Adaptation in Semantic Segmentation: A Review. *Technologies*, 2020. (Cited on page 42.)

I. O. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schölkopf. AdaGAN: Boosting Generative Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5430–5439, 2017. (Cited on pages 49, 50, and 72.)

L. T. Triess. Synthesizing realistic high-resolution LiDAR point clouds by upsampling with neural networks. Master's thesis, University of Stuttgart, 2018. Partially published in [Triess et al. 2019]. (Cited on page 68.)

L. T. Triess and D. Peter. Verfahren zur Generierung realistischer Karten von Strahlausfällen in simulierten LiDAR-Daten, 2021. Publication Date: 2021/07/08. DE. Patent DE102021002559 (A1). (Cited on page 80.)

L. T. Triess and C. B. Rist. Verfahren zum semantischen Segmentieren von ersten Sensordaten eines ersten Sensortyps, 2021. Filed: 2021/05/21. DE. Patent DE102021002684 (A1). Patent pending. (Cited on page 86.)

L. T. Triess and C. B. Rist. Computer-implementiertes Verfahren zum semantischen Segmentieren und computerimplementiertes Verfahren zum Trainieren eines computer-implementierten Algorithmus zur Bestimmung einer Szenensegmentierung, 2022. Filed: 2022/06/28. DE. Patent DE102022002324 (A1). Patent pending. (Cited on page 86.)

L. T. Triess, D. Peter, C. B. Rist, M. Enzweiler, and J. M. Zöllner. CNN-based synthesis of realistic high-resolution LiDAR data. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1512–1519, 2019. (Cited on pages 36, 48, 68, 69, 70, 74, and 179.)

L. T. Triess, D. Peter, C. B. Rist, and J. M. Zöllner. Scan-based Semantic Segmentation of LiDAR Point Clouds: An Experimental Study. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, pages 1116–1121, 2020. (Cited on pages 17, 19, 22, 23, 25, 26, and 132.)

L. T. Triess, M. Dreissig, C. B. Rist, and J. M. Zöllner. A Survey on Deep Domain Adaptation for LiDAR Perception. In *Proc. IEEE Intelligent Vehicles Symposium (IV) Workshops*, pages 350–357, 2021a. (Cited on pages 32, 33, 34, 36, 37, 38, 47, and 72.)

L. T. Triess, D. Peter, S. A. Baur, and J. M. Zöllner. Quantifying point cloud realism through adversarially learned latent representations. In *Proc. of the German Conference on Pattern Recognition (GCPR)*, pages 681–696, 2021b. (Cited on pages 46, 59, 60, 61, 62, 63, 64, 133, 136, 137, 138, 140, 141, and 142.)

L. T. Triess, D. Peter, and J. M. Zöllner. Semi-Local Convolutions for LiDAR Scan Processing. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2021c. (Cited on pages 17, 27, and 28.)

L. T. Triess, A. Bühler, D. Peter, F. B. Flohr, and J. M. Zöllner. Point Cloud Generation with Continuous Conditioning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 4462–4481, 2022a. (Cited on pages 106, 107, 111, 112, 115, 118, 119, 120, 121, 122, 146, 147, 150, 152, 154, 155, 156, and 157.)

L. T. Triess, C. B. Rist, D. Peter, and J. M. Zöllner. A Realism Metric for Generated LiDAR Point Clouds. *International Journal of Computer Vision (IJCV)*, 2022b. (Cited on pages 46, 49, 53, 57, 59, 60, 61, 62, 68, 73, 75, and 84.)

Y.-H. Tsai, W.-C. Hung, S. Schulter, K. Sohn, M.-H. Yang, and M. Chandraker. Learning to Adapt Structured Output Space for Semantic Segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7472–7481, 2018. (Cited on pages 35 and 102.)

J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger. Sparsity Invariant CNNs. In *Proc. of the International Conf. on 3D Vision (3DV)*, pages 11–20, 2017. (Cited on page 68.)

D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv.org*, 2016. (Cited on page 39.)

D. Valsesia, G. Fracastoro, and E. Magli. Learning Localized Generative Models for 3D Point Clouds via Graph Convolution. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2019. (Cited on pages 106 and 108.)

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. (Cited on page 18.)

T.-H. Vu, H. Jain, M. Bucher, M. Cord, and P. Pérez. ADVENT: Adversarial Entropy Minimization for Domain Adaptation in Semantic Segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2512–2521, 2019. (Cited on page 34.)

H. Wang, Z. Jiang, L. Yi, K. Mo, H. Su, and L. J. Guibas. Rethinking Sampling in 3D Point Cloud Generative Adversarial Networks. *arXiv.org*, 2020. (Cited on page 109.)

L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan. Graph Attention Convolution for Point Cloud Semantic Segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10288–10297, 2019a. (Cited on page 13.)

S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun. Deep Parametric Continuous Convolutional Neural Networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2589–2597, 2018. (Cited on page 13.)

Y. Wang, L. Zhang, and J. van de Weijer. Ensembles of Generative Adversarial Networks. In *Advances in Neural Information Processing Systems (NeurIPS) Workshops*, 2016. (Cited on pages 49 and 50.)

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing (TIP)*, pages 600–612, 2004. (Cited on page 49.)

Z. Wang, S. Ding, Y. Li, M. Zhao, S. Roychowdhury, A. Wallin, G. Sapiro, and Q. Qiu. Range Adaptation for 3D Object Detection in LiDAR. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops*, pages 2320–2328, 2019b. (Cited on page 39.)

G. Wilson and D. J. Cook. A Survey of Unsupervised Deep Domain Adaptation. *ACM Trans. on Intelligent System Technology*, pages 51:1–51:46, 2020. (Cited on pages 33 and 34.)

B. Wu, A. Wan, X. Yue, and K. Keutzer. SqueezeSeg: Convolutional Neural Nets with Recurrent CRF for Real-Time Road-Object Segmentation from 3D LiDAR Point Cloud. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 1887–1893, 2018. (Cited on pages 13, 14, and 17.)

B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer. SqueezeSegV2: Improved Model Structure and Unsupervised Domain Adaptation for Road-Object Segmentation from a LiDAR Point Cloud. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, pages 4376–4382, 2019. (Cited on pages 13, 27, 38, 39, 72, 74, 90, and 102.)

J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. (Cited on page 108.)

Y. Wu and K. He. Group Normalization. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 3–19, 2018. (Cited on page 39.)

S. Xiang and H. Li. On the Effects of Batch and Weight Normalization in Generative Adversarial Networks. *arXiv.org*, 2017. (Cited on pages 49 and 50.)

C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka. SqueezeSegV3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 1–19, 2020. (Cited on page 13.)

Y. Xu, F. He, B. Du, L. Zhang, and D. Tao. Self-Ensembling GAN for Cross-Domain Semantic Segmentation. *arXiv.org*, 2021. (Cited on page 46.)

Y. Yan, Y. Mao, and B. Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 2018. (Cited on page 106.)

C.-Y. Yang, C. Ma, and M.-H. Yang. Single-Image Super-Resolution: A Benchmark. In *Proc. of the European Conf. on Computer Vision (ECCV)*, pages 372–386, 2014. (Cited on page 68.)

G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan. PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 4540–4549, 2019. (Cited on page 106.)

J. Yang, A. Kannan, D. Batra, and D. Parikh. LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2017. (Cited on pages 49 and 50.)

L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A Scalable Active Framework for Region Annotation in 3D Shape Collections. In *ACM Trans. on Graphics*, pages 1–12, 2016. (Cited on pages 114 and 155.)

L. Yi, B. Gong, and T. Funkhouser. Complete & Label: A Domain Adaptation Approach to Semantic Segmentation of LiDAR Point Clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 15358–15368, 2021. (Cited on pages 36, 43, 86, 89, 99, 101, 102, 103, 143, 145, and 160.)

F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2016. (Cited on page 14.)

L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng. PU-Net: Point Cloud Upsampling Network. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2799, 2018. (Cited on page 68.)

H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 5908–5916, 2017. (Cited on page 48.)

Z. Zhang, Y. Song, and H. Qi. Decoupled Learning for Conditional Adversarial Networks. *arXiv.org*, 2018. (Cited on page 49.)

H. Zhao, R. T. D. Combes, K. Zhang, and G. Gordon. On Learning Invariant Representations for Domain Adaptation. In *Proceedings of the International Conference on Machine Learning*, pages 7523–7532, 2019a. (Cited on page 39.)

H. Zhao, L. Jiang, C.-W. Fu, and J. Jia. PointWeb: Enhancing Local Neighborhood Features for Point Cloud Processing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5560–5568, 2019b. (Cited on page 13.)

S. Zhao, Y. Wang, B. Li, B. Wu, Y. Gao, P. Xu, T. Darrell, and K. Keutzer. ePointDA: An End-to-End Simulation-to-Real Domain Adaptation Framework for LiDAR Point Cloud Segmentation. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, pages 3500–3509, 2021. (Cited on page 37.)

Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018. (Cited on pages 3, 13, and 39.)

Z. Zhou, H. Cai, S. Rong, Y. Song, K. Ren, W. Zhang, J. Wang, and Y. Yu. Activation Maximization Generative Adversarial Nets. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. (Cited on page 49.)

J.-Y. Zhu, T. Park, P. Isola, and A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, pages 2242–2251, 2017. (Cited on pages 16, 34, 37, 84, and 109.)