Closed Loop Dynamic Driving Data Mixture for Real-Synthetic Co-Training

Hongzhi Ruan [1,2] **Pei Liu** [1,3] **Weiliang Ma** [1] **Zhengning Li** [1]
Xueyang Zhang [1] **Kun Zhan** [1] **Jun Ma** [2,3] **Dan Xu** [2]

[1] Li Auto [2] HKUST [3] HKUST (GZ)
ruanhongzhicn@gmail.com, danxu@cse.ust.hk

Abstract

Data scaling is fundamental to modern deep learning, and grows increasingly critical as autonomous driving shifts to end-to-end learning. Real-world driving data is expensive to annotate and scene-biased, making real-synthetic co-training with near-infinite synthetic data a promising direction. However, naively incorporating all available synthetic data is inefficient and leads to distribution shifts, and optimizing data mixture under practical training budgets remains a critical yet under-explored problem. In this sense, we claim that the mixture of training data requires clear guidance in terms of scene types and quantities. Particularly in this work, we conceptualize the data mixture approximately as a dynamic optimization process that iteratively adjusts the training data mixture to maximize model performance, guided by closed-loop evaluation feedback, and propose AutoScale, a fully automated closed-loop data engine unifying scene representation, data mixture optimization and retrieval, as well as model training and evaluation. Specifically, we propose Graph Regularized AutoEncoder (Graph-RAE) for driving scene representations, introduce Cluster-aware Gradient Ascent (Cluster-GA) for cluster-wise importance estimation and reweighting, and perform cluster-guided vector retrieval to select high-value samples. Experiments on NavSim demonstrate that AutoScale outperforms vanilla co-training and cross-domain baselines, achieving better performance with fewer synthetic samples under constrained budgets.

1 Introduction

Data scaling has become a fundamental principle for building large modern deep learning models across language, vision, and generative tasks [3, 23, 11, 37, 2]. As autonomous driving evolves from modular pipelines toward end-to-end learning from raw sensor data to actions [20, 5], data scaling laws have grown increasingly critical in this domain.

Real-world driving data, while reliable, is inherently limited, expensive to annotate, and biased toward frequent, safe driving scenarios. To improve data diversity and coverage, recent research has begun to focus on synthetic data, which can be generated in large quantities at low cost. Leveraging generation or reconstruction techniques, recent works attempt to generate a large amount of synthetic data based on prior scenes, and further achieve performance improvement through real-synthetic co-training [38, 17, 41]. However, existing works mostly focus on generating more and better synthetic data, while few have systematically studied how to optimally mix them with real data.

Under practical training budgets, it is critical to optimize the data mixture and precisely identify valuable samples for model improvement, as the amount of data the model can effectively learn from is inherently limited [1, 53]. In practice, naively injecting all available synthetic data is inefficient and often degrades performance due to redundancy, noise, and distribution mismatch Fig. 1 (a). This problem remains under-explored because of several key challenges: (1) Diverse driving scenes

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

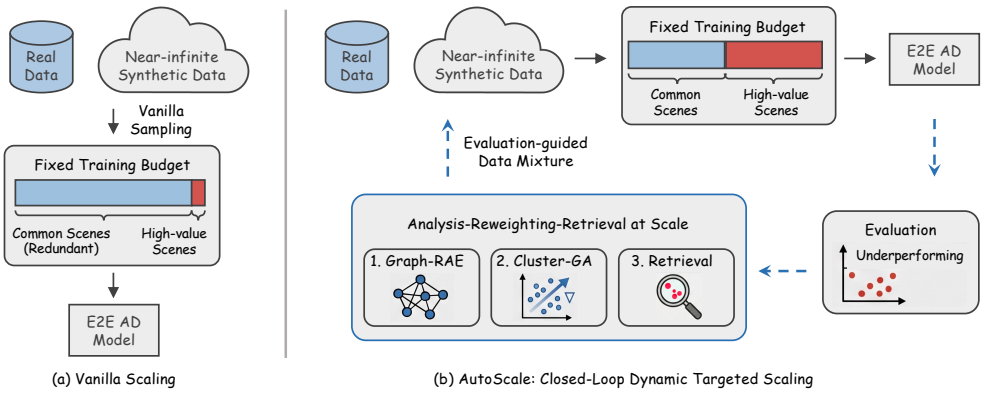


Figure 1: Scheme Comparison. (a) Existing scaling for real-synthetic co-training adopts vanilla sampling from nearly infinite unguided synthetic data. (b) Our AutoScale closed-loop framework leverages evaluation feedback to enable scalable scene analysis, cluster-wise reweighting, and targeted synthetic data retrieval, providing explicit guidance for effective data mixture optimization.

are structurally complex and difficult to analyze and localize at scale; (2) It is difficult to quantify the impact of the types and proportions of different driving scenes in the training set on model performance, which hinders the optimization of the training data mixture; (3) It is challenging to use model failure cases from evaluation to targetedly enhance the training distribution.

In this paper, we present a pioneering systematic investigation of the data mixture effect on overall model performance for real-synthetic co-training. We adopt a practical and realistic setting: a fixed, well-distributed real dataset is preserved without modification, and a large-scale pre-generated external dataset serves as an accessible, low-cost candidate pool, from which we sample a fixed-budget set of synthetic data to construct a superior mixture for real-synthetic co-training. We argue that the mixture of training data requires clear guidance in terms of scene types and quantities. With this setting, we further formulate the incorporation of synthetic data approximately as a gradient ascent optimization process toward improved model performance. In practice, we introduce AutoScale Fig. 1 (b), a closed-loop dynamic data engine that unifies scene representation, data mixture optimization and retrieval, model training, and evaluation for real-synthetic co-training.

Specifically, to precisely address the aforementioned challenges, we first propose Graph Regularized AutoEncoder (Graph-RAE), which encodes and reconstructs driving scene information and integrates contrastive and metric learning regularization to represent driving scenes as vectors. Second, we cluster driving scenes using these learned scene vectors, and further introduce Cluster-aware Gradient Ascent (Cluster-GA) to iteratively estimate cluster-wise importance weights within the training-evaluation closed loop for data reweighting. Finally, guided by the learned cluster importance, we leverage scene vectors to retrieve high-value synthetic samples that compensate for model weaknesses.

We conduct extensive experiments on the NavSim [8] benchmark. We use the training subset from the navtrain split as our fixed real dataset, and adopt the large-scale SimScale [41] dataset generated via 3D Gaussian Splatting [24, 27], from which we split a held-out calibration set for closed-loop evaluation and use the remainder as our synthetic candidate pool. Evaluated on the regression-based planner LTF [7] and diffusion-based planner DiffusionDrive [29], AutoScale utilizes simulation evaluation guidance to outperform vanilla co-training and cross-domain methods, achieving better real-world benchmark results with fewer synthetic samples. Ablation studies and comprehensive analyses verify the effectiveness of Graph-RAE, Cluster-GA, and vector-based retrieval.

Our contributions are fourfold:

- We pioneer the optimization of data mixtures for improved model performance in real-synthetic co-training, and propose a fully automated closed-loop data engine.
- We propose Graph-RAE, a Graph Regularized AutoEncoder integrated with contrastive and metric learning, enabling effective representation and efficient analysis of driving scenes at scale.
- We introduce Cluster-GA, a gradient ascent-based method for driving scene cluster reweighting.
- We perform vector-based scene retrieval to retrieve high-value synthetic samples for training.

2 Related Work

Data Centric Autonomous Driving. With the shift of autonomous driving from modular pipelines to end-to-end learning that maps raw sensor data to driving actions [20, 5], data distribution learning has become increasingly critical. Representative methods include regression-based planners [7, 22, 39] and generation-based planners [29, 50, 28, 52], both of which rely heavily on high-quality training data. On the other hand, since real-world driving data is expensive to annotate and scene-biased, recent research has turned its attention to synthetic data with inherently precise annotation and near-infinite generation at low cost. Existing synthetic data methods can be divided into two categories: generation-based approaches [19, 15, 33, 10, 46, 14, 45, 13] and reconstruction-based approaches [30, 16, 41]. Beyond using synthetic data alone, real-synthetic co-training has been proven to achieve better model performance than single-data-source training [41, 38, 17]. However, most existing works focus on generating more and higher-quality synthetic data; despite the availability of near-infinite synthetic data, few studies have systematically investigated how to optimally mix synthetic data with real data under practical training budgets, this gap is the core focus of our work.

Data Mixtures. The composition of pre-training datasets is critical in determining the generalization ability of large language models (LLMs) [3, 43]. As a result, a large body of research has focused on data mixture optimization in LLMs [48, 12, 31, 9, 51]. Similarly, the mixture of real and synthetic data is also a critical problem in action imitation learning for robotics [36, 34, 35, 40, 42]. Despite its importance, data mixture optimization remains under-explored in both robotics and autonomous driving [1, 53], primarily due to the structural complexity of representing both states and actions, which hinders large-scale analysis and localization of key data patterns. To highlight the advantages of our proposed AutoScale, we select two representative data mixture methods as baselines. Chameleon [49], a data mixture framework designed for LLMs, leverages domain text embeddings to assign mixture weights to each domain, achieving superior performance on language benchmarks. IWR [47] introduces Importance Weighted Retrieval, which estimates the importance weights of target data samples within data distributions and measures robotics task similarity in the latent space to guide sample retrieval. In this work, we first present Graph-RAE to encode driving scenes into structured vectors, addressing the challenge of complex driving scene representation. Further, unlike most existing data mixture baselines that rely solely on training data distribution analysis, we optimize data mixture through cluster-wise reweighting and targeted sample retrieval, guided by closed-loop model performance evaluation feedback.

3 Task Definition

We consider an end-to-end imitation learning setting for autonomous driving, where the model is trained on a mixed dataset consisting of fixed real-world data and selectively augmented synthetic driving scenes. Let $\mathcal{D}_{\text{real}}$ denote the real training set with $N_0 = |\mathcal{D}_{\text{real}}|$ samples, and $\mathcal{D}_{\text{syn}}$ represent a large candidate pool of synthetic data generated by specialized synthesis pipelines. Given a fixed synthetic data budget B, the total size of the final mixed training set is $N = N_0 + B$.

Each driving scene is treated as a data sample x, and its contextual information is encoded into a structured vector $\mathbf{v}_x \in \mathbb{R}^d$ via our proposed Graph-RAE (Section 4.1). A held-out calibration split $\mathcal{D}_{\text{cal}}$ is adopted to evaluate the driving policy π_θ with closed-loop driving metrics, producing a comprehensive performance score S. In our setting, all real data is fully preserved, while only synthetic samples are selected to fill the predefined budget.

Target. Our objective is to select optimal B synthetic samples from $\mathcal{D}_{\text{syn}}$ to augment $\mathcal{D}_{\text{real}}$ and refine the real-synthetic data mixture. Guided by simulation evaluation score S as quantitative feedback, we aim to enhance overall model performance on the real-world test set.

4 Methodology

This section presents AutoScale, a closed-loop dynamic data engine that unifies scene representation, data mixture optimization, retrieval, model training, and evaluation for real-synthetic co-training. The three core modules in the framework are depicted in fig. 2.

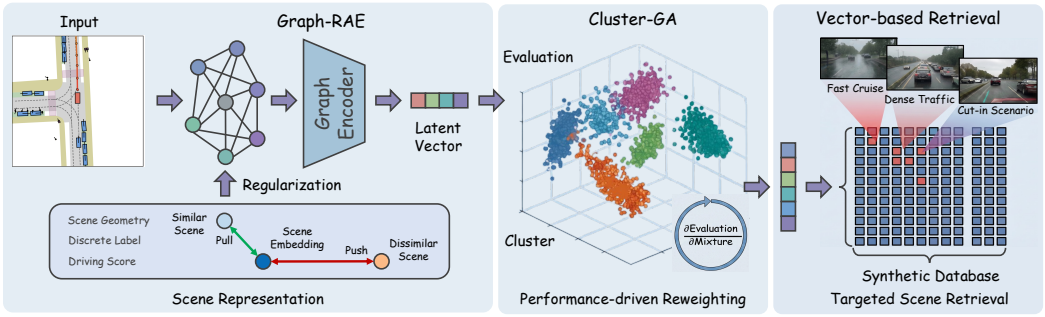


Figure 2: Overview of the proposed AutoScale with three core modules. (a) Scene representation learning: Graph-RAE encodes complex driving scenes into structured vectors. (b) Cluster-level weight optimization: Cluster-GA estimates cluster-wise data importance based on closed-loop evaluation feedback. (c) Targeted synthetic data retrieval: Vector-based retrieval selects high-value synthetic samples to adjust the training distribution for model underperformance.

4.1 Graph Regularized Autoencoder (Graph-RAE)

We learn a scene-level representation $\mathbf{v}_x \in \mathbb{R}^d$ that jointly captures the static map structure, dynamic traffic participants, and driving behavior of the ego vehicle within each traffic scene.

Graph Spatial-Temporal Autoencoder. We model each driving scenario as a heterogeneous spatio-temporal graph constructed over T_{len} sequential frames (T_h historical + T_f future). The graph consists of three distinct node categories: the ego-vehicle, dynamic traffic participants (vehicles, pedestrians, cyclists, etc.), and map structural elements (ped_crossing, divider, boundary). Each node is associated with a T_{len}-step polyline sequence comprising 2D position, heading, and curvature. Directed edge types are designed to encode pairwise spatial interactions, including map-to-dynamic, dynamic-to-dynamic, and all-to-ego relational dependencies; each edge encodes the relative position, heading difference, and curvature deviation between connected nodes across all T_{len} steps.

Node features at each step are encoded via shared multi-layer perceptrons (MLPs), augmented with learnable positional embeddings and node-type embeddings, followed by sequential aggregation to produce a node representation $\mathbf{h}_i \in \mathbb{R}^d$. Edge features are encoded through an identical pipeline. An L-layer Graph Transformer then refines all node representations via gated multi-head attention [44], formulated as:

$$\mathbf{h}_i^{(l+1)} = \mathbf{h}_i^{(l)} + \sigma\big(\mathbf{g}_i^{(l)}\big) \odot \mathrm{MHA}\big(\mathbf{h}_i^{(l)},\ \{\mathbf{h}_j^{(l)}, \mathbf{e}_{ji}\}_{j\in\mathcal{N}(i)}\big), \tag{1}$$

where $\sigma\big(\mathbf{g}_i^{(l)}\big)$ denotes a learnable gating function and MHA computes multi-head attention over neighboring node features concatenated with edge attributes. The global scene representation is derived from the ℓ_2-normalized embedding of the ego-vehicle node after the final layer and enhanced with regularization. Finally, a graph decoder reconstructs the complete polyline sequences of all nodes from the learned representations, optimized using the Huber loss.

Contrastive Learning for Regularization. To regularize the embedding space with traffic semantics, we employ two contrastive regularization objectives. First, the SimCLR [6] objective enforces invariance to geometric perturbations: two augmented views of the same scene, generated by random translation and rotation of all polylines, are treated as positive pairs, while all other scenes serve as negative samples. Second, the supervised contrastive (SupCon) [25] objective clusters scenes according to discrete semantic labels, pulling embeddings of scenes with consistent human-annotated priors closer together. Here, we simply define two semantic annotation types: a command pattern label that characterizes the ego-vehicle's high-level driving intent, and a geometric overlap label that indicates whether the ego trajectory intersects with dynamic agents or map elements. This design allows seamless integration of additional semantic labels for enhanced representation learning.

Metric Learning for Regularization. To further align the embedding space with driving task semantics, we introduce a pairwise metric regression objective. For each pair of scenes (x_i, x_j) in a batch, their ego embeddings are projected via an MLP and compared. We run training samples with evaluation metrics, and adopt a regression head to predict the element-wise absolute difference of evaluation metric vectors $|\mathbf{m}_i - \mathbf{m}_j| \in \mathbb{R}^{N_m}$, where N_m denotes the dimension of evaluation metrics. The objective is supervised by the mean squared error (MSE) loss over all valid pairs. This

objective ensures that scenes with analogous driving performance scores are embedded in proximity, directly benefiting downstream driving analysis tasks.

The overall training loss is formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda \left(\mathcal{L}_{\text{contra}} + \mathcal{L}_{\text{metric}} \right). \tag{2}$$

4.2 Gradient Ascent Cluster Reweighting

We first partition driving scenes into distinct clusters, and formulate the relationship between cluster-wise data mixture and evaluation performance as a gradient ascent optimization paradigm (Cluster-GA). A predictor is further introduced to iteratively model this optimization process.

Data Clustering. Using the scene embeddings from Graph-RAE, we partition all real training tokens into K clusters $\{C_1, \ldots, C_K\}$ via Gaussian Mixture Models (GMM) on ℓ_2-normalized vectors, yielding cluster centroids $\{\mathbf{c}_1, \ldots, \mathbf{c}_K\}$. Let $n_{0,k}$ denote the number of real tokens in cluster k, and define the data mixture vector $\mathbf{w} \in \Delta_K$, where w_k is the fraction of total samples N allocated to cluster k. The calibration set is assigned to the same clustering partition with a fixed mixture $\boldsymbol{\pi} \in \Delta_K$, enabling per-cluster evaluation feedback. The evaluation metric produces per-cluster scores $\bar{s}_k$, and the overall evaluation performance is equivalently formulated as $S(\mathbf{w}) = \boldsymbol{\pi}^\top \bar{\mathbf{s}}(\mathbf{w})$. To capture inter-cluster transfer effects, since data from one cluster can benefit semantically similar clusters, we adopt a precomputed fixed similarity matrix $\mathbf{R} \in \mathbb{R}^{K \times K}$, whose entries follow the RBF kernel: $R_{kj} = \exp\big(-(1 - \cos(\mathbf{c}_k, \mathbf{c}_j))^2 / (2\sigma^2)\big)$.

Cluster Performance Predictor. Since the evaluation score $S(\mathbf{w})$ requires a full train–evaluate closed-loop to measure, we fit a model that predicts how changes in data mixture affect per-cluster scores. Given two observed rounds (a, b) with mixtures $\mathbf{w}^{(a)}, \mathbf{w}^{(b)}$ and scores $\bar{\mathbf{s}}^{(a)}, \bar{\mathbf{s}}^{(b)}$, the per-cluster score difference is modeled as:

$$\Delta \bar{s}_k^{(a \to b)} = \sum_{j=1}^{K} R_{kj}\, \beta_j\, \Delta \log w_j + \gamma \sum_{j=1}^{K} R_{kj}\, \Delta r_j, \tag{3}$$

where $\Delta \log w_j = \log w_j^{(b)} - \log w_j^{(a)}$ and $\Delta r_j = r_j(\mathbf{w}^{(b)}) - r_j(\mathbf{w}^{(a)})$. $r_j(\mathbf{w}) = \max(0,\, 1 - n_{0,j}/(w_j N))$ denotes the synthetic data ratio for cluster j. β_j captures the marginal data return in cluster j (the logarithmic term reflects diminishing returns), and γ scales synthetic data effects.

In each iteration, given t historical rounds, we construct all $\binom{t}{2}$ pairwise samples and solve for parameters $\boldsymbol{\phi} = [\boldsymbol{\beta}^\top;\, \gamma] \in \mathbb{R}^{K+1}$ via weighted ridge regression in closed form:

$$\boldsymbol{\phi} = \left(\mathbf{X}^\top \mathbf{W} \mathbf{X} + \lambda_{\text{reg}} \mathbf{I} \right)^{-1} \mathbf{X}^\top \mathbf{W}\, \mathbf{y}. \tag{4}$$

where $\mathbf{X}$ and $\mathbf{y}$ are the feature matrix and regression target vector, $\mathbf{W}$ is a diagonal weight matrix, $\mathbf{I}$ denotes the identity matrix, and λ_{reg} controls regularization strength.

Gradient Ascent Optimization. At round 1, only the baseline observation $(\mathbf{w}^{(0)}, \bar{\mathbf{s}}^{(0)})$ is available, which is insufficient to fit the performance predictor. We set the cluster gain vector $\boldsymbol{\alpha} = \mathbf{0}$ and rely on pure token-level retrieval (Section 4.3) to select B synthetic tokens. The resulting data composition naturally induces a new mixture $\mathbf{w}^{(1)}$ that reflects the policy's failure modes: clusters with poor model performance receive more synthetic data via the score-weighted priority in (8). This observation pair $(\mathbf{w}^{(1)}, \bar{\mathbf{s}}^{(1)})$, together with the baseline, is used to train the predictor at round 2.

Starting from round 2, we adopt exponentiated gradient (EG) ascent to maximize the predictor performance $\hat{S}(\mathbf{w})$. Each optimization step is initialized from the weight vector $\mathbf{w}^{(t^*)}$ with the best historical evaluation score, where $t^* = \arg\max_{\ell < t} \boldsymbol{\pi}^\top \bar{\mathbf{s}}^{(\ell)}$. This design prevents performance drift caused by predictor inaccuracy. The gradient of $\hat{S}$ with respect to $\theta_j = \log w_j$ is formulated as:

$$g_j = (\boldsymbol{\pi}^\top \mathbf{R})_j \left(\beta_j + \gamma\, \frac{n_{0,j}}{w_j\, N} \right), \tag{5}$$

and the cluster mixture is updated in a multiplicative EG manner:

$$w_k^{(t)} \leftarrow \frac{w_k^{(\text{prev})} \exp(\eta_t\, g_k)}{\sum_j w_j^{(\text{prev})} \exp(\eta_t\, g_j)}, \tag{6}$$

Algorithm 1 AutoScale Data Engine

Require: Real data $\mathcal{D}_{\text{real}}$ with cluster counts $\mathbf{n}_0$; synthetic pool $\mathcal{D}_{\text{syn}}$; budget B; total rounds T; step schedule $\{\epsilon_t\}_{t=1}^{T}$

Ensure: Optimized policy π_{θ^*}

Extract scene embeddings via Graph-RAE; cluster embeddings with GMM to obtain K clusters $\{C_k\}$ and centroids $\{\mathbf{c}_k\}$ ▷ Clustering & Representation

$N \leftarrow N_0 + B$; $\mathbf{w}^{(0)} \leftarrow \mathbf{n}_0 / N_0$; compute $\boldsymbol{\pi}$ and $\mathbf{R}$; $\mathcal{W} \leftarrow \{\mathbf{w}^{(0)}\}$ ▷ Initialization

Train θ_0 on $\mathcal{D}_{\text{real}}$; evaluate $\bar{\mathbf{s}}^{(0)}$ ▷ Round 0: Baseline

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$ ▷ Round 1: Heuristical retrieval

$\mathcal{D}_{\text{add}}^{(1)} \leftarrow \textsc{Retrieve}(\boldsymbol{\alpha}, B, \mathcal{D}_{\text{syn}}, \mathcal{D}_{\text{cal}})$ ▷ Data Retrieval

Train θ_1 on $\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{add}}^{(1)}$; evaluate $\bar{\mathbf{s}}^{(1)}$; record $\mathbf{w}^{(1)}$; $\mathcal{W} \leftarrow \mathcal{W} \cup \{\mathbf{w}^{(1)}\}$

for $t = 2, \ldots, T$ **do** ▷ Round 2+: Cluster-GA Optimization

  Fit $(\boldsymbol{\beta}, \gamma)$ via Eq. (4) on pairs in $\mathcal{W}$ ▷ Cluster Performance Predictor

  $t^* \leftarrow \arg\max_{\ell < t} \boldsymbol{\pi}^\top \bar{\mathbf{s}}^{(\ell)}$; $\mathbf{w}^{(\text{prev})} \leftarrow \mathbf{w}^{(t^*)}$

  Compute gradient $\mathbf{g}$ via Eq. (5); update mixture $\mathbf{w}^{(t)}$ via Eq. (6) ▷ Optimize weight $\mathbf{w}^{(t)}$

  $\boldsymbol{\delta} \leftarrow \max(\lfloor \mathbf{w}^{(t)} N \rceil - \mathbf{n}_0, \mathbf{0})$; compute gain $\boldsymbol{\alpha}$ via Eq. (7)

  $\mathcal{D}_{\text{add}}^{(t)} \leftarrow \textsc{Retrieve}(\boldsymbol{\alpha}, B, \mathcal{D}_{\text{syn}}, \mathcal{D}_{\text{cal}})$ ▷ Data Retrieval

  Train θ_t on $\mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{add}}^{(t)}$; evaluate $\bar{\mathbf{s}}^{(t)}$; $\mathcal{W} \leftarrow \mathcal{W} \cup \{\mathbf{w}^{(t)}\}$

end for

return π_{θ^*}

where the step size η_t is calibrated to ensure $\|\mathbf{w}^{(t)} - \mathbf{w}^{(\text{prev})}\|_2$ follows a half-cosine schedule ϵ_t. If the updated mixture violates global data constraints, a lightweight perturbed search over the feasible set is adopted to obtain a valid candidate. The per-cluster augmentation size is finally determined as $\delta_k = \max(\lfloor w_k^{(t)} N \rceil - n_{0,k}, 0)$.

Meanwhile, the predictor yields a per-cluster performance gain $\boldsymbol{\alpha}$ in score space. It measures the predicted performance improvement from the previous mixture to the updated one, and is normalized into $[0, 1]$ via:

$$\alpha_k = \frac{\max\big(\hat{\bar{s}}_k(\mathbf{w}^{(t)}) - \hat{\bar{s}}_k(\mathbf{w}^{(\text{prev})}),\ 0\big)}{\max_j \big(\hat{\bar{s}}_j(\mathbf{w}^{(t)}) - \hat{\bar{s}}_j(\mathbf{w}^{(\text{prev})})\big)}. \tag{7}$$

This gain $\boldsymbol{\alpha}$ is fed into the downstream data retrieval stage as a weighting factor (Section 4.3).

4.3 Data Retrieval

Given the total budget B and cluster gain $\boldsymbol{\alpha}$ from Cluster-GA, we retrieve concrete synthetic tokens from $\mathcal{D}_{\text{syn}}$ via a matching strategy that combines cluster-level guidance with token-level similarity.

For each cluster k, we identify anchor tokens from calibration scenes in C_k where the current policy underperforms. For each anchor $x_i \in C_k$ and candidate $x_j \in \mathcal{D}_{\text{syn}}$, a retrieval priority is computed:

$$p(x_i, x_j) = \big(1 + \alpha_{c(x_i)}\big) \cdot \big(1 - \bar{s}(x_i)\big) \cdot \cos(\mathbf{v}_{x_i}, \mathbf{v}_{x_j}), \tag{8}$$

where $c(x_i)$ denotes the cluster assignment of anchor x_i, and $\alpha_{c(x_i)} \in [0, 1]$ is the normalized gain from Equation (7). The three factors serve complementary roles: $(1 + \alpha_{c(x_i)})$ provides a cluster-level boost from the predictor; $(1 - \bar{s}(x_i))$ amplifies the demand from low-scoring anchors within each cluster; and $\cos(\mathbf{v}_{x_i}, \mathbf{v}_{x_j})$ ensures representational proximity in the Graph-RAE embedding space.

All anchor–candidate pairs across all clusters are pooled into a single global ranking by p, and the top-B synthetic tokens are selected. The final training set for round t is $\mathcal{D}^{(t)} = \mathcal{D}_{\text{real}} \cup \mathcal{D}_{\text{add}}^{(t)}$.

5 Experiments

In this section, we evaluate the AutoScale data engine on real-synthetic co-training tasks, and compare it with vanilla co-training strategies and cross-domain data mixture methods. We also conduct experiments and discussions to verify the effectiveness of key designs of our method.

Table 1: Comparison of Real-Synthetic Co-Training in the NAVSIM-v2 `navhard` Benchmark.

Model	Settings	Real	Synthetic	Stage	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	HC↑	EC↑	EPDMS↑
TransFuser [7]	Real Data	85K	-	S1	97.3	80.2	97.8	99.3	83.4	96.2	92.9	97.8	71.1	24.4
				S2	79.4	69.0	85.6	98.5	83.8	76.7	47.9	97.0	70.6	
	SimScale-recovery	85K	147K	S1	96.4	78.4	98.9	99.8	80.5	96.2	92.7	97.6	78.2	29.8
				S2	88.9	71.1	91.8	99.0	77.3	85.5	53.8	96.8	47.5	
	SimScale-planner	85K	237K	S1	96.1	85.3	99.4	99.3	84.7	94.7	93.6	97.6	77.3	30.2
				S2	85.5	66.9	91.6	99.1	93.0	81.1	58.3	95.1	42.9	
	AutoScale	85K	50K	S1	97.4	85.1	99.6	100	83.5	96.7	96.4	97.8	76.4	31.5
				S2	84.4	69.3	92.9	98.6	88.3	79.8	58.2	96.4	45.9	
	AutoScale	85K	100K	S1	98.1	87.8	98.9	99.3	83.8	97.1	96.2	97.6	80.9	**33.3**
				S2	88.0	73.4	93.1	99.1	86.4	82.7	56.6	94.9	46.9	
DiffusionDrive [29]	Real Data	85K	-	S1	96.8	86.0	98.8	99.3	84.0	95.8	96.7	97.6	79.6	27.5
				S2	80.1	72.8	84.4	98.4	85.9	76.6	46.4	96.3	72.8	
	SimScale-recovery	85K	147K	S1	97.2	88.4	99.1	99.8	83.9	96.0	96.7	97.6	76.9	30.4
				S2	82.4	67.7	89.1	98.6	89.0	77.6	53.8	95.2	46.8	
	SimScale-planner	85K	237K	S1	97.2	88.7	99.3	99.3	82.8	96.9	98.0	97.3	59.6	32.6
				S2	82.4	72.1	92.9	98.5	92.1	80.6	60.8	95.4	31.9	
	AutoScale	85K	50K	S1	97.2	86.2	99.2	99.3	84.4	96.4	97.3	97.6	80.0	32.4
				S2	87.8	71.2	91.5	98.6	89.7	82.2	61.1	95.6	42.1	
	AutoScale	85K	100K	S1	97.9	89.1	99.2	99.3	84.3	97.1	98.0	97.8	77.8	**33.5**
				S2	83.4	72.7	91.6	98.8	90.0	77.4	61.2	94.6	46.1	

Table 2: Comparison of Real-Synthetic Co-Training in the NAVSIM-v2 `navtest` Benchmark.

Model	Settings	Real	Synthetic	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	HC↑	EC↑	EPDMS↑
TransFuser [7]	Real Data	85K	-	97.7	94.0	99.3	99.8	87.2	96.7	95.5	98.3	82.9	81.5
	SimScale-recovery	85K	147K	97.2	92.8	99.6	99.8	88.8	96.4	96.7	98.3	86.8	83.6
	SimScale-planner	85K	237K	98.3	95.6	99.6	99.8	87.1	97.5	97.2	98.3	88.2	84.4
	AutoScale	85K	50K	98.1	95.8	99.4	99.9	86.7	97.5	97.4	98.3	87.3	84.1
	AutoScale	85K	100K	97.8	96.1	99.7	99.8	88.3	97.0	97.0	98.3	87.6	**84.5**
DiffusionDrive [29]	Real Data	85K	-	98.4	95.5	99.5	99.8	87.5	97.5	96.9	98.4	87.7	84.2
	SimScale-recovery	85K	147K	98.4	96.7	99.6	99.8	87.6	97.5	97.5	98.3	87.1	85.4
	SimScale-planner	85K	237K	98.5	97.1	99.6	99.8	87.4	97.8	98.0	98.3	87.5	**85.9**
	AutoScale	85K	50K	98.5	96.8	99.6	99.8	87.4	97.7	97.6	98.3	87.7	85.6
	AutoScale	85K	100K	98.5	97.0	99.6	99.8	87.5	97.7	97.4	98.3	87.9	**85.9**

5.1 Experimental Setup

Real and Synthetic Data. We use the `navtrain` training set from NAVSIM [8] as real data, comprising approximately 85K driving scenes. We adopt SimScale [41], a reactive simulation framework that reconstructs 3DGS scenarios, generates new trajectories and rerenders sensor data. SimScale fully covers the `navtrain` training set, yielding a 384K synthetic pool. We construct a 20K held-out calibration set via cluster-stratified sampling from the synthetic pool for closed-loop evaluation in simulation, and use the remainder as our synthetic candidate pool.

Implementation Details. In AutoScale, we set synthetic budget $B = 50$K or 100K. For Graph-RAE, we set $T_{\text{len}} = 12$ with $T_h = 8$ and $T_f = 4$. We set regularization weight $\lambda = 5$ and embedding dimension 128, which is reduced to 64 via PCA for GMM clustering. For Cluster-GA, we set cluster number $K = 64$, and run $T = 5$ optimization rounds with half-cosine step schedule ($\epsilon_{\max} = 0.1$). In each round, we conduct closed-loop evaluation on the calibration set, and average results from the last 10 checkpoints to mitigate training randomness. RBF bandwidth is set to $\sigma = 0.5$, and ridge regularization coefficient to $\lambda_{\text{reg}} = 1.0$.

Details of Baselines. To illustrate the effectiveness of our AutoScale data engine, we compare it with vanilla co-training methods using larger synthetic data volumes and mainstream cross-domain data mixture methods on the real-synthetic co-training task. For a fair comparison and to ensure performance differences are solely attributable to different data mixture strategies, all experiments adopt the same pre-generated synthetic data pool and follow unified model training protocols.

We compare with the following data mixture strategies: (1) SimScale generates 147K recovery-based and 237K planner-based synthetic scenes, and directly co-trains them with 85K real driving data separately. (2) Uniform: it selects data uniformly from the synthetic pool under a given budget. (3) Chameleon [49]: a data mixture framework for LLM pretraining and finetuning that computes domain weights via kernel ridge leverage scores in the latent space. (4) IWR [47]: a retrieval method for robotic imitation learning using latent-space importance weights and feature distances. We employ a shared Graph-RAE latent space for all cross-domain data mixture methods in autonomous driving.

Table 3: Data Mixture Method Comparison on the NAVSIM-v2 `navhard` Benchmark.

Method	Stage	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	HC↑	EC↑	EPDMS↑
Uniform	S1	97.6	86.7	99.1	99.6	84.3	96.0	97.3	97.6	78.7	29.5
	S2	83.2	70.3	91.3	98.3	89.9	77.3	57.7	95.4	44.6	
Chameleon [49]	S1	98.1	87.3	98.9	99.3	84.0	97.3	98.0	97.8	80.4	30.9
	S2	82.9	71.0	90.5	98.9	90.3	76.6	60.3	95.0	42.5	
IWR [47]	S1	98.2	86.9	99.6	99.8	83.6	95.6	96.7	97.6	77.8	31.0
	S2	84.3	73.3	91.0	98.6	88.6	79.1	57.1	96.2	44.6	
AutoScale	S1	97.2	86.2	99.2	99.3	84.4	96.4	97.3	97.6	80.0	**32.4**
	S2	87.8	71.2	91.5	98.6	89.7	82.2	61.1	95.6	42.1	

All experiments are conducted on representative end-to-end planning models: the regression-based Transfuser (LTF) [7] and the diffusion-based DiffusionDrive [29]. We adopt their official implementations with two minor modifications: the input image resolution is unified to 2048×512, and LiDAR inputs are removed. All models are trained from scratch on NVIDIA A800 GPUs.

Benchmark and Metrics. We adopt two official benchmarks from NAVSIMv2 [4] for evaluation: `navtest` and `navhard`. `navtest` is a one-stage evaluation benchmark containing 12,146 diverse real-world driving scenarios. `navhard` is an officially additionally collected two-stage challenging evaluation benchmark. It consists of 244 real-world scenarios in the first stage and 4,164 corresponding synthetic scenarios generated via 3DGS in the second stage. The main evaluation metric is the Extended Predictive Driver Model Score (EPDMS).

5.2 Main Results

Real-Synthetic Co-Training Results. We compare our method with the vanilla real-synthetic co-training strategy. As shown in table 1, for LTF, our method achieves better EPDMS scores than SimScale-recovery and SimScale-planner, using only 34% and 21% additional synthetic data, respectively. For DiffusionDrive, our method similarly outperforms the vanilla co-training strategy with a larger training budget by +3.1 and +0.9 in EPDMS. In table 2, our AutoScale achieves comparable results to the vanilla real-synthetic co-training strategy on both LTF and DiffusionDrive with far less additional data. Evaluations on two distinct models across `navhard` and `navtest` demonstrate that AutoScale not only effectively boosts model performance in challenging scenarios but also improves generalization ability, achieving higher data efficiency via our data mixture strategy.

Data Mixture Method Comparison. We further compare our AutoScale with several cross-domain data mixture strategies on the DiffusionDrive model, under a fixed 50K synthetic data budget and identical training protocols. As shown in table 3, when equipped with our Graph-RAE embedding for driving scene representation, both Chameleon and IWR outperform the uniform baseline under the same synthetic data budget. This further verifies the effectiveness and generalization capability of our proposed Graph-RAE. Our AutoScale surpasses all baseline data mixture approaches, demonstrating the superiority of our targeted, evaluation-driven closed-loop data mixture scheme.

5.3 Ablation Studies and Discussions

Ablation Study on Graph-RAE. We ablate the regularizations for contrastive and metric learning in Graph-RAE separately and run the complete downstream closed-loop pipeline. Experimental results under the shared DiffusionDrive training protocols and a 50K budget in table 4 show that, compared with the full method, the model exhibits EPDMS performance drops of 0.2 and 0.3, respectively. This demonstrates that representation quality affects clustering and retrieval performance.

Effectiveness of Cluster-GA. We ablate the Cluster-GA module and perform pure retrieval with Graph-RAE representations. Experimental results in table 4 show the efficacy of the Cluster-GA module. We also investigate the selection of cluster numbers while keeping all other experimental settings unchanged. Setting the number of clusters to $K = 64$ achieves the best model performance. We also provide qualitative results. In fig. 3 (a), despite complex driving scenes and a large number of clusters, t-SNE visualizations present a compact and well-separated structure. In fig. 3 (b), low-performance clusters of the original model receive more synthetic data, and most of them achieve obvious performance gains after data mixture.

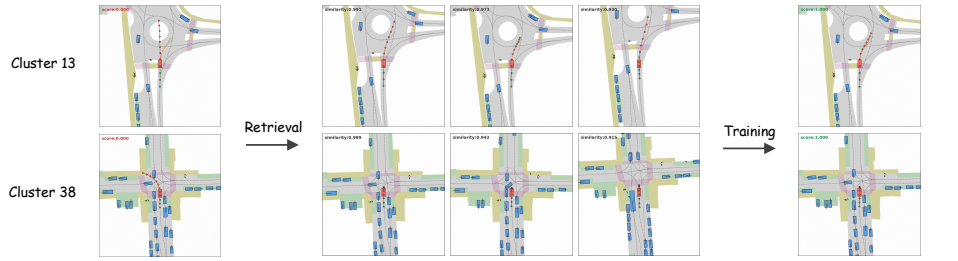
Table 4: Ablation Study Results on the NAVSIM-v2 `navtest` Benchmark.

Ablation	Setting	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	HC↑	EC↑	EPDMS↑
Abl.module	w/o Cluster-GA	98.3	96.6	99.6	99.8	87.5	97.7	97.5	98.3	87.2	85.2
	w/o Retrieval	98.5	96.6	99.6	99.8	87.5	97.6	97.5	98.3	87.4	85.4
Abl.vector	w/o Contrastive Reg.	98.3	96.7	99.6	99.8	87.5	97.6	97.7	98.3	87.6	85.4
	w/o Metric Reg.	98.3	96.6	99.6	99.8	87.6	97.5	97.6	98.3	87.6	85.3
Abl.cluster	32	98.5	96.7	99.7	99.8	87.5	97.6	97.4	98.3	87.5	85.4
	128	98.3	96.7	99.6	99.8	87.5	97.6	97.7	98.4	87.6	85.4
AutoScale	Full	98.5	96.8	99.6	99.8	87.4	97.7	97.6	98.3	87.7	**85.6**



(a) Cluster t-SNE (b) Scene Cluster Reweighting and Performance Gains

Figure 3: Qualitative results of clustering. (a) t-SNE of clusters on the training set. (b) Visualization of the scene cluster reweighting process and corresponding performance gains. From top to bottom: cluster distribution of the training set, EPDMS scores of calibration set scenes assigned to clusters of the original model, synthetic-to-real data ratio, and per-cluster performance gains after data mixture.



(a) Low-Performance Scenes (b) Top-Ranked Retrieved Scenes (c) Targeted Optimization

Figure 4: Qualitative results of retrieval. The green dotted curve, red dotted curve, and dark yellow dashed line represent historical and future trajectories, and ground truth, respectively.

Effectiveness of Retrieval. We ablate the retrieval module and perform vector-based clustering and reweighting. In each round, we assign synthetic data to clusters and uniformly sample data per cluster. Experimental results in table 4 show the efficacy of the retrieval module. We also provide qualitative results. In fig. 4, we select representative low-performance scenes from the original model corresponding to fig. 3 (b), where the ego vehicle performs poorly when navigating roundabouts and makes improper left turns at intersections, yielding an EPDMS score of 0 (shown in the top-left corner). We effectively retrieve the most similar synthetic data. After data mixture and retraining, the performance of these two scenes is significantly improved.

6 Conclusion and Limitation

In this paper, we pioneer a systematic investigation into the data mixture effect on model performance in real-synthetic co-training. We argue that training data mixture requires explicit guidance on scene types and quantities, and formulate it as an iterative optimization process to improve model evaluation performance. We propose AutoScale, a closed-loop dynamic data engine unifying scene representation, data mixture optimization and retrieval, and model training and evaluation. Experiments on NavSim show that, guided by evaluation in simulation, AutoScale achieves better real-world performance with fewer synthetic samples. To keep our investigation focused, we only conduct experiments on two representative models with limited iterative rounds. Future work includes extending the data mixture effect to more model architectures (e.g., world action models) and training objectives (e.g., reinforcement learning), and accumulating more training-evaluation rounds naturally in industrial development workflows to precisely estimate its impact on model performance.

Acknowledgment

The work is supported in part by the Early Career Scheme of the Research Grants Council (RGC) of the Hong Kong SAR under grant No. 26202321, ITF PRP/046/24FX, the Science & Technology Cooperation Program of Shandong under grant No. SDST26EG01, SAIL Research Project, and HKUST-Zeekr Collaborative Research Fund.

References

[1] Mustafa Baniodeh, Kratarth Goel, Scott Ettinger, Carlos Fuertes, Ari Seff, Tim Shen, Cole Gulino, Chenjie Yang, Ghassen Jerfel, Dokook Choe, et al. Scaling laws of motion forecasting and planning–technical report. *arXiv preprint arXiv:2506.08228*, 2025.

[2] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.

[5] 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 (TPAMI)*, 2024.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International conference on machine learning (ICML)*, 2020.

[7] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2023.

[8] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Navsim: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[9] Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, et al. Nemotron-climb: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[10] Wenhao Ding, Yulong Cao, Ding Zhao, Chaowei Xiao, and Marco Pavone. Realgen: Retrieval augmented generation for controllable traffic scenarios. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[12] Simin Fan, Matteo Pagliardini, and Martin Jaggi. Doge: Domain reweighting with generalization estimation. In *Proceedings of the International conference on machine learning (ICML)*, 2024.

[13] Ruiyuan Gao, Kai Chen, Bo Xiao, Lanqing Hong, Zhenguo Li, and Qiang Xu. Magicdrive-v2: High-resolution long video generation for autonomous driving with adaptive control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[14] Ruiyuan Gao, Kai Chen, Enze Xie, Lanqing Hong, Zhenguo Li, Dit-Yan Yeung, and Qiang Xu. MagicDrive: Street view generation with diverse 3d geometry control. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

[15] Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. Vista: A generalizable driving world model with high fidelity and versatile controllability. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[16] Guillermo Garcia-Cobo, Maximilian Igl, Peter Karkus, Zhejun Zhang, Michael Watson, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Road: Rollouts as demonstrations for closed-loop supervised fine-tuning of autonomous driving policies. *arXiv preprint arXiv:2512.01993*, 2025.

[17] Junhao Ge, Zuhong Liu, Longteng Fan, Yifan Jiang, Jiaqi Su, Yiming Li, Zhejun Zhang, and Siheng Chen. Unraveling the effects of synthetic data on end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2016.

[19] Anthony Hu, Lloyd Russell, Hudson Yeo, Zak Murez, George Fedoseev, Alex Kendall, Jamie Shotton, and Gianluca Corrado. Gaia-1: A generative world model for autonomous driving. *arXiv preprint arXiv:2309.17080*, 2023.

[20] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[21] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 1901.

[22] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[23] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[24] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 2023.

[25] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[26] Diederik P Kingma and Jimmy Ba. Adam: a method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[27] Tianyu Li, Yihang Qiu, Zhenhua Wu, Carl Lindström, Peng Su, Matthias Nießner, and Hongyang Li. Mtgs: Multi-traversal gaussian splatting. *arXiv preprint arXiv:2503.12552*, 2025.

[28] Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, et al. Recogdrive: A reinforced cognitive framework for end-to-end autonomous driving. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[29] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, et al. Diffusiondrive: Truncated diffusion model for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[30] Haohong Lin, Yunzhi Zhang, Wenhao Ding, Jiajun Wu, and Ding Zhao. Model-based policy adaptation for closed-loop end-to-end autonomous driving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[31] Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

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

[34] Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, Scott Reed, Ken Goldberg, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[35] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[36] NVIDIA, Johan Bjorck, Nikita Cherniadev Fernando Castañeda, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Proceedings of the International conference on machine learning (ICML)*, 2021.

[38] Xuanchi Ren, Yifan Lu, Tianshi Cao, Ruiyuan Gao, Shengyu Huang, Amirmojtaba Sabour, Tianchang Shen, Tobias Pfaff, Jay Zhangjie Wu, Runjian Chen, et al. Cosmos-drive-dreams: Scalable synthetic driving data generation with world foundation models. *arXiv preprint arXiv:2506.09042*, 2025.

[39] Wenchao Sun, Xuewu Lin, Yining Shi, Chuang Zhang, Haoran Wu, and Sifa Zheng. Sparsedrive: End-to-end autonomous driving via sparse scene representation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2025.

[40] GigaWorld Team, Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Haoyun Li, Jiagang Zhu, Kerui Li, Mengyuan Xu, et al. Gigaworld-0: World models as data engine to empower embodied ai. *arXiv preprint arXiv:2511.19861*, 2025.

[41] Haochen Tian, Tianyu Li, Haochen Liu, Jiazhi Yang, Yihang Qiu, Guang Li, Junli Wang, Yinfeng Gao, Zhang Zhang, Liang Wang, et al. Simscale: Learning to drive via real-world simulation at scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2026.

[42] Yang Tian, Yuyin Yang, Yiman Xie, Zetao Cai, Xu Shi, Ning Gao, Hangxu Liu, Xuekun Jiang, Zherui Qiu, Feng Yuan, et al. Interndata-a1: Pioneering high-fidelity synthetic data for pre-training generalist policy. *arXiv preprint arXiv:2511.16651*, 2025.

[43] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, 2017.

[45] Xiaofeng Wang, Zheng Zhu, Guan Huang, Xinze Chen, Jiagang Zhu, and Jiwen Lu. Drivedreamer: Towards real-world-drive world models for autonomous driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[46] Yuqing Wen, Yucheng Zhao, Yingfei Liu, Fan Jia, Yanhui Wang, Chong Luo, Chi Zhang, Tiancai Wang, Xiaoyan Sun, and Xiangyu Zhang. Panacea: Panoramic and controllable video generation for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2024.

[47] Amber Xie, Rahul Chand, Dorsa Sadigh, and Joey Hejna. Data retrieval with importance weights for few-shot imitation learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2025.

[48] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[49] Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning. In *Proceedings of the International conference on machine learning (ICML)*, 2025.

[50] Zebin Xing, Xingyu Zhang, Yang Hu, Bo Jiang, Tong He, Qian Zhang, Xiaoxiao Long, and Wei Yin. Goalflow: Goal-driven flow matching for multimodal trajectories generation in end-to-end autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2025.

[51] Jiasheng Ye, Peiju Liu, Tianxiang Sun, Jun Zhan, Yunhua Zhou, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[52] Yinan Zheng, Ruiming Liang, Kexin Zheng, Jinliang Zheng, Liyuan Mao, Jianxiong Li, Weihao Gu, Rui Ai, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Diffusion-based planning for autonomous driving with flexible guidance. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[53] Yupeng Zheng, Pengxuan Yang, Zhongpu Xia, Qichao Zhang, Yuhang Zheng, Songen Gu, Bu Jin, Teng Zhang, Ben Lu, Chao Han, et al. Data scaling laws for imitation learning-based end-to-end autonomous driving. *arXiv preprint arXiv:2412.02689*, 2024.

Technical appendices and supplementary material

This supplementary material presents implementation and evaluation details of our method and baselines in section A. We further provide experimental results and analyses in section B. Supplementary qualitative visualization results are presented in section C.

A Additional Implementation and Evaluation Details

A.1 Additional Implementation Details of Graph-RAE

Graph Definition. Each driving scenario is constructed as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote node set and edge set, respectively. All graph nodes are divided into two categories: dynamic nodes $\mathcal{V}_{\text{dyn}}$ including the ego-vehicle and surrounding traffic participants, and static map nodes $\mathcal{V}_{\text{map}}$ including pedestrian crossings, road dividers and boundary elements. Each node v_i contains a T_{len}-step sequential polyline feature $\mathbf{V}_i$. The per-time-step node feature is defined as:

$$\mathbf{v}_{i,t} = \begin{cases} [x, y, \theta, \kappa, \nu]^\top \in \mathbb{R}^5, & v_i \in \mathcal{V}_{\text{dyn}}, \\ [x, y, \theta, \kappa]^\top \in \mathbb{R}^4, & v_i \in \mathcal{V}_{\text{map}}, \end{cases} \tag{9}$$

where x, y denote 2D position, θ denotes heading angle, κ denotes trajectory curvature, and ν denotes velocity only for dynamic nodes. $\mathbf{E}_{ji}$ represents the T_{len}-step relative spatio-temporal feature sequence of each directed edge. $\mathbf{emb}_{\text{pos}}$ denotes temporal positional embedding, and $\mathbf{emb}_{\text{sem}}$ denotes unified semantic embedding for node and edge category attributes.

Feature Encoding. All raw node and edge sequential features are projected by dedicated MLPs, augmented with only positional and semantic embeddings to obtain unified hidden representations:

$$\mathbf{h}_i = \text{MLP}(\mathbf{V}_i) + \mathbf{emb}_{\text{sem}} + \mathbf{emb}_{\text{pos}}, \quad \mathbf{e}_{ji} = \text{MLP}(\mathbf{E}_{ji}) + \mathbf{emb}_{\text{pos}}. \tag{10}$$

Graph Transformer Propagation. We stack L layers of graph transformer with gated multi-head attention to aggregate spatio-temporal interaction information from neighboring nodes and connected edges. The simplified layer-wise node representation update rule is formulated as:

$$\mathbf{h}_i^{(l+1)} = \mathbf{h}_i^{(l)} + \sigma\big(\mathbf{g}_i^{(l)}\big) \odot \text{MHA}\big(\mathbf{h}_i^{(l)}, \{\mathbf{h}_j^{(l)}, \mathbf{e}_{ji}\}\big). \tag{11}$$

where $\mathbf{g}_i^{(l)}$ is the learnable gate vector at layer l, and $\sigma(\cdot)$ denotes the sigmoid activation function. After multi-layer feature refinement, the ego-vehicle node embedding is ℓ_2-normalized to generate the final global scene representation:

$$\mathbf{z}_{\text{scene}} = \mathbf{h}_{\text{ego}}^{(L)} / \|\mathbf{h}_{\text{ego}}^{(L)}\|_2. \tag{12}$$

Reconstruction Loss. The graph decoder reconstructs the original polyline sequences for all nodes, and the autoencoder training is optimized with standard Huber loss:

$$\mathcal{L}_{\text{recon}} = \sum_{v_i} \sum_{t=1}^{T_{\text{len}}} \text{Huber}(\hat{\mathbf{v}}_{i,t}, \mathbf{v}_{i,t}). \tag{13}$$

Contrastive Regularization. Two contrastive learning strategies are adopted for embedding space regularization, including SimCLR [6] for geometric invariance and SupCon [25] for semantic clustering. The total contrastive objective is $\mathcal{L}_{\text{contra}} = \mathcal{L}_{\text{simclr}} + \mathcal{L}_{\text{supcon}}$:

$$\mathcal{L}_{\text{simclr}} = -\frac{1}{N} \sum \log \frac{\exp(\text{sim}(\mathbf{z}^a, \mathbf{z}^b)/\tau)}{\exp(\text{sim}(\mathbf{z}^a, \mathbf{z}^b)/\tau) + \sum_{m \neq k} \exp(\text{sim}(\mathbf{z}^a, \mathbf{z}^m)/\tau)}, \tag{14}$$

$$\mathcal{L}_{\text{supcon}} = -\frac{1}{N} \sum \frac{1}{|P_k|} \sum \log \frac{\exp(\text{sim}(\mathbf{z}, \mathbf{z}_p)/\tau)}{\sum_{m \neq k} \exp(\text{sim}(\mathbf{z}, \mathbf{z}_m)/\tau)}. \tag{15}$$

where N is the batch size, τ denotes the temperature coefficient, a, b denote two geometrically augmented views of the same scene, k, m index individual scenes, P_k denotes the positive sample set sharing consistent semantic labels with scene k, and p enumerates positive samples within P_k.

Metric Learning Regularization. A pairwise metric regression objective is further added to align scene embeddings with actual driving task performance:

$$\mathcal{L}_{\text{metric}} = \frac{1}{N^2} \sum_{i,j} \mathbb{I}(i,j) \cdot \left\| |\mathbf{m}_i \hat{-} \mathbf{m}_j| - |\mathbf{m}_i - \mathbf{m}_j| \right\|_2^2. \tag{16}$$

where $\mathbb{I}(i,j)$ is the valid pair indicator function, $\mathbf{m}_i$ denotes the ground-truth evaluation metric vector, and $\hat{\cdot}$ represents the predicted counterpart.

Overall Training Loss. The final joint training loss balances reconstruction fitting and two regularization tasks with a weight coefficient λ:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda\big(\mathcal{L}_{\text{contra}} + \mathcal{L}_{\text{metric}}\big). \tag{17}$$

A.2 Implementation Details of Baselines

IWR [47]. We use the same implementation details as the original paper. The core idea of IWR is to retain synthetic samples whose embedding distribution most closel resembles that of the real training data. Given real embeddings $\{\mathbf{v}_x\}_{x\in\mathcal{D}_{\text{real}}}$ and synthetic embeddings $\{\mathbf{v}_x\}_{x\in\mathcal{D}_{\text{syn}}}$, we first perform a joint PCA on their concatenation and project both sets to n=16 dimensions, alleviating the curse of dimensionality for subsequent density estimation. A Gaussian KDE is then fitted separately on the projected real and synthetic embeddings, using Scott's bandwidth $h = N_0^{-1/(n+4)}$. For every synthetic sample $x \in \mathcal{D}_{\text{syn}}$, the importance weight $w_x = \exp\big(\log \hat{p}_{\text{real}}(\mathbf{v}_x) - \log \hat{p}_{\text{syn}}(\mathbf{v}_x)\big)$ measures how much more likely the sample is under the real distribution than under the synthetic one; log-weights are clipped to $[-100, 100]$ for numerical stability. The final subset $\mathcal{D}^*$ is formed by selecting the B synthetic samples with the highest importance weights, effectively prioritising synthetic scenes that fill gaps in the real data distribution.

Chameleon [49]. We use the same implementation details as the original paper. Chameleon partitions the synthetic pool into K clusters defined by cluster centroids, and allocates the budget B unevenly across clusters based on how well each cluster is already covered by the others. Concretely, we form the cluster similarity matrix $\mathbf{R} = \tilde{\mathbf{C}}\tilde{\mathbf{C}}^\top \in \mathbb{R}^{K\times K}$, where $\tilde{\mathbf{C}}$ is the ℓ_2-normalized centroid matrix and $\mathbf{R}_{kj}$ reflects the cosine similarity between cluster k and cluster j. KRLS leverage scores $\gamma_k = \big[\mathbf{R}(\mathbf{R} + K\lambda\mathbf{I})^{-1}\big]_{kk}$ are then derived from this matrix: a low score signals that cluster k is underrepresented and cannot be well explained by the remaining clusters. Cluster weights $\bar{w}_k \propto \exp(1/\gamma_k)$ therefore assign a larger synthetic allocation to underrepresented clusters, and the per-cluster budget is set to $c_k = \text{round}(\bar{w}_k B)$ with a greedy integer-correction step that enforces $\sum_k c_k = B$ exactly. Finally, each synthetic sample $x \in \mathcal{D}_{\text{syn}}$ is assigned to its nearest centroid $\mathbf{c}_k$, c_k samples are drawn uniformly at random from each cluster pool $\mathcal{P}_k$, and any shortfall caused by insufficient pool size is filled by uniform sampling from $\mathcal{D}_{\text{syn}} \setminus \mathcal{D}^*$.

A.3 Model and Training

LTF [7]. A regression-based planner that directly regresses future waypoints from fused sensory latents. We adopt the same implementation details as those in SimScale [41], it employs a pre-trained ResNet34 [18] as the image encoder for 3 camera inputs (front, front-left, front-right). The input images are concatenated to a total resolution of 2048×512 for feature extraction. The model is trained for 100 epochs on 8 GPUs with a total batch size of 512, using the Adam [26] optimizer with a learning rate of 2.83×10^{-4}.

DiffusionDrive [29]. A multi-modal generative planner that iteratively denoises diverse trajectories via anchor-conditioned truncated diffusion. Each anchor adaptively queries the encoded image features. We follow the identical experimental configurations outlined in SimScale [41], it employs a pre-trained ResNet34 [18] as the image encoder for 3 camera inputs (front, front-left, front-right). The input images are concatenated to a total resolution of 2048×512 for feature extraction, and truncated diffusion with 20 clustered anchors is applied. The model is trained for 100 epochs on 8 NVIDIA A800 GPUs with a total batch size of 512, using the AdamW [32] optimizer with a learning rate of 6×10^{-4}.

A.4 Additional Evaluation Metric Details

We use both the Predictive Driver Model Score (PDMS) and the Extended Predictive Driver Model Score (EPDMS) for evaluation.

PDMS includes No At-Fault Collision (NC), Drivable Area Compliance (DAC), Time-To-Collision (TTC), Comfort (Comf.), and Ego Progress (EP). PDMS is defined as:

$$\text{PDMS} = \text{NC} \times \text{DAC} \times \frac{5 \times \text{EP} + 5 \times \text{TTC} + 2 \times \text{Comf.}}{12} \tag{18}$$

EPDMS includes No At-Fault Collision (NC), Drivable Area Compliance (DAC), Driving Direction Compliance (DDC), Traffic Light Compliance (TLC), Ego Progress (EP), Time to Collision (TTC), Lane Keeping (LK), History Comfort (HC), and Extended Comfort (EC). The four multiplicative terms (NC, DAC, DDC, TLC) act as strict safety constraints, while the weighted sum in the numerator comprehensively evaluates overall driving performance. EPDMS is defined as:

$$\text{EPDMS} = \text{NC} \times \text{DAC} \times \text{DDC} \times \text{TLC} \times \frac{5 \cdot \text{EP} + 5 \cdot \text{TTC} + 2 \cdot \text{LK} + 2 \cdot \text{HC} + 2 \cdot \text{EC}}{16}. \tag{19}$$

B Additional Experimental Results and Discussion

B.1 Additional Results for Real-Synthetic Co-Training

Table 5: Comparison of Real-Synthetic Co-Training on the NAVSIM-v1 `navtest` Benchmark.

Model	Settings	Real	Synthetic	NC↑	DAC↑	EP↑	TTC↑	Comf.↑	PDMS↑
TransFuser [7]	Real Data	85K	-	97.5	93.7	80.0	92.6	100	84.8
	SimScale-recovery	85K	147K	98.2	95.0	80.5	94.3	100	86.4
	SimScale-planner	85K	237K	98.3	95.6	91.3	94.6	100	87.3
	AutoScale	85K	50K	98.0	95.8	82.0	93.6	100	87.2
	AutoScale	85K	100K	98.0	96.1	82.2	93.6	100	**87.5**
DiffusionDrive [29]	Real Data	85K	-	98.2	95.2	81.5	94.4	100	87.1
	SimScale-recovery	85K	147K	98.4	96.5	82.8	94.7	100	88.4
	SimScale-planner	85K	237K	98.5	97.1	83.1	94.7	100	88.9
	AutoScale	85K	50K	98.5	96.7	82.9	94.8	100	88.7
	AutoScale	85K	100K	98.5	96.9	83.1	95.0	100	**89.0**

We provide additional comparisons of our method with the vanilla real-synthetic co-training strategy NavSim-v1 `navtest` [8]. In table 5, similar to the main text, our AutoScale achieves comparable results to the vanilla real-synthetic co-training strategy on both LTF and DiffusionDrive with far less additional data, demonstrating that AutoScale effectively boosts model performance and improves generalization ability, achieving higher data efficiency via our data mixture strategy.

B.2 Additional Results for Data Mixture Methods Comparison

Table 6: Data Mixture Method Comparison in the NAVSIM-v2 `navtest` Benchmark.

Method	NC↑	DAC↑	DDC↑	TLC↑	EP↑	TTC↑	LK↑	HC↑	EC↑	EPDMS↑
Uniform	98.4	96.2	99.5	99.8	87.6	97.6	97.5	98.3	87.5	84.9
Chameleon [49]	98.3	96.5	99.5	99.8	87.5	97.5	97.5	98.3	87.4	85.2
IWR [47]	98.5	96.5	99.6	99.8	87.5	97.6	97.5	98.3	87.6	85.3
AutoScale	98.5	96.8	99.6	99.8	87.4	97.7	97.6	98.3	87.7	**85.6**

We provide additional comparisons of our AutoScale against several cross-domain data mixture strategies on the DiffusionDrive model. Evaluations are conducted on the NavSim-v2 `navtest` split [8], with a fixed 50K synthetic data budget and identical training protocols. As shown in table 6, consistent with the main text results, the experiments validate the effectiveness of our proposed Graph-RAE for driving scene representation, as well as the superiority of our data mixture scheme.

B.3 Model-Specific Data Mixture

We investigate whether the optimal data mixture in real-synthetic co-training is transferable across model architectures by quantifying mixture divergence and conducting cross-model experiments.

Mixture Divergence. Given two token sets $\mathcal{A}$ and $\mathcal{B}$ selected for different models under an identical budget from the same synthetic pool, we measure their divergence via the Jaccard similarity [21]:

$$J(\mathcal{A}, \mathcal{B}) = \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}. \tag{20}$$

Under a budget of 100K, DiffusionDrive and Transfuser share approximately 57K tokens out of a combined 143K unique selections in the total 364K synthetic data pool, yielding $J = 0.399$. This indicates that over half of each model's preferred synthetic data is unique to its own architecture.

Cross-Model Training. To verify the data mixture divergence reflects performance impact, we conduct a swap experiment: each model is trained with the data mixture optimized for the other, with a budget of 100K and all hyperparameters kept identical.

Table 7: Cross-Model Training Experiment on the NAVSIM-v2 `navtest` Benchmark.

Model	Data Mixture	EPDMS↑
Transfuser	Cross-Optimized	84.4
	Self-Optimized	**84.5**
DiffusionDrive	Cross-Optimized	85.6
	Self-Optimized	**85.9**

As shown in Table 7, each model is trained with either its own optimized data mixture (self) or that of the counterpart (cross). All models yield the best performance under self-optimized mixtures, while cross-model settings lead to performance degradation. This verifies that optimal data mixtures are model-dependent, highlighting the necessity of tailored strategies for different model architectures.

B.4 Effectiveness of Cross-Cluster Propagation

We model the effect of data mixture changes on per-cluster performance using a relative fitting formulation in (3). The matrix $\mathbf{R}$ encodes inter-cluster similarity via an RBF kernel:

$$\mathbf{R}_{kj} = \exp\left(-\frac{d_{\cos}(k,j)^2}{2\sigma^2}\right), \tag{21}$$

where $d_{\cos}(k,j) = 1 - \cos(\mathbf{c}_k, \mathbf{c}_j)$ is the cosine distance between cluster centroids.

To quantify whether cross-cluster propagation provides additional modeling power beyond independent per-cluster effects, we compare the coefficient of determination under two settings:

$$R^2 = 1 - \frac{\sum_i (\Delta s_i - \Delta \hat{s}_i)^2}{\sum_i (\Delta s_i - \overline{\Delta s})^2}, \tag{22}$$

evaluated with the full RBF kernel $\mathbf{R}$ versus the identity matrix $\mathbf{I}$ (no propagation, each cluster modeled independently).

Table 8: Cross-Cluster Propagation Effectiveness on NAVSIM-v2 `navtest`. $R^2(\mathbf{R})$: full RBF kernel; $R^2(\mathbf{I})$: no propagation (identity matrix); ΔR^2: performance improvement brought by propagation.

Model	$R^2(\mathbf{R})$	$R^2(\mathbf{I})$	ΔR^2
DiffusionDrive	0.415	0.409	+0.006
Transfuser	0.134	0.068	+0.066

As shown in table 8, under a 100K budget, cross-cluster propagation improves both models but with distinct effects. It brings a marginal gain ($\Delta R^2 = +0.006$) for DiffusionDrive, while yielding a substantial boost ($\Delta R^2 = +0.066$) for Transfuser.

C Additional Qualitative Results

Corresponding to the clustering and related performance statistics in fig. 3 (b), we further provide representative clustering results of common and long-tail scenes in the `navtrain` training split of NavSim in fig. 5, as well as additional visualizations of underperforming scenes, scene retrieval, and performance improvement in fig. 6.

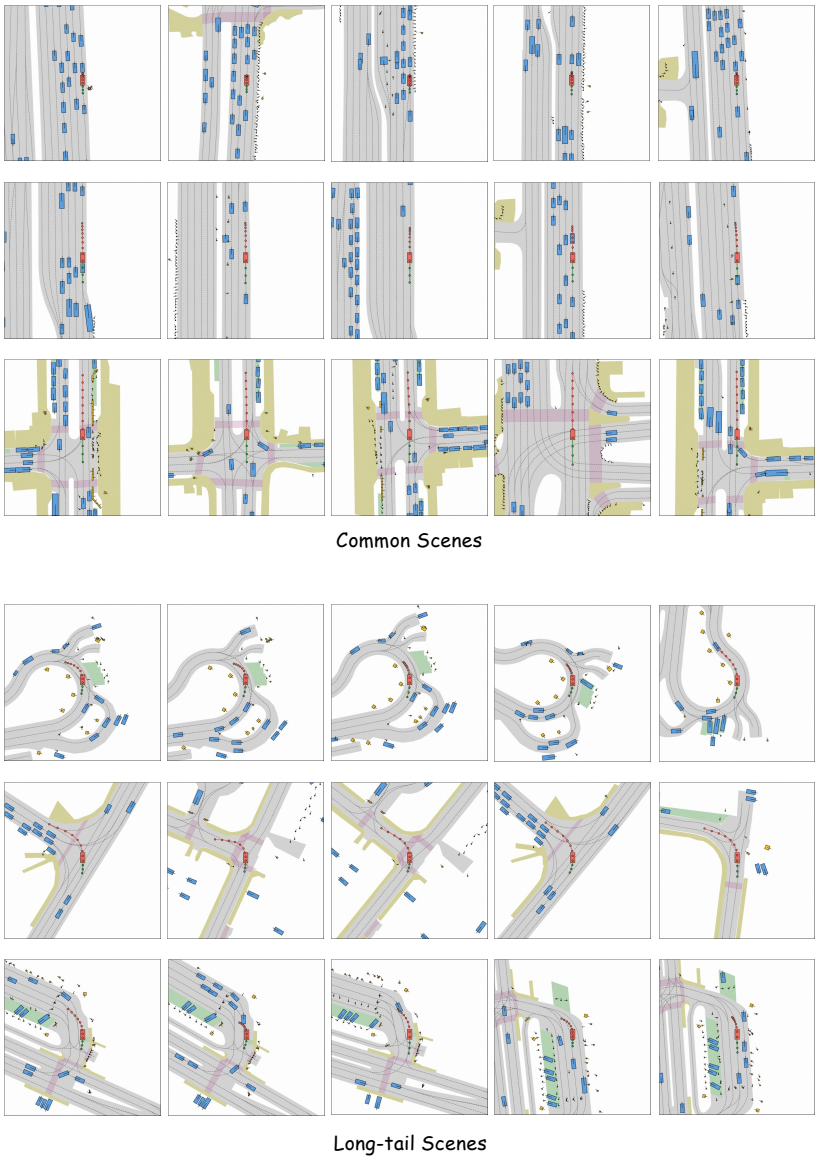


Figure 5: Clustering Visualization of Common and Long-Tail Driving Scenes in the `navtrain` Training Split of the NavSim Dataset.

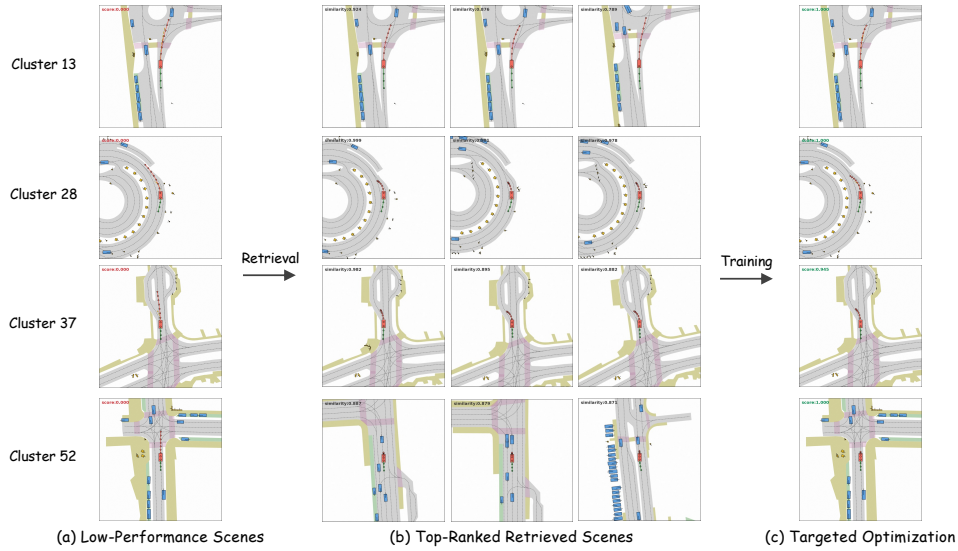


Figure 6: Additional Visualization of Underperforming Scenes, Scene Retrieval, and Performance Improvement.