PLANNER-ORIENTED SCENARIO GENERATION: UNCERTAINTY-GUIDED GENERATION OF LEARNABLE LONG-TAIL SCENARIOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Handling long-tail scenarios is one of the main challenges in autonomous driving. Several recent works have demonstrated that generative models can be a promising way to address this challenge by synthesizing diverse long-tail scenarios. However, they require domain knowledge of what makes a scene critical, such as specific traffic or safety conditions. This domain knowledge is often not available in practice and does not reflect which scenarios the planner has not learned well. Furthermore, these works also need additional components to keep their scenarios learnable. In this paper, we propose Planner-Oriented Scenario Generation (POSG), which generates long-tail scenarios that the planner is uncertain about but can learn from, without domain knowledge of what makes a scene critical. We hypothesize that a scene is worth training on if it is uncertain and learnable for the planner, not merely if it is rare or adversarial. To implement our hypothesis, POSG combines the score estimate of the diffusion model with the gradient obtained from the planner's uncertainty. POSG also encourages learnability by leveraging a generative model that synthesizes the ego trajectory together with the futures of the surrounding agents. The generative model enables us to score the generated trajectories for learnability in the context of the entire scene. Experiments show that scenarios generated by POSG induce a higher planner error than unguided samples from the same generative model, while improving the planner when it is retrained on them.

1 INTRODUCTION

Autonomous driving systems have achieved remarkable success by learning from large-scale real-world driving datasets (Sun et al., 2020). However, handling the long-tailed distribution of data remains an open problem. Long-tail scenarios occur rarely in the collected real-world data, so a planner trained on the data may perform poorly in such scenarios and can be easily biased toward frequently seen behavior (Chen et al., 2024b). Generative models, such as diffusion models, have recently gained attention as a promising way to address this challenge by synthesizing diverse long-tail scenarios. The key idea behind the generative approach is to add long-tail conditions to the sampling process. The long-tail conditions, such as surrounding vehicles braking suddenly or cutting in from neighboring lanes, are manually defined in advance.

Recent works have shown that such conditions enable a generative model to synthesize diverse long-tail scenarios (Zhong et al., 2023b; Chang et al., 2024; Xu et al., 2025; Zhong et al., 2023a; Zhang et al., 2024b). However, specifying these conditions requires domain knowledge of what makes a scene critical. This domain knowledge is rarely available in practice and does not reflect which scenarios the planner has not learned well. Furthermore, existing methods do not consider whether their scenarios are learnable or rely on additional components to avoid generating unlearnable scenarios (Rempe et al., 2022; Hanselmann et al., 2022; Zhang et al., 2024a; Chen et al., 2024a). Unlearnable scenarios should be avoided, as they provide little benefit for improving the planner and may even degrade its performance.

In this paper, we introduce Planner-Oriented Scenario Generation (POSG), a diffusion-based method that generates learnable long-tail scenarios that the planner has not learned well, without relying on

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

domain knowledge of what makes a scene critical. We hypothesize that the scenarios worth training on are those that the planner is uncertain about but can learn from, not merely those that are rare or adversarial. For example, a vehicle cutting in from a neighboring lane adds little when the planner already responds to it confidently, and is worth training on when the planner is uncertain whether to brake or to change lanes. Based on our hypothesis, POSG adds the gradient of the planner's uncertainty to the score estimate of the diffusion model. The uncertainty is estimated from the planner's hidden representations under test-time dropout.

POSG does not require additional components to generate learnable scenarios. Learnable scenarios are commonly defined as scenarios in which the ego trajectory is safe, for example by avoiding collisions and remaining within the lanes. While previous works rely on additional components to generate ego trajectories and evaluate their safety, POSG simply extends the diffusion model to generate and evaluate ego trajectories based on the context of the entire scene. In particular, along with the gradient of the planner's uncertainty, POSG also adds the gradient of the learnability costs on the ego trajectory to the score estimate. Combining the two gradients allows us to generate learnable long-tail scenarios without additional components.

The main contribution of this work is POSG, a planner-oriented scenario generation method that generates long-tail scenarios that the planner is uncertain about but can learn from. POSG contrasts with previous works that require domain knowledge to manually design long-tail conditions or additional components to generate learnable scenarios. We evaluate POSG against baselines in nuPlan (Caesar et al., 2021; Karnchanachari et al., 2024) by augmenting a planner's training data with the generated scenarios. We measure the planner's error on the generated scenarios and the open-loop error of the retrained planner on held-out recorded scenes. Experimental results demonstrate that POSG generates learnable long-tail scenarios that the planner is uncertain about, resulting in higher planner errors and enabling the planner to achieve better performance than unguided baselines.

2 Related Work

Driving datasets are long-tailed, with most drives being repetitive and safety-critical cases occurring rarely (Makansi et al., 2021). Recent works have proposed generative models trained on real-world data to synthesize new driving scenarios (Chitta et al., 2024; Rowe et al., 2025). SLEDGE (Chitta et al., 2024) generates lane graphs and agent bounding boxes with a latent diffusion model and uses them as the initial state for rule-based traffic simulation. Scenario Dreamer (Rowe et al., 2025) generates the initial scene with a vectorized latent diffusion model and simulates closed-loop agent behaviors with an autoregressive Transformer. While these methods can generate realistic and diverse driving scenarios, they rarely generate safety-critical and informative scenarios. This limitation is due to the long-tailed distribution of real-world driving data: most scenarios are repetitive and uninformative, whereas safety-critical scenarios constitute only a small fraction of the data.

Safety-critical scenario generation, which aims to produce challenging scenarios for an autonomy system, has been extensively explored (Wang et al., 2021; Rempe et al., 2022; Hanselmann et al., 2022; Ding et al., 2023; Chang et al., 2024; Zhang et al., 2024b; Xu et al., 2025). A promising approach is to make generative models controllable with conditions, such as goals, rules, or language instructions. SAFE-SIM (Chang et al., 2024) and DiffScene (Xu et al., 2025) add adversarial objectives, such as the distance between a surrounding vehicle and the ego agent, to the sampling process of a diffusion model. While these objectives are designed by hand, CTG++ (Zhong et al., 2023a) and ChatScene (Zhang et al., 2024b) convert a language description into a loss function and a scenario program, respectively, with the help of a large language model. However, both approaches rely on domain knowledge of what makes a scene critical. This domain knowledge is defined independently of what the planner has learned. GALTraj (Park et al., 2025) instead identifies tail samples on which a trajectory prediction model fails and augments these samples with a controllable diffusion model. Delphi (Ma et al., 2024) collects the failure cases of an end-to-end driving model and generates new data similar to those failure cases. Although these works reflect the failures of the model being trained, they require these failures to be observed first and generate new data only around them. To address this limitation, our work utilizes the planner's uncertainty in the sampling process to generate new scenarios based on what the planner has learned.

While most previous works do not consider whether the generated scenarios are learnable, our work aims to keep them learnable. STRIVE (Rempe et al., 2022), KING (Hanselmann et al., 2022),

asymmetric self-play (Zhang et al., 2024a), and FREA (Chen et al., 2024a), consider whether the generated scenarios are solvable. STRIVE (Rempe et al., 2022) performs an additional optimization to find an ego trajectory that avoids a collision and discards the scenario if this optimization fails. KING (Hanselmann et al., 2022) keeps only the scenarios in which a privileged rule-based expert finds a safe trajectory. Asymmetric self-play (Zhang et al., 2024a) rewards its teacher policy for proposing scenarios that the student fails to solve but that the teacher can solve without a collision when it controls all agents. FREA (Chen et al., 2024a) pre-trains a feasible value network of the ego agent on offline data and uses it to bound the adversariality of the scenarios. The experimental results of KING and asymmetric self-play demonstrate that training a planner on the generated scenarios can reduce its collision rate. However, these works rely on a separate optimization, a privileged expert, a pre-trained value network, or generator training with a learnability objective. Our work instead obtains learnability from the generative model itself.

3 Preliminaries

3.1 Guided Latent Diffusion

A latent diffusion model (Rombach et al., 2022) applies the diffusion process in the latent space of a pretrained autoencoder rather than directly in the data space, operating on latent features z that the autoencoder's decoder $\mathcal{D}$ maps back to the data space. It learns to reverse a forward process that gradually adds Gaussian noise to a clean latent z_0 over steps $t = 1, \dots, T$ (Ho et al., 2020),

$$q(z_t \mid z_0) = \mathcal{N}\big(z_t;\ \sqrt{\bar{\alpha}_t}\, z_0,\ (1 - \bar{\alpha}_t)\mathbf{I}\big), \tag{1}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ for a fixed variance schedule $\beta_1, \dots, \beta_T$. A noise-prediction network $\epsilon_\theta(z_t, t)$ is trained to predict the noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ by minimizing $\mathbb{E}_{t, z_0, \epsilon}\big[\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon,\ t)\|^2\big]$ with t drawn uniformly from $\{1, \dots, T\}$. At every step, the noise prediction gives an estimate of the clean latent by Tweedie's formula (Efron, 2011; Chung et al., 2023),

$$\hat{z}_0 = \frac{z_t - \sqrt{1 - \bar{\alpha}_t}\, \epsilon_\theta(z_t, t)}{\sqrt{\bar{\alpha}_t}}, \tag{2}$$

which estimates the posterior mean $\mathbb{E}[z_0 \mid z_t]$. Sampling starts from $z_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and draws each z_{t-1} from the forward process posterior with z_0 replaced by $\hat{z}_0$ (Ho et al., 2020), whose mean and covariance are

$$\tilde{\mu}_t(z_t, \hat{z}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\, \beta_t}{1 - \bar{\alpha}_t}\, \hat{z}_0 + \frac{\sqrt{\alpha_t}\, (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t}\, z_t, \qquad \tilde{\beta}_t \mathbf{I} = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\, \beta_t \mathbf{I}. \tag{3}$$

The noise prediction also gives an estimate of the score of the noisy latent distribution p_t (Song et al., 2021; Dhariwal & Nichol, 2021),

$$\nabla_{z_t} \log p_t(z_t) \approx s_\theta(z_t, t) = -\frac{\epsilon_\theta(z_t, t)}{\sqrt{1 - \bar{\alpha}_t}}. \tag{4}$$

Classifier guidance (Dhariwal & Nichol, 2021) combines this score estimate with the gradient of a classifier $p_\psi(y \mid z_t)$ that is trained on noisy latents to predict a label y, and shifts the mean of each reverse step by $\lambda\, \tilde{\beta}_t \nabla_{z_t} \log p_\psi(y \mid z_t)$, where λ is the gradient scale. Trajectory diffusion models perturb the predicted mean with the gradient of a differentiable objective $\mathcal{J}$ to generate trajectories with desired properties (Janner et al., 2022; Zhong et al., 2023b). Diffuser (Janner et al., 2022) trains this objective on noisy trajectories to predict their return, while CTG (Zhong et al., 2023b) writes it as an analytical loss that requires no training. For latent diffusion, MPGD (He et al., 2024) evaluates the objective on the decoded sample $\mathcal{D}(\hat{z}_0)$ and takes its gradient with respect to $\hat{z}_0$, which does not require computing the gradient with respect to z_t through ϵ_θ. Ye et al. (2024) call this mean guidance and show that MPGD is a special case of their framework. We write the mean of each reverse step with the objective as

$$\tilde{\mu}_t^{\mathcal{J}} = \tilde{\mu}_t(z_t, \hat{z}_0) + \rho_t\, \hat{g}, \tag{5}$$

where $\hat{g}$ is a direction obtained from $\nabla_{\hat{z}_0} \mathcal{J}(\mathcal{D}(\hat{z}_0))$ and ρ_t is a step size. The posterior mean $\tilde{\mu}_t(z_t, \hat{z}_0)$ can also be written as $(z_t + \beta_t s_\theta)/\sqrt{\alpha_t}$ (Ho et al., 2020), so equation 5 is equivalent to adding $(\sqrt{\alpha_t}\, \rho_t / \beta_t)\, \hat{g}$ to the score estimate s_θ. This implies that the objective takes the place of the classifier, and since it is evaluated on the clean estimate rather than on a noisy latent, any function of a decoded scene that is differentiable through $\mathcal{D}$ can serve as the objective without training a classifier on noisy latents.

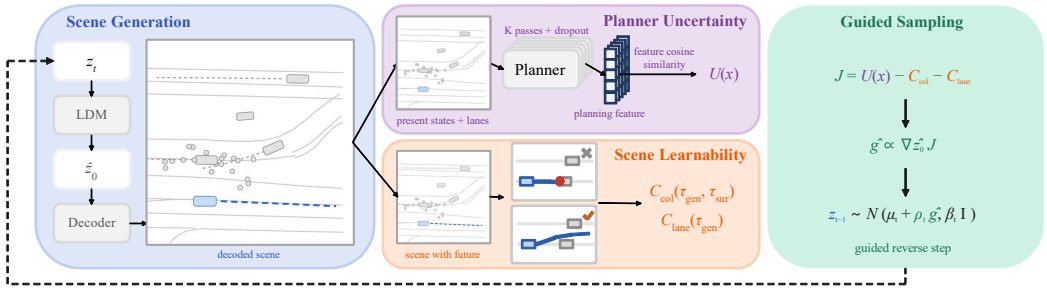


Figure 1: **Planner-oriented guidance at one reverse step.** POSG forms the clean estimate $\hat{z}_0$ from the noise prediction and decodes it into a scene, runs the frozen planner K times with dropout on that scene to obtain the uncertainty $U(x)$, and scores the generated ego trajectory τ_{gen} against the futures τ_{sur} of the surrounding agents with the collision cost C_{col} and the off-lane cost C_{lane}. The three terms form the objective $\mathcal{J}$, and the direction $\hat{g}$ obtained from its gradient shifts the mean of the reverse step, so that z_{t-1} enters the next step in the place of z_t. The ego agent is drawn in blue in this schematic, with its generated future dashed, and the surrounding agents in grey.

3.2 Monte Carlo Dropout

Monte Carlo dropout performs dropout at test time in a network trained with dropout and runs K stochastic forward passes for the same input (Gal & Ghahramani, 2016; Kendall & Gal, 2017). Gal & Ghahramani (2016) cast dropout training as approximate Bayesian inference in deep Gaussian processes, for a network with dropout applied before every weight layer. Under this view, the mean $\bar{y}(x)$ of the K outputs estimates the predictive mean, and their sample variance $\hat{V}(x)$ plus the inverse model precision estimates the predictive variance,

$$\bar{y}(x) = \frac{1}{K}\sum_{k=1}^{K} f(x; m_k), \qquad \hat{V}(x) = \frac{1}{K}\sum_{k=1}^{K} \big(f(x; m_k) - \bar{y}(x)\big)\big(f(x; m_k) - \bar{y}(x)\big)^\top, \tag{6}$$

where $f(x; m_k)$ is the output of the network f with the k-th dropout mask m_k. The dropout probabilities are variational parameters of this approximating distribution, and the stochastic forward passes sample from the same distribution (Gal & Ghahramani, 2016; Gal et al., 2017). Loquercio et al. (2020) report that the best dropout rate at test time corresponds to the rate used in training, and well-calibrated estimates depend on these probabilities, which Concrete dropout tunes with gradient methods (Gal et al., 2017). Ledda et al. (2023) instead inject dropout at test time into a network trained without it, and report that the usefulness of the resulting estimate depends on how it is scaled. The status of this Bayesian interpretation is debated. Hron et al. (2018) show that the variational objective is ill defined because the approximating distribution is singular with respect to the true posterior, and Le Folgoc et al. (2021) argue that MC dropout changes the Bayesian model rather than approximating it.

4 Planner-Oriented Scenario Generation

POSG is designed to generate learnable long-tail scenarios that the planner is uncertain about, without domain knowledge of what makes a scene critical. Similar to previous works on controllable scenario generation (Zhong et al., 2023b; Chang et al., 2024; Xu et al., 2025), POSG adds the gradient of a differentiable objective to the sampling process of a diffusion model, and both the diffusion model and the planner stay frozen. POSG operates under two assumptions. First, no user-specified condition is available, so what to aim for has to come from the models that the pipeline already contains. This contrasts with previous works that obtain their target from a rule, a cost, or a language description written by a user. Second, no additional component is available to decide whether a generated scene is learnable. This presents difficulties for previous works that establish solvability with a separate optimization, a privileged expert, or a value network trained offline.

Under these assumptions, POSG builds its objective out of the planner and the generative model alone. Whether a scene is worth training on depends on what the planner has already learned, so the

planner supplies the uncertainty. Whether a scene can be learned from depends on an ego trajectory that moves through it, which only the generative model can supply while the scene is still being sampled, so that model supplies a cost on the trajectory it produces. As shown in Figure 1, both are computed on the scene that the decoder produces from the clean estimate of a reverse step, and the gradient of the objective they form shifts the mean of that step.

4.1 Uncertainty from the Planner's Hidden Representations

POSG measures how uncertain the planner is about a scene from how much the planner's hidden representations vary under test-time dropout. This measure is built on our hypothesis that a scene is worth training on when the planner is uncertain about it. What distinguishes such a scene is that the planner does not settle on one answer for it. The planner decodes one trajectory for a scene, so the variance of its outputs under dropout rises both when the scene admits several valid futures and when the planner has not learned the scene, and the two cannot be told apart. POSG therefore uses the representations instead of the output trajectory, since an input that a network has not learned well is characterized better in its representation space than in its output space (Lee et al., 2018). To measure this, POSG runs the planner K times on the decoded scene $x = \mathcal{D}(\hat{z}_0)$ with dropout active at every dropout layer, and takes the hidden representation h_k of the ego agent at the output of the planner's encoder, before the head that decodes the trajectory. The K representations agree when the planner settles on one answer and vary when it does not, so POSG scores the scene by their average cosine distance from their mean,

$$\bar{h} = \frac{1}{K}\sum_{k=1}^{K} h_k, \qquad U(x) = \frac{1}{K}\sum_{k=1}^{K}\left(1 - \frac{\langle h_k, \bar{h}\rangle}{\|h_k\|\,\|\bar{h}\|}\right), \tag{7}$$

which is invariant to a common positive rescaling of the representations. Every pass is kept in the computation graph, so $U(x)$ is differentiable with respect to the decoded scene. POSG uses a dropout rate higher than the rate that the planner was trained with, which departs from the convention stated in Section 3 and which Appendix B.4 supports.

4.2 Learnability from the Generated Ego Trajectory

POSG keeps a generated scene learnable by scoring the ego trajectory that the generative model produces for that scene. Raising $U(x)$ alone would reward a scene in which no trajectory avoids a collision, because the planner is unsure when there is nothing to be sure of. Scoring an ego trajectory requires a scene that carries the future of every agent, since a scene of a lane graph and present states gives neither that trajectory nor a training pair for the planner. The generative model therefore generates those futures, so that every sample carries an ego trajectory τ_{gen} through the generated scene. The object latents carry that future, and the autoencoder takes the future positions of each agent over a horizon of four seconds together with a mask that marks the observed steps. The decoder has one head for the future, which outputs an acceleration and a yaw rate for eight steps of half a second and integrates them with a unicycle model, as in trajectory diffusion models that enforce dynamics (Zhong et al., 2023b), so that a decoded trajectory is kinematically feasible by construction.

POSG scores this trajectory with a soft collision cost and a soft off-lane cost, in the form used for collision avoidance and lane keeping (Rempe et al., 2022) and in trajectory diffusion models (Zhong et al., 2023b; Chang et al., 2024). The collision cost compares the ego trajectory with the future trajectories τ_{sur} of the surrounding agents at each of the $H = 8$ steps,

$$c_{j,i} = \operatorname{softplus}\left(\frac{\delta_i - d_{j,i}}{0.5}\right), \qquad C_{\text{col}}(\tau_{\text{gen}}, \tau_{\text{sur}}) = \sum_{j=1}^{H}\frac{1}{\eta}\log\sum_{i}\exp\left(\eta\, c_{j,i}\right), \tag{8}$$

where $d_{j,i}$ is the distance between the ego agent and agent i at step j, δ_i is the sum of a margin of one meter and the radii of the circles that cover the two bounding boxes, and $\eta = 2$ is a temperature with which the log-sum-exp approximates the largest cost over the agents smoothly. The off-lane cost measures how far the trajectory leaves the lane graph and rises faster once the trajectory is well

Algorithm 1 Planner-Oriented Scenario Generation

Require: diffusion model ϵ_θ, decoder $\mathcal{D}$, planner f_ϕ, passes K, weights w, step sizes ρ_t

$z_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\hat{z}_0 \leftarrow \big(z_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(z_t, t)\big)/\sqrt{\bar{\alpha}_t}$ ▷ clean estimate, equation 2
  $x \leftarrow \mathcal{D}(\hat{z}_0)$, and take τ_{gen} and τ_{sur} from the decoded scene x
  $U \leftarrow$ uncertainty of x from K passes of f_ϕ with dropout ▷ equation 7
  $C_{\text{col}}, C_{\text{lane}} \leftarrow$ costs of τ_{gen} ▷ equation 8, equation 9
  for each term $m \in \{U,\ -C_{\text{col}},\ -C_{\text{lane}}\}$ **do**
    $g_m \leftarrow \nabla_{\hat{z}_0} m$, set the rows of the ego agent to zero, normalize within the scene
  end for
  $\hat{g} \leftarrow$ normalized mixture of $\{g_m\}$ with weights w
  $z_{t-1} \sim \mathcal{N}\big(\tilde{\mu}_t(z_t, \hat{z}_0) + \rho_t\,\hat{g},\ \tilde{\beta}_t \mathbf{I}\big)$ ▷ equation 5
end for

outside it,

$$m_j = \max(d_j - 1,\, 0), \qquad C_{\text{lane}}(\tau_{\text{gen}}) = \sum_{j=1}^{H} \min\Big(\min(m_j,\, \tilde{m}) + \kappa \max(m_j - \tilde{m},\, 0),\, c_{\max}\Big), \tag{9}$$

where d_j is the distance from the ego position at step j to its projection onto the nearest lane segment and m_j is how far that distance exceeds a margin of one meter. The cost rises with unit slope up to a knee at $\tilde{m} = 1.5$ meters and with a slope of $\kappa = 5$ beyond it, and every step is clipped at $c_{\max} = 100$. Neither cost scores the trajectory of the planner, so neither rewards a scene for the planner failing on it.

4.3 Combining Uncertainty and Learnability

POSG combines the uncertainty and the two costs into a single objective, since the uncertainty says which scenes are worth training on and the costs say which of them the planner can learn from. The objective subtracts the costs from the uncertainty,

$$\mathcal{J} = U(x) - C_{\text{col}}(\tau_{\text{gen}}, \tau_{\text{sur}}) - C_{\text{lane}}(\tau_{\text{gen}}), \tag{10}$$

which is high for a scene that the planner is uncertain about and in which the generated ego trajectory avoids the surrounding agents and stays within the lanes. At every reverse step this objective is evaluated on the decoded scene and differentiated with respect to the clean estimate. The magnitude of each term differs from the others and varies over the reverse process, so the gradient of each term is normalized within the scene, and the normalized gradients are mixed with weights w into the single direction $\hat{g}$ that shifts the mean of the step in equation 5. Every term is computed from the generative model and the planner alone, so no component beyond these two is required at any step of the sampling.

4.4 Algorithm Summary

Algorithm 1 outlines the procedure of POSG. At every reverse step, POSG forms the clean estimate of equation 2 from the noise prediction and decodes it with the frozen autoencoder. It then runs the frozen planner on the decoded scene K times with dropout active to obtain the uncertainty of equation 7, and scores the decoded ego trajectory with the two costs of equation 8 and equation 9. Once the three terms are available, POSG takes the gradient of each with respect to the clean estimate and mixes them into the single direction $\hat{g}$. The mean of the reverse step is then shifted by $\rho_t\,\hat{g}$ as in equation 5, and sampling continues to the next step. The sampled scenes are then scored with the same uncertainty, and this selection step keeps the highest ones for the training set of the planner.

5 Experiments

We evaluate POSG by augmenting the training set of a planner with generated scenes and retraining it from scratch, so that those scenes are judged by the planner they produce. Three comparisons

Table 1: Average displacement error of the frozen planner on generated scenes, measured against the decoded ego trajectory, so a higher value means a behavior the planner does not already produce. The rows use 1,024 scenes, 8,192 scenes of the pool and all 50,000 selected scenes.

Scenes	Displacement error ↓
Unguided	0.9296
POSG (ours), before selection	1.2221
POSG (ours), after selection	2.0506

Table 2: Average displacement error after retraining from scratch on the ID-only data together with 50,000 generated scenes. Parentheses give the relative change against the first row, so a negative value is an improvement. **Bold** marks a result that our band separates from the next best, outside the row marked †, which is trained on the 457,648 recorded scenes of every group and bounds what the generated scenes can recover. That row follows an earlier fixed budget of 51,840 optimizer steps, which is 14.5 epochs of its training set against 40 for the other rows, so both its in-distribution error and the gap it defines carry that difference (Appendix A.4). Differences smaller than 0.042 are not separable.

Training data	ID ↓	OOD ↓	unknown ↓
No augmentation (ID-only)	0.5871	1.3343	1.0981
Unguided	0.5904 ($+0.6$)	1.1460 (-14.1)	1.0152 (-7.5)
POSG (ours)	0.5921 ($+0.9$)	**1.0876** (-18.5)	0.9978 (-9.1)
Training split†	0.6626 ($+12.9$)	0.9582 (-28.2)	0.9166 (-16.5)

carry the argument. The comparison against unguided sampling, which draws scenes from the same diffusion model without the objective, isolates the effect of the objective from the effect of adding generated scenes at all. The comparison against uncertainty only, which uses $U(x)$ alone as the objective, isolates the effect of the learnability costs. We then reverse the sign of the uncertainty term, once with the two costs in place and once without them, so that the first of the two differs from POSG in that sign alone. A planner trained on the recorded scenes of every group bounds what the generated scenes can recover. Every variant that uses generated scenes shares the same generative model, the same planner architecture and the same number of training steps, so those variants differ only in the objective.

5.1 Setup

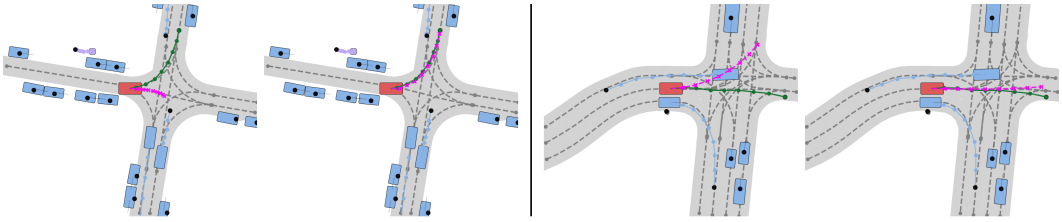
We run all experiments on nuPlan (Caesar et al., 2021; Karnchanachari et al., 2024). We adopt the 27 in-distribution scenario types of the released configuration of Wang et al. (2025), which follow scenario frequency and which Appendix C.1 names, and a scene enters that group only when every one of its tags is in-distribution, while the remaining types form the out-of-distribution group. nuPlan also labels a large share of its scenes as "Unknown", so these scenes fall outside the list, and we hold them out as a third group. The planner takes the lane graph and the present states of the agents, and predicts the future positions of the ego agent for eight steps of half a second. The base planner is trained on the 173,274 in-distribution scenes, which we refer to as the ID-only data, so it never observes an out-of-distribution or unknown scene. The generative model of Section 4.2 is trained on the 457,648 recorded scenes of the training split, which excludes the held-out scenes that every planner is evaluated on. For guidance we use $K = 10$ passes at a dropout rate of 0.5 and combine the gradients of the three terms of equation 10 with the weights and step sizes of Appendix A.3. We sample 200,000 scenes and keep the 50,000 with the highest uncertainty (Appendix A.5), which we add to the ID-only data before training the planner from scratch. We report the average displacement error over the four-second horizon on a common set of 42,852 held-out recorded scenes, which contains 16,294 in-distribution, 14,446 out-of-distribution and 12,112 unknown scenes. An earlier configuration of the same pipeline, in which two variants were each retrained over three seeds, puts the variation of their difference at 0.03. Every variant we report comes from a single seed, so we call a difference separable only when it exceeds 0.042 (Appendix B.5).

Table 3: Effect of the learnability costs, where uncertainty only keeps $U(x)$ alone and unguided sampling is the reference. Each row gives the share of the 50,000 scenes that a variant adds to the training set in which the generated ego trajectory collides or passes more than 1.5 meters from the nearest lane. The two collision rows differ in the footprint as well as in the agents counted.

	Unguided	Uncertainty only	POSG (ours)
Vehicle collision	3.9%	20.5%	9.2%
Any-agent collision	40.5%	41.4%	18.3%
Off-lane	30.7%	16.0%	7.3%

Table 4: Reversing the uncertainty term, so that sampling and selection favor the scenes the planner is most certain about. Certainty and costs keeps the two costs of equation 10 and differs from POSG in the sign of the uncertainty term alone, while certainty only drops the costs as well.

Objective	ID ↓	OOD ↓	unknown ↓
POSG (ours)	0.5921	**1.0876**	**0.9978**
Certainty and costs	0.5913	1.3705	1.1140
Certainty only	0.5772	1.3094	1.0894



(a) (b) (c) (d)

Figure 2: **Planner before and after the POSG scenes are added to its training set.** Each pair shows one held-out recorded scene, (a) and (b) for the first and (c) and (d) for the second. The ego agent is drawn in red, the recorded ego trajectory in green and the planner in magenta. Before retraining the planner leaves the recorded trajectory at the intersection, and after retraining the planner follows the recorded trajectory.

5.2 Results and Analysis

Table 1 reports the error of the frozen planner on the generated scenes themselves. The planner reaches 0.9296 on unguided samples and 1.2221 on the pool that POSG generates, and the scenes that the selection step keeps raise it further to 2.0506. POSG therefore generates scenarios on which the planner errs more than on scenes drawn from the same model without an objective.

Table 2 reports the planner trained on the ID-only data. Unguided sampling already lowers the out-of-distribution error by 14.1 percent, since the generative model is trained on the recorded scenes of every group and therefore produces out-of-distribution and unknown scenes of its own. POSG lowers that error from 1.3343 to 1.0876, a reduction of 18.5 percent, and the difference of 0.058 against unguided sampling is separable. POSG reaches the lower error on the unknown group as well, 0.9978 against 1.0152, although that difference of 0.017 is not separable. We show qualitative examples of the retrained planner in Figure 2. These results indicate that the scenes selected by the planner's uncertainty carry more of what the planner has not learned than scenes drawn from the same model without an objective.

On the group the planner was trained on, the error rises by 0.005 against no augmentation, which our band does not separate. Training on the recorded scenes of every group instead reaches 0.9582 out of distribution at the cost of a 12.9 percent increase in the in-distribution error. POSG recovers 66 percent of the gap between that bound and the planner without augmentation, against 50 percent for unguided sampling. These results indicate that generated scenes recover most of what recorded out-of-distribution scenes recover while leaving the in-distribution error nearly untouched, although the unknown group still calls for recorded scenes of its own. Unguided sampling draws from the whole distribution that the generative model learned, so its error falls as the number of generated scenes grows, and at 100,000 scenes it reaches 1.0760 out of distribution. The 50,000 scenes that POSG selects are not separable from that, so the objective reaches with half the scenes what unguided sampling reaches with twice as many (Appendix B.2). For the planner without augmentation and

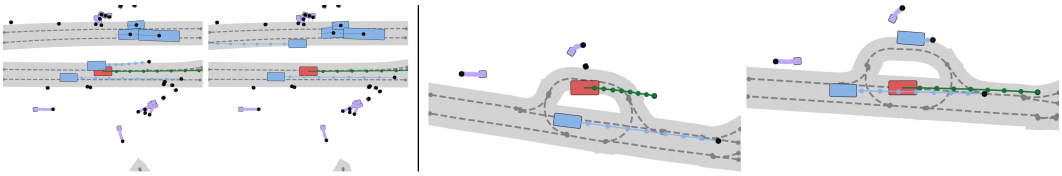


Figure 3: **Qualitative comparison of the learnability costs.** Each pair shows one scene sampled without the costs and with them, (a) and (b) for the first scene and (c) and (d) for the second. The ego agent is drawn in red, the surrounding agents in blue and purple, and the generated ego trajectory in green. Without the costs the ego agent overlaps a surrounding vehicle in (a) and the generated ego trajectory leaves the lanes in (c), and neither happens with the costs.

for every augmented variant, checkpoints are selected on the in-distribution part of the held-out scenes alone, so the errors we report out of distribution never enter that choice. We also evaluate the planner without augmentation and POSG on a test split of 66,237 scenes that takes part neither in training nor in selection, where the out-of-distribution error falls by 16.9 percent and the unknown error by 9.2 percent (Appendix B.1).

Table 3 compares POSG against uncertainty only, which keeps $U(x)$ alone as the objective. Among the 50,000 scenes that uncertainty only selects, 20.5 percent carry a generated ego trajectory that collides with a vehicle, against 3.9 percent among unguided scenes, so the uncertainty term alone concentrates contact between vehicles. The costs bring all three shares below the level that uncertainty only reaches, to 9.2 percent for a vehicle, 18.3 percent when the count includes every agent and 7.3 percent for leaving the lanes. Retraining on those scenes reaches an out-of-distribution error of 1.0809, which is not separable from the 1.0876 that POSG reaches, so the costs more than halve the share of unsafe trajectories among the selected scenes while leaving the open-loop error where it is (Appendix B.3). Figure 3 shows qualitative examples for the collision cost and the off-lane cost. The costs therefore keep the selected scenes learnable without an additional component and without a cost in open-loop error.

Finally, Table 4 reverses the uncertainty term so that both the sampling and the selection favor the scenes the planner is most certain about. We sample 100,000 scenes here and keep the lowest half, an asymmetry against the highest quarter that the other variants keep. Reversing the term while the two costs stay in place reaches 1.3705 out of distribution and dropping the costs as well reaches 1.3094, neither of which our band separates from the 1.3343 that the planner reaches without augmentation, while POSG reaches 1.0876. These results indicate that the direction of the uncertainty term is what carries the gain over unguided augmentation, and that selecting for certainty forfeits even the benefit that unguided sampling gives.

6 Conclusion

We introduce a planner-oriented scenario generation method, referred to as POSG, that generates long-tail scenarios that a planner is uncertain about and can learn from. POSG estimates the uncertainty of a planner from how much its hidden representations vary under test-time dropout, and scores the ego trajectory that the generative model decodes with a collision cost and an off-lane cost, which keeps the scenarios learnable without a component beyond the generative model and the planner. Experimental results demonstrate that POSG generates scenarios on which the planner errs more than on unguided samples, and that retraining the planner on them lowers its out-of-distribution error by 18.5 percent without a separable change in its in-distribution error. The scenarios that POSG generates are bounded by what a single sample of the generative model can express. Its rollout lasts four seconds, so a scene cannot carry a longer interaction, and its futures are fixed once a scene is decoded, so a scene that is difficult under one ego trajectory may be trivial under another. Carrying a longer rollout in the object latents and replacing the fixed futures with a reactive traffic model would both widen what a single sample can express. We also measure the retrained planner with an open-loop error, which has little correlation with closed-loop driving (Dauner et al., 2023), so closed-loop simulation is needed to establish the size of the improvement.

THE USE OF LARGE LANGUAGE MODELS

In this work, we used generative AI tools for research ideation and execution. We have not used generative AI tools for generating synthetic datasets, and proving mathematical claims is not applicable to this work. Additionally, we used generative AI tools to aid and polish writing, to retrieve and discover related work, and to draft sections of this paper. We have reviewed all AI-assisted work. AI-assisted code was verified and tested for correctness by the authors, and every AI-assisted statement about related work was checked against the cited paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Xavier Bouthillier, Pierre Delaunay, Mirko Bronzi, Assya Trofimov, Brennan Nichyporuk, Justin Szeto, Naz Sepah, Edward Raff, Kanika Madan, Vikram Voleti, Samira Ebrahimi Kahou, Vincent Michalski, Dmitriy Serdyuk, Tal Arbel, Chris Pal, Gaël Varoquaux, and Pascal Vincent. Accounting for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems (MLSys)*, volume 3, pp. 747–769, 2021.

Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

Wei-Jer Chang, Francesco Pittaluga, Masayoshi Tomizuka, Wei Zhan, and Manmohan Chandraker. SAFE-SIM: Safety-critical closed-loop traffic simulation with diffusion-controllable adversaries. In *European Conference on Computer Vision (ECCV)*, pp. 242–258. Springer, 2024.

Keyu Chen, Yuheng Lei, Hao Cheng, Haoran Wu, Wenchao Sun, and Sifa Zheng. FREA: Feasibility-guided generation of safety-critical scenarios with reasonable adversariality. In *Conference on Robot Learning (CoRL)*, volume 270, pp. 566–586. PMLR, 2024a.

Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):10164–10183, 2024b. doi: 10.1109/TPAMI.2024.3435937.

Kashyap Chitta, Daniel Dauner, and Andreas Geiger. SLEDGE: Synthesizing driving environments with generative models and rule-based traffic. In *European Conference on Computer Vision (ECCV)*, pp. 57–74. Springer, 2024.

Hyungjin Chung, Jeongsol Kim, Michael T McCann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations (ICLR)*, 2023.

Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning (CoRL)*, volume 229, pp. 1268–1281. PMLR, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 8780–8794, 2021.

Wenhao Ding, Chejian Xu, Mansur Arief, Haohong Lin, Bo Li, and Ding Zhao. A survey on safety-critical driving scenario generation—A methodological perspective. *IEEE Transactions on Intelligent Transportation Systems*, 24(7):6971–6988, 2023. doi: 10.1109/TITS.2023.3259322.

Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, volume 48, pp. 1050–1059. PMLR, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 3581–3590, 2017.

Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. KING: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *European Conference on Computer Vision (ECCV)*, pp. 335–352. Springer, 2022.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold preserving guided diffusion. In *International Conference on Learning Representations (ICLR)*, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 6840–6851, 2020.

Jiri Hron, Alex Matthews, and Zoubin Ghahramani. Variational Bayesian dropout: Pitfalls and fixes. In *International Conference on Machine Learning (ICML)*, volume 80, pp. 2019–2028. PMLR, 2018.

Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning (ICML)*, volume 162, pp. 9902–9915. PMLR, 2022.

Napat Karnchanachari, Dimitris Geromichalos, Kok Seang Tan, Nanxiang Li, Christopher Eriksen, Shakiba Yaghoubi, Noushin Mehdipour, Gianmarco Bernasconi, Whye Kit Fong, Yiluan Guo, and Holger Caesar. Towards learning-based planning: The nuPlan benchmark for real-world autonomous driving. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 629–636. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610077.

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5574–5584, 2017.

Loic Le Folgoc, Vasileios Baltatzis, Sujal Desai, Anand Devaraj, Sam Ellis, Octavio E Martinez Manzanera, Arjun Nair, Huaqi Qiu, Julia Schnabel, and Ben Glocker. Is MC dropout Bayesian? *arXiv preprint arXiv:2110.04286*, 2021.

Emanuele Ledda, Giorgio Fumera, and Fabio Roli. Dropout injection at test time for post hoc uncertainty quantification in neural networks. *Information Sciences*, 645:119356, 2023. doi: 10.1016/j.ins.2023.119356.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pp. 7167–7177, 2018.

Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020. doi: 10.1109/LRA.2020.2974682.

Enhui Ma, Lijun Zhou, Tao Tang, Zhan Zhang, Dong Han, Junpeng Jiang, Kun Zhan, Peng Jia, Xianpeng Lang, Haiyang Sun, Di Lin, and Kaicheng Yu. Unleashing generalization of end-to-end autonomous driving with controllable long video generation. *arXiv preprint arXiv:2406.01349*, 2024.

Osama Makansi, Özgün Çiçek, Yassine Marrakchi, and Thomas Brox. On exposing the challenging long tail in future prediction of traffic actors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13147–13157, 2021. doi: 10.1109/ICCV48922.2021.01290.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning (ICML)*, volume 139, pp. 8162–8171. PMLR, 2021.

Daehee Park, Monu Surana, Pranav Desai, Ashish Mehta, Reuben MV John, and Kuk-Jin Yoon. Generative active learning for long-tail trajectory prediction via controllable diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 27839–27850, 2025.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Davis Rempe, Jonah Philion, Leonidas J Guibas, Sanja Fidler, and Or Litany. Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17305–17315, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, 2022.

Luke Rowe, Roger Girgis, Anthony Gosselin, Liam Paull, Christopher Pal, and Felix Heide. Scenario Dreamer: Vectorized latent diffusion for generating driving simulation environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17207–17218, 2025.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2446–2454, 2020.

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 (CVPR)*, pp. 9909–9918, 2021.

Shuguang Wang, Qian Zhou, Kui Wu, Dapeng Wu, Wei-Bin Lee, and Jianping Wang. REDOUBT: Duo safety validation for autonomous vehicle motion planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Chejian Xu, Aleksandr Petiushko, Ding Zhao, and Bo Li. DiffScene: Diffusion-based safety-critical scenario generation for autonomous vehicles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 8797–8805, 2025. doi: 10.1609/aaai.v39i8.32951.

Haotian Ye, Haowei Lin, Jiaqi Han, Minkai Xu, Sheng Liu, Yitao Liang, Jianzhu Ma, James Zou, and Stefano Ermon. TFG: Unified training-free guidance for diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 22370–22417, 2024.

Chris Zhang, Sourav Biswas, Kelvin Wong, Kion Fallah, Lunjun Zhang, Dian Chen, Sergio Casas, and Raquel Urtasun. Learning to drive via asymmetric self-play. In *European Conference on Computer Vision (ECCV)*, pp. 149–168. Springer, 2024a.

Jiawei Zhang, Chejian Xu, and Bo Li. ChatScene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15459–15469, 2024b.

Ziyuan Zhong, Davis Rempe, Yuxiao Chen, Boris Ivanovic, Yulong Cao, Danfei Xu, Marco Pavone, and Baishakhi Ray. Language-guided traffic simulation via scene-level diffusion. In *Conference on Robot Learning (CoRL)*, volume 229, pp. 144–177. PMLR, 2023a.

Ziyuan Zhong, Davis Rempe, Danfei Xu, Yuxiao Chen, Sushant Veer, Tong Che, Baishakhi Ray, and Marco Pavone. Guided conditional diffusion for controllable traffic simulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3560–3566. IEEE, 2023b.

A Implementation Details

A.1 Generative Model

The generative model is a fork of Scenario Dreamer (Rowe et al., 2025) in which the object latents carry the future of every agent, which is the only change we make. The latent diffusion model runs over $T = 100$ steps with a cosine variance schedule (Nichol & Dhariwal, 2021) and predicts the noise, and sampling draws each step from the forward process posterior of equation 3. The lane stream is sampled at a temperature of 0.75 and every latent is clipped to ± 5 at each step. Classifier-free guidance (Ho & Salimans, 2022) is applied at a scale of 4.0 throughout, with the same setting for every variant we report. The denoiser is a factorized diffusion Transformer (Peebles & Xie, 2023) with two blocks, a lane stream of width 2048 with 16 heads and an agent stream of width 512 with 8 heads. An agent latent carries 32 dimensions and a lane latent carries 24, and a scene holds at most 61 agents and 100 lanes of 20 points each within a field of view of 64 meters. The denoiser holds 345.2 million parameters and the frozen autoencoder holds 27.9 million.

The autoencoder is trained for $357{,}450$ steps at a batch size of 128 on one GPU, which is about 100 epochs, with AdamW at a learning rate of 10^{-4} that decays linearly after $1{,}000$ warmup steps and a weight decay of 10^{-4}. The diffusion model is trained for $552{,}000$ steps at a batch size of 51 on six GPUs, and the checkpoint we sample from is taken at $550{,}160$ steps, with AdamW at a constant learning rate of 10^{-4} after 500 warmup steps, a weight decay of 10^{-5} and an exponential moving average of 0.9999 whose weights are the ones we sample with. All training runs on NVIDIA RTX A5000 cards.

The normalization constants are recomputed on the training split alone. Ten of the agent and lane fields match the released Scenario Dreamer values to rounding and the four fields of the future are new. Generated and recorded scenes share one set of constants whenever a planner is trained on both, which a hard check over fifteen keys enforces.

A.2 Planner

The planner is a set Transformer that carries no positional encoding and distinguishes its tokens only by a type embedding. It stacks four pre-norm encoder layers of width 256 with 8 attention heads and a feed-forward width of 512, with GELU activations. There are three token types, the ego agent with a seven-dimensional state and a three-dimensional type, a surrounding agent with the same ten dimensions, and a lane with a forty-dimensional polyline together with four connectivity and three type dimensions. Each type passes through its own linear projection, layer normalization, GELU and dropout, and the type embedding is added. A trajectory head on top of the encoder concatenates the encoded ego token with a masked mean over the tokens and maps the result through widths 256 and 16 to eight positions in the plane.

Dropout is applied at twenty places, three projections, three within each of the four encoder layers, one in the trajectory head, and the attention weights of the four attention modules. The rate is 0.1 during training and 0.5 at the ten passes that form $U(x)$, and the representation h_k is taken at the output of the encoder before the trajectory head.

A.3 Guidance

The three terms of equation 10 are mixed with weights of 0.5 for the uncertainty and 0.25 for each cost. The step size follows $\rho_t = \bar{\rho}\,\bar{\alpha}_t / \sum_{t'} \bar{\alpha}_{t'}$ with $\bar{\rho} = 5.0$, so the steps sum to $\bar{\rho}$ and grow toward the clean end of the reverse process. Guidance is applied at every one of the 100 reverse steps.

Each step proceeds in four stages. The rows of the ego agent in the raw gradient of each term are set to zero, each term is divided by its norm taken jointly over the agent and the lane stream within the scene, the terms are mixed with the weights renormalized over the terms that survive, and the mixture is divided by its own joint norm. The displacement of a scene at one step is therefore exactly ρ_t and the total displacement is $\bar{\rho}$. The ego rows are set to zero before the norm rather than after, so that the mass of the ego agent never enters the denominator, and only the ego row of the agent stream is affected. Zeroing those rows lets the objective shape the scene around the ego agent without moving the ego agent itself, so a cost cannot be lowered by moving the very trajectory it

scores. A variant that zeroed after the norm lost about 19 percent of the applied magnitude. A term counts as dead when its joint norm within the scene falls below 10^{-6} or when its gradient is absent, and a scene in which every term is dead takes no step.

A.4 Training Budgets

Every planner that we retrain on 50,000 generated scenes sees the same 223,274 scenes for the same 69,800 optimizer steps, and Table 5 gives that budget beside the planner without augmentation and the planner trained on the whole training split. All of them share a batch size of 128, AdamW at a constant learning rate of 10^{-4}, a weight decay of 10^{-4}, gradient clipping at 1.0, a smooth L1 loss and a single seed.

Table 5: Training budget of each planner below. The rows that use generated scenes share one budget exactly, which is the comparison that Table 2 and Table 4 rest on.

Training data	Scenes	Steps	Batch	Epochs
No augmentation (ID-only)	173,274	54,160	128	40.0
Unguided	223,274	69,800	128	40.0
POSG (ours)	223,274	69,800	128	40.0
Uncertainty only	223,274	69,800	128	40.0
Certainty only	223,274	69,800	128	40.0
Training split†	457,648	51,840	128	14.5

Those 69,800 steps are 40 epochs of the enlarged training set, so every comparison among the variants that use generated scenes is a comparison at an equal number of optimizer updates. The planner without augmentation follows the same rule of 40 epochs on its smaller training set, which gives it 54,160 steps. The row marked † comes from an earlier lineage whose principle was to hold the number of optimizer updates fixed regardless of how large the training set is, so that a difference between two runs could only come from what the training set contains. It inherits the 51,840 steps of that lineage, and spread over 457,648 scenes those steps amount to 14.5 epochs.

A.5 Generated Scenes

Every generated scene in this work is drawn from the same diffusion model, and from the same planner wherever the objective uses one. For POSG and for the two objectives of Table 3 we sample 200,000 scenes and keep the 50,000 with the highest uncertainty. For unguided sampling we set $\bar{\rho}$ to zero, sample 100,000 scenes and keep 50,000 of them at random. For the reversal of Table 4 we sample 100,000 scenes and keep the lowest half. That fraction was fixed before the run, which leaves the tail of certain scenes less extreme than the tail that the other variants keep.

B Additional Results

B.1 A Fully Held-Out Test Split

For the planner without augmentation and for every augmented variant, the checkpoint is the one that reaches the lowest error on the in-distribution part of the held-out scenes alone, so the out-of-distribution and unknown errors that we report never enter that choice. The row marked † in Table 2 is instead selected on the held-out scenes of every group. Table 6 evaluates the planner without augmentation and POSG on a separate test split of 66,237 scenes that takes part neither in training nor in that selection. The reduction out of distribution is 16.9 percent against the 18.5 percent of Table 2, and the reduction on the unknown group is 9.2 percent against 9.1 percent, so the improvement against no augmentation does not depend on the split that selected the checkpoints.

B.2 The Number of Generated Scenes

Unguided sampling draws from the distribution that the generative model learned, so its generated scenes carry more of that distribution as their number grows. Table 7 gives the out-of-distribution

Table 6: Average displacement error on a test split of 66,237 recorded scenes that is used neither for training nor for checkpoint selection. The split holds 23,678 in-distribution, 23,101 out-of-distribution and 19,458 unknown scenes.

Training data	ID ↓	OOD ↓	unknown ↓
No augmentation (ID-only)	0.5769	1.3151	1.1449
POSG (ours)	0.5838 (+1.2)	**1.0933** (−16.9)	**1.0396** (−9.2)

error of the retrained planner at two numbers of generated scenes. At 50,000 scenes the error with guidance is lower by 0.058, which our band separates, and at 100,000 scenes it is lower by 0.023, which lies within the band. The 50,000 scenes that POSG selects are not separable from the 100,000 scenes that unguided sampling draws. The objective therefore reaches with half the scenes what unguided sampling reaches with twice as many, which is what the guidance is worth per scene.

Table 7: Out-of-distribution error of the retrained planner for 50,000 and 100,000 generated scenes. Differences smaller than 0.042 are not separable.

Generated scenes	50,000	100,000
Unguided	1.1460	1.0760
POSG (ours)	**1.0876**	1.0528

B.3 Retraining on the Scenes that Uncertainty Only Selects

Table 3 compares the scenes of the two objectives by how often the generated ego trajectory is unsafe, and Table 8 compares the planners that those scenes produce. The two planners are not separable on any of the three groups, so the learnability costs leave the open-loop error unchanged and act on the generated scenes instead, where the share whose ego trajectory collides or leaves the lanes falls by more than half. An open-loop error measures how closely a planner reproduces a recorded trajectory, so we report the safety of the generated scenes as an axis of its own. The scenes that Table 4 produces by reversing the uncertainty term while the costs stay in place are the safest we sample, with 0.7 percent of them carrying a collision between vehicles, and their out-of-distribution error is the highest of the four, so the safety of the generated scenes does not by itself lower the error.

Table 8: Average displacement error after retraining on the scenes that uncertainty only selects, against those that POSG selects. Differences smaller than 0.042 are not separable.

Objective	ID ↓	OOD ↓	unknown ↓
POSG (ours)	0.5921	1.0876	0.9978
Uncertainty only	0.6017	1.0809	1.0108

B.4 The Dropout Rate at Test Time

We score a scene at a dropout rate of 0.5 while the planner was trained at 0.1. Two measurements support this rate.

A sweep over the rates 0.1, 0.3, 0.5, 0.7 and 0.9 separates the two spaces. On the two checkpoints trained at a rate of 0.1 the area under the curve for the cosine distance over representations peaks at 0.5, rising on one of them from 0.6689 at 0.1 to 0.6856 before falling to 0.4932 at 0.9, while the same quantity computed on the variance of the output trajectory falls monotonically from 0.6338 to 0.5745 to 0.4881. A third checkpoint trained at a rate of 0.5 peaks at 0.1 instead, so the rate that carries the most information differs from the training rate in every case. On the scenes we generate, the rank correlation between the cosine distance at a rate of 0.5 and the error of the frozen planner is 0.50 to 0.51, and its correlation with an ensemble of five planners trained from different seeds is 0.61 to 0.62. An ensemble over training seeds estimates uncertainty by a construction that shares

nothing with dropout, so the agreement between the two supports reading $U(x)$ as a measure of what the planner has learned. The variance of the output trajectory correlates at 0.49 to 0.52 at a rate of 0.1 and collapses to 0.27 to 0.29 at a rate of 0.5, while the cosine distance holds.

B.5 Variance Across Random Seeds

The natural scale for such a threshold is the variance of the benchmark itself, which is not known before the runs are made (Bouthillier et al., 2021). We measure that scale with six retrainings in an earlier configuration of the same pipeline, the unguided variant and a variant with an objective, each over the three seeds 0, 1 and 2. A seed changes the initialization of the weights and the order of the recorded scenes, and the sampled scenes are the same file in every run. The two variants therefore share every source of variation apart from the objective and are paired on the seed (Bouthillier et al., 2021). Out of distribution the variant with an objective reaches 1.3655, 1.3305 and 1.3483 and the unguided variant reaches 1.4190, 1.3556 and 1.3706, with standard deviations of 0.0175 and 0.0331 over the three seeds. Their paired difference is 0.0535, 0.0251 and 0.0223, with a mean of 0.0336, a standard deviation of 0.0173 and a range of 0.0312. We take 0.042 as the threshold, which is 2.4 times the standard deviation of the paired difference and lies above the range we observed.

C Data and Metrics

C.1 Scenario Types and Splits

The in-distribution group holds the twenty-seven types of Wang et al. (2025), which are stationary, near pedestrian on crosswalk, traversing intersection, on traffic light intersection, medium magnitude speed, traversing traffic light intersection, high magnitude speed, stationary in traffic, near pedestrian at pickup dropoff, stationary at traffic light without lead, near long vehicle, low magnitude speed, following lane without lead, stationary at traffic light with lead, on all way stop intersection, near high speed vehicle, near construction zone sign, following lane with slow lead, behind long vehicle, starting protected noncross turn, near pedestrian on crosswalk with ego, near multiple pedestrians, accelerating at traffic light, stopping with lead, following lane with lead, stopping at traffic light without lead, and waiting for pedestrian to cross. The training split of $457{,}648$ scenes holds $173{,}274$ in-distribution, $158{,}473$ out-of-distribution and $125{,}901$ unknown scenes. Of the out-of-distribution scenes, $47{,}422$ carry only out-of-distribution tags and $111{,}051$ carry a mixture, which the rule of Section 5.1 places outside the in-distribution group.

C.2 Safety of the Generated Ego Trajectory

The two collision rows of Table 3 differ in the footprint and in the agents they count, and neither is a subset of the other. The first covers a vehicle with five circles of radius $W/2$ placed evenly along its heading between $-L/2 + W/2$ and $L/2 - W/2$, and calls a pair a collision when the distance between two centers falls below $(W_i + W_j)/\sqrt{3.8}$. It counts vehicles alone and judges each of the eight future steps, taking the heading from the direction of the step. The second covers every agent with a single disk of radius $\frac{1}{2}\sqrt{L^2 + W^2}$ and calls a pair a collision on contact, with no margin and with partners of every type. The off-lane row counts a scene when the distance from the ego position to the nearest lane segment exceeds 1.5 meters at any of the eight steps. All three rows are reported with a Wilson interval at 95 percent.