Appendix

A. Model Implementation

A.1. Architecture Overview

The cornerstone of our framework, Retrieval-Augmented Generation for Trajectory Prediction (RAG-TP), is a two-stage learning process designed to decouple knowledge encoding from utilization. The first stage is self-supervised pre-training, which focuses on training a set of feature encoders to distill structured, high-dimensional representations from vast, heterogeneous driving data. These representations are then used to build a knowledge base. The second stage is supervised fine-tuning, where a complete prediction model is optimized end-to-end, leveraging the pre-trained knowledge base to enhance final trajectory prediction accuracy. The hidden dimension for all features in the model is uniformly set to 128.

A.2. Input Feature Representation

To ensure the model comprehensively understands the scene, all input data is standardized, feature-engineered, and transformed into an agent-centric vehicle coordinate system.

- **Historical Trajectory Features (X_{hist}):** For each timestep within the observation horizon, an agent's state is encoded as a 4-dimensional vector $(\Delta x, \Delta y, \Delta v_x, \Delta v_y)$, representing its relative displacement and velocity changes. This vector is computed relative to the previous timestep to capture fine-grained dynamics.
- **Scene Map Features (M):** Vectorized map information is processed into a series of lane segments. Each lane segment is represented by sampling 20 equidistant points. In the agent-centric coordinate system, each sampled point is characterized by a 2-dimensional vector (x, y). Consequently, a scene with L lane lines is represented as a tensor of dimension $(L, 20, 2)$. This representation preserves the core topological structure while maintaining input conciseness.

A.3. Module Design in the Self-Supervised Pre-training Stage

During the pre-training stage, our primary objective is to learn features that can clearly distinguish scene dynamics, static topology, and future trends. To this end, we have designed an autoencoding framework with three independent branches, corresponding to historical trajectories, map information, and future trajectories.

History Encoder (E_{hist}). This module processes the agent's historical motion sequence within the standard observation horizon defined by the dataset. We employ a 4-layer standard Transformer encoder to capture temporal dependencies within the trajectory, ultimately outputting an embedding vector e_h that represents the dynamic characteristics of the current scene. The Transformer encoder contains 8 attention heads and a feed-forward network with a hidden dimension of 128. We add learnable positional encodings to the input sequence and apply layer normalization and residual connections after each sub-layer.

Scene Encoder (E_{scene}). This module encodes the vectorized map information. Each lane segment is preprocessed into an unordered set of 20 sampled points. To effectively process such data, we construct a specialized feature extraction network. This network uses a shared-parameter Multi-Layer Perceptron (MLP) to independently extract features from each point. The MLP consists of 3 linear layers with ReLU activation functions. Subsequently, a global max-pooling operation aggregates all point-level features into a high-dimensional vector e_m that represents the entire segment's topology.

Future Encoder (E_{future}). This module's architecture is symmetric to the history encoder. It also employs a 4-layer Transformer structure with hyperparameters identical to those of E_{hist}, including the number of attention heads and the feed-forward network dimension. It encodes the agent's ground-truth motion trajectory in the corresponding prediction horizon, aiming to capture features related to driving intent and motion trends, and outputs an embedding vector e_f. These three encoders are jointly optimized through a joint reconstruction loss function $\mathcal{L}_{\text{joint}} = \lambda_h \mathcal{L}_{\text{hist}} + \lambda_m \mathcal{L}_{\text{scene}} + \lambda_f \mathcal{L}_{\text{fut}}$, where all loss weights are set to 1.0. The reconstruction task is performed by corresponding simple two-layer MLP decoders. This process compels each encoder to focus on its specific modality, thereby learning information-rich and highly disentangled representations that lay the foundation for subsequent stages.

A.4. Module Design and Training in the Supervised Fine-tuning Stage

In the fine-tuning stage, we build the final prediction model around a *retrieve-fuse-decode* pipeline.

Dynamic Knowledge Fusion Module. As the core of the framework, this module is designed based on the Mixture-of-Experts (MoE) paradigm. It aims to intelligently integrate retrieved prior knowledge with current scene information. The module receives the query embedding e_q generated by the frozen query encoder E_{query} and the N most relevant knowledge vectors retrieved from the knowledge base. The fusion process follows a cross-attention-based gating mechanism: a lightweight gating network uses e_q as the Query and the N retrieved knowledge vectors as Keys. It dynamically computes normalized attention weights $\alpha \in \mathbb{R}^N$ for these N experts through

a linear layer and a Softmax activation function. Finally, these weights are used to perform a weighted sum of the N knowledge vectors, fusing the multi-source information into a unified, high-information-density context vector e_{RAG}. This design ensures that retrieved knowledge is not simply averaged but is intelligently fused through a dynamic reasoning process contingent on the current context.

Two-Stage Trajectory Decoder. Our backbone architecture is based on the MTR framework [35]. We adapt this backbone into our two-stage learning paradigm, consisting of self-supervised pre-training and supervised fine-tuning, as described in Sec. A.1. To fully leverage the retrieved information within this paradigm, the decoder itself follows MTR's effective *propose-and-refine* process. Specifically, D_{propose} is a small MLP that generates K initial trajectory anchors from the history embedding e_h. Following this, a refinement decoder D_{refine}, also an MLP, concatenates each trajectory anchor with the more information-dense enhanced context e_{RAG}. It then predicts a residual correction for the anchor and the final trajectory probability, thereby outputting the final K predicted trajectories.

Optimization Objective and Training Strategy. The model is trained end-to-end using the AdamW optimizer for a total of 100 epochs. The initial learning rate is set to 0.001, with a linear warmup over the first 10 epochs, followed by a cosine annealing schedule. We apply a weight decay of 1e-4 and gradient clipping at a threshold of 5.0 to ensure training stability. The overall optimization objective is a composite loss function $\mathcal{L}_{\text{total}} = \lambda_{\text{prop}}\mathcal{L}_{\text{prop}} + \lambda_{\text{ref}}\mathcal{L}_{\text{ref}} + \mathcal{L}_{\text{cls}}$. Based on empirical tuning, we set the regression loss weights $\lambda_{\text{prop}} = 1.0$ and $\lambda_{\text{ref}} = 1.0$ to balance learning between the proposal and refinement stages. Consistent with our theoretical design, no MoE load-balancing loss is applied, preserving the natural relevance ranking of the retrieved experts.

B. Knowledge Base Construction

B.1. Unification of Heterogeneous Datasets

Large-scale, diverse datasets are fundamental to building models with high generalization capability. However, major industry datasets, such as Argoverse 2 and the Waymo Open Motion Dataset (WOMD), exhibit significant differences in data format, coordinate systems, sampling frequencies, and annotation paradigms. Therefore, data unification is an indispensable prerequisite for building a universal knowledge base. The cross-domain experiments in this study utilize the open-source frameworks ScenarioNet and UniTraj [8] to process all scenes into standardized 9-second segments, comprising a 4-second history and a 5-second future.

B.2. Knowledge Organization and Indexing

After extracting high-quality scene embeddings (e_h, e_m, e_f) from the unified dataset using the self-supervised encoders detailed in Section A.3, we construct the knowledge base through unsupervised learning and efficient indexing techniques.

Knowledge Organization: We apply the K-Means clustering algorithm to the concatenated history and scene embeddings $\{[e_h; e_m]\}$ for unsupervised pattern discovery. The goal is to discretize the continuous embedding space into semantically meaningful behavioral clusters. We ultimately determine the number of cluster centers to be $C = 64$ and the number of representative instances per cluster to be $N_s = 200$, based on the hyperparameter sensitivity analysis detailed in Section C.1. These selected representative instances form the raw content of the knowledge base. We use each instance's history embedding e_h as the key and the concatenation of its corresponding scene embedding e_m and future embedding e_f as the value.

Integrated High-Efficiency Indexing and Knowledge Organization: To achieve millisecond-level online retrieval, we use the FAISS library [17] to build an efficient Approximate Nearest Neighbor (ANN) index. To maximally integrate knowledge organization with retrieval efficiency, we adopt a unified strategy. Specifically, we select the `IndexIVFFlat` index type and directly use the history embedding portion of the 64 behavioral cluster centroids from the K-Means stage to initialize the inverted file cells of the FAISS index. Therefore, the core index parameter `nlist` is set to 64, perfectly corresponding to our number of behavioral clusters 'C'. During querying, the model only needs to perform an exact search within the few cells nearest to the query vector. This approach significantly enhances retrieval speed while ensuring relevance, thereby meeting the demands of real-time prediction.

C. Supplemental Experiments and Analysis

C.1. Hyperparameter Sensitivity Analysis

As shown in Figure 6, this section focuses on three core hyperparameters governing our retrieval mechanism and knowledge base structure: the number of retrieved knowledge items N for online retrieval, the number of cluster centers C in the knowledge base, and the number of representative instances per cluster N_s.

Impact of Retrieved Knowledge Quantity N: Subfigure (a) illustrates the effect of the number of retrieved prior knowledge items N on prediction accuracy. The results show that as N increases from 1, all evaluation metrics, including minADE_6, minFDE_6, and MR_6, improve significantly, reaching a performance peak at $N = 8$. This trend suggests that a smaller amount of knowledge fails to provide sufficient contextual information for prediction.

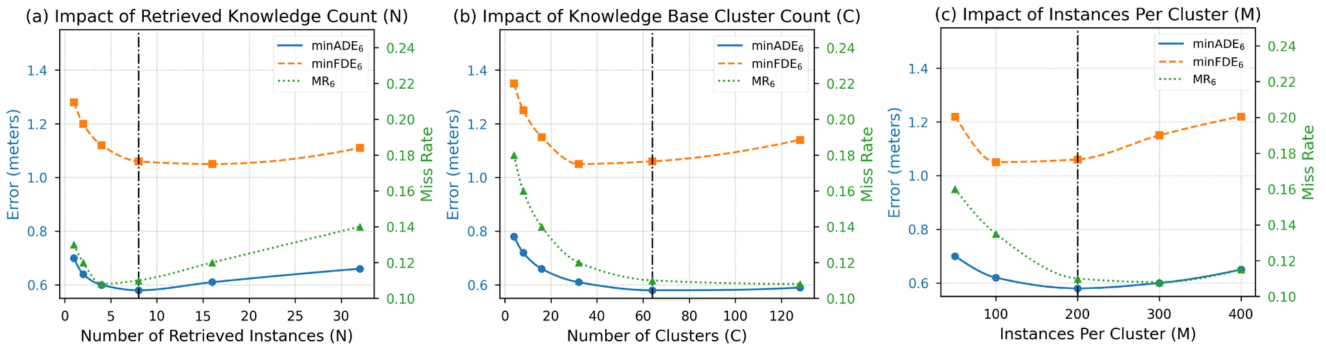

Figure 6. Sensitivity analysis of key hyperparameters in the RAG framework.

However, as N further increases to 16 or 32, model performance begins to slightly decline. This may be because an excessive amount of retrieved knowledge introduces irrelevant or conflicting guidance from different experts, increasing the burden on the MoE fusion module and thereby introducing noise into the final decision. Therefore, we select $N = 8$ as the optimal configuration, as it strikes an ideal balance between information richness and decision noise.

Impact of Knowledge Base Cluster Count C: Sub-figure (b) explores how the number of behavioral clusters C in the knowledge base affects performance. The results show that performance steadily improves as C increases, with a notable decrease in all error metrics as C grows from 16 to 64. This indicates that a finer-grained clustering can partition the driving behavior space with greater semantic specificity, making the retrieved knowledge more relevant to the current scene. When C exceeds 64, the performance improvement plateaus, possibly because the clustering is already sufficiently fine, and further increasing the number of clusters yields diminishing marginal returns. Considering both performance and computational efficiency, we identify $C = 64$ as an ideal setting that ensures semantic distinctiveness of knowledge without over-partitioning.

Impact of Instances Per Cluster N_s: Sub-figure (c) analyzes the role of the number of representative instances N_s within each behavioral cluster. This parameter directly determines the diversity of knowledge within each behavioral pattern in the knowledge base. As shown, model performance significantly improves as N_s increases from 50 to 200. This reveals the critical importance of providing a sufficient variety of exemplars within each semantic cluster to help the model generalize better. However, when N_s exceeds 200, performance begins to decline. This is likely because including too many instances far from the cluster center dilutes the homogeneity of the cluster's core behavioral pattern and introduces additional variance. Thus, choosing $N_s = 200$ provides an adequate and high-quality set of representative knowledge for each behavioral cluster.

C.2. Qualitative Visualization Analysis

Knowledge Base Clustering Results: Figure 7 provides an intuitive visualization of the unsupervised organization achieved by applying K-Means clustering to the concatenated history and scene embeddings $\{[e_h; e_m]\}$. Each column in the figure represents a distinct cluster, for which we display four different scene instances. We can clearly observe that this automated process successfully discovers and separates groups of scenes with significantly different kinematic characteristics. Scenes within a cluster exhibit high internal consistency in the morphology of their historical trajectories. For instance, scenes in Cluster 1 and Cluster 2 predominantly feature straight-line trajectories. In sharp contrast, scenes in Cluster 5 uniformly exhibit historical trajectories with a sharp left-turning curvature. This indicates that the model has autonomously identified driving straight and turning left as two fundamentally different motion states. More interestingly, the method also captures subtler differences. For example, the model distinguishes between the wide turns in Cluster 5 and the slight turns in Cluster 6. It also separates the lane-changing behaviors with lateral displacement in Cluster 3 from pure straight-line motion and turning maneuvers. This capability of automatically categorizing historical trajectory dynamics is central to the precise retrieval mechanism of the RAG-TP framework. It ensures that when the model encounters a new scene, it can recall truly similar historical experiences in terms of kinematics from the knowledge base, thereby providing the most relevant dynamic priors for prediction.

Challenging Scenario Analysis: To further demonstrate the practical performance of the RAG-TP model across various typical driving scenarios, Figure 8 presents a series of qualitative prediction results in map-free mode. The figure covers a range of core driving behaviors, including straight driving, lane changes, left turns, and right turns. We can observe that in all these scenarios, the predicted trajectories generated by the model are highly consistent with the ground-truth future trajectories. RAG-TP demon-

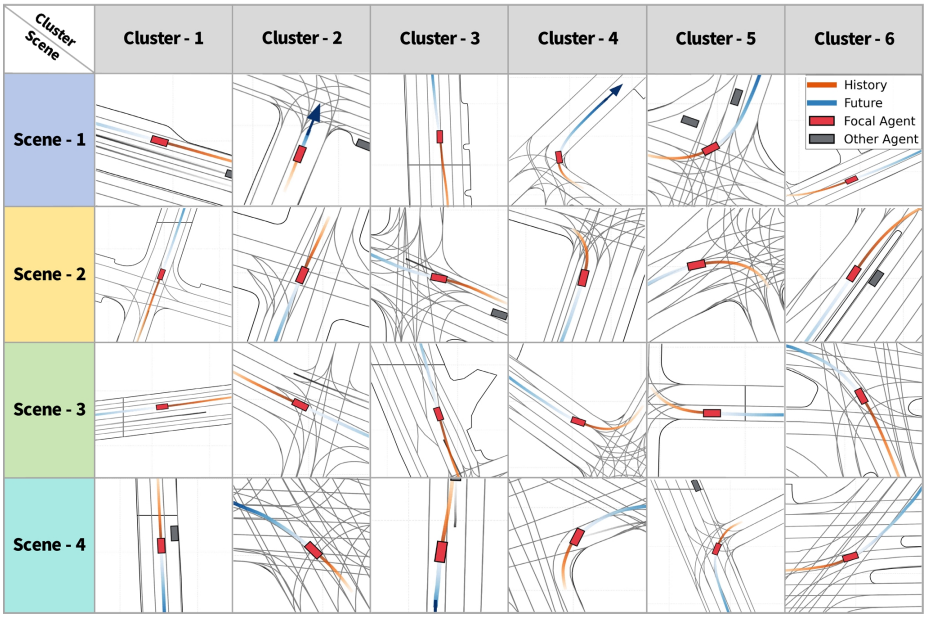

Figure 7. Visualization of scenes from different behavioral clusters in the knowledge base.

strates outstanding accuracy, whether in relatively simple tasks like straight driving and lane changes or in left and right turn scenarios that require an implicit understanding of complex intersection structures. Particularly in multiple turning cases, despite the varying geometry of each intersection, the model is able to generate smooth and natural trajectories that adhere to road constraints. These results strongly prove that by retrieving and fusing similar historical experiences from the knowledge base, our RAG module effectively compensates for the loss of scene structure and dynamic priors that results from the absence of real-time high-definition maps. This enables the model to make reliable and accurate trajectory predictions even in the challenging map-free setting, thus validating the effectiveness of our framework design.

Robustness Analysis for Mismatched Priors. To evaluate the framework's resilience against inevitable noise in a large-scale knowledge base, we analyze its performance when provided with mismatched or contradictory priors. Figure 9 illustrates four such challenging cases. In the first scenario involving a right turn and the third scenario featuring a straight-line crossing, the top retrieved prior is correct, but the second retrieved instance introduces contradictory semantics. Specifically, they retrieve a straight drive and a right turn respectively. More severely, the second scenario is a right-turn query where the top two retrieved priors are completely incorrect left turns. Finally, the fourth scenario features a slight left-veering maneuver, which retrieves a straight-driving prior alongside the correct slight left turn. Crucially, these retrieval inconsistencies do not lead to prediction failures. As shown in the final output, the predicted trajectories remain accurate and closely aligned with the ground truth across all four cases. This demonstrates the exceptional robustness of our RAG-TP framework. The dynamic MoE fusion module acts as an intelligent semantic gatekeeper. It leverages the online context to evaluate the retrieved experts, assigning near-zero attention weights to semantically incorrect priors while effectively utilizing the correct ones or relying solely on the online scene encoding. This analysis strongly validates the necessity of the dynamic MoE mechanism over naive fusion approaches for ensuring reliable map-free predictions.

D. Discussion on Scalability and Future Work

D.1. Potential for Continual Learning

The core design of this framework, which decouples parameterized inference and the non-parametric knowledge base, offers unique theoretical and architectural advantages for its evolution into a continual learning system. This decoupling is a key strength of RAG-TP in this direction, transforming the grand challenge of lifelong learning from periodic, high-cost retraining of the entire prediction model into a more manageable problem of dynamically updating the external knowledge base. However, fully realizing this potential requires addressing a series of cutting-edge research topics tailored to the RAG-TP architecture. For instance, the challenge of efficient knowledge integration translates into how to perform incremental, low-cost online updates to the behavioral clusters and their ANN index proposed in this work. The problem of knowledge selection and forgetting

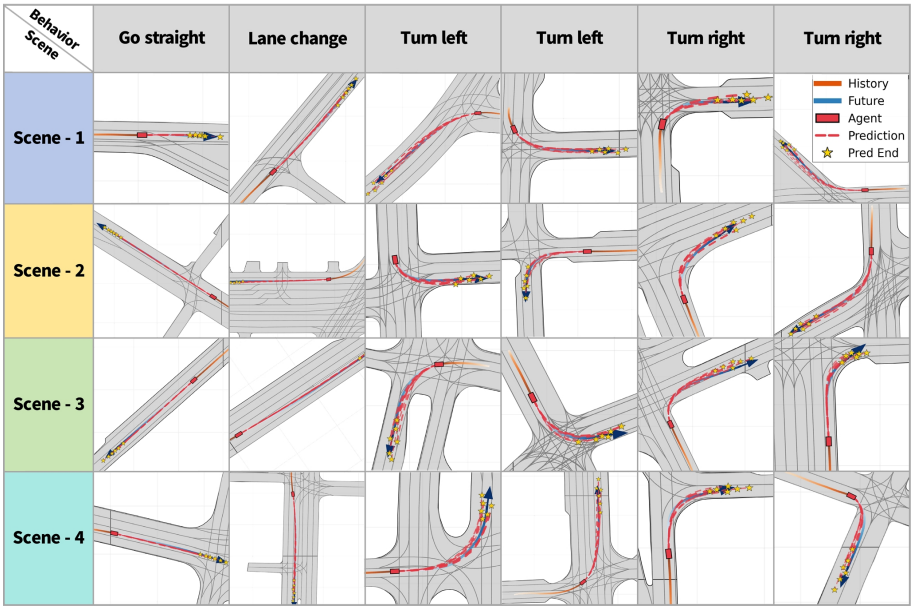

Figure 8. Qualitative analysis of RAG-TP in challenging scenarios.

can be addressed using RAG-TP's own components: the dynamic weighting of different knowledge sources by its MoE fusion module during inference can itself serve as a signal for evaluating knowledge value, with guidance from experts that is consistently assigned low weights being candidates for deprecation. Furthermore, the framework's explicit retrieval mechanism provides a clear path for adapting to concept drift: experiences corresponding to changes in road structure or new driving patterns can be directly injected into the knowledge base as new knowledge and immediately utilized by the model in relevant scenarios. Therefore, leveraging these inherent architectural advantages of RAG-TP to tackle the aforementioned challenges will be key to achieving a truly efficient and robust lifelong learning system for autonomous driving.

D.2. Model's Inherent Interpretability

In safety-critical domains such as autonomous driving, model interpretability is paramount. Building robust predictive models that are self-aware is a central challenge [16]. Unlike black-box models that completely embed knowledge within their network parameters, the RAG-TP framework offers a unique solution. Its core advantage lies in the fact that every decision made by the model is partly derived from explicitly retrieved, human-understandable driving instances from the knowledge base. This provides a pathway to establishing a clear decision attribution chain. Future work could focus on developing a system for visual attribution: when the model makes a critical decision, such as executing a left turn at a complex intersection, this system would not only output the predicted trajectory but also present the top-N historical experiences that informed this decision. These retrieved real-world scenarios form a clear chain of evidence, enabling developers and safety auditors to intuitively understand the basis of the model's decision-making process.

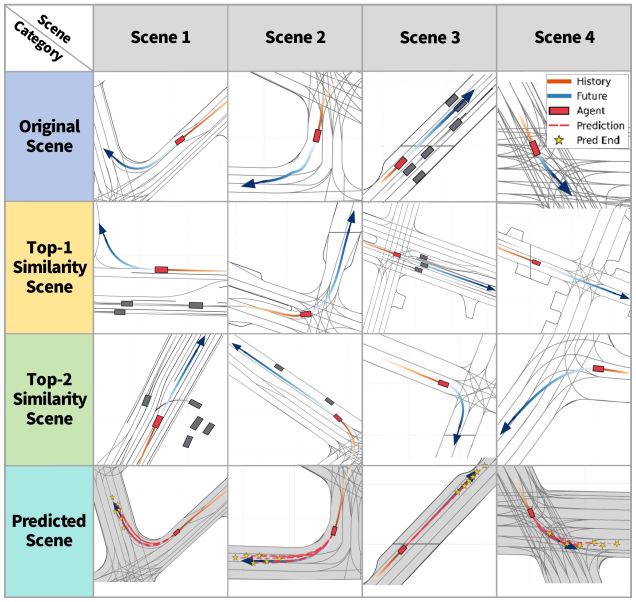

Figure 9. Robustness to retrieval failure. The dynamic MoE module successfully filters semantically incorrect priors, ensuring the final prediction remains correct even when retrieved knowledge is contradictory.