Generative Active Learning for Long-tail Trajectory Prediction via Controllable Diffusion Model

Supplementary Material

1. Long-tail definition

Tail agents are defined dynamically by the current model error, so their characteristics shift during training. Nonetheless, we analyzed tails of the initial model (Fig. 1): tail (orange) vs. head (blue) histograms for agent density, Δ-heading, speed, and a speed + heading composite. Density barely differs, but tails show larger headings/speeds; the composite confirms this. However, many tail cases do not exhibit these extremes (red circled), implying the presence of semantic uniqueness. Leveraging such hard-to-explain tails during training is a key strength of our framework.

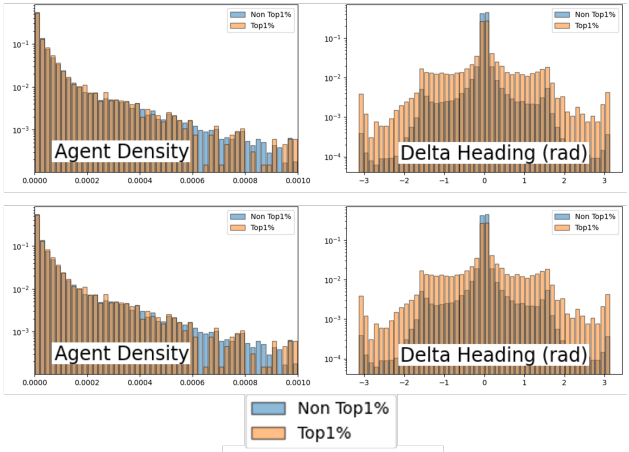

Figure 1. Distribution comparison on the GT trajectories of tail and head agents.

2. Random time window shift

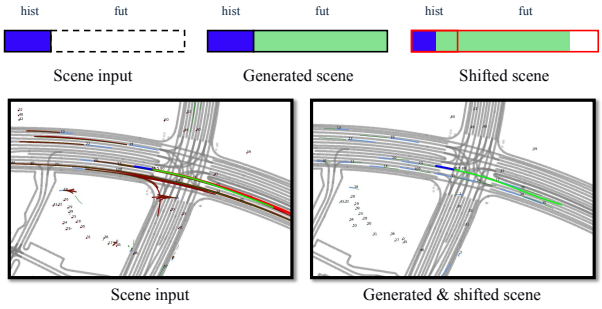

Figure 2. The left side shows input scene, and the right side shows generated and shifted scene. Blue line indicates historical trajectory while green line indicates future trajectory.

The random time window shift technique applies a random shift to the current time $t = 0$ in the trajectory of all agents within a scene. The shift, denoted as δt, is sampled from an exponential distribution with a scale parameter of 0.5, ensuring smaller values are sampled more frequently. To normalize the shift, δt is scaled to a range of $[0, 1]$ by applying:

$$\delta t_{\text{norm}} = \frac{\delta t}{\delta t + 1}.$$

The normalized shift is then multiplied by half the future time length $T/2$ to calculate the final shift value. Figure 2 illustrates the results of scenario generation after applying the random time window shift. Notably, a portion of the generated scenario is utilized as input, demonstrating how shifted trajectories integrate into the process.

3. Metric definition

3.1. minADE$_6$ and minFDE$_6$

The metrics minADE$_6$ (minimum Average Displacement Error) and minFDE$_6$ (minimum Final Displacement Error) are widely used to evaluate the accuracy of trajectory prediction models in a multimodal context.

minADE$_6$: Measures the average displacement error across the trajectory for the most accurate prediction among six modes.

minFDE$_6$: Measures the displacement error at the final time step for the most accurate prediction among six modes. Mathematically, they are defined as follows:

$$\text{minADE}_6 = \min_{F \in 1,\ldots,6} \frac{1}{T} \sum_{t=1}^{T} \left|\left| \hat{\mathbf{y}}_t^{(F)} - \mathbf{y}_t \right|\right|_2 \tag{1}$$

$$\text{minFDE}_6 = \min_{F \in 1,\ldots,6} \left|\left| \hat{\mathbf{y}}_T^{(F)} - \mathbf{y}_T \right|\right|_2 \tag{2}$$

Where:
$\hat{\mathbf{y}}_t^{(F)}$: Predicted position at time t for the F-th mode.
$\mathbf{y}_t$: Ground truth position at time t .
T : Total number of time steps in the trajectory.

3.2. Difference between FPR and VaR

Fig. 4 illustrates the differences between the FPR and VaR metrics discussed in the main text. Both metrics evaluate the quality of the model's error distribution. However, the distinction lies in their definitions: FPR measures the proportion of agents whose prediction error exceeds a predefined threshold, while VaR quantifies the prediction error corresponding to the 1-α quantile of the error distribution.

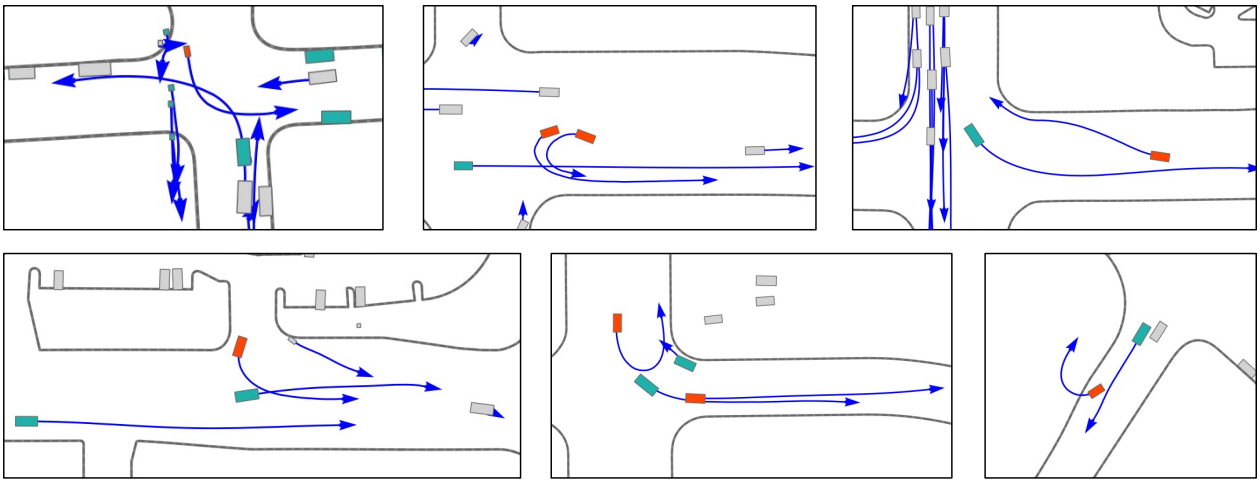

Figure 3. Visualization of the agent type classification results. ood, relevant, normal agents are highlighted with red, turquoise, and gray bounding boxes, respectively. This figure illustrates that multiple OOD agents can exist in a single scene. Relevant agents are effectively identified as those with the highest interaction at future timestamps.

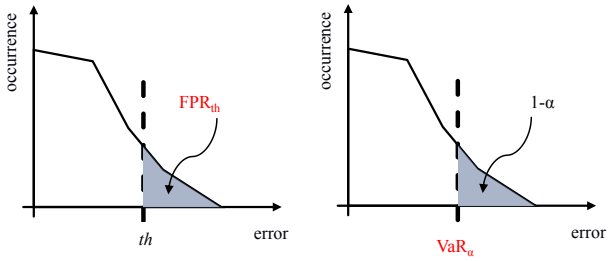

Figure 4. Illustration of the FPR and VaR metrics with respect to the error distribution. The x-axis represents the prediction error, specifically minADE_6, and the y-axis indicates the count (occurrence) of agents within each error bin. Both metrics provide distinct yet complementary perspectives for assessing the quality of the error distribution.

These two metrics provide complementary and intuitive criteria for assessing the quality of the error distribution.

4. Implementation details

We utilize 4 $\times$ A6000 GPUs for model training. Due to modifications in the training procedure, we employ Pytorch Lightning Fabric [2]. Our predictor retains the original QCNet [6] architecture without any structural changes. The hyperparameters used for training align with those in the original paper. For the LCSim model [5], the encoder architecture remains unchanged from the original paper. However, we introduce a future-future attention module to the denoiser to enhance its functionality, inspired by [3] and [4]. This module uses a MessagePassing layer, akin to other attention modules within the denoiser. Its purpose is to calculate attention across generated agents' trajectories, ensuring inter-agent relationship consideration and scenario plausibility. For identifying relevant agents via attention scores, we utilize scores from the future-future attention module as described in the main text. The attention score from the last denoising step in the entire reverse process is employed.

For generative active learning, we set the sampling weight decay to 0.7 and the sampling weight clipping minimum value to 0.5. The maximum train set size is set to twice the size of the original train set. If the train set exceeds this size, the earliest generated data is excluded to maintain the limit. For the random time window shift, we use an exponential distribution with a scale parameter of 0.5 to sample the shift. The sampled value is normalized to the range $[0, 1]$ and then multiplied by 40 to compute the final shift value.

5. Definition of traffic rules

No-Off-Road Constraint To ensure that generated trajectories remain within road boundaries, we define the no-off-road objective as a penalty function that discourages deviations from the drivable area. Let $\mathbf{y}^n = \{\boldsymbol{p}_t^n\}_{\Delta t:T_f}$ denote the future trajectory of agent n , where $\boldsymbol{p}_t^n = (x_t^n, y_t^n)$ represents the agent's position at time t . The objective is formulated as follows:

$$\mathcal{C}_{\text{no-off-road}}(\mathbf{y}^n) = \sum_{t=\Delta t:T_f} \max(0, d_{\text{road}}(\boldsymbol{p}_t^n) - \epsilon_{\text{off}}), \quad (3)$$

where $d_{\text{road}}(\boldsymbol{p}_t^n)$ represents the Euclidean distance between $\boldsymbol{p}_t^n$, and the closest drivable area, and ϵ_{off} is a tolerance threshold. The function penalizes points that lie outside the road region, enforcing adherence to traffic lanes.

Repeller Constraint To prevent collisions between generated trajectories, we introduce a repeller objective that enforces a minimum safe distance between agents. For a pair of agents n and m , we define the repulsion function:

$$\mathcal{C}_{\text{repeller}}(\mathbf{y}^n, \mathbf{y}^m) = \sum_{t=\Delta t:T_f} \max(0, \epsilon_{\text{rep}} - ||\boldsymbol{p}_t^n - \boldsymbol{p}_t^m||), \quad (4)$$

where d_{min} is the minimum safety distance threshold, and $\|\boldsymbol{p}_t^n - \boldsymbol{p}t^m\|$ denotes the Euclidean distance between agents n and m at time t . This objective imposes a penalty when the inter-agent distance falls below ϵ_{rep} , encouraging safer, collision-free trajectories.

Integration of gradient guidance During inference, these objectives are incorporated into the gradient-based guidance framework. The total guidance function is given by the sum of two guidance terms. The gradient is then used to adjust the predicted mean at each denoising step, ensuring that the generated trajectories comply with traffic rules.

6. Agent type classification results

We illustrate the classification results of our proposed method, where agents are categorized into three types: *ood* (out-of-distribution), *relevant*, and *normal*. As shown in Fig. 3, ood agents are represented with red bounding boxes, relevant agents with turquoise bounding boxes, and normal agents with gray bounding boxes. The blue arrows indicate the generated scenarios. As depicted in the figure, multiple ood agents can exist within a single scene. Relevant agents are accurately classified as those with the highest interaction at future timestamps. This demonstrates the effectiveness of the proposed method in capturing inter-agent dynamics and interactions.

(1) Stability: We follow the LCSim decoder and add a future-future attention module that computes attention over the to-be-generated future, not past interactions. Therefore, our method adapts to behavior changes during scene generation, and Fig. 3 shows stable classification results across generated scenes.

(2) Threshold: We use a threshold of 1/N, where N is the number of adjacent agents. From our observations, attention scores for non-relevant ("other") agents are nearly zero, while relevant agents tend to evenly share the attention (approximately 1/n, where n is the number of true relevant agents). Since 1/n is much greater than 1/N, this threshold effectively separates relevant agents, and we have found the classification to be robust under this rule.

Table 1. Additional experimental results on nuScenes dataset with different backbones and evaluation metrics. In this experiment, we further investigate the performance of different training methods on an additional dataset (nuScenes), an additional backbone (MTR), and additional evaluation metrics (Top 5%, VaR_{97}).

	Method	Long-tail metrics			Overall metric
		Top 1%	VaR_{99}	FPR_5	minFDE_6
QCNet	Vanilla	12.59	9.60	4.11%	1.691
	resampling	10.39	8.91	3.31%	1.889
	cRT	10.96	9.02	3.33%	1.730
	contrastive	10.33	8.74	3.21%	1.864
	GALTraj	**8.53**	**7.04**	**2.08%**	**1.513**
	Method	Top 5%	VaR_{97}	FPR_{10}	minFDE_6
MTR	Vanilla	6.54	7.25	0.93%	2.174
	resampling	5.34	5.72	0.60%	2.192
	cRT	5.53	6.01	0.64%	2.140
	contrastive	4.96	5.54	0.53%	2.181
	GALTraj	**4.11**	**4.70**	**0.48%**	**2.098**

7. Additional experiments: backbone, metric and dataset

To further validate the robustness and generalization capability of our approach, we conducted additional experiments on a new dataset (nuScenes), an additional backbone (MTR), and new evaluation metrics (Top 5% and VaR_{97}). For MTR, we use the github repositories of UniTraj [1]. The results of these experiments are summarized in Table 1. We extended our evaluation to nuScenes, a dataset with different distribution characteristics and challenges. Additionally, we introduced MTR as an alternative backbone to examine the effect of different architectures on long-tail performance. Furthermore, we incorporated new evaluation criteria: 5% (Top 5%), 3%(VaR_{97}), and FPR_{10} to capture a broader range of failure cases and assess the overall stability of trajectory prediction models under long-tail scenarios.

The results in Table 1 demonstrate that GALTraj consistently achieves the best performance across all settings, outperforming all baselines under both QCNet and MTR backbones. Notably, when using QCNet, GALTraj reduces the FPR_5 error rate by over 20% compared to other methods, achieving the lowest failure rate of 2.08%. Similarly, under the MTR backbone, GALTraj maintains the lowest prediction error, indicating strong generalization capabilities. While Resampling, cRT, and Contrastive learning techniques outperform the baseline (Vanilla), they still fall short of GALTraj's superior performance. Among the baselines, Contrastive learning performs the best, yet it remains over 10% worse than GALTraj in long-tail scenarios. This highlights the effectiveness of GALTraj's specialized design in handling imbalanced trajectory distributions.

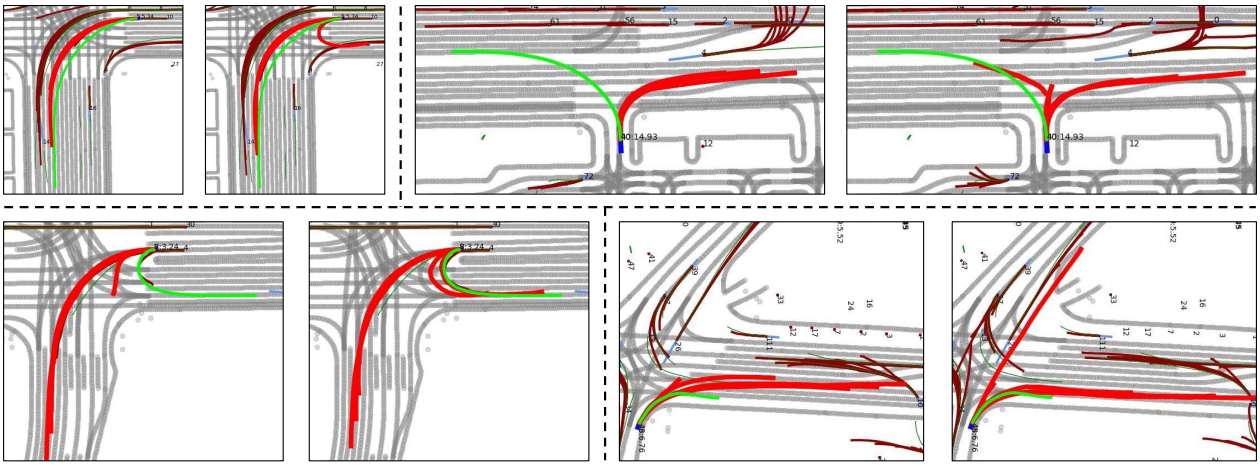

Figure 5. Additional visualization for the main experiment. In each image pairs, the left one indicates prediction results from the vanilla method, and the right one shows prediction results from the proposed method. Red lines represent multi-modal predictions, blue lines indicate the history, green lines show the ground truth future motion, and gray lines represent lane centerlines. Bright, thick lines highlight the trajectories of out-of-distribution (*ood*) agents.

8. Dataset details

We conduct experiments on three datasets: Waymo Open Motion Dataset (WOMD), Argoverse 2, and nuScenes. For WOMD and Argoverse 2, we adopt the temporal configuration defined in their respective benchmarks without modification. Specifically, in WOMD, the history length, future length, and time interval are set to 1 second, 8 seconds, and 0.1 seconds, respectively. Similarly, for Argoverse 2, we strictly follow its predefined benchmark settings: 3-second history, a 6-second future, and a time interval of 0.1 seconds.

However, training a diffusion-based model on nuScenes is impractical due to its relatively small dataset size. To enable the reuse of our diffusion model pre-trained on WOMD, we reorganized the temporal configuration of the nuScenes dataset to align it with WOMD. The total duration remains the same at 9 seconds. The original nuScenes benchmark defines a 3-second history, a 6-second future, and a 0.5-second time interval, which differs significantly from WOMD. To bridge this gap, we perform two modifications: We reorganize the temporal configuration of the nuScenes dataset to match that of WOMD. Notably, the total duration of nuScenes data is the same as WOMD, i.e., 9 seconds. First, since the temporal interval in nuScenes is larger, we interpolate the data to achieve a time interval of 0.1 seconds. Next, we shift the current time ($t = 0$) backward by 2 seconds in the original nuScenes configuration. As a result, the temporal configuration of nuScenes data is reorganized to match WOMD's configuration, i.e., 1-second history, 8-second future, and a 0.1-second time interval.

9. Error distribution analysis

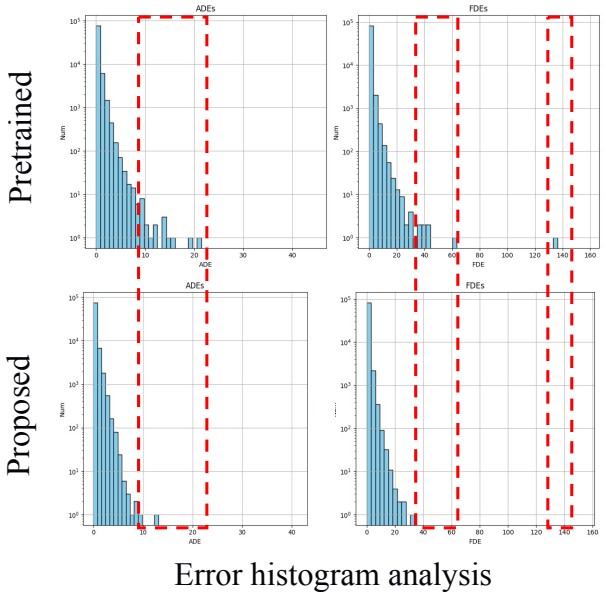

Figure 6. Histogram comparison of minADE_6 and minFDE_6 before and after applying the proposed training method. The x-axis corresponds to the error magnitude, and the y-axis shows the agent count for each error bin. The red dashed boxes highlight the regions where the proposed method effectively reduces large errors, showcasing its ability to improve performance on challenging cases.

In Fig. 6, we compare the histograms of minADE_6 and minFDE_6 before and after applying the proposed training method. The x-axis represents the magnitude of the error, while the y-axis indicates the number of agents (occur-

rences) within each bin. As highlighted by the red boxes in the figure, the proposed training method significantly improves the performance for agents that originally exhibited large errors. This demonstrates the effectiveness of the method in addressing high-error scenarios.

10. More qualitative results

Figure 5 provides additional qualitative results to demonstrate the effectiveness of the proposed method. The visualizations show that the proposed method not only performs robust predictions for unique maneuvers but also learns a diverse distribution of maneuvers across various challenging scenarios. Compared to the vanilla method, the proposed approach demonstrates better coverage of plausible trajectory distributions. This ensures that the predictions align more closely with the complexity and diversity required in real-world applications.

References

[1] Lan Feng, Mohammadhossein Bahari, Kaouther Messaoud Ben Amor, Éloi Zablocki, Matthieu Cord, and Alexandre Alahi. Unitraj: A unified framework for scalable vehicle trajectory prediction. In *European Conference on Computer Vision*, pages 106–123. Springer, 2024. 3

[2] Lightning-AI. Lightning fabric: Expert control over pytorch training loops. https://github.com/Lightning-AI/pytorch-lightning, 2023. 2

[3] Daehee Park, Hobin Ryu, Yunseo Yang, Jegyeong Cho, Jiwon Kim, and Kuk-Jin Yoon. Leveraging future relationship reasoning for vehicle trajectory prediction. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[4] Sungmin Woo, Minjung Kim, Donghyeong Kim, Sungjun Jang, and Sangyoun Lee. Fimp: Future interaction modeling for multi-agent motion prediction. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14457–14463, 2024. 2

[5] Yuheng Zhang, Tianjian Ouyang, Fudan Yu, Cong Ma, Lei Qiao, Wei Wu, Jian Yuan, and Yong Li. Lcsim: A large-scale controllable traffic simulator, 2024. 2

[6] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-centric trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17863–17873, 2023. 2