SLEDGE: Synthesizing Simulation Environments for Driving Agents with Generative Models

Kashyap Chitta* Daniel Dauner* Andreas Geiger

University of Tübingen Tübingen AI Center https://github.com/autonomousvision/sledge

Abstract. SLEDGE is the first generative simulator for vehicle motion planning trained on real-world driving logs. Its core component is a learned model that is able to generate agent bounding boxes and lane graphs. The model's outputs serve as an initial state for traffic simulation. The unique properties of the entities to be generated for SLEDGE. such as their connectivity and variable count per scene, render the naive application of most modern generative models to this task non-trivial. Therefore, together with a systematic study of existing lane graph representations, we introduce a novel raster-to-vector autoencoder (RVAE). It encodes agents and the lane graph into distinct channels in a rasterized latent map. This facilitates both lane-conditioned agent generation and combined generation of lanes and agents with a Diffusion Transformer. Using generated entities in SLEDGE enables greater control over the simulation, e.g. upsampling turns or increasing traffic density. Further, SLEDGE can support 500m long routes, a capability not found in existing data-driven simulators like nuPlan. It presents new challenges for planning algorithms, evidenced by failure rates of over 40% for PDM. the winner of the 2023 nuPlan challenge, when tested on hard routes and dense traffic generated by our model. Compared to nuPlan, SLEDGE requires $500 \times$ less storage to set up (<4GB), making it a more accessible option and helping with democratizing future research in this field.

1 Introduction

While recent breakthroughs in generative AI have revolutionized natural image synthesis [4, 16], generative models are yet to find widespread adoption in autonomous driving. In contrast to the regular pixel grid of images, self-driving planners typically require abstract bird's eye view (BEV) representations as input which characterize the most important scene elements (e.g., lanes, traffic lights, static and dynamic objects) in a compact, vectorized format (see Fig. 2). These representations are a key component of data-driven simulators which are necessary for rigorous evaluation of planners [11, 12, 25, 50]. However, learning a generative model on such irregular vectorized representations is hard.

Consequently, many existing data-driven simulators [17, 22] are initialized by simply replaying logs of abstract representations. They extract the local lane

^{*}equal contribution



Fig. 1: SLEDGE. We show state snapshots of simulation environments generated by our approach in 4 cities, with the lanes, ego-vehicle, other vehicles, pedestrians, obstacles, and traffic lights. Our supplementary video visualizes clips with more examples.

graph from a High Definition (HD) map and object bounding boxes from prerecorded annotation logs. They test planning algorithms by simulating short scenarios (~15 seconds long) in which the algorithms must safely traverse the route followed in the original log. However, to provide sufficient diversity among routes for comprehensive testing, these simulators require huge databases, e.g., nuPlan [22] consists of 1300 hours of driving logs which require over 2TB of storage. Such high resource requirements heighten the barrier for entry into the field of vehicle motion planning.

In this paper, we study the task of generating simulation-ready abstract representations of driving scenes. Using generative models as an alternative to log replays has the potential for significant compression [39]. Furthermore, abstract representations are far lower dimensional than visual data, simplifying the generation task. However, the unique characteristics of abstract representations in driving scenes pose new challenges to modeling them, e.g., they require accurate topological connectivity and precise modeling of geometry (e.g., parallel/perpendicular lines) in order to be useful for simulation.

To tackle these complexities, we first perform a systematic study of the existing representations of lane graphs used in autonomous driving. We then propose a novel representation based on a raster-to-vector autoencoder (RVAE). It represents a driving scene with a fixed-size BEV rasterized latent map (RLM). We learn to generate these RLMs with a Diffusion Transformer (DiT) [35]. Our model generates high-fidelity results enabling both lane-conditioned agent generation or joint lane and agent generation within a single flexible and scalable framework (Fig. 1). Independent to its use of generative models, SLEDGE also provides the previously missing functionality of simulating only agents within a certain radius of the ego-vehicle. By doing so, we can test on routes that are significantly longer than those currently used for evaluating planners. We find previously ignored failure modes of the state-of-the-art PDM-Closed planner [12], which is unable to complete 25-50% of our new 500m long test routes despite having failure rates below 10% on existing benchmarks. **Contributions.** (1) We formalize the task of abstract scene generation for autonomous driving with a challenging benchmark and corresponding metrics. (2) We perform a systematic exploration of modern generative models (with various architectures and representations) and propose a novel latent diffusion model for synthesizing abstract driving scenes that largely outperforms other baselines. (3) We present a simulation framework, SLEDGE, which is nearly 3 orders of magnitude more storage efficient than nuPlan yet enables more rigorous testing of motion planning algorithms. SLEDGE will be made publicly available.

2 Related Work

Diffusion Models. Best known for their success in generative modeling of images [16,38] and video [3,4,48], diffusion models have recently found widespread adoption in diverse domains, including point clouds [30,33,49], floor plans [7,41], molecules [47], robot policies [9], traffic patterns [51], and many others. Scenario Diffusion [36], a pioneering approach in generating vehicles conditioned on HD maps with diffusion, is the closest existing approach to ours. This method uses latent diffusion with a raster-based vehicle decoder unlike our proposed transformer decoder head. Importantly, we offer significantly increased capabilities in comparison: generation of lane graphs (enabling settings both with and without map conditioning), support for pedestrians, static obstacles, and traffic lights, as well as long-horizon simulation environments with reactive agents.

Generating Lane Graphs. Lane graphs, being the most important component of HD maps, are well-studied in the autonomous driving domain. They are often constructed through an expensive offline mapping process often involving human annotators [15]. However, a surge of recent work on predicting lane graphs from sensor data [24, 26] has sparked interest on generative modeling of these lane graphs. The first and only existing research study on this task, HDMap-Gen [32], proposes an autoregressive approach for generating lane graphs nodeby-node [10]. Our experiments show that this achieves reasonable results, but is unable to match the high quality and scalability offered by diffusion models. Unlike HDMapGen, our approach jointly generates agents with lanes, and efficiently generates all scene elements in parallel. A concurrent project, DriveScene-Gen [42], generates lanes and vehicles with image-space diffusion. Our approach covers agent types beyond vehicles which are critical for simulation, uses less heuristics, and is more efficient via the use of compact latent representations. Additionally, we show the successful integration of our model into a simulator. which we then use for the downstream task of evaluating planning algorithms.

Data-driven Simulation. Developing an autonomous driving system necessitates rigorous testing which is costly and risky if conducted in the real world. Driving simulators are a valuable alternative [14,22,46]. However, simulators face challenges in ensuring realism while initializing traffic scenes, simulating traffic, and providing sensor data, for which they primarily depend on rule-based solutions [8]. Instead, data-driven simulators address these challenges by replaying traffic scenes from real-world recordings [1, 17, 22, 23]. The simulator can mine specific situations or even optimize the initial parameters for safety-critical scenarios [13, 18, 45]. Leveraging modern generative models, we take a step further and learn the underlying distribution of the real-world data. This leads to a flexible, compact and controllable simulator that goes beyond the replay-based nuPlan framework [5, 22] which we build upon.

3 SLEDGE

Our goal is to design a driving scene synthesis framework that can be trained using real-world driving logs and incorporated into SLEDGE, our generative simulator. We base this framework on LDMs [38] as: (1) latent diffusion shows excellent training stability and scalability with compute. (2) One can easily construct a fixed-size latent space for diffusion that can be mapped to the variable sized set representation for simulation (Section 3.1) using detection-based transformer architectures [6, 26]. Our LDM is trained in two stages: an autoencoder (Section 3.2) followed by a diffusion model (Section 3.3). We detail the simulation of scenes generated by the LDM in Section 3.4.

3.1 nuPlan Vector Representation

We represent a scene state abstractly in the format native to the nuPlan simulator. The representation combines several sets of entities, as outlined below.

Lanes. Our focus is on the generation of lanes, the central element of HD maps used in data-driven simulation. Each lane $\mathbf{L} \in \mathbb{R}^{20 \times 2}$ is geometrically represented by a polyline, i.e., a fixed set of 20 bird's eye view (BEV) points. These are bounded by two endpoints and form the lane centerline, connected along the driving direction. A lane may share endpoint(s) with predecessor and successor lanes. This information is encoded in an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, where N is the number of lanes in a certain field of view (FOV). The set of all lane polylines \mathcal{L} form the lane graph of the local map $\mathcal{M} = \{\mathcal{L}, \mathbf{A}\}$.

Traffic Lights. We then augment the lane graph with polylines representing traffic lights. These share the same 20×2 format as lanes, and come in two types (red and green). The set of red polylines (\mathcal{R}) contains the lane regions that are currently not traversable due to a red traffic light. The set of green polylines (\mathcal{G}), on the other hand, contains entities indicating lane segments where the road is safe to proceed along due to the presence of a green traffic signal.

Agents. Further, we expand the scene representation using oriented bounding boxes for the agents. Each bounding box is defined by a 2D center position, heading, 2D extent and optional speed. We consider three types of agent sets: pedestrians (\mathcal{P}) , vehicles (\mathcal{V}) and static objects (\mathcal{O}) . Pedestrians and vehicle boxes are assigned a speed attribute, whereas static objects are not.

Ego Velocity. Finally, initializing a simulation requires the BEV ego velocity $\mathbf{v} \in \mathbb{R}^2$. Overall, we denote the scene state as $\mathcal{S} = \{\mathcal{M}, \mathcal{R}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{O}, \mathbf{v}\}.$



Fig. 2: Rasterized State Image (RSI). We encode S into a 12-channel image, with 2 channels per entity type. We visualize these encodings as optical flow fields.

3.2 Raster-to-Vector Autoencoder

Each entity type in S is unique. To maintain overall scene consistency, we would like to model them with a single architecture, instead of creating several independent entity-specific generative models. Furthermore, most existing tools in the literature have been developed and optimized for 2D input domains [38]. To this end, we propose the raster-to-vector autoencoder (RVAE) which unifies all entity types in S into a compact, shared 2D representation well-suited for diffusion modeling. The RVAE is inspired by work in object detection [6] and online mapping for autonomous driving [26].

Rasterization. We first define a function $\rho: S \to \mathbf{I}$ that encodes the scene state into a rasterized state image (RSI) $\mathbf{I} \in \mathbb{R}^{W \times H \times 12}$. Our design of ρ is motivated by (and closely resembles) techniques used in motion planners [2, 37]. As shown in Fig. 2, ρ maps the three polyline entity types ($\mathcal{L}, \mathcal{R}, \mathcal{G}$) and three bounding box entity types ($\mathcal{P}, \mathcal{V}, \mathcal{O}$) to image pixel locations, assigning 2 channels to each entity type which encode all their attributes. For polylines, we use a 2D directional vector $\Delta = [dx, dy]$, which points from any point to its successor, indicating the presence of a polyline traversing a specific pixel. A background value (i.e., [0,0]) is assigned to other regions. For a bounding box type entity, we rasterize it in BEV according to its position, extent, and heading. For dynamic bounding boxes (\mathcal{P}, \mathcal{V}), the values in the two channels within the box region represent the entity's 2D velocity. For static obstacles, we fill the rasterized region with the obstacle's orientation vector. We use a square field of view centered at and oriented as per the ego vehicle's pose for rasterization. The ego velocity \mathbf{v} is encoded at the origin of \mathbf{I} as an extra rasterized vehicle in \mathcal{V} .

Vectorization. This step is necessary to decode the unified RSI representation I back into the per-entity attributes (e.g., polylines, bounding boxes). We use a learned vectorization pipeline consisting of a raster encoder π and vector decoder head ϕ . The ResNet-50 [19] encoder takes the RSI and outputs a rasterized latent map (RLM) $\mathbf{M} = \pi(\mathbf{I})$ of shape $W' \times H' \times C$, where $H' = H/2^d$ and $W' = W/2^d$



Fig. 3: Raster-to-Vector Autoencoder (RVAE). We represent scenes with a rasterized latent map (RLM) consisting of two channel groups. The 'Lanes' group is decoded into lane segments and the 'Agents' group into all other scene entities, via a transformer decoder with attention masking. The RVAE is trained to predict polylines, bounding boxes, and the ego velocity in a simulation-compatible vectorized format.

for a downsampling factor $d \in \mathbb{N}$. C is a chosen channel dimension. In practice, the RLMs we use are compact, H' = W' = 8 and C = 64. Additionally, as shown in Fig. 3, we split the RLM's channels into 2 groups, $C = C_L + C_A$ for the lanes and agents respectively. For each group, we tokenize the latent vectors spatially, resulting in $W' \times H'$ lane tokens and $W' \times H'$ agent tokens. Following the DETR [6] paradigm, we use these tokens as keys and values for our transformer decoder ϕ . The decoder uses a fixed number of learnable queries of each entity type, which we cap to a maximum count per entity, based on statistics from our dataset. The final decoder layer is unique per entity type and outputs the attributes specific to that entity, e.g. a 20×2 set of point coordinates for a polyline, or a 6-dimensional descriptor (2D position, orientation, 2D extent, and speed) for a bounding box. It also outputs an existence attribute $p \in [0, 1]$ for both polylines (p_{line}) and bounding boxes (p_{box}), which is used to handle variable counts of ground truth entities with a fixed number of queries.

Channel Group Masking. The motivation behind our design with two token groups is to enable agent generation conditioned on known lanes. To this end, the tokens for agents should contain no information about lanes. We implement a binary mask in the cross-attention mechanism of ϕ to achieve this. Specifically, queries for lanes \mathcal{L} are prevented from attending to the keys and values of the agents tokens, and all other queries (i.e., $\mathcal{R}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{O}, \mathbf{v}$) cannot attend to the lanes tokens. Our experiments show the effectiveness of this approach.

Training. The RVAE is optimized using both reconstruction and existence losses on the decoder outputs, and a KL divergence loss on the RLM. For reconstruction, we first match generated and ground truth entities using the Hungarian algorithm, as in [6]. We use a matching score based on the L1 error of the entity's position attributes. We then use the L1 error summed over all attributes and averaged over all matches as the training loss. For the existence variable, we use a binary cross entropy loss based on whether the query was matched to a ground truth entity. We provide additional details in the supplementary material.



Fig. 4: Route Extrapolation by Inpainting. We show an example scenario generated by our DiT, where we iteratively sample poses along a route, warp the previous tile's RSI to this pose, and generate a new tile conditioned on the warped RSI.

3.3 Diffusion Transformer

We obtain RLMs **M** for each training example via the frozen, pretrained encoder π and use them to train a diffusion model δ with the DDPM algorithm [21].

Training. For each scene, we sample a noise scaling factor σ from a log-normal distribution and create a noisy sample $\hat{\mathbf{M}} = \mathbf{M} + \sigma \boldsymbol{\mathcal{E}}$, with $\boldsymbol{\mathcal{E}}$ being sampled from a standard normal distribution of the same shape as \mathbf{M} . We model $\delta(\hat{\mathbf{M}}; \mathbf{c}, \sigma)$ as a Transformer [44], following DiT [35], where \mathbf{c} is a conditioning vector. For our experiments, we choose \mathbf{c} to be a one-hot class label indicating the city to which the example belongs, which is readily available in nuPlan. However, it is easily possible to adapt our framework to other conditioning, e.g., images, text descriptions, or learned cluster labels based on RVAE features. The DiT architecture is simple, scalable, and free from down- or upsampling operations, making it compatible with RLMs of any spatial resolution. It applies a series of self-attention blocks to the tokenized input $\hat{\mathbf{M}}$, with the conditioning on \mathbf{c} and σ implemented using AdaLN-Zero [35]. To train the model, we optimize the L2 reconstruction loss between $\boldsymbol{\mathcal{E}}$ and $\delta(\hat{\mathbf{M}}; \mathbf{c}, \sigma)$.

Generation. During inference, we begin with an initial noisy sample $\mathbf{M} \sim \mathcal{N}(0, \sigma_{\max}^2 \mathbf{I})$, which undergoes iterative refinement from $\sigma = \sigma_{\max}$ to $\sigma = 0$ based on the reverse PDE defined via δ [21]. We then use the trained vector decoder ϕ from Section 3.2 to predict $\mathcal{L}, \mathcal{R}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{O}$, and \mathbf{v} . Only entities with an existence probability above a threshold τ are retained, and for overlapping bounding boxes, those with the highest existence probability are kept. The process of recovering the full scene state \mathcal{S} further involves extracting the adjacency matrix \mathbf{A} of the lane graph, which is not an explicit output of our model. We do this by simply matching lanes whose start and endpoints lie within a range of 1.5 meters with orientations differing by less than 60 degrees, which we found to be robust in practice given our highly accurate lane polyline predictions.

Conditional Generation via Inpainting. Diffusion models excel at inpainting, even without explicit training for this task [29]. Specifically, by executing the denoising process on only a subset of the tokens in the noisy sample $\hat{\mathbf{M}}$, with the remaining tokens extracted from a known and encoded scene, we can inpaint RLMs. We use this capability to perform two tasks: (1) lane conditioned agent generation and (2) route extrapolation. Lane conditioned agent generation involves encoding all lane tokens from a known map, and denoising all agent tokens. Route extrapolation, as illustrated in Fig. 4, involves encoding a subset of tokens within a known spatial region to denoise the unknown region. Specifically, we can iteratively sample poses along a generated route, warp the previously generated scene's RSI to the new pose with an affine transformation, and use a known region as conditioning for completing a newly created tile. We provide implementation details in the supplementary material.

3.4 SLEDGE Simulation Environments

Finally, we initialize a reactive simulation in SLEDGE using the generated initial scene state S. In the following, we provide an overview of the steps involved.

Hard Routes and Traffic. To evaluate a planner in ambiguous situations, we must specify the driver intention, e.g. whether to turn left or right at an intersection. Existing replay-based simulators offer limited controllability over this, since they are unable to extract agents to simulate if the planner diverges significantly from the route followed by the human driver from the log recording. However, for generated scenes, we can extract multiple valid routes from the lane graph, e.g., we define 'hard' routes by selecting the route with the highest number of turns. In addition, our approach also provides a degree of control over traffic configurations along the desired route, and selecting the configuration with the largest number of generated agents. Our experiments show that these 'hard' settings provide new challenges to the state of the art for planning.

Behavior Simulation. We simulate non-ego vehicles in SLEDGE by projecting each to the center of a lane based on proximity and heading, from which it then laterally follows the lane centerlines. For longitudinal control, we use a simple policy called the Intelligent Driver Model (IDM) [43]. Upon choosing a connected lane sequence as a driving path, IDM calculates a longitudinal trajectory, iteratively adjusting acceleration based on current position, velocity, and distance to the leading vehicle. For pedestrians, we assume a constant velocity and heading while unrolling the simulation. Traffic lights are hard-coded to change states every 15 seconds. While these choices are simplistic, they are in line with the capabilities of today's best data-driven simulators [17, 22]. SLEDGE is not incompatible by design with other types of policies for vehicle and pedestrian simulation, but we leave this exploration to future work.

Simulation Radius. By default, existing data-driven simulators like nuPlan simulate all the initialized agents at all timesteps. This severely limits the scalability of these simulators to long simulation horizons or large scenes. We propose a simple modification for SLEDGE wherein we only simulate agents at a distance below α =64m to the ego vehicle at a given timestep, while holding the



Fig. 5: Long Route Simulation. SLEDGE supports (a) replayed scenarios, (b) laneconditioned agent generation, and (c) joint lane and agent generation. Importantly, we enable testing on arbitrarily long routes by dynamically simulating agents near the ego vehicle while keeping the state of distant agents fixed.

state of all other agents to be constant (Fig. 5). We demonstrate the scalability this provides by conducting experiments on simulations that are $10 \times$ longer than those in nuPlan, i.e., up to 150 seconds as opposed to nuPlan's 15 seconds.

4 Experiments

We now present our experimental results. (1) We demonstrate the suitability of the RLM as a lane graph representation. (2) We benchmark a series of models for lane graph generation. (3) We showcase the effectiveness of SLEDGE environments for evaluating planners. For all experiments, we present concise descriptions of baselines, metrics and implementations in the main paper. Additional details can be found in the supplementary material.

Dataset. We use nuPlan [22], the largest publicly available dataset for vehicle motion planning. It comprises 1300 hours of logs from 4 cities. We sample 450k train and 50k validation frames from these logs with sampling intervals of 30s, 1s, 2s and 2s between frames for Las Vegas, Boston, Pittsburgh and Singapore respectively, in order to obtain a balanced distribution while achieving high map coverage. Each city offers unique challenges for generative modeling, e.g. Las Vegas has large and dense intersections, while Singapore involves left-hand traffic. Each frame is limited to a $64m \times 64m$ FOV centered at the ego vehicle.

4.1 Lane Graph Representations

In our first experiment, we evaluate various representations based on their ability to reconstruct the complete directed lane graph $\mathcal{M} = \{\mathcal{L}, \mathbf{A}\}$.

Baselines. We consider three baseline representations. (1) **RSI:** we use the skan library [34] to extract vector polylines from the $256 \times 256 \times 2$ RSI representation of the lane graph. Skan is a highly optimized image processing pipeline for graph extraction (details and visualizations in supplementary). (2) **RLM w/o mask:** we train the RVAE without the channel group masking proposed in Section 3.2,

which entangles information about agents and lanes into a single $8 \times 8 \times 64$ latent map instead of an $8 \times 8 \times 32$ tensor for the lane graph and an independent $8 \times 8 \times 32$ tensor for agents. (3) Vector: As an upper bound, we additionally compute our metrics for the respresentation used as target labels for the RVAE during training. This is a set of polylines shaped $N \times 20 \times 2$, where we cap N at 30 in our experiments. Note that the ground truth for evaluation in this experiment uses all polylines in the scene (which is sometimes greater than 30).

Metrics. Our metrics are adapted from the street map extraction literature [20]. We use three base metrics, which all operate on point sets sampled along graphs at a resolution of one point every 1.5m. (1) F1: measures the harmonic mean of the precision and recall, which are estimated using Hungarian matching between point sets with a distance threshold of 1.5m. Intuitively, this penalizes large structural errors, while ignoring small positional offsets. (2) Lateral L2 (Lat.): averaged over all true positive matched points, measures the lateral offset of each such point from its nearest ground truth lane centerline. In contrast to F1, it penalizes positional errors, while ignoring structurally incorrect and unmatched lanes. (3) Chamfer: averaged over all points in two point sets, this is the distance of each point to the closest in the other set. It requires both precise structure and details. These three base metrics are further applied in two settings. (1) GEO uses point sets sampled from the complete graph, making it independent of the predicted adjacency matrix A. On the other hand, (2) **TOPO** uses **A** to extract sets of fully-connected sub-graphs corresponding to every tenth node of the graph (i.e., every 15 meters). The base metrics are computed on these sub-graphs and averaged. Errors in A can lead to large missing sections of such sub-graphs, making TOPO suitable for evaluating connectivity.

Results. As shown in Table 1, the RSI is unable to match the RLM on all TOPO and both the F1 and Chamfer GEO metrics. Despite the significantly larger representation size of 524 KB per $64m \times 64m$ scene, the reliance on heuristics to convert this representation back to its constituent entities serves as a major limiting factor for the RSI. Qualitatively, we observe that the key issue is dense groups of lines that nearly overlap at forks or cross over in intersections. For the 2 variants of RLM, we obtain similar reconstruction quality, largely closing the gap towards the upper bound vector representation. Importantly, the proposed channel group masking for the RLM (i.e., 'Split') disentangles the lane graph from the agents with no impact on the graph reconstruction fidelity. Finally, as we cap the maximum number of polylines to 30, we observe a very minor error rate (<1% drop in F1) in the upper bound vector representation, corresponding to the negligible fraction of scenes with over 30 lanes. However, the size of the vector representation varies significantly per scene, ranging from 2 to 30 lanes. The RLM achieves an ideal balance of high quality with a fixed size.

4.2 Lane Graph Generation

Next, we compare our proposed DiT to several generative models for lane graphs.

Table 1: $64m \times 64m$ Lane Graph Reconstruction. We show the F1 score, lateral displacement and Chamfer distance for graphs extracted from each representation. We additionally include qualitative results (more in supplementary material). The RSI struggles with nearly overlapping segments at the beginning of forks (FNs) and overlaps at intersections (FPs). The RLM closes the gap towards the upper bound (vector).

Rep.	Fixed?	Split?	Size		GEO	торо			
reept		opnor	(KB)	$F1\uparrow$	$ $ Lat. $\downarrow $ Ch. \downarrow	$\left \begin{array}{c c} F1\uparrow & Lat.\downarrow & Ch.\downarrow \end{array}\right.$			
RSI	1	1	524.3	0.933	0.133 0.423	0.851 0.438 64.824			
RLM	1	×	$\begin{array}{c} 16.0\\ 8.0\end{array}$	$0.981 \\ 0.980$	$\left \begin{array}{cc c} 0.161 & 0.399 \\ 0.164 & 0.411 \end{array}\right $	$ \begin{vmatrix} 0.945 & 0.282 & 20.096 \\ 0.944 & 0.288 & 20.624 \end{vmatrix} $			
Vector	×	1	4.8	0.997	0.005 0.070	0.990 0.010 4.174			
		RSI		RSI • TP • FP × FN		RLM RLM			

Baselines. We select four diverse baselines. (1) VAE: we train a convolutional VAE with a 2D decoder head to generate RSIs. (2) RVAE: we sample from the decoder ϕ of our proposed RVAE. (3) HDMapGen: an autoregressive hierarchical graph neural network for lane graph generation [32], reimplemented for nuPlan. (4) DiT (RSI): similar to the concurrent DriveSceneGen [42], this is an image diffusion model that generates the RSI.

Metrics. Lane graph generation does not have established evaluation protocols. Therefore, we use a comprehensive set of seven metrics. (1) Route Length measures the mean and std of the longest valid ego vehicle route in the $64m \times 64m$ FOV for 1k generated graphs. The remaining six metrics are Frechet distances in different feature spaces measured using 50k generated and ground truth graphs. Two of these involve rasterizing lane graphs into the RSI, and using the penultimate feature vector of a ResNet-50 for the Frechet distance. (2) ImNet uses a ResNet trained on ImageNet, and (3) RVEnc uses the encoder π of our RVAE. The remaining metrics are taken from [10, 32], based on graph features used in urban planning. They operate on all nodes of the generated lane graphs with degree $\neq 2$, which are referred to in [32] as key points. (4) Connectivity: this uses the degrees of all key points. (5) Density: the number of key points in the $64m \times 64m$ FOV. (6) Reach: the number of valid paths found from key points to others. (7) Convenience: Lengths of all valid paths from all key points. The Frechet metrics are scaled by suitable powers of 10 for readability.

Results. Our results are shown in Table 2. All DiT variants generate more plausible layouts than the VAE, RVAE and HDMapGen, with significantly better

Table 2: 64m×64m Lane Graph Generation. We show the route lengths and
various Frechet distances between samples from the validation set and model outputs.
We additionally include qualititve results (more examples in supplementary material).
Latent diffusion combining DiT with an RLM obtains the best results. *Trained with
$\sim 6 \times$ more compute than others, which already converge at the lower compute budget.

Arch.	Repr.	Route	$ $ Frechet (CNN) $\downarrow $ Frechet (Urban Planning) \downarrow								
	respri	$\mathbf{Length} \uparrow$	ImNet	RVEnc	Conne.	Densi.	Reach	Conve.			
VAE	RSI	2.68 ± 3.66	21.06	14.77	9.45	0.99	2.86	13.06			
RVAE	Vector	23.79 ± 9.96	9.27	14.52	15.63	12.57	3.08	17.72			
HDMapGen	Vector	28.17 ± 14.81	9.18	13.97	7.02	3.03	2.49	18.10			
D:T I	RSI	24.78 ± 10.38	1.14	7.27	6.11	15.33	1.90	3.95			
DII-L	RLM	32.51 ± 9.93	1.30	2.36	2.35	3.52	0.88	3.10			
DiT-XL*	RLM	$\textbf{35.37} \pm \textbf{10.28}$	2.08	3.38	0.27	2.47	0.20	0.47			

VAE (RSI)	RVAE (Vec.)	HDMapGen (Vec.)	DiT-L (RSI)	DiT-L (RLM)	DiT-XL (RLM)
度引					
		X			

metrics. For DiT-L, we observe higher visual fidelity in intersection areas when using the RLM representation instead of the RSI. In particular, the RLM-based models excel at creating coincident endpoints between connected lanes, which is crucial for smooth simulation. Scaling to DiT-XL provides further gains, leading to results competitive to the upper bound of replay. Among the metrics used in our study, we find the CNN Frechet distances do not value connected drivable lanes. In particular, the ImNet Frechet distance increases when moving from the RSI to RLM or from DiT-L to DiT-XL, despite visual evidence of reduced fidelity (see supplementary for larger sets of randomly generated samples from each model). Therefore, we focus on Reach in our subsequent experiments.

Scaling. We run a systematic analysis of scaling for our DiT with the RLM representation. We conduct a grid of experiments, considering (1) 2 model sizes: DiT-B and DiT-L, (2) 3 dataset sizes: $1\times$, $0.5\times$, and $0.25\times$ our full dataset, and (3) 3 compute budgets: 24, 48 and 96 GPU hours. As shown in Fig. 6, the performance scales significantly with increased compute. While it also scales with more parameters, more data does not have a large impact. This is possibly because data diversity is more valuable than scale, and all of our 3 datasizes have similar diversity. This scaling behavior shows the potential of further improvements for SLEDGE with more training resources.



Fig. 6: Scaling. The DiT's performance scales significantly with increased compute. For our task, dataset size is less crucial, with all settings performing similarly.

4.3 SLEDGE Simulation of PDM-Closed

In our final experiment, we use the inpainting capabilities of DiT-XL in SLEDGE to demonstrate its utility for testing vehicle motion planners.

Tasks and Settings. We consider the two inpainting-based tasks described in Section 3.3: lane conditioned agent generation (Lane \rightarrow Agent) and joint lane and agent generation via route extrapolation (Lane & Agent). For the Lane \rightarrow Agent task, we consider 100 existing nuPlan logs and use the original route extracted from these logs as the 'easy' route. 'Hard' routes are generated from the same initial pose while maximizing the number of turns. For the **Lane** & Agent task, as we no longer rely on the nuPlan maps, we dynamically adjust the difficulty of generated routes during DiT inpainting. We generate 100 scenarios. For each, we perform iterative inpainting starting from an initial $64m \times 64m$ area. and perform a depth-first search on the updated lane graph at every inpainting step. 'Hard' routes correspond to selecting the the next point to inpaint from to be the endpoint reachable with the most new turns, and 'easy' routes are generated from endpoints with the fewest turns. We evaluate on route lengths of 100m (with a simulation time of 30 seconds) and 500m (150 seconds). We consider both 'easy' and 'hard' traffic densities: 'easy' is a single sample, and 'hard' is the scene with the largest number of agents among 8 DiT samples.

Metrics. We evaluate the Planner Failure Rate (PFR) in SLEDGE using PDM-Closed [12]. This is the winner of the 2023 nuPlan challenge, and the state of the art for motion planning in the short, 15 second scenarios possible with existing data-driven simulators. The planner 'fails' if it achieves less than 20% of the route's total progress, goes in the wrong driving direction for more than 6m, goes off-road, or causes an at-fault collision. We also measure the **#Turns** and **#Agents**, which are proxies of the difficulty of the route and traffic.

Results. We show our results in Table 3. Importantly, we note that setting up an evaluation with SLEDGE only requires a 3GB download of our DiT-L checkpoint for the Lane & Agent mode, and an additional 1GB download of the nuPlan maps for the Lane \rightarrow Agent mode, in contrast to the 2TB of logs needed to set up nuPlan. When using easy routes and traffic, we observe similar PFRs for both replay-based and generative simulation despite this large compression factor. For

Tasl	$\leftarrow \rightarrow$		$\mathbf{Lane} ightarrow \mathbf{Agent}$						Lane & Agent					
${f Length} o$			00 meters 5			500 meters		100 meters			500 meters			
Routes	Traffic	Turns	Agents	\mathbf{PFR}	Turns	Agents	\mathbf{PFR}	Turns	Agents	\mathbf{PFR}	Turns	Agents	\mathbf{PFR}	
Rep	lay	0.89	57.40	0.06	3.29	102.34	0.26	-	-	-	-	-	-	
Easy	Easy Hard	0.89	$44.61 \\ 56.44$	$\begin{array}{c} 0.07\\ 0.11\end{array}$	3.29	$125.23 \\ 167.47$	$0.25 \\ 0.39$	$\begin{array}{c} 0.61 \\ 0.57 \end{array}$	$27.30 \\ 39.11$	$0.22 \\ 0.20$	$\begin{vmatrix} 2.22\\ 2.30 \end{vmatrix}$	$110.51 \\ 173.91$	$\begin{array}{c} 0.39\\ 0.44 \end{array}$	
Hard	Easy Hard	1.18	$44.66 \\ 57.65$	$\begin{array}{c} 0.14\\ 0.11\end{array}$	4.20	$128.79 \\ 170.87$	$\begin{array}{c} 0.28\\ 0.44\end{array}$	$1.23 \\ 1.07$	$27.14 \\ 39.03$	$\begin{array}{c} 0.29 \\ 0.30 \end{array}$	$\begin{vmatrix} 3.66 \\ 3.82 \end{vmatrix}$	$\begin{array}{c} 107.11\\ 169.66 \end{array}$	$\begin{array}{c} 0.45 \\ 0.49 \end{array}$	

Table 3: Simulation of PDM-Closed in SLEDGE. Our simulator offers control over the route length, difficulty and traffic density. In several settings, we present new challenges for the existing state-of-the-art, leading to high failure rates of over 40%.

replay, we observe over a $4 \times$ rise in PFR from 0.06 to 0.26 when extending the route from 100m to 500m. These are primarily due to PDM-Closed's inability to make lane changes or overtake slow vehicles, which are important planning behaviors that are not strongly penalized on current benchmarks like Val14 [12]. In all settings, switching to hard routes increases the number of turns, which is more challenging than straight driving and in turn deteriorates the planning performance. In the most challenging settings with hard routes and traffic, the PFR increases to over 40%. We visualize some snapshots of states from 100m SLEDGE simulations in Fig. 1. Our supplementary video contains clips from such reactive simulations, where we highlight the weaknesses of PDM-Closed that were not sufficiently penalized in prior benchmarks.

5 Conclusion

We present SLEDGE, a generative simulator for vehicle motion planning based on latent diffusion. We conduct several experiments to show that it is more realistic, compact, controllable, and diverse than other generative and replaybased approaches. Additionally, we establish several baselines and metrics for the generative simulation task. We hope our work can lay the foundation for accelerating progress in data-driven simulation and vehicle motion planning.

Limitations. Evaluating simulators is hard. We provide metrics for the lane graph generation sub-task, and preliminary experiments on testing rule-based planning, but using the simulator for other downstream tasks, such as reinforcement learning, will be important to showcase its full potential. For efficiency, we use a relatively small FOV and simulation radius, and a simplistic lane representation consisting of only the centerline (assuming constant lane widths). These issues could be alleviated by further scaling of our model and compute. However, like other diffusion models, the compute requirements of our approach are already high. We see value in improving the efficiency of SLEDGE through relevant techniques for accelerating diffusion models [27, 28, 31, 40]. Acknowledgments. This work was supported by the ERC Starting Grant LEGO-3D (850533), the DFG EXC number 2064/1 - project number 390727645, the German Federal Ministry of Education and Research: Tübingen AI Center, FKZ: 01IS18039A and the German Federal Ministry for Economic Affairs and Climate Action within the project NXT GEN AI METHODS. We thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Kashyap Chitta and Daniel Dauner. We also thank Agniv Sharma for providing his reimplementation of HDMapGen, Bernhard Jaeger for proofreading, and the nuPlan team for open-sourcing their dataset and simulation tools to the community.

References

- Althoff, M., Koschi, M., Manzinger, S.: Commonroad: Composable benchmarks for motion planning on roads. In: Proc. IEEE Intelligent Vehicles Symposium (IV) (2017) 4
- Bansal, M., Krizhevsky, A., Ogale, A.S.: Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In: Proc. Robotics: Science and Systems (RSS) (2019) 5
- Blattmann, A., Dockhorn, T., Kulal, S., Mendelevitch, D., Kilian, M., Lorenz, D., Levi, Y., English, Z., Voleti, V., Letts, A., Jampani, V., Rombach, R.: Stable Video Diffusion: Scaling Latent Video Diffusion Models to Large Datasets. arXiv.org 2311.15127 (2023) 3
- Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., Ramesh, A.: Video generation models as world simulators (2024), https://openai.com/research/videogeneration-models-as-world-simulators 1, 3
- Caesar, H., Kabzan, J., Tan, K.S., Fong, W.K., Wolff, E.M., Lang, A.H., Fletcher, L., Beijbom, O., Omari, S.: nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops (2021) 4
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: Endto-end object detection with transformers. In: Proc. of the European Conf. on Computer Vision (ECCV) (2020) 4, 5, 6
- Chen, J., Deng, R., Furukawa, Y.: Polydiffuse: Polygonal shape reconstruction via guided set diffusion models. In: Advances in Neural Information Processing Systems (NeurIPS) (2023) 3
- Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., Li, H.: End-to-end autonomous driving: Challenges and frontiers. arXiv preprint arXiv:2306.16927 (2023) 3
- Chi, C., Feng, S., Du, Y., Xu, Z., Cousineau, E., Burchfiel, B., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. In: Proc. Robotics: Science and Systems (RSS) (2023) 3
- Chu, H., Li, D., Acuna, D., Kar, A., Shugrina, M., Wei, X., Liu, M.Y., Torralba, A., Fidler, S.: Neural Turtle Graphics for Modeling City Road Layouts. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV) (2019) 3, 11
- Contributors, N.: Navsim: Data-driven non-reactive autonomous vehicle simulation. https://github.com/autonomousvision/navsim (2024) 1

- Dauner, D., Hallgarten, M., Geiger, A., Chitta, K.: Parting with misconceptions about learning-based vehicle motion planning. In: Proc. Conf. on Robot Learning (CoRL) (2023) 1, 2, 13, 14
- 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 (RA-L) (2021) 4
- 14. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proc. Conf. on Robot Learning (CoRL) (2017) 3
- Elhousni, M., Lyu, Y., Zhang, Z., Huang, X.: Automatic Building and Labeling of HD Maps with Deep Learning. arXiv.org 2006.00644 (2020) 3
- Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., Podell, D., Dockhorn, T., English, Z., Lacey, K., Goodwin, A., Marek, Y., Rombach, R.: Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. arXiv.org 2403.03206 (2024) 1, 3
- 17. Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., Co-Reyes, J.D., Agarwal, R., Roelofs, R., Lu, Y., Montali, N., Mougin, P., Yang, Z., White, B., Faust, A., McAllister, R., Anguelov, D., Sapp, B.: Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. In: Advances in Neural Information Processing Systems (NIPS) Track on Datasets and Benchmarks (2023) 1, 4, 8
- Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., Geiger, A.: King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In: Proc. of the European Conf. on Computer Vision (ECCV) (2022) 4
- He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2016)
 5
- He, S., Balakrishnan, H.: Lane-level street map extraction from aerial imagery. In: Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV) (2022) 10
- Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. arXiv.org 2006.11239 (2020) 7
- Karnchanachari, N., Geromichalos, D., Seang Tan, K., Li, N., Eriksen, C., Yaghoubi, S., Mehdipour, N., Bernasconi, G., Kit Fong, W., Guo, Y., Caesar, H.: Towards learning-based planning: The nuPlan benchmark for real-world autonomous driving. In: Proc. IEEE International Conf. on Robotics and Automation (ICRA) (2024) 1, 2, 3, 4, 8, 9
- Li, Q., Peng, Z., Feng, L., Zhang, Q., Xue, Z., Zhou, B.: Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 45(3), 3461–3475 (2022) 4
- Li, T., Jia, P., Wang, B., Chen, L., Jiang, K., Yan, J., Li, H.: LaneSegNet: Map Learning with Lane Segment Perception for Autonomous Driving. In: Proc. of the International Conf. on Learning Representations (ICLR) (2024) 3
- Li, Z., Yu, Z., Lan, S., Li, J., Kautz, J., Lu, T., Alvarez, J.M.: Is ego status all you need for open-loop end-to-end autonomous driving? arXiv.org 2312.03031 (2023)
 1
- Liao, B., Chen, S., Wang, X., Cheng, T., Zhang, Q., Liu, W., Huang, C.: Maptr: Structured modeling and learning for online vectorized hd map construction. In: Proc. of the International Conf. on Learning Representations (ICLR) (2023) 3, 4, 5
- Lin, S., Wang, A., Yang, X.: SDXL-Lightning: Progressive Adversarial Diffusion Distillation. arXiv.org 2402.13929 (2024) 14

- Liu, X., Gong, C., Liu, Q.: Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. arXiv.org 2209.03003 (2022) 14
- Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., Van Gool, L.: Re-Paint: Inpainting using Denoising Diffusion Probabilistic Models. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2022) 7
- Luo, S., Hu, W.: Diffusion Probabilistic Models for 3D Point Cloud Generation. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2021) 3
- Luo, S., Tan, Y., Huang, L., Li, J., Zhao, H.: Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference. arXiv.org 2310.04378 (2023) 14
- Mi, L., Zhao, H., Nash, C., Jin, X., Gao, J., Sun, C., Schmid, C., Shavit, N., Chai, Y., Anguelov, D.: Hdmapgen: A hierarchical graph generative model of high definition maps. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2021) 3, 11
- Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-E: A System for Generating 3D Point Clouds from Complex Prompts. arXiv.org 2212.08751 (2022) 3
- Nunez-Iglesias, J., Blanch, A.J., Looker, O., Dixon, M.W.A., Tilley, L.: A new python library to analyse skeleton images confirms malaria parasite remodelling of the red blood cell membrane skeleton. PeerJ (2018) 9
- Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proc. of the IEEE International Conf. on Computer Vision (ICCV) (2023) 2, 7
- Pronovost, E., Ganesina, M.R., Hendy, N., Wang, Z., Morales, A., Wang, K., Roy, N.: Scenario Diffusion: Controllable Driving Scenario Generation With Diffusion. In: Advances in Neural Information Processing Systems (NeurIPS) (2023) 3
- Renz, K., Chitta, K., Mercea, O.B., Koepke, S., Akata, Z., Geiger, A.: Plant: Explainable planning transformers via object-level representations. In: Proc. Conf. on Robot Learning (CoRL) (2022) 5
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2022) 3, 4, 5
- Santurkar, S., Budden, D., Shavit, N.: Generative Compression. arXiv.org 1703.01467 (2017) 2
- Sauer, A., Boesel, F., Dockhorn, T., Blattmann, A., Esser, P., Rombach, R.: Fast High-Resolution Image Synthesis with Latent Adversarial Diffusion Distillation. arXiv.org 2403.12015 (2024) 14
- Shabani, M.A., Hosseini, S., Furukawa, Y.: Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2023) 3
- Sun, S., Gu, Z., Sun, T., Sun, J., Yuan, C., Han, Y., Li, D., Ang, Marcelo H., J.: DriveSceneGen: Generating Diverse and Realistic Driving Scenarios from Scratch. arXiv.org 2309.14685 (2023) 3, 11
- Treiber, M., Hennecke, A., Helbing, D.: Congested traffic states in empirical observations and microscopic simulations. Physical review E (2000) 8
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 5998–6008 (2017) 7
- Wang, J., Pun, A., Tu, J., Manivasagam, S., Sadat, A., Casas, S., Ren, M., Urtasun, R.: Advsim: Generating safety-critical scenarios for self-driving vehicles. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2021) 4

- Wymann, B., Dimitrakakisy, C., Sumnery, A., Espié, E., Guionneauz, C.: Torcs: The open racing car simulator (2015) 3
- Xu, M., Powers, A., Dror, R., Ermon, S., Leskovec, J.: Geometric latent diffusion models for 3d molecule generation. In: Proc. of the International Conf. on Machine learning (ICML) (2023) 3
- Yang, J., Gao, S., Qiu, Y., Chen, L., Li, T., Dai, B., Chitta, K., Wu, P., Zeng, J., Luo, P., Zhang, J., Geiger, A., Qiao, Y., Li, H.: Generalized Predictive Model for Autonomous Driving. arXiv.org 2403.09630 (2024) 3
- Zeng, X., Vahdat, A., Williams, F., Gojcic, Z., Litany, O., Fidler, S., Kreis, K.: Lion: Latent point diffusion models for 3d shape generation. In: Advances in Neural Information Processing Systems (NeurIPS) (2022) 3
- Zhai, J.T., Feng, Z., Du, J., Mao, Y., Liu, J.J., Tan, Z., Zhang, Y., Ye, X., Wang, J.: Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. arXiv.org 2305.10430 (2023) 1
- Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., Pavone, M.: Guided Conditional Diffusion for Controllable Traffic Simulation. In: Proc. IEEE International Conf. on Robotics and Automation (ICRA) (2023) 3