Semantically Adversarial Driving Scenario Generation with Explicit Knowledge Integration

Wenhao Ding[1], Haohong Lin[1], Bo Li[2], Kim Ji Eun[3], and Ding Zhao[1]

[1] Carnegie Mellon University
{wenhaod,haohongl,dingzhao}@andrew.cmu.edu
[2] University of Illinois at Urbana-Champaign
lbo@illinois.edu
[3] Robert Bosch LLC
JiEun.Kim@us.bosch.com

Abstract. Generating adversarial scenarios, which have the potential to fail autonomous driving systems, provides an effective way to improve the robustness. Extending purely data-driven generative models, recent specialized models satisfy additional controllable requirements such as embedding a traffic sign in a driving scene by manipulating patterns *implicitly* in the neuron level. In this paper, we introduce a method to incorporate domain knowledge *explicitly* in the generation process to achieve the *Semantically Adversarial Generation (SAG)*. To be consistent with the composition of driving scenes, we first categorize the knowledge into two types, the property of objects and the relationship among objects. We then propose a tree-structured variational auto-encoder (T-VAE) to learn hierarchical scene representation. By imposing semantic rules on the properties of nodes and edges in the tree structure, explicit knowledge integration enables controllable generation. We construct a synthetic example to illustrate the controllability and explainability of our method in a succinct setting. We further extend to realistic environments for autonomous vehicles: our method efficiently identifies adversarial driving scenes against different state-of-the-art 3D point cloud segmentation models and satisfies the traffic rules specified as the explicit knowledge.

Keywords: Adversarial Generation, Autonomous Driving, Knowledge Integration

1 Introduction

According to the report published by the California Department of Motor Vehicle [1], there were at least five companies (Waymo, Cruise, AutoX, Pony.AI, Argo.AI) that made their autonomous vehicles (AVs) drive more than 10,000 miles without disengagement. It is a great achievement that current AVs succeed in normal cases trained by hundreds of millions of miles of training. However, we are still unsure about their safety and robustness in adversarial driving scenarios. For example, the perception system may fail to detect a surrounding vehicle that is partially blocked by another vehicle. Due to the fidelity and structure of driving scenarios, the biggest difficulty of generating such adversarial scenarios is incorporating traffic rules and physical laws to make the generation controllable.

The recent breakthrough in machine learning enables us to learn complex distributions with sophisticated models, which uncover the data generation process so as to realize controllable data generation [2, 64, 17]. Deep Generative Models (DGMs) [25, 36], approximating the data distribution with neural networks (NN), are representative methods to generate data targeting a specific style or category. However, existing controllable generative models focus on manipulating implicit patterns in the neuron or feature level. For instance, [8] dissects DGMs to build the relationship between neurons and generated data, while [54] interpolates in the latent space to obtain vectors that control the poses of objects. One main limitation is that they cannot explicitly incorporate semantic rules, e.g., cars follow the direction of lanes, which may lead to meaningless data that violates common sense. In light of the limitation, we aim to develop a structural generative framework to integrate explicit knowledge [16] during the generation process and thus control the generated scene to be compliant with semantic rules.

Driving scenes can be described with objects and their various relationships [6]. Thus, in this paper, we categorize the semantic knowledge that describes scenes into two types, where the first type denoted as *node-level knowledge* represents the properties of single objects, and the second type denoted as *edge-level knowledge* represents the relationship among objects. We observe that tree structure is highly consistent with this categorization for constructing scenes, where nodes of the tree represent objects and edges the relationship. By automatically controlling the tree structure during the generation, we explicitly integrate the *node-level* and *edge-level* knowledge.

In detail, we propose Semantically Adversarial Generation (SAG), a general framework shown in Figure 1. The framework contains two stages to separate the learning of data distribution of real-world driving scenes and the searching of adversarial scenarios with knowledge as constraints. In the *training stage*, we train a tree-structured generative model that parameterizes nodes and edges of trees with NN to learn the representation of structured data. In the *generation stage*, explicit knowledge is applied to different levels of the learned tree model to achieve knowledge-guided generation for reducing the performance of victim algorithms.

To verify SAG, we first construct a synthetic scene reconstruction example to illustrate its advantages and provide analysis on its controllability and explainability. With SAG, it is possible to generate natural scenes that follow semantic rules, e.g., boxes with the same color should be positioned close to each other. To demonstrate the practicality of SAG, we conduct extensive experiments on adversarial LiDAR scene generation against state-of-the-art 3D segmentation models. We show that our generated safety-critical driving scenarios successfully attack victim models and meanwhile follow the specified traffic rules. In addition, compared with traditional attack methods, scenes generated by our method achieve stronger adversarial transferability across different victim models. Our technical contributions are summarized below:

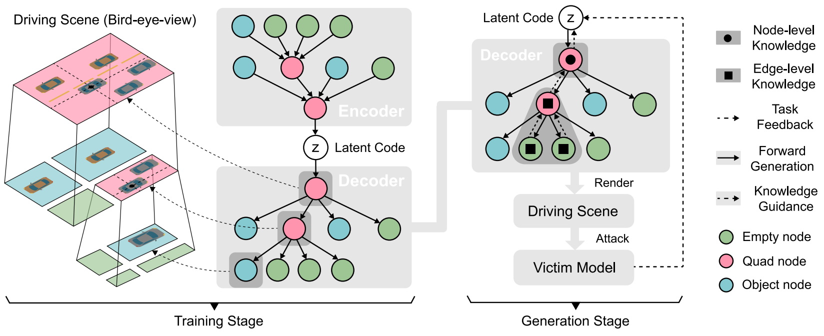

Fig. 1. Diagram of SAG. **Training stage.** Train the T-VAE model to learn the representation of structured data. **Generation stage.** Integrate node-level and edge-level knowledge during the generation to create adversarial samples for the victim model.

– We propose a semantically adversarial generative framework (SAG) via integrating explicit knowledge and categorizing the knowledge into two types according to the composition of scenes.
– We propose a tree-structured generative model based on our knowledge categorization and construct a synthetic example to demonstrate the effectiveness of our knowledge integration.
– We propose the *first* semantic adversarial point cloud attack based on SAG, named *Scene Attack*, against state-of-the-art segmentation algorithms, demonstrating several essential properties.

2 Semantically Adversarial Generation Framework

We define the driving scenario $x \in \mathcal{X}$ in the physical world, which contains a group of objects and their properties such as positions and colors. The goal of this framework is to create x so that to reduce the performance $\mathcal{L}_t(x)$ of the victim model $t \in \mathcal{T}$, as well as satisfying semantic loss $\mathcal{L}_{\mathbb{K}}(x)$:

$$x = \arg\min_{x} \mathcal{L}_t(x), \quad s.t. \;\; \mathcal{L}_{\mathbb{K}}(x) \leq 0, \tag{1}$$

where $\mathbb{K}$ is knowledge rules. Due to the structure of driving scenarios, it is usually difficult to directly search x in the data space, so we consider a generative model that creates x with learnable parameters.

In this section, we first describe the tree-based generative model for learning the hierarchical representations of x, which is important and necessary for applying knowledge to achieve semantic controllability (Section 2.1). Then we explain the two types of knowledge to be integrated into the generative model together with the generation stage that uses explicit knowledge as constraints (Section 2.2).

2.1 Tree-structured Variational Auto-encoder (T-VAE)

VAE [36] is a powerful model that combines auto-encoder and variational inference [10]. It estimates a mapping between data point $x \in \mathcal{X}$ and latent code $z \in \mathcal{Z}$ to find the low-dimensional manifold of the data space. The objective function of training VAE is to maximize a lower bound of the log-likelihood of training data, which is so-called Evidence Lower Bound (ELBO),

$$\mathrm{ELBO} = \mathbb{E}_{q(z|x;\phi)}\left[\log p(x|z;\theta)\right] - \mathbb{KL}(q(z|x;\phi)||p(z)), \tag{2}$$

where $\mathbb{KL}$ is Kullback–Leibler (KL) divergence. $q(z|x;\phi)$ is the encoder with parameters ϕ, and $p(x|z;\theta)$ is the decoder with parameters θ. The prior distribution of the latent code $p(z)$ is usually a Gaussian distribution for simplification of KL divergence calculation.

Tree structure design. One typical characteristic of natural scenes is that the data dimension varies with the variable number of objects. Thus, it is challenging to represent objects with a fixed number of parameters as in traditional models [36].

Graphs are commonly used to represent structured data [42] but are too complicated to describe the hierarchy and inefficient to generate. As a special case of graphs, trees naturally embed hierarchical information via recursive generation with depth-first-search traversal [31, 48]. This hierarchy is highly consistent with natural physical scenes and makes it easier to apply explicit knowledge, supported by previous works in cognition literature [47].

In this work, we propose a novel tree-structured generative model, which is inspired by the stick-breaking approach [57]: assume we have a stick with length W and we recursively break it into segments $w^{(n,i)}$ with:

$$W = w^{(1,1)} = w^{(2,1)} + w^{(2,2)} = \cdots = \sum_{i=1}^{K_n} w^{(n,i)}, \tag{3}$$

where (n,i) means the i-th segment of the n-th level. K_n is the total number of segments in the n-th level. The index starts from 1 and the entire stick has index $(1,1)$. The recursive function of breaking the stick follows

$$w^{(n+1,j)} = \alpha^{(n,i)} w^{(n,i)}, \quad w^{(n+1,j+1)} = (1-\alpha^{(n,i)}) w^{(n,i)}, \tag{4}$$

where $\alpha^{(n,i)} \in [0,1]$ is the splitting ratio for segment $w^{(n,i)}$. Segment $w^{(n+1,j)}$ is the first child of $w^{(n,i)}$ in the $(n+1)$-th level. Intuitively, this breaking process creates a tree structure where the i-th level is corresponding to the i-th layer of the tree and segments are nodes in the tree. We extend the above division to 2D space as shown in the left of Figure. 1. To generate a tree for driving scenarios, we define three types of nodes, namely Quad (generates 4 child nodes), Object (describes one kind of object), and Empty (works as a placeholder). When there is more than one object in the region, the Quad node is used to divide the region and expand the tree to one more depth. Since the expansion will always have four child nodes but not all nodes contain objects, the Empty node will be used for filling the empty region. If there is only one object in the region, the Object node is used to represent the property of this object and end the expansion of the tree. Different types of nodes can appear in the same layer and we follow Recursive Neural Networks (RvNN) [60] to build the tree structure recursively. Please refer to Appendix.B for a detailed example.

Model Implementation. Assuming there are M types of objects (including Quad and Empty) in the scene, we use a set of encoders $\{E_m\}_{m=1}^{M}$ and decoders $\{D_m\}_{m=1}^{M}$ to construct each kind of nodes in the entire N layer tree. $\{E_m\}_{m=1}^{M}$ create the *encoder tree* in a bottom-up manner and end at the latent variables z, while $\{D_m\}_{m=1}^{M}$ reconstruct the *decoder tree* in a top-down manner. The relationship between the n-th layer and the $(n+1)$-th layer in the *encoder tree* and the *decoder tree* are respectively:

$$\begin{aligned} f^{(n,i)} &= E_m([f^{(n+1,j)}, \cdots, f^{(n+1,j+3)}, g^{(n,i)}]; \phi_m), \\ [\hat{f}^{(n+1,j)}, \cdots, \hat{f}^{(n+1,j+3)}, \hat{g}^{(n,i)}] &= D_m(\hat{f}^{(n,i)}; \theta_m), \end{aligned} \tag{5}$$

where $f^{(n,i)}$ is named as the feature vector that passes the messages through the tree structure. $g^{(n,i)}$ is named as the property vector of node (n,i) that stories properties such as color of the object generated by node (n,i). In the bottom-up *encoder tree*, the selection of node type is accessible in the structured data, while in the top-down *decoder tree*, the selection does not have reference. A *Classifier* is required to determine the child node type $\hat{c}^{(n,i)}$:

$$\hat{c}^{(n,i)} = \text{Classifier}(f^{(n,i)}; \theta_c). \tag{6}$$

Between the encoders and decoders, the latent code z is sampled according to parameters $[z_\mu, z_\sigma]$, which are estimated by a *Sampler* by the reparameterization trick [10]:

$$[z_\mu, z_\sigma] = \text{Sampler}(f^{(1,1)}; \phi_s). \tag{7}$$

Then, we can summarize all model parameters with $q(z|x; \boldsymbol{\phi})$ and $p(x|z; \boldsymbol{\theta})$, where $\boldsymbol{\phi} = \{\phi_1, \cdots, \phi_m, \phi_s\}$ and $\boldsymbol{\theta} = \{\theta_1, \cdots, \theta_m, \theta_c\}$.

Model Training. The final connections of the encoders and decoders depend on the tree structure of the data point. Thus, the input scene x to the *encoder tree* can be represented by the node type c and property g of all nodes.

$$x = \{c, g\} = \{c^{(1,1)}, \cdots, c^{(N,K_N)}, \cdots, g^{(1,1)}, \cdots, g^{(N,K_N)}\}, \tag{8}$$

Correspondingly, the output from the *decoder tree* is $\hat{x} = \{\hat{c}, \hat{g}\}$ Assume c and g are conditionally independent given z, we get the objective of T-VAE following the ELBO of VAE (2):

$$\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \; \underbrace{\mathbb{E}_q\left[log\, p(c|z;\theta)\right]}_{-\mathcal{L}_C(\hat{c},c)} + \underbrace{\mathbb{E}_q\left[log\, p(g|z;\theta)\right]}_{-\mathcal{L}_R(\hat{g},g)} - \mathbb{KL}\left(\mathcal{N}(z_\mu, z_\sigma) \| \, \mathcal{N}(0, \mathbf{I})\right), \tag{9}$$

The $\mathcal{L}_C$ term represents the cross-entropy loss (CE) of the *Classifier*,

$$\mathcal{L}_C(\hat{c}, c) = \frac{1}{\sum_n^N K_n} \sum_{n=1}^{N} \sum_{i=1}^{K_n} \text{CE}(\hat{c}^{(n,i)}, c^{(n,i)}), \tag{10}$$

To make the *decoder tree* have the same structure as the *encoder tree*, we use Teacher Forcing [66] during the training stage. However, in the generation stage, we select the node with maximum probability as the child to expand the tree. Another term $\mathcal{L}_R$ uses mean square error (MSE) to approximate the log-likelihood of node property for all decoders,

$$\mathcal{L}_R(\hat{g}, g) = \sum_{m=1}^{M} \frac{1}{N_m} \sum_{n=1}^{N} \sum_{i=1}^{K_n} \mathbb{1}\left[c^{(n,i)} = m\right] \|\hat{g}^{(n,i)} - g^{(n,i)}\|_2^2, \tag{11}$$

where N_m is the times that node type m appears in the tree and $\mathbb{1}[\cdot]$ is the indicator function. In (11), we normalize the MSE with N_m instead of $\sum_n^N K_n$ to avoid the influence caused by imbalanced node type in the tree.

The advantage of this hierarchical structure is that the root node stores the global information, and other nodes only contain local information, making it easier for the model to capture the feature from multiple scales in the scene. In addition, this tree structure makes it possible to explicitly apply semantic knowledge in the generation stage, which will be explained in Section 2.2.

2.2 Knowledge-guided Generation

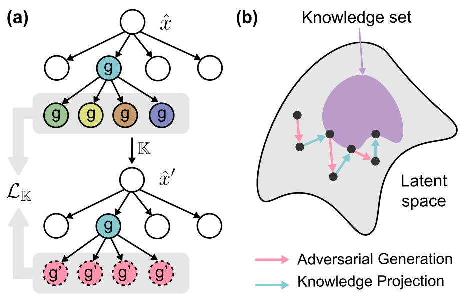

Fig. 2. (a) The knowledge integration example described in Section 2.2: the child nodes of the blue color node should be red. **(b)** Illustration of the knowledge-guided generation process by proximal optimization.

In this generation stage, we aim to create an adversarial scenario x to decrease $\mathcal{L}_t(x)$ by searching in the latent space of the *decoder tree* obtained in the previous training stage. Meanwhile, we also use the knowledge $\mathbb{K}$, representing traffic rules, to guide the searching with a low $\mathcal{L}_{\mathbb{K}}$. We formulate this process as a constraint optimization problem in the latent space that uses the knowledge $\mathcal{L}_{\mathbb{K}} \leq 0$ as constraints to minimize $\mathcal{L}_t(x)$. The general idea is shown in Figure. 2(b).

Knowledge representation. We first provide a formal definition to describe knowledge that we use in the decoder tree. Suppose there is a function set $\mathcal{F}$, where the function $f(A) \in \mathcal{F}$ returns true or false for a given input node A of a tree x. Then, we define the two types of propositional knowledge $\mathbb{K}$ for a particular victim model t using the first-order logic [59] as follows.

Algorithm 1: SAG Framework

```
Input: Dataset $\mathcal{D}$, Task loss $\mathcal{L}_t(x)$,
       budget $B$, Knowledge set $\mathbb{K}$
Output: Generated scene $\hat{x}$
Stage 1: Train T-VAE
    Initialize model parameters $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$
    for $x$ in $\mathcal{D}$ do
        Encode $z \leftarrow q(z|x; \boldsymbol{\phi})$
        Decode $\hat{x} \leftarrow p(x|z; \boldsymbol{\theta})$
        Update parameters $\{\boldsymbol{\theta}, \boldsymbol{\phi}\}$ by
          maximizing ELBO (9)
    end
Store the learned decoder $p(x|z; \boldsymbol{\theta})$
Stage 2: Knowledge-guided Generation
    Initialize latent code $z \sim \mathcal{N}(0, \mathbf{I})$
    while $B$ is not used up do
        if $\mathcal{L}_t(x)$ is differentiable then
            $z \leftarrow z - \eta \nabla \mathcal{L}_t(p(x|z; \boldsymbol{\theta}))$
        else
            $z \leftarrow$ Black-box Optimization
        end
        $\hat{x}' \leftarrow$ ApplyK($\mathbb{K}$, $\hat{x}$), $\hat{x} \leftarrow p(x|z; \boldsymbol{\theta})$
        $z \leftarrow \mathbf{prox}_{\mathcal{L}_{\mathbb{K}}}(z, \hat{x}, \hat{x}')$ with (13)
    end
Decode the scene $\hat{x} = p(x|z; \boldsymbol{\theta})$
```

Definition 1 (Knowledge Set). *The node-level knowledge k_n is denoted as $f(A)$ for a function $f \in \mathcal{F}$, where A is a single node. The edge-level knowledge k_e is denoted as $f_1(A) \to \forall j \ f_2(B_j)$ for two functions $f_1, f_2 \in \mathcal{F}$, where we apply knowledge f_2 to all child nodes B_j of A. Then, The knowledge set is constructed as $\mathbb{K} = \{k_n^{(1)}, \cdots, k_e^{(1)}, \cdots\}$.*

In the tree context, k_n describes the properties of a single node, and k_e describes the relationship between the parent node and its children. Specifically, in order to satisfy $f(A)$ in k_n, we locate node A in the tree x and change the property vector from g to g'. Similarly, in order to satisfy $f_1(A) \to f_2(B_j)$, after traversing x to find node A that satisfy f_1 in k_e, we change the type vector from c to c' or the property vector from g to g' for all A's children so that $f_2(B_j)$ holds true. The reference vector c' and g' are pre-defined by the knowledge set $\mathbb{K}$. We summarized the process of applying knowledge in **Algorithm 2**.

Algorithm 2: Apply Knowledge

```
Input: $\mathbb{K}$, Decoder tree $\hat{x}$
Output: Modified decoder tree $\hat{x}'$
Function ApplyK($\mathbb{K}$, $\hat{x}$)
for each knowledge $k^{(n)} \in \mathbb{K}$ do
    $\hat{x}' \leftarrow$ modify $\hat{x}$ according to $k^{(n)}$
end
if $x$ has child nodes then
    for all child nodes $\hat{x}_i$ of $\hat{x}$ do
        $\hat{x}'_i \leftarrow$ ApplyK($\mathbb{K}$, $\hat{x}_i$)
        Add node $\hat{x}'_i$ as a child to $\hat{x}'$
    end
end
return $\hat{x}'$
```

One running example is that the explicit knowledge described as *"if one node represents blue, its child nodes should represent red"* is implemented by the following operations. Starting from the root, we find all nodes whose colors are blue and collect the property vectors g of its child nodes; then we change g to g', representing the red color. This process is illustrated in Figure 2(a).

Adversarial Generation. To minimize the adversarial loss and satisfy the constraints of knowledge, we combine them to the new objective $\mathcal{L}(x) = \mathcal{L}_t(x) + \mathcal{L}_{\mathbb{K}}$, where the second term represents the mismatch between the original decoder tree $\hat{x}$ and the modified tree $\hat{x}'$ (shown in Figure 2(a)):

$$\mathcal{L}_{\mathbb{K}}(\hat{x}, \hat{x}') = \mathrm{MSE}(\hat{g}_i, \hat{g}_i') + \mathrm{CE}(\hat{c}_i, \hat{c}_i'), \quad \forall \hat{x}_i \neq \hat{x}_i' \tag{12}$$

Usually, $\mathcal{L}_t(x)$ requires large computations, while $\mathcal{L}_{\mathbb{K}}$ is efficient to evaluate since it only involves the inference of $p(x|z;\boldsymbol{\theta})$. Therefore, we resort to Proximal algorithms [52], which alternatively optimize $\mathcal{L}_t(x)$ and $\mathcal{L}_{\mathbb{K}}$. In our setting, the explicit knowledge is treated as the trusted region to guide the optimization of $\mathcal{L}_t(x)$. The knowledge loss and adversarial objective are alternatively optimized under the proximal optimization framework as shown in Figure 2(b). In the step of optimizing $\mathcal{L}_t(x)$ (pink arrow), we can either use gradient descent for a differentiable $\mathcal{L}_t(x)$ or change to black-box optimization methods [7] when $\mathcal{L}_t(x)$ is non-differentiable. Then, in the step of the optimizing $\mathcal{L}_{\mathbb{K}}$ (blue arrow), we use **Algorithm 2** to get the modified *decoder tree* $\hat{x}'$ and use the following proximal operator

$$z' = \mathbf{prox}_{\mathcal{L}_{\mathbb{K}}}(z, \hat{x}, \hat{x}') = \arg\min_{z'} \left(\mathcal{L}_{\mathbb{K}}(\hat{x}, \hat{x}') + \frac{1}{2}\|z - z'\|_2^2 \right), \tag{13}$$

to project the latent code z to z' so that $p(x|z';\boldsymbol{\theta})$ satisfies the knowledge rules. The second term in (13) is a regularize to make the projected point also close to the original point. The equation (13) can be solved by gradient descent since the decoder $p(x|z;\boldsymbol{\theta})$ of T-VAE is differentiable. In summary, The entire training and generation stages are shown in **Algorithm 1**.

3 Experiments

First, we design a synthetic scene to illustrate the controllability and explainability of the proposed framework. The synthetic physical scene provides a simplified setting to unveil the essence of the knowledge-guided generation. After that, we evaluate the performance of SAG on realistic driving scenes represented by point clouds. Based on SAG, we propose a new adversarial attack method, *Scene Attack*, against multiple point cloud segmentation methods.

3.1 Synthetic Scene Reconstruction

Task description. We aim to reconstruct a scene to match a given image as shown in Figure 4(a). The objective is the reconstruction error $\mathcal{L}_t(x) = \|S - \mathcal{R}(x)\|_2$, where $\mathcal{R}$ is a differentiable image renderer [35] and S is the image of target scene. Under this succinct setting, it is possible to analyze and compare the contribution of explicit knowledge integration since we can access the optimal solution, which usually cannot be obtained in an adversarial attack. According to the understanding of the target scene, we define three knowledge rules: ① The scene has at most two plates; ② The boxes that belong to the same plate should have the same color; ③ The boxes belong to the same plate should have distance smaller than a threshold γ.

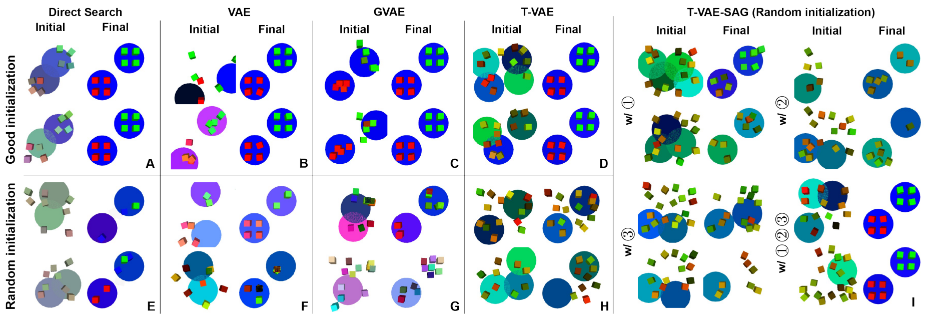

Fig. 3. **A-I** show results of synthetic scene reconstruction experiment from 5 methods with random and good initialization. **I** shows the results of T-VAE using SAG. With the combination of knowledge ①②③, we can almost reach the optimal solution even from a random initialization, while baseline methods can realize the target *only* when starting from the good initialization.

Experiment settings. We synthesize the dataset by randomly generating 10,000 samples with a varying number of boxes and plates. We also inject the target scene 10 times into the dataset to make it accessible to all models. We compare our method with the following baselines:

- Direct Search (DS) directly optimizes the positions and colors of boxes and plates in the data space.
- Direct Search with constraints (DS-C) modifies DS by adding knowledge constraints ①② to the objective function.
- VAE [36] is a well-known generative model that supports the latent space searching for sample generation.
- VAE-WR [64] simultaneously updates the shape of latent space during the searching process.
- SPIRAL [24] generates one object at one time to create the scene in an autoregressive manner.
- L2C [18] uses autoregressive structure to generate objects in the scene.
- Grammar-VAE (GVAE) [39] uses pre-defined rules (shown in Appendix.D) to generate the structural scene.
- T-VAE only uses the tree structure to build the model; the searching is done in the latent space without any knowledge integration.

Table 1. Reconstruction Error

Method	Initialization: Random	Initialization: Good
DS	86.0±9.4	**7.9±1.2**
DS-C	90.6±13.1	8.1±1.4
VAE [36]	110.4±10.6	13.4±6.1
VAE-WR [64]	105.9±24.6	13.2±8.4
SPIRAL [24]	95.2±21.9	23.6±5.5
L2C [18]	115.4±13.8	14.1±7.1
GVAE [39]	123.7±9.5	19.7±10.2

Among these methods, DS, DS-C, VAE, and VAE-WR need to access the number of boxes and plates in the target image (e.g., two plates and eight boxes) to fix the dimension of the input feature. To get the good initial points for DS and DS-C, we add a small perturbation to the positions and colors of all objects in the target scene. Similarly,

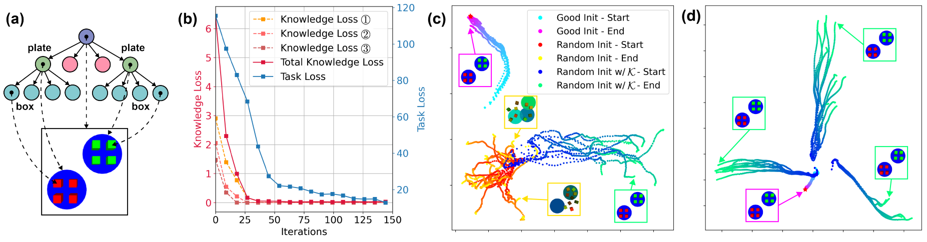

Fig. 4. (a) Target scene. **(b)** Knowledge losses of integrating semantic rules ①②③ separately. **(c)** The influence of knowledge to the trajectories with same initialization. **(d)** After applying explicit knowledge, optimization trajectories are diverse when start from different initialization.

we add the perturbation to the optimal latent code for other methods, which is obtained by passing the target scene to the encoder to get good initialization.

Evaluation results. We display the generated samples from five representative methods in Figure 3 and show the final errors of all methods in Table. 1. With good initialization, all models find a similar scene to the target one, while with the random initialization, all models are trapped in local minimums. However, obtaining good initialization is not practical in most real-world applications, indicating that this task is non-trivial and all models without knowledge cannot solve it. After integrating the knowledge into the T-VAE model, we obtain **I** of Figure 3. We can see that all three knowledge have positive guidance for the optimization, e.g., the boxes concentrate on the centers of plates with knowledge ③ When combining the three knowledge, even from a random initialization, our T-VAE can finally find the target scene, leading to a small error in Table. 1. We also want to mention that it is also possible to apply simple knowledge to GVAE during the generation. However, the advantages of our method are that (1) we can integrate any constraints as long as they can be represented by Definition 1. In contrast, GVAE can only apply hard constraints to objects with co-occurrences.

Analysis of knowledge and controllability. To analyze the contribution of each knowledge, we plot the knowledge losses of ①②③ in Figure 4(b) together with the adversarial loss. All knowledge losses decrease quickly at the beginning and guide the searching in the latent space. Next, we made ablation studies to explore why knowledge helps the generation. In Figure 4(b), we compare the optimization trajectories of T-VAE (red$\rightarrow$yellow) and T-VAE-SAG (blue$\rightarrow$green) with the same initialization. For T-VAE, the generated samples are diverse but totally different from the target scene, while for T-VAE-SAG, the trajectories go in another direction, and the generated samples are good. However, the interesting thing is that although the knowledge helps us find good scenes, it does not reach

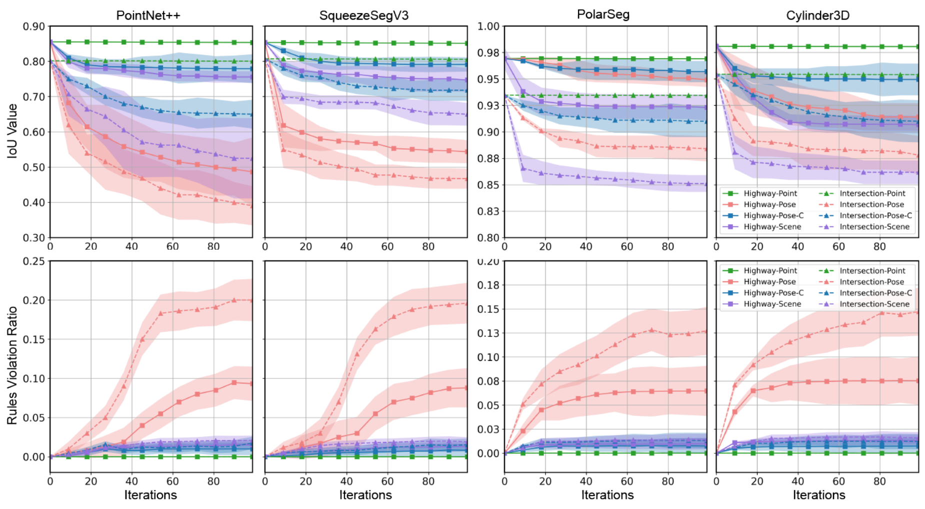

Fig. 5. Top: The IoU values during the attack process of four victim models on two backgrounds. **Bottom:** The ratio of rules violation of all methods, on two backgrounds. Although *Pose Attack* outperforms our *Scene Attack* in terms of the IoU value in some cases, it also has a large ratio of rules violation, which means the scenes generated by *Pose Attack* are not realistic as shown in Figure 6.

the same point in the latent space with the trajectories from good initialization (cyan→purple).

This result can be explained by the entanglement of the latent space [44], which makes multiple variables control the same property. To further study this point, we plot Figure 4(c), where we use 3 different initialization for T-VAE-SAG. The result shows that all three cases find the target scene but with totally different trajectories, which supports our conjecture. *In summary, we believe the contribution of knowledge can be attributed to the entanglement of the latent space, which makes the searching easily escape the local minimum and find the nearest optimal points.*

3.2 Adversarial Driving Scenes Generation

Task description. We aim to generate *realistic* driving scenes against segmentation algorithms as well as satisfy specific semantic knowledge rules. The adversarial scenes are defined as scenes that reduce the performance of victim models. To generate adversarial LiDAR scenes containing various fore-/background rather than the point cloud of a single 3D object as existing studies [40, 62], a couple of challenges should be considered: First, LiDAR scenes with millions of points are hard to be directly operated; Second, generated scenes need to be realistic and follow traffic rules. Since there are no existing methods to compare with directly, we compare three methods: (1) *Point Attack*: a point-wise attack baseline [68] that adds small disturbance to points; (2) *Pose Attack*: a scene generation method

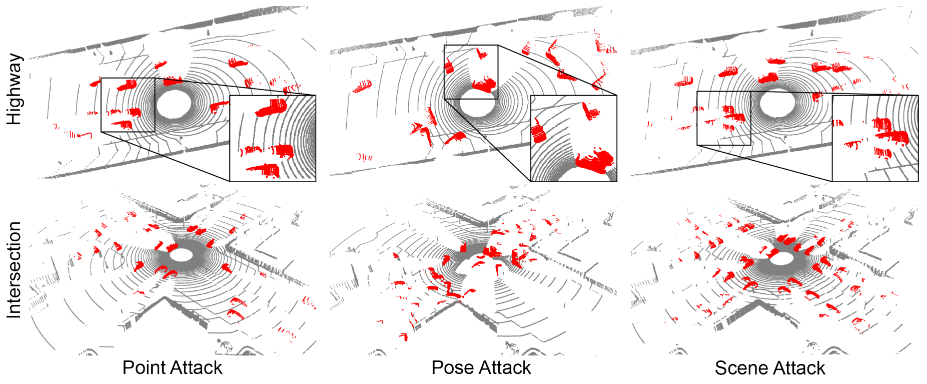

Fig. 6. Scenarios from three adversarial generation methods for PointNet++ model under *Highway* and *Intersection* background. Red points represent vehicles. Scenes generated by *Scene Attack* are complicated and follow basic traffic rules, while scenes generated by *Pose Attack* violate basic physical laws and traffic rules.

developed by us that searches pose of vehicles in the scene; (3) *Scene Attack*: a semantically controllable generative method based on our T-VAE and SAG.

We explore the attack effectiveness against different models of these methods, and their transferability. For *Pose Attack* and *Scene Attack*, we implement an efficient LiDAR model $\mathcal{R}(x, B)$ [49] (refer to Appendix.A for details) to convert the generated scene x to a point cloud scene with a background B. The task objective $\min \mathcal{L}_t(x) = \max \mathcal{L}_P(\mathcal{R}(x, B))$ is defined by maximizing the loss function $\mathcal{L}_P$ of segmentation algorithms P. We design three explicit knowledge rules: ① roads follow a given layout (location, width, and length); ② vehicles on the lane follow the direction of the lane; ③ vehicles should gather together but keep a certain distance. ①② ensure generated vehicles follow the layout of the background B and ③ makes the scene contain more vehicles.

Experiment settings. We select 4 segmentation algorithms (PointNet++ [56], PolarSeg [73], SqueezeSegV3 [70], Cylinder3D [74]) as our victim models, all of which are pre-trained on Semantic Kitti dataset [9]. We collect two backgrounds B (Highway and Intersection) in the CARLA simulator [19]. Since it is usually unable to access the parameters of segmentation algorithms, we focus on the black-box attack in this task. The *Point Attack* optimizes $\mathcal{L}_t(x)$ with SimBA [27], while *Pose Attack* and *Scene Attack* optimizes $\mathcal{L}_t(x)$ with Bayesian Optimization (BO) [53]. For the training of T-VAE, we build a dataset by extracting the pose information of vehicles together with road and lane information from the Argoverse dataset [12].

Evaluation results. We show the Intersection over Union (IoU) metric for the vehicle and the ratio of rules violation (ratio of objects that violate knowledge ①②) during the attack in Figure 5. Generally, it is harder to find adversarial scenes in the highway background than in the intersection background since the latter has much more vehicles. Within 100 iterations, the *Point Attack*

Table 2. Transferability of Adversarial Scenes (Point Attack IoU / Scene Attack IoU). *Scene Attack* has lower IoU for all evaluation pairs, which demonstrates its better adversarial transferability.

Source \ Target	PointNet++	SqueezeSegV3	PolarSeg	Cylinder3D
PointNet++[*]	- / -	0.916 / **0.768**	0.936 / **0.854**	0.955 / **0.918**
SqueezeSegV3	0.954 / **0.606**	- / -	0.932 / **0.855**	0.956 / **0.892**
PolarSeg	0.952 / **0.528**	0.904 / **0.753**	- / -	0.953 / **0.908**
Cylinder3D	0.951 / **0.507**	0.903 / **0.688**	0.934 / **0.877**	- / -

[*] The IoU for *Point Attack* is obtained after 20,000 iterations.

method nearly has no influence on the performance since it operates in very high dimensions. In contrast, *Pose Attack* and *Scene Attack* efficiently reduce the IoU value. Although *Pose Attack* achieves comparable results to our method, scenes generated by it (shown in Figure 6) are unrealistic due to the overlaps between vehicles; therefore, the ratio of rules violation is high. In contrast, scenes generated by our method only modify the vehicles within the traffic constraints. In Table 2, we explore the transferability of *Point Attack* and *Scene Attack*. Transferability, which means using generated samples from the *Source* model to attack other *Target* models, is crucial for evaluating adversarial attack algorithms. Better transferability indicates that the samples carry essential patterns ignored in most victim systems. Although *Point Attack* dramatically reduces the performance of all four victims, the generated scenes have weak transferability and cannot decrease the performance of other victim models. However, scenes generated by *Scene Attack* successfully attack all models, even those not targeted during the training, which demonstrates strong adversarial transferability. More generated scenes can be found in Appendix.E

4 Related Work

Semantically adversarial attacks. Early adversarial attack methods focused on the pixel-wise attack in the image field, where L_p-norm is used to constrain the adversarial perturbation. For the sake of the interpretability of the adversarial samples, recent studies begin to consider *semantic attacks*. They attack the rendering process of images by modifying the light condition [43, 72] or manipulating the position and shape of objects [5, 69, 30]. This paper explores the generation of adversarial point cloud scenes, which already have similar prior works [65, 4, 61]. [65, 4] modify the environment by adding objects on the top of existing vehicles to make them disappear. [61] create a ghost vehicle by adding an ignorable number of points; however, they modify a single object without considering the structural relationship of the whole scene.

Semantic driving scenario generation. Traditional ways of scene generation focus on sampling from pre-defined rules and grammars, such as probabilistic scene graphs used in [55] and heuristic rules applied in [19]. These methods rely on domain expertise and cannot be easily extensible to large-scale scenes.

Recently, data-driven generative models [15, 63, 51, 41, 38] are proposed to learn the distribution of objects and decouple the generation of scenes into sequence [63] and graphs [51, 41] of objects. Although they reduce the gap between simulation and reality, generated scenes cannot satisfy specific constraints. Another substantial body of literature [22, 37, 26] explored learning scene graphs from images or generating scene images directly via an end-to-end framework. Their generalization to high-dimensional data is very challenging, making them less effective than modularized methods proposed by [38, 67, 15].

Structural deep generative models. The original DGMs, such as GAN [25] and VAE [36], are mostly used for unstructured data. They leverage the powerful feature extraction of neural networks to achieve impressive results [34, 11]. However, the physical world is complex since objects have diverse and structural relationships. Domain-specific structural generative models are developed via tree structure (RvNN-VAE [41]) or graph structure (Graph-VAE [58]). Rule-based generative models are also explored by sampling from pre-defined rules [39, 33, 15]. Typical applications of this kind of model are molecular structure generation [28, 31, 32], natural scene generation [51, 14], and automatic program generation [13]. Unlike these existing methods, our approach explicitly integrates knowledge during the generation process. One practical application of DGMs is generating samples that meet the requirements of downstream tasks [21, 64]. [2, 3] searches in the latent space of StyleGAN [34] to obtain images that are similar to a given image. For structured data, such a searching framework transforms discrete space optimization to continuous space optimization, which was shown to be more efficient [45]. However, it may not guarantee the rationality of generated structured data due to the loss of interpretability and constraints in the latent space [13].

Incorporating knowledge into neural networks. Integrating knowledge to data-driven models has been explored in various forms from training methods, meta-modeling, embedding to rules used for reasoning. [29] distills logical rules with a teacher-student framework under *Posterior Regularization* [23]. Another way of knowledge distillation is encoding knowledge into vectors then refining the features from the model that are in line with the encoded knowledge [26]. These methods need to access *Knowledge Graphs* [20] during the training, which heavily depends on human experts. Meta-modeling of complex fluid is integrated into the NN to improve the performance of purely data-driven networks in [46]. In addition, [71] restricts the output of generative models to satisfy physical laws expressed by partial differential equations. In the reinforcement learning area, reward shaping [50] is recognized as one technique to incorporate heuristic knowledge to guide the training.

5 Conclusion

In this paper, we explore semantically adversarial generation tasks with explicit knowledge integration. Inspired by the categorization of knowledge for the driving scenario description, we design a tree-structured generative model to represent structured data. We show that the two types of knowledge can be explicitly injected into the tree structure to guide and restrict the generation process efficiently and effectively. After considering explicit semantic knowledge, we verify that the generated data contain dramatically fewer semantic constraint violations. Meanwhile, the generated data still maintain the diversity property and follow its original underlying distribution. Although we focus on the scene generation application, the SAG framework can be extended to other structured data generation tasks, such as chemical molecules and programming languages, showing the hierarchical properties. One assumption of this work is that the knowledge is helpful or at least harmless as they are summarized and provided by domain experts, which needs examination in the future.

Bibliography

[1] California Department of Motor Vehicle Disengagement Report. https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/disengagement-reports/ (2022), [Online]

[2] Abdal, R., Qin, Y., Wonka, P.: Image2stylegan: How to embed images into the stylegan latent space? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4432–4441 (2019)

[3] Abdal, R., Qin, Y., Wonka, P.: Image2stylegan++: How to edit the embedded images? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8296–8305 (2020)

[4] Abdelfattah, M., Yuan, K., Wang, Z.J., Ward, R.: Towards universal physical attacks on cascaded camera-lidar 3d object detection models. arXiv preprint arXiv:2101.10747 (2021)

[5] Alcorn, M.A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.S., Nguyen, A.: Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4845–4854 (2019)

[6] Amizadeh, S., Palangi, H., Polozov, A., Huang, Y., Koishida, K.: Neuro-symbolic visual reasoning: Disentangling" visual" from" reasoning". In: ICML. pp. 279–290 (2020)

[7] Audet, C., Hare, W.: Derivative-free and blackbox optimization (2017)

[8] Bau, D., Zhu, J.Y., Strobelt, H., Lapedriza, A., Zhou, B., Torralba, A.: Understanding the role of individual units in a deep neural network. Proceedings of the National Academy of Sciences **117**(48), 30071–30078 (2020)

[9] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In: Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV) (2019)

[10] Blei, D.M., Kucukelbir, A., McAuliffe, J.D.: Variational inference: A review for statisticians. Journal of the American statistical Association **112**(518), 859–877 (2017)

[11] Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)

[12] Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3d tracking and forecasting with rich maps. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8748–8757 (2019)

[13] Dai, H., Tian, Y., Dai, B., Skiena, S., Song, L.: Syntax-directed variational autoencoder for structured data. arXiv preprint arXiv:1802.08786 (2018)

[14] Deng, F., Zhui, Z., Lee, D., Ahn, S.: Generative scene graph networks. In: International Conference on Learning Representations (2021)

[15] Devaranjan, J., Kar, A., Fidler, S.: Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In: European Conference on Computer Vision. pp. 715–733. Springer (2020)

[16] Dienes, Z., Perner, J.: A theory of implicit and explicit knowledge. Behavioral and brain sciences **22**(5), 735–808 (1999)

[17] Ding, W., Chen, B., Li, B., Eun, K.J., Zhao, D.: Multimodal safety-critical scenarios generation for decision-making algorithms evaluation. IEEE Robotics and Automation Letters **6**(2), 1551–1558 (2021)

[18] Ding, W., Chen, B., Xu, M., Zhao, D.: Learning to collide: An adaptive safety-critical scenarios generating method. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2243–2250. IEEE (2020)

[19] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)

[20] Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. SEMANTiCS (Posters, Demos, SuCCESS) **48**, 1–4 (2016)

[21] Engel, J., Hoffman, M., Roberts, A.: Latent constraints: Learning to generate conditionally from unconditional generative models. arXiv preprint arXiv:1711.05772 (2017)

[22] Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Kavukcuoglu, K., Hinton, G.E.: Attend, infer, repeat: Fast scene understanding with generative models. arXiv preprint arXiv:1603.08575 (2016)

[23] Ganchev, K., Graça, J., Gillenwater, J., Taskar, B.: Posterior regularization for structured latent variable models. The Journal of Machine Learning Research **11**, 2001–2049 (2010)

[24] Ganin, Y., Kulkarni, T., Babuschkin, I., Eslami, S.A., Vinyals, O.: Synthesizing programs for images using reinforced adversarial learning. In: International Conference on Machine Learning. pp. 1666–1675. PMLR (2018)

[25] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. arXiv preprint arXiv:1406.2661 (2014)

[26] Gu, J., Zhao, H., Lin, Z., Li, S., Cai, J., Ling, M.: Scene graph generation with external knowledge and image reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1969–1978 (2019)

[27] Guo, C., Gardner, J., You, Y., Wilson, A.G., Weinberger, K.: Simple black-box adversarial attacks. In: International Conference on Machine Learning. pp. 2484–2493. PMLR (2019)

[28] Guo, X., Zhao, L.: A systematic survey on deep generative models for graph generation. arXiv preprint arXiv:2007.06686 (2020)

[29] Hu, Z., Ma, X., Liu, Z., Hovy, E., Xing, E.: Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318 (2016)

[30] Jain, L., Chandrasekaran, V., Jang, U., Wu, W., Lee, A., Yan, A., Chen, S., Jha, S., Seshia, S.A.: Analyzing and improving neural networks by generating semantic counterexamples through differentiable rendering. arXiv preprint arXiv:1910.00727 (2019)

[31] Jin, W., Barzilay, R., Jaakkola, T.: Junction tree variational autoencoder for molecular graph generation. In: International Conference on Machine Learning. pp. 2323–2332. PMLR (2018)

[32] Jin, W., Barzilay, R., Jaakkola, T.: Hierarchical generation of molecular graphs using structural motifs. In: International Conference on Machine Learning. pp. 4839–4848. PMLR (2020)

[33] 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. pp. 4551–4560 (2019)

[34] Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4401–4410 (2019)

[35] Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3907–3916 (2018)

[36] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)

[37] Kosiorek, A.R., Kim, H., Posner, I., Teh, Y.W.: Sequential attend, infer, repeat: Generative modelling of moving objects. arXiv preprint arXiv:1806.01794 (2018)

[38] Kundu, A., Li, Y., Rehg, J.M.: 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3559–3568 (2018)

[39] Kusner, M.J., Paige, B., Hernández-Lobato, J.M.: Grammar variational autoencoder. In: International Conference on Machine Learning. pp. 1945–1954. PMLR (2017)

[40] Lang, I., Kotlicki, U., Avidan, S.: Geometric adversarial attacks and defenses on 3d point clouds. arXiv preprint arXiv:2012.05657 (2020)

[41] Li, M., Patil, A.G., Xu, K., Chaudhuri, S., Khan, O., Shamir, A., Tu, C., Chen, B., Cohen-Or, D., Zhang, H.: Grains: Generative recursive autoencoders for indoor scenes. ACM Transactions on Graphics (TOG) **38**(2), 1–16 (2019)

[42] Liao, R., Li, Y., Song, Y., Wang, S., Nash, C., Hamilton, W.L., Duvenaud, D., Urtasun, R., Zemel, R.S.: Efficient graph generation with graph recurrent attention networks. arXiv preprint arXiv:1910.00760 (2019)

[43] Liu, H.T.D., Tao, M., Li, C.L., Nowrouzezahrai, D., Jacobson, A.: Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. arXiv preprint arXiv:1808.02651 (2018)

[44] Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., Bachem, O.: Challenging common assumptions in the unsupervised learning of disentangled representations. In: international conference on machine learning. pp. 4114–4124. PMLR (2019)

[45] Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.Y.: Neural architecture optimization. arXiv preprint arXiv:1808.07233 (2018)

[46] Mahmoudabadbozchelou, M., Caggioni, M., Shahsavari, S., Hartt, W.H., Karniadakis, E., Jamali, S., et al.: Data-driven physics-informed constitutive metamodeling of complex fluids: A multifidelity neural network (mfnn) framework. Journal of Rheology **65**(2) (2021)
[47] Malcolm, G.L., Groen, I.I., Baker, C.I.: Making sense of real-world scenes. Trends in Cognitive Sciences **20**(11), 843–856 (2016)
[48] Mo, K., Wang, H., Yan, X., Guibas, L.: Pt2pc: Learning to generate 3d point cloud shapes from part tree conditions. In: European Conference on Computer Vision. pp. 683–701. Springer (2020)
[49] Möller, T., Trumbore, B.: Fast, minimum storage ray-triangle intersection. Journal of graphics tools **2**(1), 21–28 (1997)
[50] Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: Icml. vol. 99, pp. 278–287 (1999)
[51] Para, W., Guerrero, P., Kelly, T., Guibas, L., Wonka, P.: Generative layout modeling using constraint graphs. arXiv preprint arXiv:2011.13417 (2020)
[52] Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends in optimization **1**(3), 127–239 (2014)
[53] Pelikan, M., Goldberg, D.E., Cantú-Paz, E., et al.: Boa: The bayesian optimization algorithm. In: Proceedings of the genetic and evolutionary computation conference GECCO-99. vol. 1, pp. 525–532. Citeseer (1999)
[54] Plumerault, A., Borgne, H.L., Hudelot, C.: Controlling generative models with continuous factors of variations. arXiv preprint arXiv:2001.10238 (2020)
[55] 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: 2019 International Conference on Robotics and Automation (ICRA). pp. 7249–7255. IEEE (2019)
[56] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
[57] Sethuraman, J.: A constructive definition of dirichlet priors. Statistica sinica pp. 639–650 (1994)
[58] Simonovsky, M., Komodakis, N.: Graphvae: Towards generation of small graphs using variational autoencoders. In: International Conference on Artificial Neural Networks. pp. 412–422. Springer (2018)
[59] Smullyan, R.M.: First-order logic. Courier Corporation (1995)
[60] Socher, R., Lin, C.C.Y., Ng, A.Y., Manning, C.D.: Parsing natural scenes and natural language with recursive neural networks. In: ICML (2011)
[61] Sun, J., Cao, Y., Chen, Q.A., Mao, Z.M.: Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 877–894 (2020)
[62] Sun, J., Koenig, K., Cao, Y., Chen, Q.A., Mao, Z.M.: On the adversarial robustness of 3d point cloud classification. arXiv preprint arXiv:2011.11922 (2020)

[63] Tan, S., Wong, K., Wang, S., Manivasagam, S., Ren, M., Urtasun, R.: Scenegen: Learning to generate realistic traffic scenes. arXiv preprint arXiv:2101.06541 (2021)

[64] Tripp, A., Daxberger, E., Hernández-Lobato, J.M.: Sample-efficient optimization in the latent space of deep generative models via weighted retraining. Advances in Neural Information Processing Systems **33** (2020)

[65] Tu, J., Ren, M., Manivasagam, S., Liang, M., Yang, B., Du, R., Cheng, F., Urtasun, R.: Physically realizable adversarial examples for lidar object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13716–13725 (2020)

[66] Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural computation **1**(2), 270–280 (1989)

[67] Wu, J., Tenenbaum, J.B., Kohli, P.: Neural scene de-rendering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 699–707 (2017)

[68] Xiang, C., Qi, C.R., Li, B.: Generating 3d adversarial point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9136–9144 (2019)

[69] Xiao, C., Yang, D., Li, B., Deng, J., Liu, M.: Meshadv: Adversarial meshes for visual recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6898–6907 (2019)

[70] Xu, C., Wu, B., Wang, Z., Zhan, W., Vajda, P., Keutzer, K., Tomizuka, M.: Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In: European Conference on Computer Vision. pp. 1–19. Springer (2020)

[71] Yang, Y., Perdikaris, P.: Physics-informed deep generative models. arXiv preprint arXiv:1812.03511 (2018)

[72] Zeng, X., Liu, C., Wang, Y.S., Qiu, W., Xie, L., Tai, Y.W., Tang, C.K., Yuille, A.L.: Adversarial attacks beyond the image space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4302–4311 (2019)

[73] Zhang, Y., Zhou, Z., David, P., Yue, X., Xi, Z., Gong, B., Foroosh, H.: Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9601–9610 (2020)

[74] Zhou, H., Zhu, X., Song, X., Ma, Y., Wang, Z., Li, H., Lin, D.: Cylinder3d: An effective 3d framework for driving-scene lidar semantic segmentation. arXiv preprint arXiv:2008.01550 (2020)