Generating Useful Accident-Prone Driving Scenarios via a Learned Traffic Prior

Davis Rempe[1,2] Jonah Philion[2,3,4] Leonidas J. Guibas[1] Sanja Fidler[2,3,4] Or Litany[2]
[1]Stanford University [2]NVIDIA [3]University of Toronto [4]Vector Institute

nv-tlabs.github.io/STRIVE

Abstract

Evaluating and improving planning for autonomous vehicles requires scalable generation of long-tail traffic scenarios. To be useful, these scenarios must be realistic and challenging, but not impossible to drive through safely. In this work, we introduce STRIVE, a method to automatically generate challenging scenarios that cause a given planner to produce undesirable behavior, like collisions. To maintain scenario plausibility, the key idea is to leverage a learned model of traffic motion in the form of a graph-based conditional VAE. Scenario generation is formulated as an optimization in the latent space of this traffic model, perturbing an initial real-world scene to produce trajectories that collide with a given planner. A subsequent optimization is used to find a "solution" to the scenario, ensuring it is useful to improve the given planner. Further analysis clusters generated scenarios based on collision type. We attack two planners and show that STRIVE successfully generates realistic, challenging scenarios in both cases. We additionally "close the loop" and use these scenarios to optimize hyperparameters of a rule-based planner.

1. Introduction

The safety of contemporary autonomous vehicles (AVs) is defined by their ability to safely handle complicated near-collision scenarios. However, these kinds of scenarios are rare in real-world driving, posing a data-scarcity problem that is detrimental to both the development and testing of data-driven models for perception, prediction, and planning. Moreover, the better models become, the more rare these events will be, making the models even harder to train.

A natural solution is to synthesize difficult scenarios in simulation, rather than relying on real-world data, making it easier and safer to evaluate and train AV systems. This approach is especially appealing for *planning*, where the appearance domain gap is not a concern. For example, one can manually design scenarios where the AV may fail by inserting adversarial actors or modifying trajectories, either from scratch or by perturbing a small set of real scenarios. Unfortunately, the manual nature of this approach quickly becomes prohibitively expensive when a large set of scenarios is necessary for training or comprehensive evaluation.

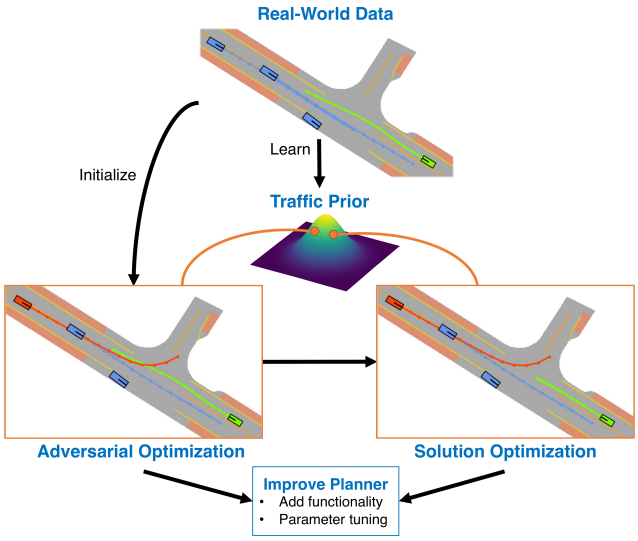

Figure 1. STRIVE generates challenging scenarios for a given planner. An *adversarial optimization* perturbs a real-world scene in the latent space of a learned traffic model, causing an adversary (red) to collide with the planner (green). A subsequent *solution optimization* finds a planner trajectory to avoid collisions, verifying a scenario is useful for identifying planner improvements.

Recent work looks to *automatically generate* challenging scenarios [1, 11, 12, 26, 36, 50, 53]. Generally, these approaches control a single or small group of "adversaries" in a scene, define an objective (*e.g.* cause a collision with the AV), and then optimize the adversaries' behavior or trajectories to meet the objective. While most methods demonstrate generation of only 1 or 2 scenarios [1, 8, 26, 36], recent work [53] has improved scalability by starting from real-world traffic scenes and perturbing a limited set of prechosen adversaries. However, these approaches *lack expressive priors over plausible traffic motion*, which limits the realism and diversity of scenarios. In particular, adversarial entities in a scenario are a small set of agents heuristically chosen ahead of time; surrounding traffic will not be reactive and therefore perturbations must be careful to avoid implausible situations (*e.g.* collisions with auxiliary agents). Furthermore, less attention has been given to determining if

a scenario is "unsolvable" [18], *i.e.*, if even an oracle AV is incapable of avoiding a collision. In this degenerate case, the scenario is not *useful* for evaluating/training a planner.

In this work, we introduce STRIVE – a method for generating challenging scenarios to **S**tress-**T**est d**RIVE** a given AV system. STRIVE attacks the prediction, planning, and control subset of the AV stack, which we collectively refer to as the *planner*. As shown in Fig. 1, our approach perturbs an initial real-world scene through an optimization procedure to cause a collision between an arbitrary adversary and a given planner. Our core idea is to measure the plausibility of a scenario during optimization by its likelihood under a learned generative model of traffic motion, which encourages scenarios to be challenging, yet realistic. As a result, STRIVE does not choose specific adversaries ahead of time, rather it jointly optimizes all scene agents, enabling a diverse set of scenarios to arise. Moreover, in order to accommodate for non-differentiable (or inaccessible) planners, which are widely used in practice, the proposed optimization uses a differentiable proxy representation of the planner within the learned motion model, thus allowing standard gradient-based optimization to be used.

We propose to identify and characterize generated scenarios that are *useful* for improving a given planner. We first search for a "solution" to generated scenarios to determine if they are degenerate, and then cluster solvable scenarios based on collision properties. We test STRIVE on two AV planners, including a new rule-based planner, and show that it generates plausible and diverse collision scenarios in both cases. We additionally use generated scenarios to improve the rule-based planner by identifying fundamental limitations of its design and tuning hyperparameters.

In short, our contributions are: (i) a method to automatically generate plausible challenging scenarios for a given planner, (ii) a solution optimization to ensure scenario utility, and (iii) an analysis method to cluster scenarios by collision type. Supplementary videos and material for this work are available on the project webpage.

2. Related Work

Traffic Motion Modeling. Scenario replay is insufficient for testing and developing AV planners as the motion of non-ego vehicles is strongly coupled to the actions chosen by the ego planner. Advances in deep learning have allowed us to replace traditional dynamic and kinematic models [27, 29, 52] or rule-based simulators [14, 33] with neural counterparts that better capture traffic complexity [2, 37]. Efforts to predict future trajectories from a short state history and an HD map can generally be categorized according to the encoding technique, modeling of multi-modality, multi-agent interaction, and whether the trajectory is estimated in a single step or progressively. The encoding of surrounding context of each agent is often done via a bird's-eye view (BEV) raster image [7, 9, 16], though some work [17, 31] replaces the rasterization-based map encoding with a lane-graph representation. SimNet [3] increased the diversity of generated simulations by initializing the state using a generative model conditioned on the semantic map. To account for multi-modality, multiple futures have been estimated either directly [9] or through trajectory proposals [7, 16, 32, 39]. Modeling multi-actor interactions explicitly using dense graphs has proven effective for vehicles [5, 47], lanes [31], and pedestrians [20, 28, 45]. Finally, step-by-step prediction has performed favorably to one-shot prediction of the whole trajectory [13]. We follow these works and design a traffic model that uses an inter-agent graph network [21] to represent agent interaction and is variational, allowing us to sample multiple futures.

Our model builds on VAE-based approaches [5, 47] that provide a learned prior over a controllable latent space [41]. Among other design differences, we incorporate a penalty for environment collisions and structure predictions through a bicycle model to ensure physical plausibility.

Challenging Scenario Generation. Generating scenarios has the potential to exponentially increase scene coverage compared to relying exclusively on recorded drives. Advances in photo-realistic simulators like CARLA [14] and NVIDIA's DRIVE Sim, along with the availability of large-scale datasets [4, 15, 19, 23, 46], have been instrumental to methods that generate plausible scene graphs to improve perception [10, 22, 42] and planning [3, 5, 24, 47]. Our work focuses on generating challenging – or "adversarial"[1] – scenarios, which are even more crucial since they are so rare in recorded data. While most works assume perfect perception and attack the planning module [8, 11, 12, 18, 26, 50], recent efforts exploit the full stack, including image or point-cloud perception [1, 30, 36, 48, 53]. Our work focuses on attacking the planner only, though our scene parameterization as a learned traffic model could be incorporated into end-to-end methods. Unlike our approach, which uses gradient-based optimization enabled by the learned motion model, most adversarial generation works rely heavily on black-box optimization which may be slow and unreliable.

Our scenario generation approach is most similar to AdvSim [53], however instead of optimizing acceleration profiles of a simplistic bicycle model we use a more expressive data-driven motion prior. This remedies the previous difficulty of controlling many adversarial agents simultaneously in a plausible manner. Moreover, we avoid constraining the attack trajectory to not collide with the playback AV by proposing a "solution" optimization stage to filter worthwhile scenarios. Prior work [8] clusters lane-change scenarios based on trajectories of agents, while we cluster based on collision properties between the adversary and planner.

[1] we use "challenging" to denote generation procedures that do not explicitly attack a specific module in the perception or planning stack

AV Planners. Despite the recent academic interest in end-to-end learning-based planners and AVs [2, 6, 43, 44, 54], rule-based planners remain the norm in practical AV systems [51]. Therefore, we evaluate our approach on a rule-based planner similar to the lane-graph-based planners used by successful teams in the 2007 DARPA Urban Challenge [34, 49] detailed in Sec. 4.2.

3. Challenging Scenario Generation

STRIVE aims to generate high-risk traffic situations for a given *planner*, which can subsequently be used to improve that planner (Fig. 1). For our purpose, the planner encapsulates prediction, planning, and control, *i.e.* we're interested in scenarios where the system misbehaves even with perfect perception. The planner takes as input past trajectories of other agents in a scene and outputs the future trajectory of the vehicle it controls (termed the *ego* vehicle). It is assumed to be black-box: STRIVE has no knowledge of the planner's internals and cannot compute gradients through it. Undesirable behavior includes collisions with other vehicles and non-drivable terrain, uncomfortable driving (*e.g.* high accelerations), and breaking traffic laws. We focus on generating **accident-prone scenarios** involving vehicle-vehicle collisions with the planner, though our formulation is general and in principle can handle alternative objectives.

Similar to prior work [53], scenario generation is formulated as an optimization problem that perturbs agent trajectories in an initial scenario from real-world data. The input is a planner f, map $\mathcal{M}$ containing semantic layers for drivable area and lanes, and a sequence from a pre-recorded real-world scene that serves as initialization for optimization. This initial scenario contains N agents with trajectories represented in 2D BEV as $Y = \{\mathbf{Y}^i\}_{i=1}^N$, where $\mathbf{Y}^i = [\mathbf{y}_1^i, \mathbf{y}_2^i, \ldots, \mathbf{y}_T^i]$ is the sequence of states for agent i. We let $\mathbf{Y}_t = [\mathbf{y}_t^1, \mathbf{y}_t^2, \ldots, \mathbf{y}_t^N]$ be the state of all agents at a single timestep. An agent state $\mathbf{y}_t^i = [x_t, y_t, \theta_t, v_t, \dot{\theta}_t]$ at time t contains the 2D position (x_t, y_t), heading θ_t, speed v_t, and yaw rate $\dot{\theta}_t$. When rolled out within a scenario, at each timestep the planner outputs the next **ego** state $\mathbf{y}_t^{\text{plan}} = f(\mathbf{y}_{<t}^{\text{plan}}, \mathbf{Y}_{<t}, \mathcal{M})$ based on the *past* motion of itself and other agents. For simplicity, we will write the rolled out planner trajectory as $\mathbf{Y}^{\text{plan}} = f(Y, \mathcal{M})$ where $\mathbf{Y}^{\text{plan}} = [\mathbf{y}_1^{\text{plan}}, \mathbf{y}_2^{\text{plan}}, \ldots, \mathbf{y}_T^{\text{plan}}]$ for the remainder of this paper. Scenario generation perturbs trajectories for all non-ego agents to best meet an adversarial objective $\mathcal{L}_{\text{adv}}$ (*e.g.* cause a collision with the planner):

$$\min_Y \mathcal{L}_{\text{adv}}(Y, \mathbf{Y}^{\text{plan}}), \quad \mathbf{Y}^{\text{plan}} = f(Y, \mathcal{M}). \tag{1}$$

One may optimize a single or small set of "adversaries" in Y explicitly, *e.g.* through the kinematic bicycle model parameterization [27, 40, 53]. While this enforces plausible single-agent dynamics, **interactions** must be constrained to avoid collisions between non-ego agents and, even then, the resulting traffic patterns may be unrealistic. We propose to instead *learn* to model traffic motion using a neural network and then use it at optimization time **(i) to parameterize** all trajectories in a scenario as vectors in the latent space, and **(ii) as a prior** over scenario plausibility. Next, we describe this traffic model, followed by the "adversarial" optimization that produces collision scenarios.

3.1. Modeling "Realism": Learned Traffic Model

We wish to generate accident-prone scenarios that are assumed to develop over short time periods (<10 sec) [35]. Therefore, traffic modeling is formulated as *future forecasting*, which predicts future trajectories for all agents in a scene based on their past motion. We learn $p_\theta(Y|X, \mathcal{M})$ to enable sampling a future scenario Y conditioned on the fixed past $X = \{\mathbf{X}^i\}_{i=1}^N$ (defined similar to Y) and the map $\mathcal{M}$. Two properties of the traffic model make it particularly amenable to downstream optimization: a low-dimensional latent space for efficient optimization, and a prior distribution over this latent space to determine the plausibility of a given scenario. Inspired by recent work [5, 47], we design a conditional variational autoencoder (CVAE), shown in Fig. 2, that meets these criteria while learning accurate and scene-consistent joint future predictions. We briefly introduce the architecture and training procedure here, and refer to the supplement for specific details.

Architecture. To sample future motions at test time, the *conditional prior* and *decoder* are used; both are graph neural networks (GNN) operating on a fully-connected scene graph of all agents. The *prior* models $p_\theta(Z|X, \mathcal{M})$ where $Z = \{\mathbf{z}^i\}_{i=1}^N$ is a set of agent latent vectors. Each node in the input scene graph contains a context feature $\mathbf{h}^i$ extracted from that agent's past trajectory, local rasterized map, bounding-box size, and semantic class. After message passing, the *prior* outputs parameters of a Gaussian $p_\theta(\mathbf{z}^i|X, \mathcal{M}) = \mathcal{N}(\mu_\theta^i(X, \mathcal{M}), \sigma_\theta^i(X, \mathcal{M}))$ for each agent in the scene, forming a "distributed" latent representation that captures the variation in possible futures.

The deterministic *decoder* $Y = d_\theta(Z, X, \mathcal{M})$ operates on the scene graph with both a sampled latent $\mathbf{z}^i$ and past context $\mathbf{h}^i$ at each node. Decoding is performed autoregressively: at timestep t, one round of message passing resolves interactions before predicting accelerations $\dot{v}_t, \ddot{\theta}_t$ for each agent. Accelerations immediately go through the kinematic bicycle model [27, 40] to obtain the next state $\mathbf{y}_{t+1}^i$, which updates $\mathbf{h}^i$ before continuing rollout. The determinism and graph structure of the decoder encourages scene-consistent future predictions even when agent $\mathbf{z}$'s are independently sampled. Importantly for latent optimization, the decoder ensures plausible vehicle dynamics by using the kinematic bicycle model, even if the input Z is unlikely.

Training. Training is performed on pairs of (X, Y_{gt}) using

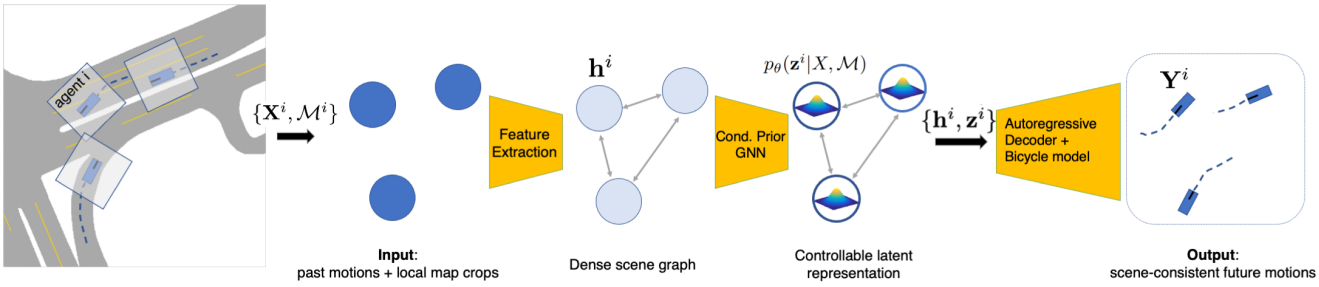

Figure 2. Test-time architecture of the learned traffic model. To jointly sample future trajectories for all agents in a scene, past motion and local map context is first processed individually for each agent. The *conditional prior*, then outputs a latent distribution at each node that can be sampled and fed through the *autoregressive decoder* to predict future agent trajectories.

a modified CVAE objective:

$$\mathcal{L}_{\text{cvae}} = \mathcal{L}_{\text{recon}} + w_{\text{KL}}\mathcal{L}_{\text{KL}} + w_{\text{coll}}\mathcal{L}_{\text{coll}}. \quad (2)$$

To optimize this loss, a *posterior* network $q_\phi(Z|Y_{\text{gt}}, X, \mathcal{M})$ is introduced similar to the prior, but operating jointly on past and future motion. Future trajectory features are extracted separately, while past features are the same as used in the *prior*. The full training loss uses trajectory samples from both the posterior Y_{post} and prior Y_{prior}:

$$\mathcal{L}_{\text{recon}} = \sum_{i=1}^{N} ||\mathbf{Y}^i_{\text{post}} - \mathbf{Y}^i_{\text{gt}}||^2 \quad (3)$$

$$\mathcal{L}_{\text{KL}} = D_{\text{KL}}(q_\phi(Z|Y_{\text{gt}}, X, \mathcal{M})||p_\theta(Z|X,\mathcal{M})) \quad (4)$$

$$\mathcal{L}_{\text{coll}} = \mathcal{L}_{\text{agent}} + \mathcal{L}_{\text{env}} \quad (5)$$

where $\mathbf{Y}^i_{\text{post}} \in Y_{\text{post}}$, $\mathbf{Y}^i_{\text{gt}} \in Y_{\text{gt}}$, and D_{KL} is the KL divergence. Collision penalties $\mathcal{L}_{\text{agent}}$ and $\mathcal{L}_{\text{env}}$ use a differentiable approximation of collision detection as in [47], which represents vehicles by sets of discs to penalize Y_{prior} for collisions between agents or with the non-drivable map area.

3.2. Adversarial Optimization

To leverage the learned traffic model, the real-world scenario used to initialize optimization is split into the past X and future Y_{init}. Throughout optimization, past trajectories in X (including that of the planner) are *fixed* while the future is perturbed to cause a collision with the given planner f. This perturbation is done in the learned latent space of the traffic model – as described below, we optimize the set of latents for all N *non-ego* agents $Z = \{\mathbf{z}^i\}_{i=1}^N$ along with a latent representation of the planner $\mathbf{z}^{\text{plan}}$.

Latent scenario parameterization encourages plausibility in two ways. First, since the decoder is trained on real-world data, it will output realistic traffic patterns if Z stays near the learned manifold. Second, the learned prior network gives a distribution over latents, which is used to penalize unlikely Z. This strong prior on behavioral plausibility enables jointly optimizing *all agents* in the scene rather than choosing a small set of specific adversaries in advance.

At each step of optimization (Fig. 3), the perturbed scenario is decoded with $d_\theta(Z, \mathbf{z}^{\text{plan}}, X, \mathcal{M})$ and non-ego trajectories Y are passed to the (black-box) planner, which rolls out the ego motion before losses can be computed. Adversarial optimization seeks two simultaneous objectives:

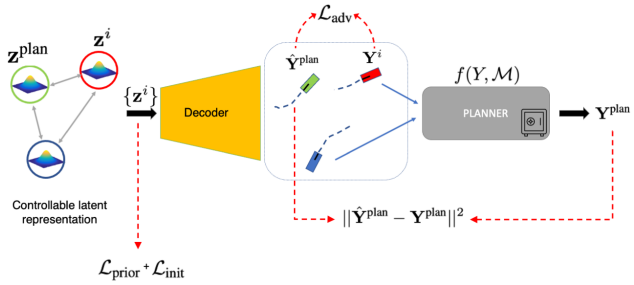

Figure 3. At each step of adversarial optimization, latent representations of both the planner and non-ego agents are decoded with the learned decoder and non-ego trajectories are given to the planner for rollout within the scenario. Finally, losses are computed.

1. Match Planner. Although optimization has no direct control over the planner's behavior – an external function that is queried only when required – it is still necessary to represent the planner within the traffic model (*i.e.* include it in the scene graph with an associated latent $\mathbf{z}^{\text{plan}}$) so that interactions with other agents are realistic. In doing this, future predictions from the decoder include an estimate of the planner trajectory $\hat{\mathbf{Y}}^{\text{plan}}$ that, ideally, is close to the true planner trajectory $\mathbf{Y}^{\text{plan}} = f(Y, \mathcal{M})$. Note that this gives a *differentiable approximation* of the planner, enabling typical gradient-based optimization to be used for the second objective described below. To encourage matching the real planner output with this "internal" approximation, we use

$$\min_{\mathbf{z}^{\text{plan}}} ||\hat{\mathbf{Y}}^{\text{plan}} - \mathbf{Y}^{\text{plan}}||^2 - \alpha \log p_\theta(\mathbf{z}^{\text{plan}}|X,\mathcal{M}) \quad (6)$$

where the right term lightly regularizes $\mathbf{z}^{\text{plan}}$ to stay likely under the learned prior and α balances the two terms.

2. Collide with Planner. The goal for non-ego agents is to cause the planner to collide with another vehicle:

$$\min_{Z} \mathcal{L}_{\text{adv}} + \mathcal{L}_{\text{prior}} + \mathcal{L}_{\text{init}} + \mathcal{L}_{\text{coll}}. \quad (7)$$

The *adversarial* term encourages a collision by minimizing the positional distance between controlled agents and the current traffic model approximation of the planner:

$$\mathcal{L}_{\text{adv}} = \sum_{i=1}^{N} \sum_{t=1}^{T} \delta_t^i \cdot ||\mathbf{y}_t^i - \hat{\mathbf{y}}_t^{\text{plan}}||^2 \tag{8}$$

$$\delta_t^i = \frac{\exp(-||\mathbf{y}_t^i - \hat{\mathbf{y}}_t^{\text{plan}}||)}{\sum_j \sum_t \exp(-||\mathbf{y}_t^j - \hat{\mathbf{y}}_t^{\text{plan}}||)} \tag{9}$$

where $\mathbf{y}_t$ here only includes the 2D position. Intuitively, the δ_t^i coefficients defined by the *softmin* in Eq. (9) are finding a candidate agent and timestep to collide with the planner. The agent with the largest δ_t^i is the most likely "adversary" based on distance, and Eq. (8) prioritizes causing a collision between this adversary and the planner while still allowing gradients to reach other agents. This weighting helps $\mathcal{L}_{\text{prior}}$ to avoid all agents unrealistically colliding with the planner.

The *prior* term encourages latents to stay likely under the learned prior network:

$$\mathcal{L}_{\text{prior}} = -\frac{1}{N} \sum_{i=1}^{N} \gamma^i \cdot \log p_\theta(\mathbf{z}^i | X, \mathcal{M}) \tag{10}$$

$$\gamma^i = 1 - \sum_t \delta_t^i. \tag{11}$$

The γ^i coefficient will weight likely adversaries near zero, *i.e.* agents close to colliding with the planner are allowed to deviate from the learned traffic manifold to exhibit rare and challenging behavior. Because the traffic model training data does not contain collisions, we found it difficult for an agent to collide with the planner using a large prior loss, thus motivating the weighting in Eq. (11). Note that even when γ^i is small, agents will maintain physical plausibility since the decoder uses the kinematic bicycle model.

$\mathcal{L}_{\text{init}}$ encourages staying close to the initialization in latent space, since it is already known to be realistic:

$$\mathcal{L}_{\text{init}} = \frac{1}{N} \sum_{i=1}^{N} \gamma^i \cdot ||\mathbf{z}^i - \mathbf{z}_{\text{init}}^i||^2 \tag{12}$$

where $\mathbf{z}_{\text{init}}^i \in Z_{\text{init}}$ are the latents that initialize optimization. Finally, similar to CVAE training, $\mathcal{L}_{\text{coll}}$ discourages non-ego agents from colliding with each other and the non-drivable area. In practice, all loss terms are balanced by manual inspection of a small set of generated scenarios.

Initialization and Optimization. Given a real-world scene, Z_{init} is obtained through the posterior network q_ϕ, then further refined with an initialization optimization that fits to the input future trajectories of all agents (similar to Eq. (6)), including the initial planner rollout. Optimization is implemented in PyTorch [38] using ADAM [25] with a learning rate of 0.05. Runtime depends on the planner and number of agents; for our rule-based planner (see Sec. 4.2), a 10-agent scenario takes 6-7 minutes.

4. Analyzing and Using Generated Scenarios

4.1. Filtering and Collision Classification

Solution Optimization. Adversarial optimization produces plausible scenarios, but it cannot guarantee they are *solvable* and *useful*: *e.g.* a scenario in which the ego is squeezed by multiple cars produces an unavoidable collision and is therefore uninformative for evaluating or improving a planner. Therefore, we perform an additional optimization to identify an ego trajectory that avoids collision; if this optimization fails, the scenario is discarded for downstream tasks. This solution optimization is initialized from the output of the adversarial optimization and essentially inverts the objectives described in Sec. 3.2: non-ego latent Z are tuned to maintain the adversarial trajectories while $\mathbf{z}^{\text{plan}}$ is optimized to avoid collisions and stay likely under the prior.

Clustering and Labeling. To gain insight into the distribution of collision scenarios and inform their downstream use, we propose a simple approach to cluster and label them. Specifically, scenarios are characterized by the explicit relationship between the planner and adversary at the time of collision: the relative direction and heading of the adversary are computed in the frame of the planner and concatenated to form a collision feature for each scenario. These features are clustered with k-means to form semantically similar groups of accidents that are labeled by visual inspection. Their distribution can then be visualized as in Fig. 6.

4.2. Improving the Planner

With a large set of labeled collision scenarios, the planner can be improved in two main ways. First, discrete improvements to functionality may be needed if many scenarios of the same type are generated. For example, a planner that strictly follows lanes is subject to collisions from head on or behind as it fails to swerve, indicating necessary new functionality to leave the lane graph. Second, scenarios provide data for tuning hyperparameters or learned parameters.

Rule-based Planner. To demonstrate how STRIVE scenarios are used for these kinds of improvements, we introduce a simple, yet competent, rule-based planner that we use as a proxy for a real-world planner. Our planner is ideal for evaluating STRIVE as it is easily interpretable, uses a small set of hyperparameters, and has known failure modes. In short, it relies entirely on the lane graph to both predict future trajectories of non-ego vehicles and generate candidate trajectories for the ego vehicle. Among these candidates, it chooses that which covers the most distance with a low "probability of collision." Planner behavior is affected by hyperparameters such as maximum speed/acceleration and how collision probability is computed. This planner has the additional limitation that it cannot change lanes, which scenarios generated by STRIVE exposes in Sec. 5.3. Full details of this planner are included in the supplementary.

5. Experiments

We next highlight the new capabilities that STRIVE enables. Sec. 5.1 demonstrates the ability to generate challenging and useful scenarios on two different planners; these scenarios contain a diverse set of collisions, as shown through analysis in Sec. 5.2. Generated scenarios are used to improve our rule-based planner in Sec. 5.3.

Dataset. The nuScenes dataset [4] is used both to train the traffic model and to initialize adversarial optimization. It contains $20s$ traffic clips annotated at 2 Hz, which we split into $8s$ scenarios. Only *car* and *truck* vehicles are used and the traffic model operates on the rasterized *drivable area*, *carpark area*, *road divider*, and *lane divider* map layers. We use the splits and settings of the nuScenes prediction challenge which is $2s$ (4 steps) of past motion to predict $6s$ (12 steps) of future, meaning collision scenarios are $8s$ long, but only the future $6s$ trajectories are optimized.

Planners. Scenario generation is evaluated on two different planners. The *Replay* planner simply plays back the ground truth ego trajectory from nuScenes data. This is an open-loop setting where the planner's $6s$ future is fully rolled out without re-planning. The *Rule-based* planner, described in Sec. 4.2, allows a more realistic closed-loop setting where the planner reacts to the surrounding agents during future rollout by re-planning at 5 Hz.

Metrics. The **collision rate** is the fraction of optimized initial scenarios from nuScenes that succeed in causing a planner collision, which indicates the sample efficiency of scenario generation. **Solution rate** is the fraction of these colliding scenarios for which a solution was found, which measures how often scenarios are *useful*. **Acceleration** indicates how comfortable a driven trajectory is; challenging scenarios should generally increase acceleration for the planner, while the adversary's acceleration should be reasonably low to maintain plausibility. If a scenario contains a collision, acceleration (and other trajectory metrics) is only calculated up to the time of collision. **Collision velocity** is the relative speed between the planner and adversary at the time of collision; it points to the severity of a collision.

5.1. Scenario Generation Evaluation

First, we demonstrate that STRIVE generates challenging, yet solvable, scenarios causing planners to collide and drive uncomfortably. Moreover, compared to an alternative generation approach that *does not* leverage the learned traffic prior, STRIVE scenarios are more plausible. Scenario generation is initialized from 1200 $8s$ sequences from nuScenes. Before adversarial optimization, scenes are prefiltered heuristically by how likely they are to produce a useful collision, leaving $<$500 scenarios to optimize.

Planner-Specific Scenarios. Tab. 1 shows that compared to rolling out a given planner on "regular" (unmodified) nuScenes scenarios, challenging scenarios from STRIVE produce more collisions and less comfortable driving. For the *Rule-based* planner, metrics on challenging scenarios are compared to the corresponding set of regular scenarios from which they originated (regular scenarios for *Replay* are omitted since nuScenes data contains no collisions and, by definition, planner behavior does not change).

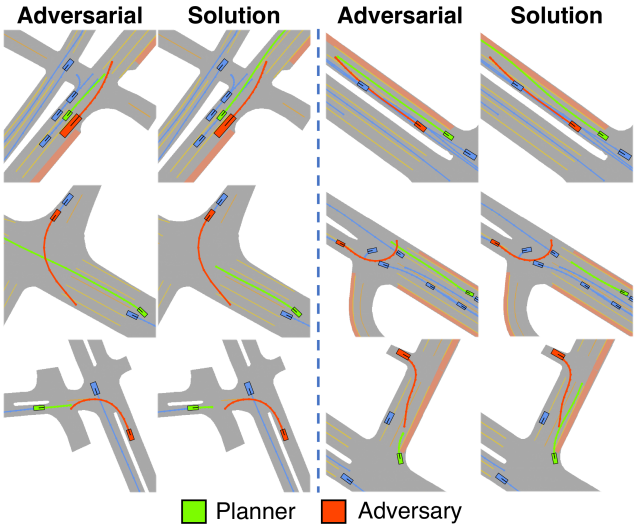

Figure 4. Qualitative results on the *Rule-based* planner. Adversarial and solution optimization results are shown. STRIVE produces diverse collision scenarios including lane changes, (u-)turning in front of the planner, and pulling into oncoming traffic.

Collision and solution rates indicate that generated scenarios are accident-prone and *useful* (solvable). For the *Rule-based* planner, adversarial optimization causes collisions in 27.4% of scenarios compared to only 1.2% in the regular scenarios. Generated scenarios also contain more severe collisions in terms of velocity, and elicit larger accelerations, *i.e*. less comfortable driving. The position and angle errors between approximate ($\hat{\mathbf{Y}}^{\text{plan}}$) and true ($\mathbf{Y}^{\text{plan}}$) planner trajectories at the end of adversarial optimization are shown on the right (see Sec. 3.2). The largest position error of $1.23m$ is reasonable relative to the $4.084m$ length of the planner vehicle. Qualitative results for the *Rule-based* planner visualize 2D waypoint trajectories (Fig. 4); though not shown, STRIVE also generates speed and heading.

Baseline Comparison. STRIVE is next compared to a baseline approach to demonstrate that leveraging a learned traffic model is key to realistic and useful scenarios. Previous works are not directly comparable as they focus on small-scale scenario generation (*e.g*. [1, 8]) and/or attack the full AV stack rather than just the planner [36, 53]. Therefore, in the spirit of AdvSim [53] we implement the *Bicycle* baseline, which explicitly optimizes the kinematic bicycle model parameters (acceleration profile) of a single prechosen adversary in the scenario to cause a collision. Rather than using the learned traffic model, it relies on the bicycle

Planner	Scenarios	Collision (%)	Solution (%)	Planner Trajectory Accel (m/s^2)	Planner Trajectory Coll Vel (m/s)	Match Planner Err Pos (m)	Match Planner Err Ang (deg)
Replay	Challenging	43.7 (**+43.7**)	82.4	0.85	7.82	0.28	1.32
Rule-based	Regular	1.2	–	1.63	8.48	–	–
Rule-based	Challenging	27.4 (**+26.2**)	86.8	1.91 (**+0.28**)	9.65 (**+1.17**)	1.23	3.79

Table 1. Evaluation of generated *challenging* scenarios. Generated scenarios contain far more collisions compared to the corresponding *regular* (unmodified) scenes, as well as higher acceleration and collision speeds. Acceleration is measured in the forward direction (*i.e.* change in speed), since the *Rule-based* planner cannot change lanes. Rightmost columns show small errors between $\hat{\mathbf{Y}}^{\text{plan}}$ and $\mathbf{Y}^{\text{plan}}$.

Scenarios	Plausibility of Adversary Trajectory ↓ Accel (m/s^2)	Env Coll (%)	NN Dist (m)	NLL	Usefulness ↑ Solution (%)
Bicycle	2.00	16.5	0.97	962.9	73.4
STRIVE	**0.98**	**10.8**	**0.72**	**323.4**	**83.5**

Table 2. Scenario generation for *Replay* planner compared to the *Bicycle* baseline, which does not leverage a learned traffic model.

model, collision penalties, and acceleration regularization to maintain plausibility. This precludes using the differentiable planner approximation from the traffic model, thus requiring gradient estimation (*e.g.* finite differences) for the closed-loop setting, which we found is $\approx 40\times$ slower and requires several hours to generate a single scenario. Therefore, comparison is done only on the *Replay* planner.

Tab. 2 shows that scenarios generated by *Bicycle* exhibit more unrealistic adversarial driving, and are more difficult to find a solution for. All metrics are reported only for scenarios where both methods successfully caused a collision. In addition to higher accelerations, the *Bicycle* adversary collides with the non-drivable area more often (*Env Coll*), and exhibits less typical trajectories as measured by the distance to the nearest-neighbor ego trajectory in the nuScenes training split (*NN Dist*). After fitting the *Bicycle*-generated scenarios with our traffic model, we see the adversary's behavior is also less realistic as measured by the negative log-likelihood (NLL) of its latent $\mathbf{z}$ under the learned prior. These observations are supported qualitatively in Fig. 5.

5.2. Analyzing Generated Scenarios

Before improving a given planner, the analysis from Sec. 4.1 is used to identify useful scenarios by filtering out unsolvable scenarios and classifying collision types. For classification, collision features are clustered with $k = 10$ and clusters are visualized to manually assign the semantic labels shown in Fig. 6. The distribution of generated collision scenarios for both planners in Sec. 5.1 is shown in Fig. 6(a) (see supplement for visualized examples from clusters). STRIVE generates a diverse set of scenarios with solvable scenes found in all clusters. "Head On" is the most frequently generated scenario type, likely because *Replay* is non-reactive and *Rule-based* cannot change lanes. "Behind" exhibits the highest rate of unsolvable scenarios since being hit from behind is often the result of a negligent following vehicle, rather than undesirable planner behavior.

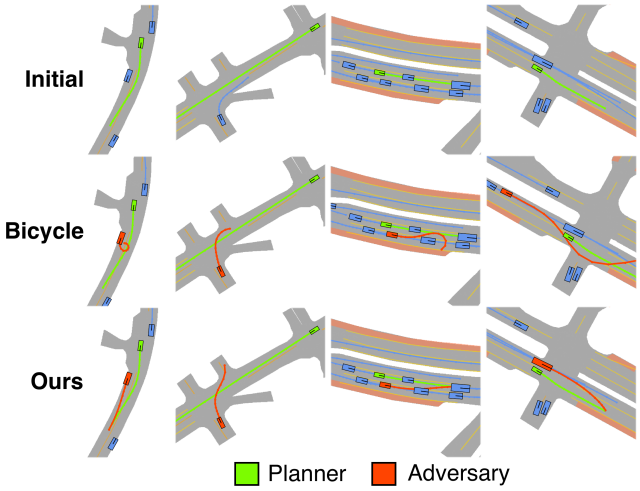

Figure 5. Qualitative comparison of generated scenarios for the *Replay* planner. *Bicycle* often produces semantically unrealistic trajectories as no learned traffic model is leveraged.

Replay is much more susceptible to being cut off since it is open-loop, while the closed-loop *Rule-based* can successfully react to avoid such collisions.

5.3. Improving Rule-Based Planner

Now that we have a large set of labeled collision scenarios, in addition to the original nuScenes data containing "regular" scenarios (with few collisions), we can improve the *Rule-based* planner to be better prepared for challenging situations. Besides uncovering fundamental flaws that lead us to add new functionality, improvement is based on hyperparameter tuning via a grid search over possible settings. For each set of hyperparameters, the planner is rolled out within all scenarios of a dataset, and the optimal tuning is chosen based on the minimum collision rate.

The planner is first tuned on regular scenarios before adversarial optimization is performed to create a set of challenging scenarios to guide further improvements. Performance of this initial regular-tuned planner on held out regular (*Reg*) and collision (*Coll*) scenarios is shown in the top row of Tab. 3. Before any improvements, the planner collides in 68.6% of challenging and 4.6% of regular scenarios. Note that avoiding collisions altogether on regular scenarios is not possible: even if we choose optimal hyperparameters for each scenario separately, the collision rate is still 3.2%.

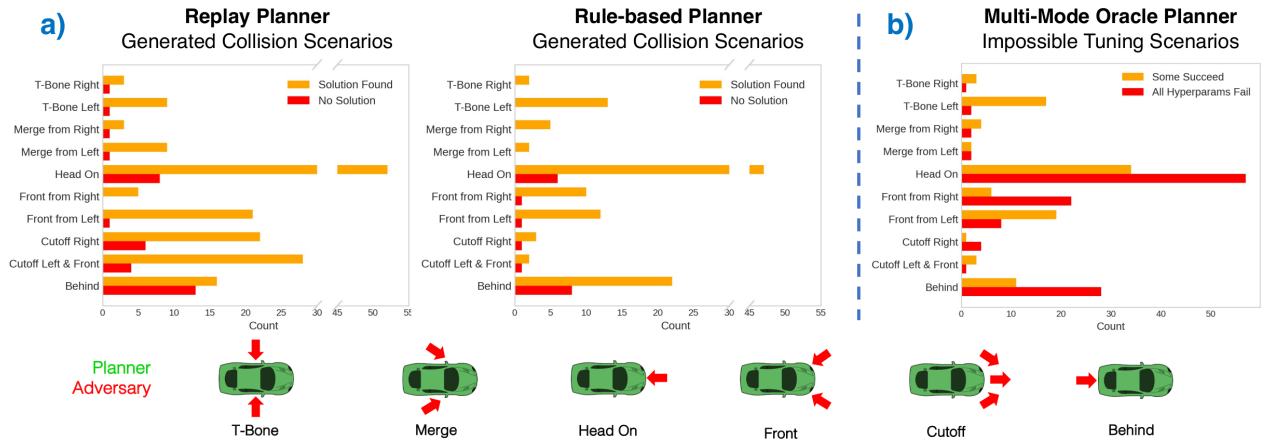

Figure 6. (Bottom) Collision types are depicted; the arrow indicates the position/direction of the adversary. **(a)** Distribution of generated scenarios for both planners. **(b)** Scenarios used to tune the multi-mode *oracle* planner. Scenarios where all parameter settings cause a collision are in red. A majority of *Head On*, *Front from Right*, and *Behind* scenarios always fail due to the inability to change lanes.

Improvement	**Collision** (%) Reg / Coll	**Coll Vel** (m/s) Reg / Coll	**Accel** (m/s^2) Reg / Coll
None (regular-tuned)	4.6 / 68.6	4.59 / 10.48	1.96 / 2.26
+ Challenging data	6.0 / 51.4	5.48 / 13.88	2.29 / 2.50
+ Extra *learned* mode	4.6 / 54.3	4.60 / 10.86	2.02 / 2.55
+ Extra *oracle* mode	4.6 / 54.3	4.59 / 10.40	1.96 / 2.39

Table 3. Improving *Rule-Based* planner. Including challenging tuning data and adding an extra "mode" improves performance on collision scenarios (*Coll*) while maintaining performance in regular scenarios (*Reg*). Acceleration is in the forward direction.

Tuning on Challenging Scenes. The first improvement, shown in the second row of Tab. 3, is to naïvely combine regular and challenging scenarios for tuning. Combined tuning greatly reduces the collision rate on challenging scenarios, but negatively impacts performance on regular driving. This points to a first *fundamental issue*: the planner uses a single set of hyperparameters for all driving situations, causing it to drive too aggressively in regular scenarios when tuned on challenging ones.

Multi-Mode Operation. To address this, we add a second set of parameters such that the planner has one for regular and one for accident-prone situations. Using this second "accident mode" of operation requires a binary classification of the current scene during rollout. For this, we augment the planner with a learned component that decides which parameter set to use based on a moving window of the past $2s$ of traffic; it is trained on scenarios generated by STRIVE. The extra parameter set is tuned on collision scenarios only. As shown in the third row of Tab. 3, this learned extra mode reduces the collision rate on challenging scenarios by 14.3% compared to the vanilla planner without hindering performance on regular scenes. We compare it to an *oracle* version (bottom row) that automatically switches into accident mode $2s$ before a collision is supposed to happen on generated scenarios, showing the learned version is achieving near-optimal performance.

Lane Change Limitation. The inability of the planner to switch lanes is another fundamental issue exposed by collision scenarios. Fig. 6(b) shows the distribution of tuning scenarios for the *oracle* multi-mode version; red bars indicate "impossible" scenarios where all sets of evaluated hyperparameters collide. A majority of "Head On" and "Behind" scenarios are impossible, pointing out the lane change limitation. Adversarial optimization has indeed exploited the flaw and the proposed analysis made it visible.

6. Discussion

STRIVE enables automatic and scalable generation of plausible, accident-prone scenarios to improve a given planner. However, remaining limitations offer potential future directions. Our method assumes perfect perception and only attacks the planner, but using our traffic model to additionally attack detection and tracking is of great interest. STRIVE generates scenarios from existing data and only considers collisions between vehicles, but other incidents involving pedestrians and cyclists are also important, and other kinds of adversaries like adding/removing assets and changing map topology will uncover additional AV weaknesses. Our method is intended to make AVs safer by exposing them to challenging and rare scenarios similar to the real world. However, our experiments expose the difficulty of properly balancing regular and challenging data when tuning a planner. Care must be taken to integrate generated scenarios into AV testing and to design unified planners that robustly address highly variable driving conditions.

Acknowledgments. Author Guibas was supported by a Vannevar Bush faculty fellowship. The authors thank Amlan Kar for the fruitful discussion and feedback throughout the project.

References

[1] Yasasa Abeysirigoonawardena, Florian Shkurti, and Gregory Dudek. Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277. IEEE, 2019. 1, 2, 6

[2] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics: Science and Systems (RSS)*, 2019. 2, 3

[3] Luca Bergamini, Yawei Ye, Oliver Scheel, Long Chen, Chih Hu, Luca Del Pero, Błażej Osiński, Hugo Grimmet, and Peter Ondruska. Simnet: Learning reactive self-driving simulations from real-world observations. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021. 2

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 2, 6

[5] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 624–641. Springer, 2020. 2, 3

[6] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021. 3

[7] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning (CoRL)*, pages 86–99. PMLR, 2020. 2

[8] Baiming Chen, Xiang Chen, Qiong Wu, and Liang Li. Adversarial evaluation of autonomous vehicles in lane-change scenarios. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 1, 2, 6

[9] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 2

[10] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *European Conference on Computer Vision*, pages 715–733. Springer, 2020. 2

[11] Wenhao Ding, Baiming Chen, Bo Li, Kim Ji Eun, and Ding Zhao. Multimodal safety-critical scenarios generation for decision-making algorithms evaluation. *IEEE Robotics and Automation Letters*, 6(2):1551–1558, 2021. 1, 2

[12] Wenhao Ding, Baiming Chen, Minjun Xu, and Ding Zhao. Learning to collide: An adaptive safety-critical scenarios generating method. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2243–2250. IEEE, 2020. 1, 2

[13] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, Nitin Singh, and Jeff Schneider. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2095–2104, 2020. 2

[14] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 2

[15] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. *ICCV*, 2021. 2

[16] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020. 2

[17] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 2

[18] Zahra Ghodsi, Siva Kumar Sastry Hari, Iuri Frosio, Timothy Tsai, Alejandro Troccoli, Stephen W Keckler, Siddharth Garg, and Anima Anandkumar. Generating and characterizing scenarios for safety testing of autonomous vehicles. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 157–164. IEEE, 2021. 2

[19] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. https://level-5.global/level5/data/, 2020. 2

[20] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019. 2

[21] Daniel Johnson, Hugo Larochelle, and Daniel Tarlow. Learning graph structure with a finite-state automaton layer. *Advances in Neural Information Processing Systems*, 33:3082–3093, 2020. 2

[22] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4551–4560, 2019. 2

[23] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Level 5 perception dataset 2020. https://level-5.global/level5/data/, 2019. 2

[24] Seung Wook Kim, , Jonah Philion, Antonio Torralba, and Sanja Fidler. DriveGAN: Towards a Controllable High-Quality Neural Simulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021. 2

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 5

[26] Moritz Klischat and Matthias Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2352–2358. IEEE, 2019. 1, 2

[27] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)*, pages 1094–1099. IEEE, 2015. 2, 3

[28] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[29] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal*, 1(1):1–14, 2014. 2

[30] Yiming Li, Congcong Wen, Felix Juefei-Xu, and Chen Feng. Fooling lidar perception via adversarial trajectory perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7898–7907, 2021. 2

[31] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020. 2

[32] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7577–7586, 2021. 2

[33] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE, 2018. 2

[34] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008. 3

[35] Wassim G Najm, John D Smith, and Mikio Yanagisawa. *Pre-Crash Scenario Typology for Crash Avoidance Research*. U.S. Department of Transportation, National Highway Traffic Safety Administration, 2007. 3

[36] Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *NeurIPS*, 2018. 1, 2, 6

[37] Avik Pal, Jonah Philion, Yuan-Hong Liao, and Sanja Fidler. Emergent road rules in multi-agent driving environments. In *International Conference on Learning Representations*, 2020. 2

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems*, 2017. 5

[39] Jonah Philion, Amlan Kar, and Sanja Fidler. Learning to evaluate perception models using planner-centric metrics. In *CVPR*, 2020. 2

[40] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 812–818, 2017. 3

[41] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11488–11499, 2021. 2

[42] Cinjon Resnick, Or Litany, Amlan Kar, Karsten Kreis, James Lucas, Kyunghyun Cho, and Sanja Fidler. Causal bert: Improving object detection by searching for challenging groups. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2972–2981, October 2021. 2

[43] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *European Conference on Computer Vision*, pages 414–430. Springer, 2020. 3

[44] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3949–3956. IEEE, 2019. 3

[45] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020. 2

[46] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 2

[47] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021. 2, 3, 4

[48] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for li-

dar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020. 2

[49] Christopher Urmson, Joshua Anhalt, J. Andrew (Drew) Bagnell, Christopher R. Baker, Robert E. Bittner, John M. Dolan, David Duggins, David Ferguson, Tugrul Galatali, Hartmut Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Alonzo Kelly, David Kohanbash, Maxim Likhachev, Nick Miller, Kevin Peterson, Raj Rajkumar, Paul Rybski, Bryan Salesky, Sebastian Scherer, Young-Woo Seo, Reid Simmons, Sanjiv Singh, Jarrod M. Snider, Anthony (Tony) Stentz, William (Red) L. Whittaker, and Jason Ziglar. Tartan racing: A multi-modal approach to the darpa urban challenge. Technical report, Carnegie Mellon University, Pittsburgh, PA, April 2007. 3

[50] Sai Vemprala and Ashish Kapoor. Adversarial attacks on optimization based planners. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9943–9949. IEEE, 2021. 1, 2

[51] Matt Vitelli, Yan Chang, Yawei Ye, Maciej Wołczyk, Błażej Osiński, Moritz Niendorf, Hugo Grimmett, Qiangui Huang, Ashesh Jain, and Peter Ondruska. Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. *arXiv preprint arXiv:2109.13602*, 2021. 3

[52] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. Ieee, 2000. 2

[53] Jingkang Wang, Ava Pun, James Tu, Sivabalan Manivasagam, Abbas Sadat, Sergio Casas, Mengye Ren, and Raquel Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021. 1, 2, 3, 6

[54] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 3