applied sciences

MDPI

Article

Using Auto-ML on Synthetic Point Cloud Generation

Moritz Hottong [1], Moritz Sperling [1] and Christoph Müller [1,2,*]

[1] Fraunhofer Institute for Physical Measurement Techniques IPM, 79110 Freiburg, Germany
[2] Faculty of Digital Media, Furtwangen University, 78120 Furtwangen, Germany
* Correspondence: christoph.mueller@ipm.fraunhofer.de; Tel.: +49-761-8857-236

Abstract: Automated Machine Learning (Auto-ML) has primarily been used to optimize network hyperparameters or post-processing parameters, while the most critical component for training a high-quality model, the dataset, is usually left untouched. In this paper, we introduce a novel approach that applies Auto-ML methods to the process of generating synthetic datasets for training machine learning models. Our approach addresses the problem that generating synthetic datasets requires a complex data generator, and that developing and tuning a data generator for a specific scenario is a time-consuming and expensive task. Being able to reuse this data generator for multiple purposes would greatly reduce the effort and cost, once the process of tuning it to the specific domains of each task is automated. To demonstrate the potential of this idea, we have implemented a point cloud generator for simple scenes. The scenes from this generator can be used to train a neural network to semantically segment cars from the background. The simple composition of the scene allows us to reuse the generator for several different semantic segmentation tasks. The models trained on the datasets with the optimized domain parameters easily outperform a model without such optimizations, while the optimization effort is minimal due to our Auto-ML approach. Although the development of such complex data generators requires considerable effort, we believe that using Auto-ML for dataset creation has the potential to speed up the development of machine learning applications in domains where high-quality labeled data is difficult to obtain.

Keywords: machine learning; Auto-ML; synthetic training data; data generator; domain parameter optimization; high quality labeled data; semantic segmentation; artificial neural networks

check for updates

Citation: Hottong, M.; Sperling, M.; Müller, C. Using Auto-ML on Synthetic Point Cloud Generation. *Appl. Sci.* **2024**, *14*, 742. https://doi.org/10.3390/app14020742

Academic Editor: Eui-Nam Huh

Received: 7 December 2023
Revised: 4 January 2024
Accepted: 12 January 2024
Published: 15 January 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Data is a crucial component of modern machine learning algorithms. However, obtaining high-quality data can be a challenging and time-consuming task, especially in areas where specific data is limited or difficult to collect. Synthetic data generation is a promising approach that can address this challenge by generating large amounts of task-specific data without the need for expensive field measurements [1]. However, using synthetically trained machine learning models for real-world tasks often suffers from the domain gap problem, which leads to a loss of model performance. Finding optimal simulation parameters is a time-consuming and iterative process of setting parameters, generating data, training on the dataset, and determining the model's performance.

To address this problem, we propose the integration of automated machine learning (Auto-ML) techniques to enhance the efficiency of synthetic data generation. Specifically, an Auto-ML system is introduced for automating the adaptation of training data by fine-tuning simulator parameters. The aim is to generate optimized synthetic data that can effectively substitute real data in the training for semantic segmentation in point clouds. This approach follows the principles of Auto-ML, seeking to automate the dataset generation process, analogous to its role in streamlining hyperparameter tuning in machine learning algorithms. By employing Auto-ML techniques in the synthetic data generation process, the we aim to reduce the reliance on extensive human intervention to identify optimal simulation parameter settings. The primary research questions center around the

feasibility of automating synthetic point cloud generation for semantic segmentation, the identification of advantageous simulator parameters, and the evaluation of performance using optimized synthetic data. Notably, the system is implemented using Unreal Engine 5 [2], a sophisticated game development platform, and Open3D/Open3D-ML [3], a versatile toolkit for point cloud processing and machine learning tasks. The reference real-world test dataset is a set of subsamples from the Hessigheim3D LiDAR dataset [4].

1.1. Data Generators and Synthetic Training Data

Synthetic training data is data generated by a computer program and not collected from real sources. Synthetic data can be used to create training datasets, fill gaps in data, and increase the diversity of the data. Due to progress in computer graphics, simulation environments gather a growing interest for industrial and scientific purposes and are recently used in a variety of applications including object detection [5–7], semantic segmentation [8,9], data augmentation [10,11], pose estimation [12,13] or robot control [14,15]. Current game engines such as Unreal Engine provide state-of-the-art technologies for realistic visualization and physics simulation that can be used to produce almost photo-realistic data.

A domain gap is a mismatch between the distribution of data in the synthetic dataset and the distribution of data in the real-world dataset. This can lead to poor performance when a synthetically trained model is applied to real-world data. Domain adaptation techniques are used to address this issue by modifying the synthetic data to better match the distribution of the real-world data. Some common domain adaptation techniques include transfer learning [16], data augmentation, domain randomization and generative adversarial networks (GANs) [17]. For our approach we chose to generate synthetic data with variations of real-world properties of point clouds to better fit the target distribution. The parameters of these properties are being tuned with the presented Auto-ML technique. However, there are overlapping approaches to synthetic 2D image generation. One of them is Meta-Sim, a method proposed for automatically generating synthetic labeled datasets relevant for a downstream task [18]. The method learns a generative model of synthetic scenes and obtains images as well as their corresponding ground-truth via a graphics engine. The proposed method aims to improve the synthetic scene content in a dataset over a human-engineered probabilistic scene grammar. Another approach is Structured Domain Randomization (SDR), which adds context to the generated data by randomly placing objects and distractors according to specific problem probability distributions [19]. SDR generates images that allow neural networks to consider the context around an object during detection. We argue that the aforementioned work is limited to optimize the synthetic scene content, but to successfully bridge the domain gap, more domain characteristics such as lighting, coloring, or sensor behaviours are relevant.

1.2. Auto-ML

Auto-ML, or automated machine learning, refers to the process of automating the design and implementation of machine learning models. Auto-ML tools can perform tasks such as selecting the appropriate algorithm, tuning hyperparameters, and optimizing the architecture of the model. This can save time and reduce the need for specialized expertise in machine learning.

Auto-ML tools typically use a combination of search techniques and optimization algorithms to explore the space of possible models and parameters to find the best one for a given task. These tools can also incorporate techniques such as ensemble learning and transfer learning to improve the performance of the resulting models. A good overview and detailed description of state-of-the-art Auto-ML techniques can be found in [20–22].

While approaches such as the automatic selection of algorithms, hyperparameters and model architecture significantly increase the effectiveness of Auto-ML methods, it is important to emphasise that the availability of high quality training data is also crucial for optimal model performance. Consequently, the success of Auto-ML is closely linked

to the quality of the training data and the availability of a sufficient amount of it. To our knowledge, current Auto-ML paradigms do not appear to include procedures for generating new data for tasks without a prior training set, limiting the possibilities for automated, task-specific data generation that would further improve model performance.

1.3. Auto-ML System Overview

Figure 1 presents the idea of an Auto-ML system that includes data generation as an optimization loop. At the start of the system, a defined set of labeled synthetic scenes is generated by a data generator. The scenes are constructed using simulator settings from a defined parameter search space. The parameter search space describes the synthetic domain. After training a semantic segmentation model, the performance is evaluated on a test dataset. Based on the evaluation results, an optimizer decides on new values for the simulator settings, and the data generation process is restarted. When the Auto-ML system finishes, it outputs the combination of domain parameters that produced the best model performance on the real-world test dataset. With the proposed method Auto-ML approaches would be complemented by adding automated, task-specific data generation with the goal of improving model performance.

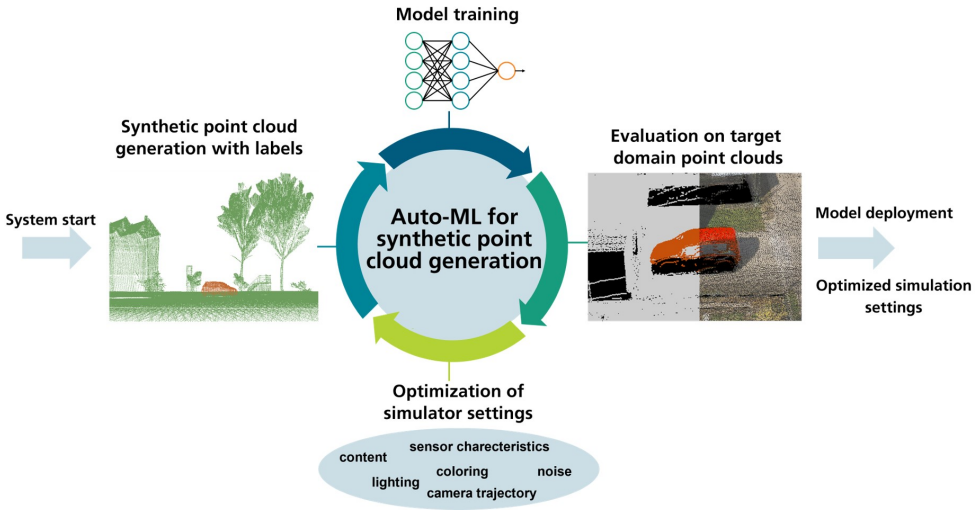


Figure 1. Visualization of the complete Auto-ML process loop. The gray arrows indicate the entry and exit point of the loop. The circle with arrows indicates the sequence of actions during an iteration, including data generation, model training, model evaluation and optimization.

2. Materials and Methods

2.1. System Description

Our system is based on an existing machine learning pipeline for semantic segmentation (Figure 1, step of model training) and extended with self-developed Auto-ML functionalities and communication interfaces (Figure 1, steps of: evaluation on target domain point clouds, optimization of simulator settings and synthetic point cloud generation with labels).

The machine learning pipeline used for semantic segmentation is provided by Open3D-ML [23]. We use the TensorFlow version of the semantic segmentation pipeline with a RandLANet model for semantic segmentation in point clouds. The functionality of the pipeline remains unchanged throughout the research. It is extended by a controller script that acts as an interface coordinating the different components of the system.

The synthetic point clouds with labels are generated using an Unreal Engine 5 project that includes a simulation platform for generating synthetic point clouds for urban environments.

Figure 2 shows the workflow of the Unreal Engine data generator. At system startup, the parameter file contains random or preset domain parameters. The parameter file is frequently checked by an Unreal Engine data generator process. When the process reads new parameters, the simulator settings are set accordingly and the synthetic data generation

begins. For point cloud generation, Unreal Engine provides simulated sensors such as LiDAR, radar and ultrasound for sensor data in modelled scenes. With the LiDAR point cloud plugin [24], it becomes a powerful tool for importing, editing, and saving point clouds to create custom synthetic point cloud data for machine learning. When a scene is processed, the point cloud is saved to an external dataset folder. Upon reaching a scene limit, an Unreal Engine process writes the name of the generated dataset to an external dataset logfile.

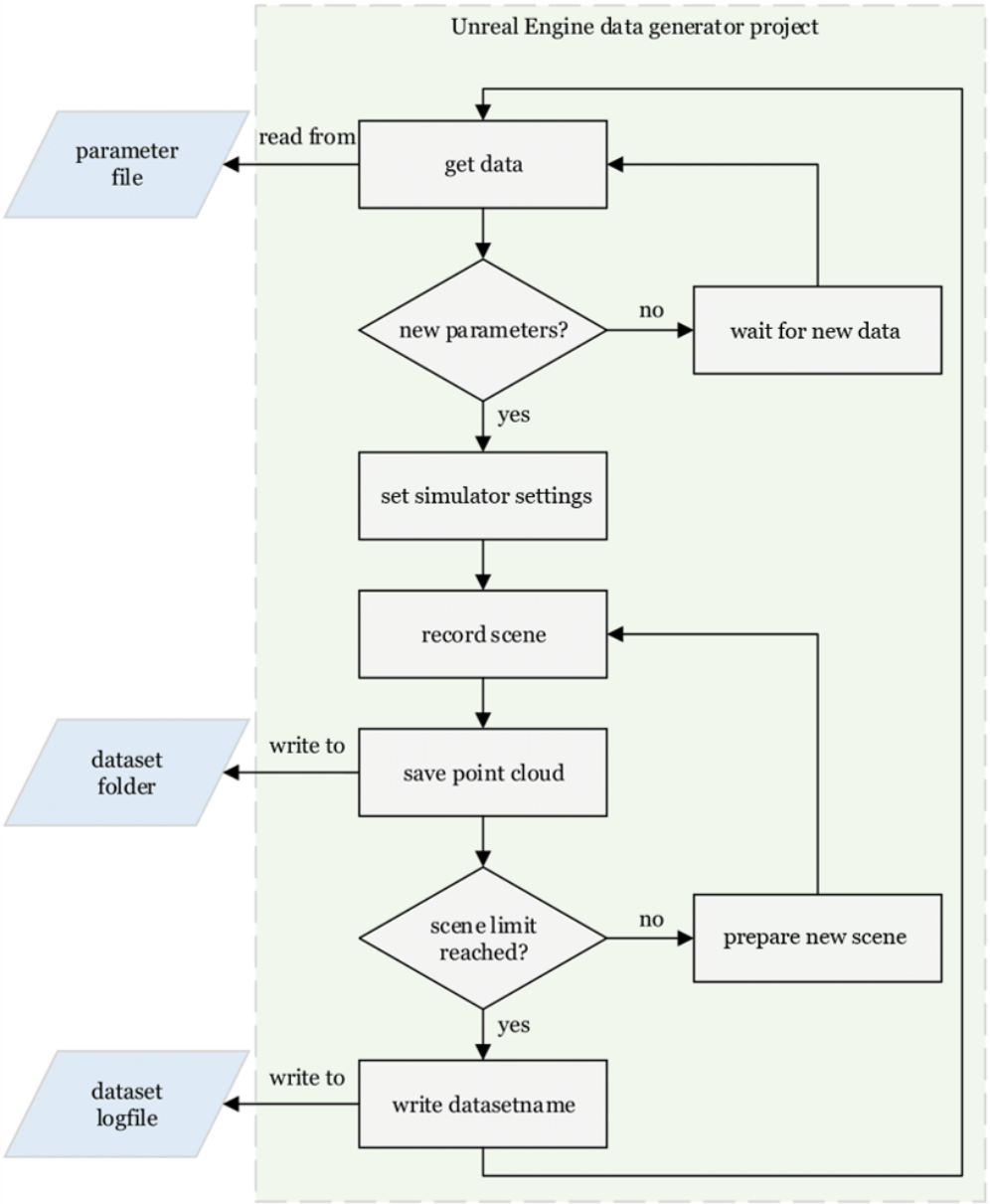


Figure 2. Flowchart of the Unreal Engine data generation process.

The developed controller script frequently checks the dataset logfile for updates, and if a new dataset is available, the data is processed and model training begins using the Open3D-ML pipeline. After the training process, the model's performance is determined on a real-world test dataset. When the test is evaluated, the results are sent to the optimizer, which calculates a new parameter configuration from the search space. During the experiments, a random optimizer is used that randomly chooses domain parameter settings from the search space. The new settings are then saved to the external parameter file supervised by the data generator and a new data generation with new simulator settings

is triggered. The system stops when the optimizer reaches the stopping criterion or if all possible parameter combinations have been used for data generation. Stopping criteria can be set in the optimizer. They can be either a fixed number of iterations in the optimization process or a performance threshold. The output of the system is the combination of domain parameter settings that resulted in the best model performance on the real-world test set, as well as the name of the best model and several log files. Files such as the domain parameter settings file or log files are formatted in JSON file format.

2.2. Experiments

The goal of the experiments is to demonstrate that the developed system is capable of automatically finding simulation parameter settings to generate useful synthetic training data. The target domain for which the optimization is to be performed is represented by subsamples of the Hessigheim3D dataset. The task of the network is to perform semantic segmentation of cars in point clouds. Consequently, the class car IoU is the performance metric of concern. To evaluate and compare the quality of the synthetic data generated, the same machine learning pipeline with the same settings is used for all experiments performed. During the experiments, three characteristic parameters describing the synthetic domain are investigated.

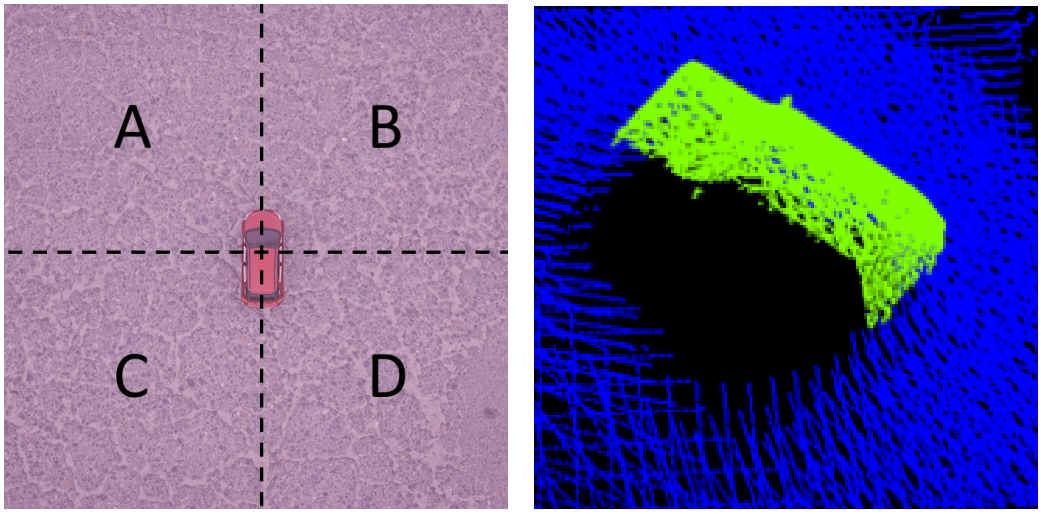
Table 1 states the domain parameters to investigate and their range of values. The parameters are limited to have two discrete values each. Since we want to show that not only the scene content is relevant for the model performance, we decided to investigate two other parameters describing the domain. The *Objects in Environment* parameter determines whether there are other objects in the scene besides a car (Figure 3). The *Angular Loss* parameter represents the phenomenon in which certain camera angles could not be captured (Figure 4). The *Camera Trajectory* parameter defines what kind of camera trajectory is used when capturing a scene with the virtual LiDAR (Figure 5).

Table 1. Domain parameters and their value range

Parameter	1st Value	2nd Value
Objects in Environment	False	True
Angular Loss	off	on
Camera Trajectory	grid	spiral

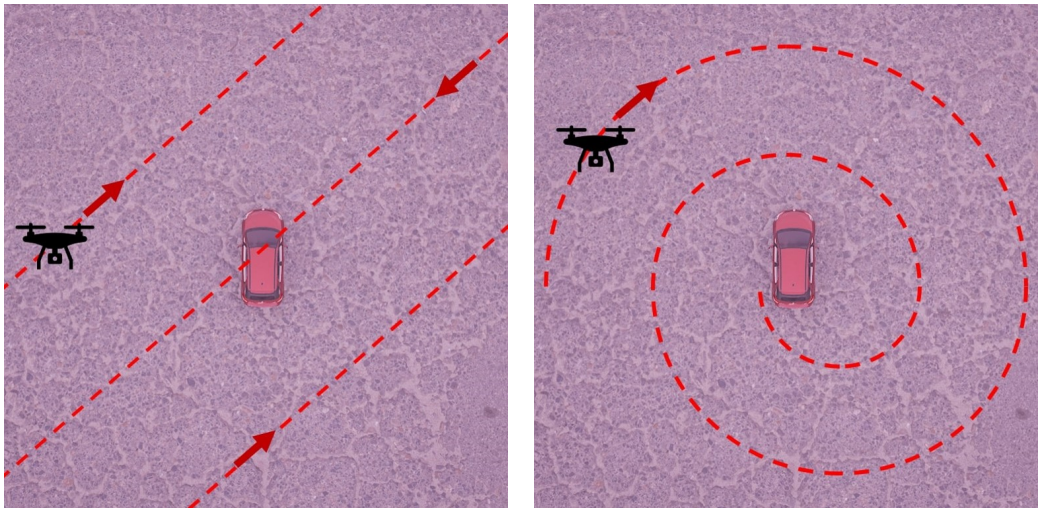
(**a**) (**b**)

Figure 3. Visualization of the *Objects in Environment* parameter. (**a**) *Objects in Environment* with value *False*; (**b**) *Objects in Environment* with value *True*.



(**a**) (**b**)

Figure 4. *Angular Loss* parameter. (**a**) *Angular Loss* space declaration. Division of the synthetic environment into quadrants A–D, with origin in the center of the car. One to three sectors are blocked if parameter is set to *on*; (**b**) Occuring shadow effects of setting *Angular Loss* to *on*.



(**a**) (**b**)

Figure 5. Visualization of the implemented *Camera Trajectory* settings. The red dashed lines and the arrows indicate the trajectory of the camera. (**a**) Camera with *grid* trajectory; (**b**) Camera with *spiral* trajectory.

In a first experiment, the system is used to examine the entire parameter space resulting from Table 1. Since the training data can differ in content and scene arrangement even when using the same domain parameter settings, the performance of the resulting models can differ. To make a more general statement, the system is set up to generate ten datasets per domain parameter combination. For each dataset, a semantic segmentation model is trained and tested on data from the Hessigheim3D target domain. During runtime, the performance of the system can be monitored through several log files. On system exit, the random optimizer checks its memory, then sets and displays the best found domain parameter settings. In a second experiment, a model trained on the best found synthetic dataset competes with a model trained on real data from the Hessigheim3D target domain. The synthetic model uses 60 synthetic scenes for model training, while the real trained model uses 43 subsamples of the Hessigheim3D point clouds. Due to the

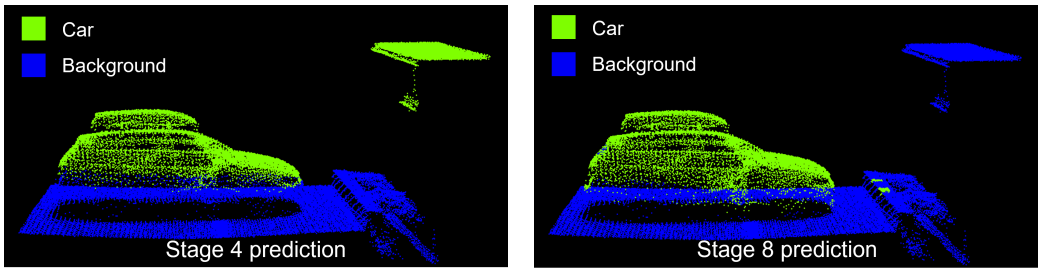
performance scatter, 50 models are trained for each approach and the best ones are selected for comparison.

3. Results and Discussion

Table 2 states the results of the first experiment. It shows the parameters chosen for data generation as well as the corresponding performance with confidence. On system exit, the random optimizer displays the best domain parameter settings, the resulting model, and some statistics. The best found setting for the investigated domain parameters leads to an average model performance of 0.806 with respect to the class car IoU. Synthetic training data containing several objects in the simulated scenes are generally more useful than synthetic training data containing only cars. Setting the virtual LiDAR to a spiral trajectory is more appropriate than using a grid trajectory. The best setting of the Angular Loss parameter depends on the setting of the other domain parameters. Setting Objects in Environment to True will cause models to perform more differently than setting the parameter to False. The domain parameters under investigation have an impact on the performance of a model of up to 53% with respect to the class car IoU. Figure 6 shows the prediction results on a test scene of the Hessigheim3D dataset for two models trained on synthetic data generated with different domain parameter settings.

Table 2. Stages of the Auto-ML system

Stage	Objects in Environment	Angular Loss	Camera Trajectory	Mean Car-IoU	Confidence IoU (p = 0.95)
1	False	off	grid	0.528	±0.005
2	False	off	spiral	0.547	±0.005
3	False	on	grid	0.530	±0.003
4	False	on	spiral	0.548	±0.005
5	True	off	grid	0.727	±0.028
6	True	off	spiral	0.770	±0.020
7	True	on	grid	0.721	±0.021
8	True	on	spiral	0.806	±0.026



(**a**) (**b**)

Figure 6. Prediction of models trained with different synthetic training data on a Hessigheim3D test scene. The incorrectly labeled roof part in the stage 4 prediction leads to a poor recognition result. (**a**) Model trained with synthetic data generated with setting from stage 4. Car-IoU: 0.542; (**b**) Model trained with synthetic data generated with setting from stage 8. Car-IoU: 0.936.

Figure 6 highlights the need for a system that automatically finds suitable synthetic domain parameter settings from a given search space. It can be clearly seen that the selection of appropriate domain parameter settings for synthetic data generation is critical for achieving good model performance. Because of performance scatter among models trained with the same domain parameter settings, the optimizer can only relate domain parameter settings to a performance range. The range is determined by the Gaussian distribution of the performances of models trained with the same domain parameter settings.

Figure 7 shows the mapping of domain parameter settings to a performance range. If domain parameter settings are mapped to overlapping performance ranges this can

potentially affect the traceability of parameter settings that are suitable for data generation. The best solution found may then no longer be described by a single combination of parameter settings, but rather by a set of appropriate parameter settings.

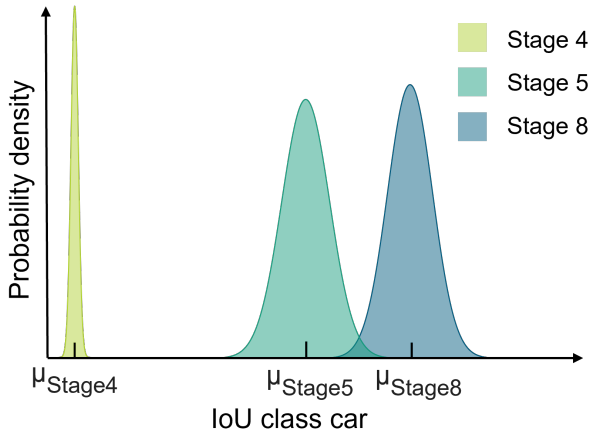


Figure 7. Visualization of the performance scatter

Table 3 provides a summary of the results from the second experiment, which compares the performance of the best model trained with synthetic data to that of the best model trained with real data. Both models underwent training with augmentation techniques, including scaling, rotation, added noise, and color variations. For each test scene, the model with the better class car IoU prediction is highlighted in green. In the conducted test the synthetically trained model performed 9.1% better than the model trained with data from the target domain. Thus, the approach of using the Auto-ML system to find suitable training data is superior to the approach of creating a model based on the existing real data for the given task. The overall performance of the synthetically trained model suffers mainly from the poor prediction on Scene 8 (Figure 8).

Table 3. Best performing synthetically trained model compared to the best performing real trained model. The model with the best performance per scene and the best overall performance is highlighted in green. The overall performance (in bold) is the average across all scenes.

Test Scene	Model Trained Synthetically Class Car IoU	Model Trained with Real Data Class Car IoU
1	0.983	0.566
2	0.967	0.928
3	0.973	0.803
4	0.974	0.962
5	0.934	0.915
6	0.951	0.857
7	0.745	0.917
8	0.395	0.742
9	0.933	0.585
10	0.714	0.713
11	0.993	0.984
12	0.942	0.748
13	0.924	0.797
Overall	**0.879**	**0.809**

This is the only test scene with significant uneven terrain. Since uneven terrain is not taken into account in the synthetic data generator, the scene may be difficult for the network to predict because the training data did not contain this environmental property.

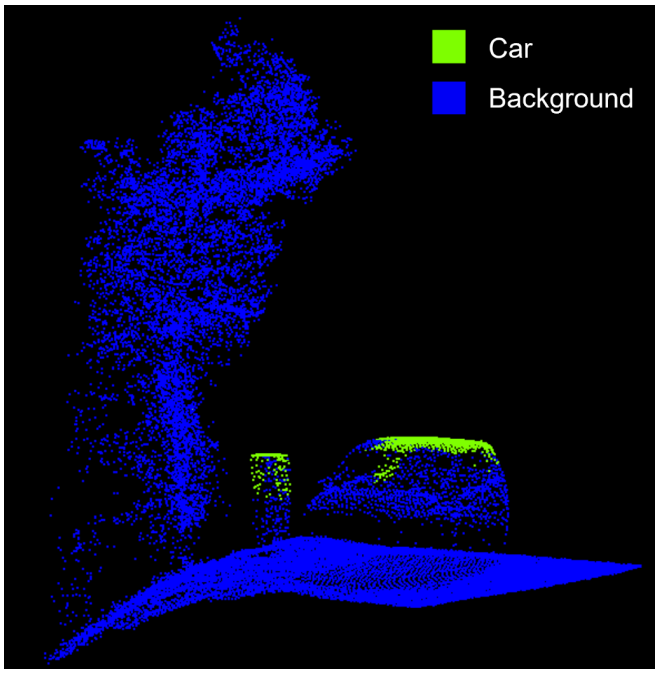


Figure 8. Hessigheim3D subsample—test scene 8 prediction.

4. Conclusions

The experiments demonstrate the immense impact that the choice of synthetic training data has on model performance in the real-world domain. The developed system finds domain parameter setting to generate useful synthetic data and thus successfully adapts to the given target domain. Determination of simulation environment parameters with the developed system allows their reuse as a basis for generating data for similar domains. The experiments further demonstrate that models trained on the adapted data can outperform models trained with real-world data from the target domain. Determining precise domain parameter settings is limited by the scatter in model performance that occurs when data is generated with the same parameter settings. Compared to similar approaches, the system's capabilities go beyond optimizing the content of the synthetic training data. The system can also handle domain characteristics such as sensor behavior, illumination, or color in the same optimization process. Once set up, the developed system removes the human almost completely from the loop, requiring no manual intervention in the scene generation or training process.

The developed Auto-ML system follows the approach of domain adaptation via a simulator. Transfer Learning and Autoencoders are domain adaptation approaches that do not rely on the generation of new data for domain adaptation and are therefore not comparable to the developed system. Generative Adversarial Networks (GANs), on the other hand, adapt to a target domain by generating data. The developed system and GANs are both types of automated approaches. While GANs only automate the data generation part, the developed Auto-ML system is an end-to-end system that covers the entire process from data generation to model deployment. The main differences between GANs and the developed system in terms of data generation are that GANs are based on complex models for adapting data to another domain, while domain adaptation in the developed system is based on tuning simulator parameters. The advantage of the developed system is that the adaptation is comprehensible and traceable, since the parameter settings in the simulator can be uniquely assigned to properties of the domain. As a result, the found parameters suitable for describing a domain can be reused for related domains. Due to the black-box behavior of a GAN, the adaptation processes within a GAN are not comprehensible and difficult to trace. As the explainability of artificial intelligence has become a topic of

growing interest [25–27], the developed system solves the problem of explainability for domain adaptation tasks.

The domain randomization approach is based on using randomly generated environments as training data, so that the target domain is covered. However, domain randomization does not guarantee that the trained model will perform well in the target domain, since only a limited number of variations of the training domain can be generated [28]. If the target domain is not well covered by the training data, the model performance will decrease. In terms of scene generation, this is similar to the Auto-ML system's approach (when using a random optimizer), where the parameter settings for point cloud generation are randomly chosen from the search space. The main difference is that the Auto-ML system does not change the parameter settings during the generation of one dataset. Thus it is able to keep track of good parameter settings. As a result, the domain randomization approach does not allow for the precise determination of an advantageous parameter setting, and thus the reusability of parameter settings for related domains is lost. The Auto-ML system, on the other hand, is able to determine the best possible parameter settings and thus is able to provide the found settings for tasks in related domains. When the *Objects in Environment* parameter is set to True, some randomization is added to the scenes within the generation of one dataset, but unlike the domain randomization approach, the randomization is limited to the position of the spawned objects and can be traced to only one simulation parameter.

As the approach breaks new ground in the field of Auto-ML for synthetic point cloud generation, further research is needed to verify its ability to automatically generate suitable training data for other target domains. With the implementation of state-of-the-art optimizers, tasks with a larger parameter space could be investigated more efficiently than with the currently used random optimizer. Since the scatter of the model performance is the main limitation for precise parameter determination, an attempt should be made to minimize the scatter and thus increase the resolution of the system. As a result, the system could consider more domain parameters with a wider range of values to make the system more universally applicable.

Author Contributions: Conceptualization, M.S. and M.H.; methodology, M.H. and M.S.; software, M.H.; writing—original draft preparation, M.H. and M.S.; writing—review and editing, M.H. and C.M.; supervision, M.S. and C.M.; funding acquisition, C.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Fraunhofer Gesellschaft.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from other research projects and are available from the corresponding authors with the permission of the respective owner.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. de Melo, C.M.; Torralba, A.; Guibas, L.; DiCarlo, J.; Chellappa, R.; Hodgins, J. Next-generation deep learning based on simulators and synthetic data. *Trends Cogn. Sci.* **2022**, *26*, 174–187. [CrossRef] [PubMed]
2. Website of Unreal Engine. Available online: https://www.unrealengine.com/en-US/unreal-engine-5 (accessed on 11 January 2024).
3. Zhou, Q.Y.; Park, J.; Koltun, V. Open3D: A Modern Library for 3D Data Processing. *arXiv* **2018**, arXiv:1801.09847. Available online: http://arxiv.org/abs/1801.09847 (accessed on 3 January 2024).
4. Kölle, M.; Laupheimer, D.; Schmohl, S.; Haala, N.; Rottensteiner, F.; Wegner, J.D.; Ledoux, H. The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo. *ISPRS Open J. Photogramm. Remote. Sens.* **2021**, *1*, 100001. [CrossRef]
5. Shermeyer, J.; Hossler, T.; Van Etten, A.; Hogan, D.; Lewis, R.; Kim, D. Rareplanes: Synthetic data takes flight. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Virtual, 5–9 January 2021; pp. 207–217.
6. Abu Alhaija, H.; Mustikovela, S.K.; Mescheder, L.; Geiger, A.; Rother, C. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *Int. J. Comput. Vis.* **2018**, *126*, 961–972. [CrossRef]

7. Tremblay, J.; Prakash, A.; Acuna, D.; Brophy, M.; Jampani, V.; Anil, C.; To, T.; Cameracci, E.; Boochoon, S.; Birchfield, S. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–23 June 2018; pp. 969–977.
8. Chen, Y.; Li, W.; Chen, X.; Gool, L.V. Learning semantic segmentation from synthetic data: A geometrically guided input-output adaptation approach. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 1841–1850.
9. Sharma, S.; Ball, J.E.; Tang, B.; Carruth, D.W.; Doude, M.; Islam, M.A. Semantic segmentation with transfer learning for off-road autonomous driving. *Sensors* **2019**, *19*, 2577. [CrossRef]
10. Jaipuria, N.; Zhang, X.; Bhasin, R.; Arafa, M.; Chakravarty, P.; Shrivastava, S.; Manglani, S.; Murali, V.N. Deflating dataset bias using synthetic data augmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 772–773.
11. Bhattarai, B.; Baek, S.; Bodur, R.; Kim, T.K. Sampling strategies for GAN synthetic data. In Proceedings of the ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 2303–2307.
12. Doersch, C.; Zisserman, A. Sim2real transfer learning for 3d human pose estimation: Motion to the rescue. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 12929–12941.
13. Basak, S.; Corcoran, P.; Khan, F.; Mcdonnell, R.; Schukat, M. Learning 3D head pose from synthetic data: A semi-supervised approach. *IEEE Access* **2021**, *9*, 37557–37573. [CrossRef]
14. Devaranjan, J.; Kar, A.; Fidler, S. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 715–733.
15. Hämäläinen, A.; Arndt, K.; Ghadirzadeh, A.; Kyrki, V. Affordance learning for end-to-end visuomotor robot control. In Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Venetian Macao, Macau, 3–8 November 2019; pp. 1781–1788.
16. Kouw, W.M.; Loog, M. An introduction to domain adaptation and transfer learning. *arXiv* **2018**, arXiv:1812.11806.
17. Ferreira, A.; Li, J.; Pomykala, K.L.; Kleesiek, J.; Alves, V.; Egger, J. GAN-based generation of realistic 3D data: A systematic review and taxonomy. *arXiv* **2022**, arXiv:2207.01390.
18. Kar, A.; Prakash, A.; Liu, M.Y.; Cameracci, E.; Yuan, J.; Rusiniak, M.; Acuna, D.; Torralba, A.; Fidler, S. Meta-Sim: Learning to Generate Synthetic Datasets. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
19. Prakash, A.; Boochoon, S.; Brophy, M.; Acuna, D.; Cameracci, E.; State, G.; Shapira, O.; Birchfield, S. Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 7249–7255. [CrossRef]
20. He, X.; Zhao, K.; Chu, X. AutoML: A survey of the state-of-the-art. *Knowl. Based Syst.* **2021**, *212*, 106622. [CrossRef]
21. Yao, Q.; Wang, M.; Chen, Y.; Dai, W.; Li, Y.F.; Tu, W.W.; Yang, Q.; Yu, Y. Taking human out of learning applications: A survey on automated machine learning. *arXiv* **2018**, arXiv:1810.13306.
22. Chen, Y.W.; Song, Q.; Hu, X. Techniques for automated machine learning. *ACM SIGKDD Explor. Newsl.* **2021**, *22*, 35–50. [CrossRef]
23. Website Open3D 0.18.0 Documentation. Available online: https://www.open3d.org/docs/release/open3d_ml.html (accessed on 3 January 2024).
24. Website of the UE Lidar Pointcloud Plugin. Available online: https://docs.unrealengine.com/5.1/en-US/lidar-point-cloud-plugin-for-unreal-engine (accessed on 3 January 2024).
25. Gunning, D.; Stefik, M.; Choi, J.; Miller, T.; Stumpf, S.; Yang, G.Z. XAI—Explainable artificial intelligence. *Sci. Robot.* **2019**, *4*, eaay7120. [CrossRef] [PubMed]
26. Das, A.; Rad, P. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv* **2020**, arXiv:2006.11371.
27. Došilović, F.K.; Brčić, M.; Hlupić, N. Explainable artificial intelligence: A survey. In Proceedings of the 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 21–25 May 2018; pp. 0210–0215.
28. Scheiderer, C.; Dorndorf, N.; Meisen, T. Effects of domain randomization on simulation-to-reality transfer of reinforcement learning policies for industrial robots. In *Proceedings of the Advances in Artificial Intelligence and Applied Cognitive Computing: Proceedings from ICAI'20 and ACC'20*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 157–169.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.