

# Streaming Real-Time Trajectory Prediction Using Endpoint-Aware Modeling

Alexander Prutsch      David Schinagl      Horst Possegger  
Institute of Visual Computing, Graz University of Technology  
{alexander.prutsch, david.schinagl, possegger}@tugraz.at

## Abstract

*Future trajectories of neighboring traffic agents have a significant influence on the path planning and decision-making of autonomous vehicles. While trajectory forecasting is a well-studied field, research mainly focuses on snapshot-based prediction, where each scenario is treated independently of its global temporal context. However, real-world autonomous driving systems need to operate in a continuous setting, requiring real-time processing of data streams with low latency and consistent predictions over successive timesteps. We leverage this continuous setting to propose a lightweight yet highly accurate streaming-based trajectory forecasting approach. We integrate valuable information from previous predictions with a novel endpoint-aware modeling scheme. Our temporal context propagation uses the trajectory endpoints of the previous forecasts as anchors to extract targeted scenario context encodings. Our approach efficiently guides its scene encoder to extract highly relevant context information without needing refinement iterations or segment-wise decoding. Our experiments highlight that our approach effectively relays information across consecutive timesteps. Unlike methods using multi-stage refinement processing, our approach significantly reduces inference latency, making it well-suited for real-world deployment. We achieve state-of-the-art streaming trajectory prediction results on the Argoverse 2 multi-agent and single-agent benchmarks, while requiring substantially fewer resources.*

## 1. Introduction

Trajectory prediction, *i.e.*, forecasting the future movements of nearby traffic participants, is a key component of the autonomy stack in self-driving vehicles. Trajectory prediction is essential for efficient and reliable navigation, as it enables autonomous vehicles to anticipate the behavior of surrounding agents and plan maneuvers proactively in response to predicted motions. The latency of the trajectory prediction model directly contributes to the total delay between sensor observations and vehicle control actions as it is typically performed between the perception and motion planning mod-

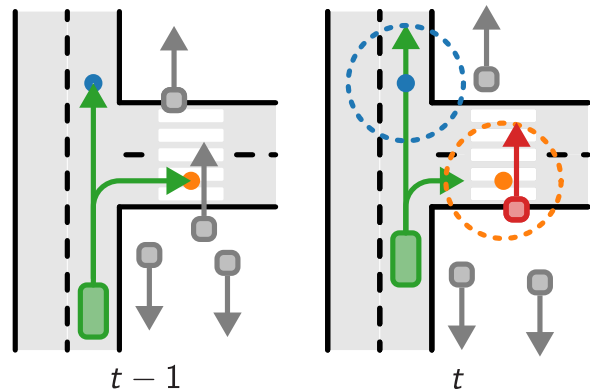


Figure 1. Traffic scenes are constantly changing, which is often neglected in trajectory prediction methods operating on snapshots rather than in a continuous setting. For example, for the trajectory prediction of the turning process of the focal agent (green), a possible interaction with a pedestrian (red) near the endpoint of the previous estimation (at time  $t - 1$ ) is important context that needs to be considered in the current prediction ( $t$ ). We model this temporal context propagation with a novel endpoint streaming mechanism, achieving accurate predictions at minimal latencies.

ules. Thus, for practical deployment in autonomous vehicles, trajectory prediction models must produce highly accurate predictions with minimal latency.

One of the key challenges in trajectory forecasting is identifying the most influential map elements and the most relevant neighboring agents for motion prediction, especially in complex scenarios with long prediction horizons, often exceeding 100 meters. While the initial agent position provides a strong prior for relevant context elements, scene elements along the future path play a crucial role, as illustrated in Figure 1. Recent trajectory prediction approaches commonly use refinement schemes [25, 39, 40] to incorporate this knowledge: Initial coarse future trajectories are iteratively refined to enhance prediction accuracy. For instance, SmartRefine [39] proposes an adaptive multi-stage refinement scheme with scene context re-encoding to improve predictions of the strong baseline model QCNet [40]. Methods that use such refinement iterations achieve highly accurate results on large-scale benchmarks like Argoverse 2 (AV2) [34]. However, they impose high computational cost and latency.

This limits their applicability in real-world autonomous systems, where fast and efficient decision-making is critical and methods with higher latency can lead to less reliable results due to outdated sensor observations.

In addition, current research in trajectory prediction focuses on processing and evaluating snapshots of individual scenarios in benchmark datasets [9, 34]. This snapshot-based treatment is in sharp contrast to the application in real autonomous vehicles, where trajectory prediction systems operate in a continuous environment. In this real-world scenario, additional constraints must be considered. In particular, the consistency across successive predictions, which is essential for integration into downstream decision-making systems. On the other hand, continuous streaming-based processing also offers advantages, such as using contextual information across multiple prediction frames to improve the accuracy of future motion estimates. Recently, pioneering streaming-based approaches [21, 27] have emerged, laying the foundation for addressing the challenges of continuous trajectory prediction. RealMotion [27] demonstrates that a simple backbone architecture combined with streaming mechanisms achieve promising results on the AV2 dataset [34]. However, their approach does not use all available temporal information to encode the scene context, for example by not taking into account past predictions.

To advance towards consistent and efficient continuous real-world motion forecasting, we introduce a novel streaming-based trajectory prediction approach that uses temporal information for targeted scene context extraction. In particular, we propose a streaming endpoint-aware modeling (SEAM) architecture to effectively incorporate information across consecutive frames. By leveraging prediction endpoints from the previous prediction frame as prior information, we directly integrate target region information into our modeling without requiring resource-intensive refinement iterations. In-depth evaluations on AV2 [34], both single and multi-agent benchmarks, show that our new streaming mechanism provides significant advantages over previously proposed streaming techniques [27]. We demonstrate that our approach outperforms more complex and resource-demanding models, while also addressing real-world considerations like reducing fluctuations across multiple predictions for the same agent. It combines high accuracy with minimal inference latency, achieving state-of-the-art results for streaming-based trajectory prediction on the AV2 benchmarks. In summary, our main contributions include:

- We propose a new trajectory prediction model, effectively leveraging the consecutive nature of scenarios for target-centric context encoding without any refinement steps.
- Despite achieving minimal inference latency, we demonstrate that our model outputs excellent results on complex trajectory prediction tasks and sets a new state-of-the-art in streaming processing on the AV2 multi-agent benchmark.

- Our ablation study highlights the effectiveness of our endpoint-aware modeling in propagating information from the previous prediction timestep.

## 2. Related Work

Trajectory prediction methods typically take agent states and map information as input. Map data is predominantly modeled as vectorized data [12, 29] and encoded by PointNet-like architectures [23], while agent time series information is commonly processed by recurrent architectures [18, 29], attention-based modules [4, 15, 22, 27] or state-space models [36]. To capture relationships between agents and map elements, recent methods utilize graph neural networks [6, 12, 14, 16, 28, 35] or attention-based encoders [4, 11, 17, 19, 20, 25, 38]. Trajectory decoding is performed using either multilayer perceptrons (MLPs) [4, 27] or more advanced cross-attention-based designs [25, 26, 36].

### 2.1. Refinement Modules for Trajectory Prediction

Iterative refinement is a common strategy in state-of-the-art trajectory prediction approaches [11, 25, 31, 39, 40]. Long-horizon benchmarks like Argoverse 2 [34] and WOMB [9] involve predictions spanning over 100 meters, resulting in a large number of scene context elements. While the initial agent position helps to identify the relevant context, detecting critical elements at long distances remains challenging based solely on the given past motion of the agent. To address this, multiple refinement steps can be applied to progressively extract scene context elements that have a strong influence on future agent behavior [25, 39]. Initial predictions provide strong guidance for identifying key scene elements which can be used to iteratively refine the trajectory hypotheses.

MTR [25] and its follow-up works [7, 11, 26] iteratively refine trajectory predictions, using dynamic intention queries for aggregating scene context during decoding. R-Pred [5] proposes a two-stage forecasting approach where initial predictions define scene context tubes which are then utilized by a refinement network to generate improved predictions. QCNet [40] employs a two-stage decoding process, where initial predictions are used as priors in a refinement stage by incorporating them into the mode query of a cross-attention decoder. Additionally, at each inference step, it divides future trajectory prediction into shorter temporal segments, which are predicted recurrently. SmartRefine [39] proposes an adaptive refinement iteration strategy, which is demonstrated for QCNet. At each inference step, an adaptive number of trajectory decoding iterations are performed to progressively refine predictions based on updated scene encoding. Recently, DeMo [36] introduces a decoupled query strategy, first modeling motion state consistency and multi-modal intents separately before decoding the final trajectory output.

Overall, methods like DeMo [36] and SmartRefine [39] provide strong evidence that multi-stage decoding improves

trajectory prediction quality. However, the gains in accuracy are often only small, especially considering the significantly higher computing costs required for executing a multi-stage refinement scheme at each prediction time step. This diminishing return underscores the need for novel strategies to balance computational cost and accuracy, when designing trajectory prediction models for real-world applications.

## 2.2. Streaming Trajectory Prediction

Current trajectory prediction research primarily focuses on snapshot-based prediction, where individual scenarios are processed without considering global temporal context across evolving scenes. In practice, however, autonomous systems operate in a continuous environment and process data in a streaming manner, introducing additional requirements like temporal consistency, as well as opportunities for improvement by leveraging additional temporal information.

Pang *et al.* [21] present an early contribution to motion forecasting in continuous, streaming scenarios. Their predictive streamer extends snapshot-based trajectory prediction models with a custom occlusion reasoning module and a differentiable filter to enhance temporal consistency. RealMotion [27] shows that incorporating context and predictions from previous frames enables accurate results even with a simple backbone [4]. They introduce a context stream applying cross-attention on ego-motion-aligned context from current and previous frames, and a trajectory relay mechanism that refines current predictions via offsets from embedded past predictions. DeMo [36] also integrates both mechanisms into their approach. HPNet [28] includes the exploration of a basic historical prediction attention module on the comparatively simple AV1 dataset [3]. QCNet [40] explores a key-value cache for past observations, touching on concepts related to streaming processing. However, it does not consider past predictions or ensure alignment across prediction steps, and its compute-intensive decoder poses significant challenges for real-time deployment.

Overall, existing streaming-based methods incorporate mechanisms to maintain the consistency of successive predictions [21, 27] and leverage the propagation of agent-centric scene context across timesteps [27] to address the continuous nature of real-world autonomous driving. However, they do not explore the use of the previously predicted trajectories as prior knowledge for context encoding. Given the importance of these regions for extracting relevant contextual features [25, 39], we propose an efficient strategy that includes this information in a single decoding step, thereby avoiding the high computational overhead of refinement modules.

## 3. Streaming Trajectory Prediction Using Endpoint-Aware Modeling

We propose SEAM, an accurate yet lightweight trajectory prediction method based on a streaming-based processing

paradigm. Unlike refinement-based approaches [25, 39], which use multi-stage decoding to iteratively refine predictions by aggregating relevant scene context, SEAM introduces a novel mechanism that propagates predictions across timesteps to extract endpoint-centered target region features. This design enables SEAM to achieve high accuracy with low latency by avoiding the computational overhead of refinement iterations. Furthermore, by leveraging past predictions to identify potential target regions, SEAM addresses a key limitation of prior streaming-based methods [21, 27], which ignore this strong temporal prior. Our approach aligns well with the continuous nature of real-world autonomous driving: as future prediction timesteps unfold, interactions with other agents may evolve and must be dynamically considered. For example, recall Figure 1: when predicting a turn, a potentially crossing pedestrian on the sidewalk must be considered. As illustrated in overview Figure 2, we leverage past prediction endpoints to guide the model towards relevant regions, allowing us to achieve highly accurate future trajectories in a single decoding step, eliminating the need for computationally expensive refinement iterations.

The core component of our approach is the streaming-based dual-region encoding mechanism that uses the endpoints from past predictions to enhance scene context aggregation. In addition to conventional agent-centric context encoding, which is centered at the current position of the focal agent, we introduce a second set of context features: We extract *target-centric* features by retrieving scene elements closely around the endpoints of the previously predicted trajectory as these regions are likely to have high relevance for the current frame’s prediction. We encode them using a shallow target encoder with the trajectory endpoints as coordinate system origins. During decoding, we execute attention to both agent-centric and target-centric features.

We describe the baseline encoding in Sections 3.1 and 3.2. Next, we introduce our *endpoint-aware modeling* (EAM), which includes the novel target-centric feature encoding (Section 3.3) and its integration into the proposed dual-context decoder (Section 3.4). Section 3.5 outlines the integration of baseline streaming mechanisms [27]. We present our extension to the multi-agent setting in Section 3.6.

**Input Representation:** Trajectory prediction refers to forecasting future trajectories  $F$  given historical agent states  $A$  and lane data  $L$ . We follow the common input representation of state-of-the-art work [4, 22, 27, 36] and initially sample all agents and map elements within a fixed radius around the focal agent. A tensor  $A \in \mathbb{R}^{N_a \times T_h \times D_a}$  represents the historical trajectories, where  $N_a$  is the number of agents,  $T_h$  represents the number of historical timesteps and  $D_a$  the feature dimension of the motion states, *i.e.* positions and velocity. Each lane is represented by sampling  $P_l$  points along its centerline. We define lane data as a tensor  $L \in \mathbb{R}^{N_l \times P_l \times D_l}$ , where  $N_l$  denotes the number of lane

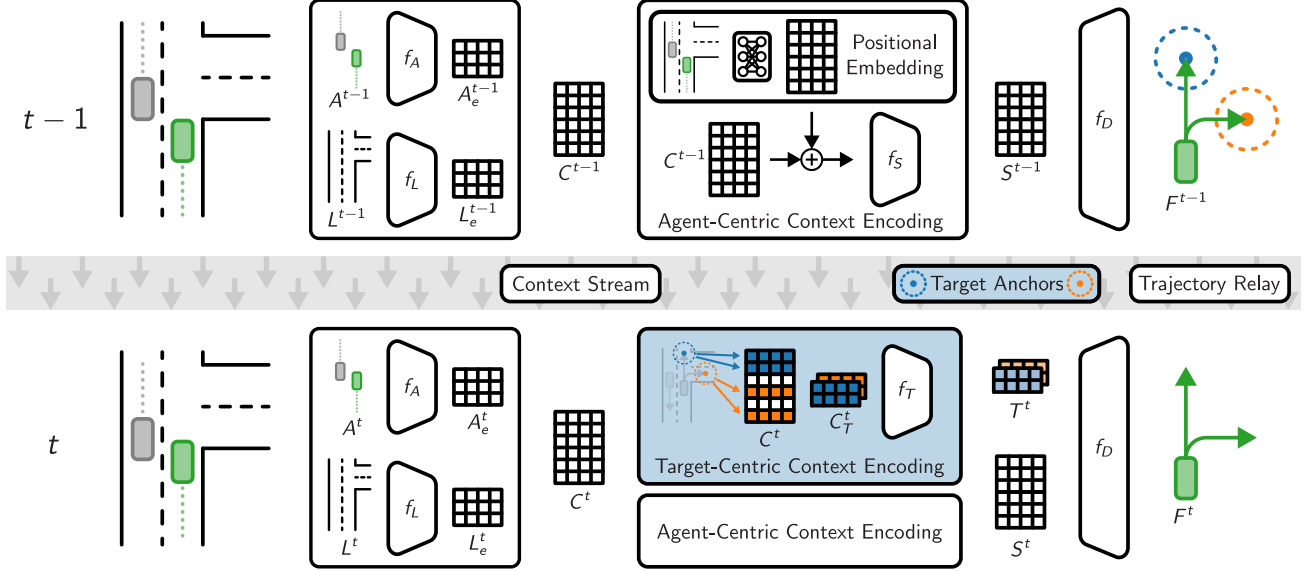


Figure 2. Overview of our streaming-based trajectory prediction architecture SEAM. We assume that for frame  $t - 1$  no previous prediction exists for our focal agent. Thus, only a standard trajectory prediction model pass is executed. In the next frame  $t$ , we incorporate the endpoints from the previous predictions to aggregate target-centric context information. We encode them using a second encoder path and provide it as additional input to our novel dual-context decoder.

segments and  $D_l$  represents the feature dimension, *i.e.* the  $xy$ -coordinates of the centerline. To facilitate efficient encoding of historical movements and lane data, we first transform each element in a local coordinate system [4]. We normalize the historical trajectories w.r.t. their initial pose while coordinates of lane centerlines are given w.r.t. their center position. This enables the model to learn motion patterns and lane segment shapes independently of global positions.

**Output Representation:** Our model predicts for a focal agent the multi-modal future trajectory hypotheses  $F \in \mathbb{R}^{K \times T_f \times 2}$ , where  $T_f$  denotes the number of future timesteps and  $K$  the number of motion modes, along with associated probability scores  $P \in \mathbb{R}^K$ . We represent future trajectories in  $xy$ -coordinates. Thus, trajectory prediction can be formulated as  $(F, P) = f(A, L)$ , where  $f$  is our trajectory prediction model. More specifically, our model consists of an encoder  $f_E$  generating the scene context  $S = f_E(A, L)$  and a trajectory decoder  $f_D$  using the scene context  $S \in \mathbb{R}^{N_a + N_l \times D}$  to predict trajectories  $F$  and probability scores  $P$  as  $(F, P) = f_D(S)$ .

**Streaming Processing:** To align with real-world autonomous driving applications, we adopt a streaming processing scheme to predict the future trajectories at the current timestep  $t$ . We leverage previously encoded context information  $S^{t-1}$  and past trajectory predictions  $F^{t-1}$  to generate highly accurate predictions and ensure temporally consistent trajectories. Consequently, our trajectory prediction at time  $t$  can be formulated as  $(F^t, P^t) = f(A^t, L^t, S^{t-1}, F^{t-1})$ .

### 3.1. Agent and Lane Encoding

We follow [15, 22] to encode the historical agent movement by projecting the agent data  $A^t$  into the  $D$ -dimensional feature space  $\mathbb{R}^{N_a \times T_h \times D}$  using a linear layer. Next, we apply multi-head self-attention [30] along the temporal dimension to capture the agent motion dynamics and use a pooling operation to aggregate temporal information. This yields an agent representation  $A_e^t \in \mathbb{R}^{N_a \times D}$  where the movement of each agent is encoded in a feature vector  $\mathbb{R}^{1 \times D}$ . To encode the lane data, we follow the default state-of-the-art approach [4, 22, 25–27, 36] of using a mini-PointNet-like [23] network. This results in a lane context representation  $L_e^t \in \mathbb{R}^{N_l \times D}$ . Thus, our encoders  $f_A$  and  $f_L$  generate agent feature tokens  $A_e^t = f_A(A^t)$  and lane feature tokens  $L_e^t = f_L(L^t)$  for further encoding.

### 3.2. Agent-Centric Context Encoding

After encoding each scene context element individually, the relationships among all scene elements must be learned. To achieve this, we concatenate the agent  $A_e^t$  and lane  $L_e^t$  tokens to form a scene context representation  $C^t \in \mathbb{R}^{N_c \times D}$ , where  $N_c = N_a + N_l$ . To incorporate categorical information, like different agent and lane types, we add learnable type embeddings [4] to the scene tokens. A common way for encoding relations among map elements and agents is self-attention across scene tokens [4, 15, 22, 36, 39]. To this end, a global positional encoding must be established, which is commonly done using a focal agent-centric coordinate system. We encode the global poses using the position  $(x, y)$  and the rotation *yaw* of each local token coordinate system,



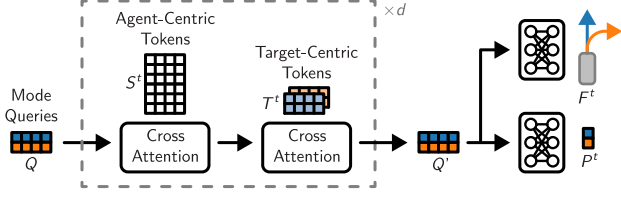


Figure 3. Architecture of our novel dual-context attention decoder, leveraging both agent-centric  $S^t$  and target-centric  $T^t$  features.

*i.e.* agent track origin and lane segment center. We apply a sine and cosine transformation to the yaw and process the pose information  $\mathbb{R}^{N_c \times 4}$  using a two-layer multilayer perceptron (MLP). Following, we add the positional embeddings to our context tokens  $C^t \in \mathbb{R}^{N_c \times D}$  and apply multi-head attention in our scene encoder  $f_S$ . This yields the encoded scene context  $S^t = f_S(C^t)$  with  $S^t \in \mathbb{R}^{N_c \times D}$ .

### 3.3. Target-Centric Context Encoding

The agent-centric encoding implicitly guides the model to focus on the surroundings of the focal agent, where nearby context elements are highly relevant for predicting future motion. However, as highlighted in Figure 1, also the areas where the agent is heading to are highly important for accurate predictions. Since we operate in a streaming-based setting, each agent that was visible in the previous frame has multi-modal past trajectory predictions, denoted as  $F^{t-1} \in \mathbb{R}^{K \times T_f \times 2}$ . These previous predictions  $F^{t-1}$  provide a strong prior for scenario regions that are also crucial for context extraction. We use the endpoints of these trajectories as anchors to define  $K$  target regions. Next, we gather tokens from our scene context  $C^t$  within a region-of-interest radius  $r$  around each target anchor, forming  $K$  individual target context token sets  $C_T^t \in \mathbb{R}^{K \times N'_c \times D}$ . In practice,  $N'_c$  is significantly smaller than  $N_c$  making our target encoding computationally less expensive than the main agent-centric encoding.

Following the same process as for the agent-centric scene context  $S$ , we encode each target-centric context  $C_T^t$  individually in our target encoder  $f_T$ , obtaining  $T^t \in \mathbb{R}^{K \times N'_c \times D}$ . To capture the relationship between the global coordinate root and each target anchor coordinate frame, we also add positional embeddings for the transformation to the target anchor of each region (see our supplementary for details).

### 3.4. Decoding with Dual-Context Attention

Our decoder  $f_D$  generates future trajectories  $F^t$  and probability scores  $P^t$  based on the agent-centric scene context  $S^t$  and our novel target-centric scene context  $T^t$ , formulated as  $(F^t, P^t) = f_D(Q, S^t, T^t)$ . We employ learnable queries  $Q \in \mathbb{R}^{K \times D}$  to obtain multi-modal outputs and implement a DETR-like [2] decoding approach. Figure 3 illustrates the architecture of our dual-context attention decoder.

We iteratively execute cross-attention  $d$ -times to our

scene context  $S \in \mathbb{R}^{N_c \times D}$ , and the target-specific tokens  $T \in \mathbb{R}^{K \times N'_c \times D}$ . Thus, information from both feature sets is available in each decoder stage and integrated into our final output. The agent-centric context  $S$  provides local information, while our target context  $T$  gives the model fine-grained information on long-horizon dependencies with other agents and map elements. Each mode query only attends to one associated set of target tokens  $T \in \mathbb{R}^{1 \times N'_c \times D}$ . Finally, we project the decoded queries  $Q'$  to future trajectories  $F^t \in \mathbb{R}^{K \times T_f \times 2}$  and probability scores  $P^t \in \mathbb{R}^K$  using a dedicated two-layer MLP for each output.

### 3.5. Context Referencing and Trajectory Relaying

In addition to our novel endpoint-aware modeling, we leverage context referencing and trajectory relaying [27] to model the streaming setting through information flow from the previous frame  $t-1$ . We follow the setup of [27] for both mechanisms to enable fair comparison: To account for focal agent motion between  $t-1$  and  $t$ , we apply motion-aware layer normalization (MLN) [32] to the scene context  $S^{t-1}$  before integrating it into the current encoding  $C^t$  via agent-to-scene and map-to-map cross-attention. Trajectory relay employs a cross-attention-based interaction module to refine the current predictions  $F^t$  using past predictions  $F^{t-1}$ .

### 3.6. Extension to Multi-Agent Settings

To predict accurate trajectories for all agents  $N_a$  in a scene, we first generate marginal predictions for each agent individually. To this end, we treat each agent as focal agent once for our single-agent model, stack the input data and perform an inference pass with a batch size of  $N_a$ . Next, we introduce a lightweight global consistency module, to fuse the marginal predictions to joint, globally consistent world predictions.

Our consistency module takes the mode features  $Q' \in \mathbb{R}^{N_a \times K \times D}$  for each agent and their global positions  $R \in \mathbb{R}^{N_a \times 4}$  as input. A shallow MLP encodes the global positions into positional embeddings. We then apply self-attention across all modes per agent (over  $K$ ) as well as self-attention across all agents per mode (over  $N_a$ ) to capture intra- and inter-agent relationships. Following our single agent decoder design, we decode the final world trajectory predictions using two shallow MLPs. As output, we obtain  $F_w \in \mathbb{R}^{K \times N_a \times T_f \times 2}$  world predictions with associated confidence scores  $P_w \in \mathbb{R}^K$ .

## 4. Experiments

We present experimental results on the single-agent and multi-agent benchmarks of the Argoverse 2 (AV2) [34] dataset, along with a latency analysis for both settings to assess the practical applicability of our approach. Additionally, we provide an ablation study for our endpoint-aware modeling on both AV2 and Argoverse (AV1) [3], with further results included in the supplementary material.

Method	Streaming	minADE <sub>1</sub>	minFDE <sub>1</sub>	MR <sub>6</sub>	minADE <sub>6</sub>	minFDE <sub>6</sub>	brier-minFDE <sub>6</sub>
SIMPL [37]	✗	2.03	5.50	0.19	0.72	1.43	2.05
HPTR [38]	✗	1.84	4.61	0.19	0.73	1.43	2.03
Forecast-MAE [4]	✗	1.74	4.36	0.17	0.71	1.39	2.03
MTR [25]	✗	1.74	4.39	0.15	0.73	1.44	1.98
EMP-D [22]	✗	1.75	4.35	0.17	0.71	1.37	1.98
GANet [31]	✗	1.77	4.48	0.17	0.72	1.34	1.96
ProIn [8]	✗	1.77	4.50	0.18	0.73	1.35	1.93
RealMotion [27] *	✓	1.65	4.12	0.15	0.67	1.29	1.93
QCNet [40]	✗	1.69	4.30	0.16	0.65	1.29	1.91
RealMotion [27] (Train/Val)*	✓	<u>1.59</u>	<u>3.93</u>	0.15	0.66	1.24	1.89
Tamba [13]	✗	1.66	4.24	0.17	0.64	1.24	1.89
ProphNet [33]	✗	1.80	4.74	0.18	0.68	1.33	1.88
MTR++ [26]	✗	1.64	4.12	<u>0.14</u>	0.71	1.37	1.88
DyMap [10]	✗	-	-	-	0.71	1.29	1.87
SmartRefine [39]	✗	1.65	4.17	0.15	<u>0.63</u>	<u>1.23</u>	<u>1.86</u>
DeMo [36]	✓	<b>1.49</b>	<b>3.74</b>	<b>0.13</b>	<b>0.61</b>	<b>1.17</b>	<b>1.84</b>
SEAM (Ours)	✓	1.60	3.95	0.15	0.66	1.24	<b>1.84</b>

Table 1. Single-agent trajectory prediction results on the Argoverse 2 test set. For all metrics lower indicate better, table sorted in descending order by brier-minFDE<sub>6</sub>. We report results for published works on the official leaderboard without model ensembling. The streaming approach of Pang *et al.* [21] is not included (their custom dataset prevents direct comparison). \*For RealMotion [27] we report the results for their official checkpoint, as well as the results from their paper, which is trained on the train and validation set.

#### 4.1. Experimental Settings

**Dataset:** We evaluate our model on the Argoverse 2 (AV2) [34] trajectory prediction dataset. It has a sampling rate of 10 Hz and contains 199,908 training, 24,988 validation, and 24,984 test scenarios collected in six U.S. cities. In each scenario the first 5 s serve as historical context and the goal is to predict the trajectories for the subsequent 6 s. The long historical context makes AV2 ideal for evaluation in streaming processing, as it allows a reasonable sliding-window processing (see below). In contrast, WOMD [9] provides only 1 s of history and nuScenes [1] offers 2 s but at only 2 Hz. Further reducing this few historical data points to multiple windows limits the agent encoder’s ability to learn motion patterns. Both are also not considered for streaming processing by related work [21, 27, 36]. We evaluate on the single-agent (predict trajectories for one agent per scenario) and multi-agent (predict globally consistent trajectories for multiple agents per scenario) benchmarks of AV2. Additionally, we provide an ablation study on Argoverse 1 (AV1) [3] which features 2 s history and considers a future of 3 s.

**Metrics:** We evaluate our approach using the standard AV2 [34] benchmark metrics. For the single-agent case, we use the miss rate (MR<sub>k</sub>), minimum average displacement error (minADE<sub>k</sub>), minimum final displacement error (minFDE<sub>k</sub>), and Brier minimum final displacement error (brier-minFDE<sub>k</sub>). Following the official leaderboard, we evaluate using the top  $k \in \{1, 6\}$  scoring trajectories. For the multi-agent case we utilize the actorMR<sub>k</sub>, avgMinADE<sub>k</sub>, avgMinFDE<sub>k</sub> and avgBrierMinFDE<sub>k</sub> which account for the mean errors across all actors in each world.

**Streaming Processing:** To enable streaming processing

using existing benchmarks, we follow [27, 36] by limiting the historical model input  $T_h$  to a fraction of the available history. We then operate using a sliding window and conduct multiple model executions. In particular, for AV2 we use a historic input of  $h = 3$  s and execute three predictions at  $t \in \{3, 4, 5\}$  s. The final prediction is made at the same time point as in snapshot-based methods, allowing a fair comparison. We provide a detailed comparison of streaming and snapshot-based processing in our supplementary material.

#### 4.2. Argoverse 2 Single-Agent Results

Table 1 shows the results for the Argoverse 2 single-agent test set. Our SEAM, along with DeMo [36], achieves the best score on the main metric (brier-minFDE<sub>6</sub>), outperforming recent work like Tamba [13] and even more resource-intensive methods, such as QCNet [40] and SmartRefine [39]. This strong performance without any refinement iterations highlights the effectiveness of our endpoint-aware modeling. We also outperform RealMotion [27], a prior streaming-based prediction method, while also achieving lower inference latency (Sec. 4.4). DeMo [36] achieves slightly lower displacement errors since their decoupled decoding yields more diverse trajectories, but at a significantly higher latency (Sec. 4.4). The small gap between minFDE<sub>6</sub> to brier-minFDE<sub>6</sub> (induced by the main metric’s probability penalty) demonstrates our model’s strong confidence estimation, which is also crucial for downstream integration into the planning module. This highlights that directly integrating target-centric features helps assess maneuver likelihood.

Method	Streaming	avgMinADE <sub>1</sub>	avgMinFDE <sub>1</sub>	actorMR <sub>6</sub>	avgMinADE <sub>6</sub>	avgMinFDE <sub>6</sub>	avgBrierMinFDE <sub>6</sub>
FJMP [24]	✗	1.52	4.00	0.23	0.81	1.89	2.59
Forecast-MAE [4]	✗	1.30	3.33	0.19	0.69	1.55	2.24
RealMotion [27]	✓	1.14	2.97	0.18	0.62	1.32	2.01
DeMo [36]	✓	1.12	2.78	0.16	0.58	1.24	1.93
SEAM (Ours)	✓	1.05	2.57	0.16	0.62	1.21	1.85
QCNeXt [41]	✗	1.03	2.55	0.14	0.54	1.13	1.79

Table 2. Multi-agent trajectory prediction results on the Argoverse 2 test set. For all metrics lower values indicate better performance, table sorted in descending order by avgBrierMinFDE<sub>6</sub>. For all methods we report results without model ensembling.

Method	Streaming	Latency ( $B = 1$ )		Latency ( $B = 32$ )		Latency ( $B = 64$ )		Model Parameters	AV2 Test Set brier-minFDE <sub>6</sub>
		Offline	Online	Offline	Online	Offline	Online		
SmartRefine [39]	✗	>118 ms*		>680 ms*		-		8.0M	1.86
QCNet [40]	✗	118 ms		680 ms		-		7.7M	1.91
RealMotion [27]	✓	86 ms	24 ms	247 ms	88 ms	436 ms	150 ms	2.9M	1.93
DeMo [36]	✓	139 ms	39 ms	275 ms	83 ms	466 ms	142 ms	5.9M	1.84
SEAM (Ours)	✓	91 ms	28 ms	140 ms	50 ms	246 ms	74 ms	4.6M	1.84

Table 3. Latency analysis for predicting  $B$  Argoverse 2 scenarios in the single-agent setting using a NVIDIA V100 GPU. *Online* (for streaming-based methods) is the time required to process a single sliding window; *Offline* is the time required to process the full stream (to generate the benchmark predictions). For streaming methods the *online* latency is relevant for practical applications. \*SmartRefine does not provide an implementation for AV2—since its main results add refinement iterations to QCNet, latency is higher than for QCNet.

### 4.3. Argoverse 2 Multi-Agent Results

Previous methods, *e.g.* [13, 39], often report only single-agent results. However, the multi-agent benchmark is more relevant for practical systems: models deployed in real-world settings must be capable to predict trajectories for more than one agent in a scene at the same time. The results in Table 2 show that, in this setting, our SEAM outperforms previous streaming-based methods [27, 36] in the main metrics. The current best-performing method is the snapshot method QCNeXt [41], an extension of QCNet [40]. However, the displacement errors differ only by a few centimeters, whereas its extensive use of recurrence and refinement iterations in decoding poses significant challenges for real-time deployment. Since model latency directly impacts the delay between sensor observations and prediction output, models with computational overhead may lead to worse results in real-world scenarios due to delayed response. For reference, an agent traveling at 50 km/h covers approximately 0.7 m in 50 ms. Thus, marginally lower displacement errors at increased processing delay offer no benefit in practice.

### 4.4. Latency Analysis

**Argoverse 2 Single-Agent Setting:** Table 3 compares latencies for predicting various batch sizes of AV2 single-agent scenarios. Our approach has favorable runtime compared to related work, especially since it scales well with larger batch sizes which are relevant in practice as the single-agent benchmark evaluates only one agent per scenario. SEAM achieves minimal latency by using only a single decoder pass, unlike DeMo [36], QCNet [40] and SmartRefine [39], which

employ multiple decoding stages. Additionally, it predicts the complete future trajectory without splitting it into segments, as done in QCNet and SmartRefine. In comparison to RealMotion [27] our more efficient decoder design leads to a latency advantage. The fast processing speed makes our approach highly suitable for real-world deployment, as it combines streaming modeling with computational efficiency — two key considerations for practical use.

**Argoverse 2 Multi-Agent Setting:** Related work on streaming-based processing (RealMotion [27] and DeMo [36]) does not provide details on their multi-agent implementations. Moreover, no official implementation is available for QCNeXt [41]. As a result, a direct latency comparison is not possible. However, based on the nature of the task, we can reasonably assume that per-scenario latency in a multi-agent setting exceeds the single-agent latencies reported in Table 3. For our model, we measure an average online latency of 38 ms per scenario in the multi-agent setting on the AV2 validation set using a single NVIDIA V100 GPU (batch size  $B = 1$ ). Notably, this is even faster than the single-agent prediction latencies reported for DeMo and QCNet, highlighting the high efficiency of our approach.

### 4.5. Ablation Studies

**Information Streaming Mechanisms:** In Table 4 we compare our endpoint-aware modeling (EAM), which includes the target-centric feature encoding and the dual-context decoder, to the context stream (CS) and trajectory relay (TR) streaming mechanisms introduced by previous work [27]. The top row in each group presents the results

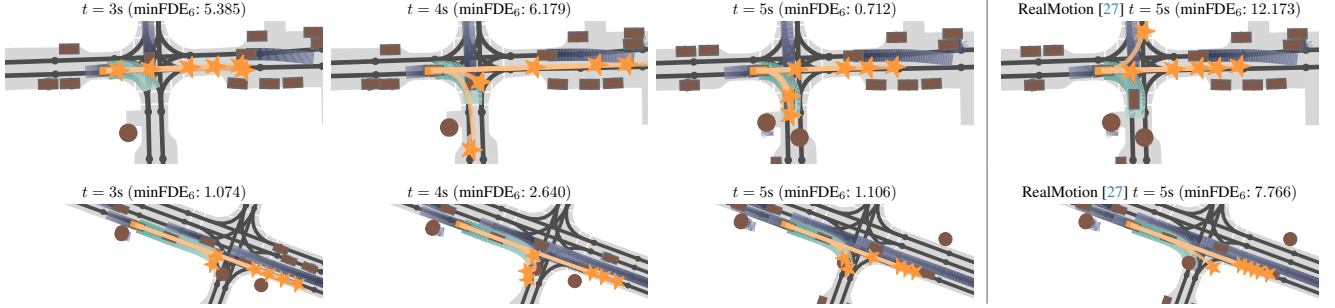


Figure 4. Qualitative results on two Argoverse 2 scenarios. We show the **predictions** of our streaming-based method at  $t \in \{3, 4, 5\}$ s. The visualizations also show **ground truth future**, **agent histories**, and **neighboring agents**. The right column shows the final predictions at  $t = 5$ s for RealMotion [27] in the streaming-based setting. Top row: the focal agent is approaching an intersection where other traffic is currently passing by, making it difficult to identify possible movement. Bottom row: a pedestrian crosses the street at an intersection to the right. Our approach correctly predicts that the vehicle can either continue straight or turn right, either waiting before the crosswalk or proceeding directly if the pedestrian has already left the crosswalk.

Dataset	$h$	Stream	CS+TR [27]	EAM	mADE <sub>6</sub>	mFDE <sub>6</sub>	b-mFDE <sub>6</sub>
AV1	2s	×	×	×	0.64	1.04	1.67
	1s	✓	✓	×	0.62	0.98	1.62
	1s	✓	×	✓	0.60	0.95	1.55
	1s	✓	✓	✓	<b>0.59</b>	<b>0.93</b>	<b>1.52</b>
AV2	5s	×	×	×	0.77	1.39	2.00
	3s	✓	✓	×	0.75	1.29	1.93
	3s	✓	×	✓	0.70	1.27	1.88
	3s	✓	✓	✓	<b>0.66</b>	<b>1.25</b>	<b>1.85</b>

Table 4. Comparison of different information forwarding mechanisms for streaming trajectory prediction. We compare the integration of context stream (CS) and trajectory relay (TR) introduced by [27] to our endpoint-aware modeling (EAM). The first line shows the traditional snapshot-based use. We highlight **best** values.

of our backbone architecture operating in the traditional snapshot-based prediction setting without any streaming mechanism. Comparing streaming-based processing using CS and TR to our approach highlights the effectiveness of our EAM in propagating information from the previous frame. Our prediction endpoint-aware modeling achieves a larger improvement than the two streaming mechanisms proposed by [27]. Leveraging endpoint information in the streaming setting proves to be highly beneficial for enhancing trajectory prediction accuracy. Overall, combining our EAM with CS and TR leads to the best results.

**Target Context Radius:** We evaluate the influence of different target context radii  $r$  in Table 5. Performance saturates at a radius of 30 meters, as incorporating further context at larger radii does not provide relevant information for our endpoint-aware modeling. A larger radius increases the number of target context tokens and thus latency, making a radius of 30 meters an optimal trade-off between performance and efficiency for practical applications. Notably, our approach remains effective even with a smaller radius of 10 meters, as injecting previous endpoint positions into the decoder already provides a strong prior. Incorporating the endpoint location during decoding also guides cross-attention toward

Dataset	$r$	mADE <sub>6</sub>	mFDE <sub>6</sub>	b-mFDE <sub>6</sub>
AV2	10m	0.68	1.27	1.89
	15m	0.67	1.26	1.87
	30m	<b>0.66</b>	<b>1.25</b>	<b>1.85</b>
	45m	<b>0.66</b>	<b>1.25</b>	<b>1.85</b>

Table 5. Comparison of different radii  $r$  to select context elements for our target-centric region. We highlight **best** values.

agent-centric features, further underscoring the importance of endpoint information in streaming processing.

#### 4.6. Qualitative Results

Figure 4 shows predictions for two AV2 validation set scenarios, illustrating the evolving nature of streaming processing at  $t \in \{3, 4, 5\}$ s. To demonstrate the benefits of our endpoint-aware modeling against previous work on streaming trajectory prediction, we also show the output of RealMotion [27] at  $t = 5$ s. Additional qualitative results, including failure cases, are available in the supplementary material.

### 5. Conclusion

We propose a novel endpoint-aware modeling for streaming-based trajectory prediction. Our dual-context approach enables us to effectively incorporate the continuously evolving relevant scene context, eliminating the need for costly trajectory refinement iterations to achieve highly accurate predictions. We significantly outperform inference speed of highly iterative schemes [26, 39] and previous streaming based methods [27, 36] for practical scenario sizes, while achieving competitive results on the Argoverse 2 single-agent dataset and setting new state-of-the-art for streaming-based trajectory prediction on the multi-agent benchmark.

**Acknowledgments.** This work was partially funded by Addisafety (923936), a COMET Project funded by BMIMI, BMWET as well as the co-financing federal province of Styria. The COMET programme is managed by the FFG.



## References

- [1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *CVPR*, 2020. 6
- [2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020. 5
- [3] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3D Tracking and Forecasting with Rich Maps. In *CVPR*, 2019. 3, 5, 6
- [4] Jie Cheng, Xiaodong Mei, and Ming Liu. Forecast-MAE: Self-supervised Pre-training for Motion Forecasting with Masked Autoencoders. In *ICCV*, 2023. 2, 3, 4, 6, 7
- [5] Sehwan Choi, Jungho Kim, Junyong Yun, and Jun Won Choi. R-Pred: Two-Stage Motion Prediction Via Tube-Query Attention-Based Trajectory Refinement. In *ICCV*, 2023. 2
- [6] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. GoRela: Go Relative for Viewpoint-Invariant Motion Forecasting. In *ICRA*, 2023. 2
- [7] Tobias Demmler, Lennart Hartung, Andreas Tamke, Thao Dang, Alexander Hegai, Karsten Haug, and Lars Mikelsons. Dynamic Intent Queries for Motion Transformer-based Trajectory Prediction. In *IEEE IV*, 2025. 2
- [8] Yinke Dong, Haifeng Yuan, Hongkun Liu, Wei Jing, Fangzhen Li, Hongmin Liu, and Bin Fan. ProIn: Learning to Predict Trajectory Based on Progressive Interactions for Autonomous Driving. *arXiv*, 2024. 6
- [9] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large Scale Interactive Motion Forecasting for Autonomous Driving: The Waymo Open Motion Dataset. In *ICCV*, 2021. 2, 6
- [10] Bin Fan, Haifeng Yuan, Yinke Dong, Zhengyu Zhu, and Hongmin Liu. Bidirectional Agent-Map Interaction Feature Learning Leveraged by Map-Related Tasks for Trajectory Prediction in Autonomous Driving. *IEEE Trans. Automation Science and Eng.*, 2025. 6
- [11] Yiqian Gan, Hao Xiao, Yizhe Zhao, Ethan Zhang, Zhe Huang, Xin Ye, and Lingting Ge. MGTR: Multi-Granular Transformer for Motion Prediction with LiDAR. In *ICRA*, 2024. 2
- [12] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. VectorNet: Encoding HD Maps and Agent Dynamics From Vectorized Representation. In *CVPR*, 2020. 2
- [13] Yizhou Huang, Yihua Cheng, and Kezhi Wang. Trajectory Mamba: Efficient Attention-Mamba Forecasting Model Based on Selective SSM. In *CVPR*, 2025. 6, 7
- [14] Xiaosong Jia, Penghao Wu, Li Chen, Yu Liu, Hongyang Li, and Junchi Yan. HDGT: Heterogeneous Driving Graph Transformer for Multi-Agent Trajectory Prediction via Scene Encoding. *IEEE TPAMI*, 2023. 2
- [15] Zhiqian Lan, Yuxuan Jiang, Yao Mu, Chen Chen, Shengbo Eben Li, Hang Zhao, and Keqiang Li. SEPT: Towards Efficient Scene Representation Learning for Motion Prediction. In *ICLR*, 2023. 2, 4
- [16] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning Lane Graph Representations for Motion Forecasting. In *ECCV*, 2020. 2
- [17] Yicheng Liu, Jinghui Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal Motion Prediction with Stacked Transformers. *CVPR*, 2021. 2
- [18] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-Head Attention for Multi-Modal Joint Vehicle Motion Forecasting. In *ICRA*, 2020. 2
- [19] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratharth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In *ICRA*, 2023. 2
- [20] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. SceneTransformer: A Unified Architecture for Predicting Multiple Agent Trajectories. *ICLR*, 2022. 2
- [21] Ziqi Pang, Deva Ramanan, Mengtian Li, and Yu-Xiong Wang. Streaming Motion Forecasting for Autonomous Driving. In *IROS*, 2023. 2, 3, 6
- [22] Alexander Prutsch, Horst Bischof, and Horst Possegger. Efficient Motion Prediction: A Lightweight & Accurate Trajectory Prediction Model With Fast Training and Inference Speed. In *IROS*, 2024. 2, 3, 4, 6
- [23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 2, 4
- [24] Luke Rowe, Martin Ethier, Eli-Henry Dykhne, and Krzysztof Czarnecki. FJMP: Factorized Joint Multi-Agent Motion Prediction over Learned Directed Acyclic Interaction Graphs. In *CVPR*, 2023. 7
- [25] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. Motion Transformer with Global Intention Localization and Local Movement Refinement. In *NeurIPS*, 2022. 1, 2, 3, 4, 6
- [26] Shaoshuai Shi, Li Jiang, Dengxin Dai, and Bernt Schiele. MTR++: Multi-Agent Motion Prediction With Symmetric Scene Modeling and Guided Intention Querying. *IEEE TPAMI*, 2024. 2, 6, 8
- [27] Nan Song, Bozhou Zhang, Xiatian Zhu, and Li Zhang. Motion Forecasting in Continuous Driving. In *NeurIPS*, 2024. 2, 3, 4, 5, 6, 7, 8
- [28] Xiaolong Tang, Meina Kan, Shiguang Shan, Zhilong Ji, Jinfeng Bai, and Xilin Chen. HPNet: Dynamic Trajectory Forecasting with Historical Prediction Attention. In *CVPR*, 2024. 2, 3
- [29] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. MultiPath++: Efficient Information Fusion and Trajectory Aggregation for Behavior Prediction. In *ICRA*, 2022. 2
- [30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017. 4

- [31] Mingkun Wang, Xinge Zhu, Changqian Yu, Wei Li, Yuexin Ma, Ruochun Jin, Xiaoguang Ren, Dongchun Ren, Mingxu Wang, and Wenjing Yang. GANet: Goal Area Network for Motion Forecasting. In *ICRA*, 2023. [2](#), [6](#)
- [32] Shihao Wang, Yingfei Liu, Tiancai Wang, Ying Li, and Xiangyu Zhang. Exploring Object-Centric Temporal Modeling for Efficient Multi-View 3D Object Detection. In *CVPR*, 2023. [5](#)
- [33] Xishun Wang, Tong Su, Fang Da, and Xiaodong Yang. ProphNet: Efficient Agent-Centric Motion Forecasting with Anchor-Informed Proposals. In *CVPR*, 2023. [6](#)
- [34] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting. In *NeurIPS Datasets and Benchmarks*, 2021. [1](#), [2](#), [5](#), [6](#)
- [35] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. LaneRCNN: Distributed Representations for Graph-Centric Motion Forecasting. In *IROS*, 2021. [2](#)
- [36] Bozhou Zhang, Nan Song, and Li Zhang. DeMo: Decoupling Motion Forecasting into Directional Intentions and Dynamic States. In *NeurIPS*, 2024. [2](#), [3](#), [4](#), [6](#), [7](#), [8](#)
- [37] Lu Zhang, Peiliang Li, Sikang Liu, and Shaojie Shen. SIMPL: A Simple and Efficient Multi-agent Motion Prediction Baseline for Autonomous Driving. *RAL*, 2024. [6](#)
- [38] Zhejun Zhang, Alexander Liniger, Christos Sakaridis, Fisher Yu, and Luc Van Gool. Real-Time Motion Prediction via Heterogeneous Polyline Transformer with Relative Pose Encoding. In *NeurIPS*, 2023. [2](#), [6](#)
- [39] Yang Zhou, Hao Shao, Letian Wang, Steven L. Waslander, Hongsheng Li, and Yu Liu. SmartRefine: A Scenario-Adaptive Refinement Framework for Efficient Motion Prediction. In *CVPR*, 2024. [1](#), [2](#), [3](#), [4](#), [6](#), [7](#), [8](#)
- [40] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-Centric Trajectory Prediction. In *CVPR*, 2023. [1](#), [2](#), [3](#), [6](#), [7](#)
- [41] Zikang Zhou, Zihao Wen, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. QCNeXt: A Next-Generation Framework For Joint Multi-Agent Trajectory Prediction. *arXiv*, 2023. [7](#)