# SHARP: Short-Window Streaming for Accurate and Robust Prediction in Motion Forecasting

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

## Abstract

*In dynamic traffic environments, motion forecasting models must be able to accurately estimate future trajectories continuously. Streaming-based methods are a promising solution, but despite recent advances, their performance often degrades when exposed to heterogeneous observation lengths. To address this, we propose a novel streaming-based motion forecasting framework that explicitly focuses on evolving scenes. Our method incrementally processes incoming observation windows and leverages an instance-aware context streaming to maintain and update latent agent representations across inference steps. A dual training objective further enables consistent forecasting accuracy across diverse observation horizons. Extensive experiments on Argoverse 2, nuScenes, and Argoverse 1 demonstrate the robustness of our approach under evolving scene conditions and also on the single-agent benchmarks. Our model achieves state-of-the-art performance in streaming inference on the Argoverse 2 multi-agent benchmark, while maintaining minimal latency, highlighting its suitability for real-world deployment.*

## 1. Introduction

Trajectory prediction is a core component of the autonomous vehicle (AV) control stack, providing hypotheses on the future motions of surrounding agents based on perception outputs, typically detections and tracks. Accurate and robust predictions are critical for safe and efficient motion planning which enables the AV to anticipate and respond to dynamic behaviors in complex traffic scenarios. In real-world driving, traffic scenes are constantly changing, as illustrated in Fig. 1: Agents entering the field-of-view of the AV have been observed only briefly, whereas for other agents a more comprehensive motion history is available. Motion forecasting models must therefore be able to *leverage heterogeneous historical observations* effectively while operating under real-time constraints in continuously evolving scenes.

Trajectory prediction has been extensively studied [6, 27, 34, 50, 55], with state-of-the-art methods achieving high accuracy on large-scale datasets [2, 4, 11, 44]. However, these benchmarks consider only fixed-size historical and future windows, while in practice, the historical context can range from a few frames up to several seconds. Methods that rely on extensive contexts typically achieve the most accurate results, but must delay predictions until a sufficient amount of observations are accumulated for newly detected agents. Only a few works, *e.g.* [22, 46], evaluate varying observation lengths, usually by partially masking agent histories. To handle the constantly evolving context of real-world traffic scenes, however, we require models that can efficiently propagate motion features as long as the agents are visible. This challenge has not yet been sufficiently addressed.

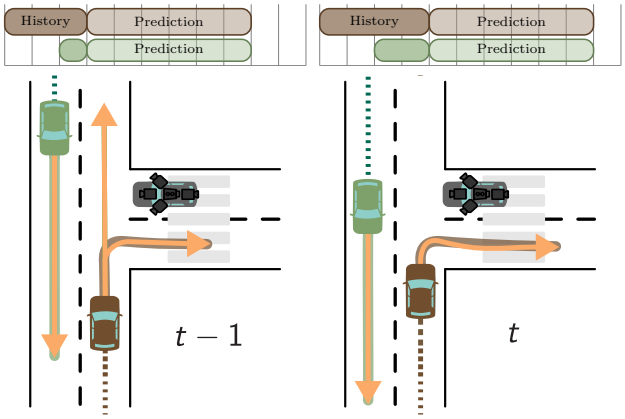



Figure 1. In real-world driving scenes, the available context history for different agents is heterogeneous. Therefore, forecasting models should be able to make accurate predictions for both long-term (brown car) and short-term (green car) contexts, where agents have been observed for a long time or have recently entered the field of view. While existing models often struggle with such varying context lengths, our SHARP is explicitly designed to provide accurate forecasts in such dynamically evolving scenes. Dotted lines indicate the available observations, while transparent paths show the corresponding future trajectory.

To enable reliable motion forecasting under heterogeneous observation lengths, we propose a novel trajectory prediction framework and training scheme. Our model,

SHARP (*short-window streaming for accurate and robust predictions)*, operates in a streaming mode and processes short observation windows while effectively propagating context information across multiple time steps. We introduce a new instance-aware context streaming module and jointly optimize for both long-context and single-chunk prediction.

Our SHARP employs an efficient transformer-based architecture that demonstrates strong robustness across varying context lengths on the Argoverse 2 (AV2) [44] dataset, which features long and complex driving scenarios. We conduct a comprehensive evaluation using different context lengths to assess performance across diverse temporal settings. Additional experiments on Argoverse 1 (AV1) [4] and nuScenes [2] further demonstrate the generality and effectiveness of our method. Compared to prior streaming methods, *e.g*. [36, 50], which relay information but remain constrained to fixed-size training inputs, our design generalizes more effectively to varying lengths. In contrast to models trained on variable histories, *e.g*. [46], SHARP incorporates the gradual evolution of traffic scenes which leads to favorable performance under varying observation lengths. Our method matches state-of-the-art performance in the standard single-agent evaluation setting and establishes a new state-of-the-art for streaming multi-agent prediction. In summary, our main contributions are:

- We propose SHARP, a novel trajectory forecasting approach, effectively predicting accurate trajectories on various observation lengths in continuously evolving scenes, tackling key challenges in real-world autonomous driving.
- Extensive evaluations on AV2, nuScenes, and AV1 show that our method excels across various datasets providing a lightweight yet effective motion forecasting model.
- We demonstrate robustness across a wide range of variable observation lengths, a highly relevant evaluation that should be thoroughly addressed by our community.

## 2. Related Work

State-of-the-art trajectory prediction models typically encode historical trajectories of agents and map information as a first processing step. While historical agent tracks are commonly encoded using temporal self-attention [6, 21, 30, 36] or state-space models [18, 50], map information is usually represented in vectorized form [15] and encoded via PointNet-like architectures [32]. To model relationships between all scene elements, individual encodings are combined either with attention blocks [14, 24, 27, 28, 34, 35, 52] or graph neural networks [8, 15, 20, 37, 55]. DETR-based architectures [3] have shown strong performance [30, 34, 55] for decoding multi-modal future trajectories. Recent advances in trajectory prediction focus on improving various aspects, like scene modeling [55], refinement decoders [53], or improved model architectures [18, 50]. While existing methods achieve high accuracy on benchmark datasets [2, 4, 11, 44], they largely rely on fixed-size inputs and overlook real-world challenges such as varying observation lengths and continuous operation in evolving scenes. Our approach explicitly addresses these aspects by emphasizing flexible input sizes, efficiency, and temporal consistency.

**Trajectory Prediction with Varying Observation Lengths.** *FlexiLength Network (FLN)* [46] presents a method for handling observation lengths shifts using two components: *FlexiLength Calibration (FLC)*, enabling an observation length invariant feature representation, and *FlexiLength Adaptation (FLA)*, which then adapts to specific observation lengths. POP [42] uses self-supervised learning for mask history reconstruction as a pretext task to handle dynamic observation lengths. Feature distillation is then used to transfer knowledge from a fully observed teacher model to a student model with limited observations. LaKD [22] introduces a dynamic knowledge distillation method where a single encoder, acting as both teacher and student, adaptively transfers knowledge across trajectories of different lengths. It also employs a dynamic soft-masking mechanism to mitigate knowledge conflicts during this self-distillation process.

Processing sequences of varying input lengths offers greater flexibility compared to models trained on fixed-size inputs. In real-world driving, however, models must handle streams of observations that gradually extend over time. While existing approaches perform well across different sequence lengths, they lack information-streaming mechanisms, resulting in consecutive predictions that are independent. We address this limitation by processing sequences in chunks, mirroring the progressive nature of real-world inputs, while introducing a streaming mechanism that promotes temporal continuity and captures long-term dependencies.

**Streaming Trajectory Prediction.** RealMotion [36] introduces two information streaming mechanisms for motion forecasting in continuous settings. Unlike traditional snapshot-based methods [18, 34, 53], which predict each frame independently, these mechanisms leverage information across multiple observation windows, enabling more consistent predictions. Specifically, a cross-attention-based scene context streamer integrates context of the previous step into the current scene. A motion-aware layer normalization (MLN) [41] compensates the coordinate frame displacement. In addition, a trajectory relay mechanism incorporates past predictions into the current inference. DeMo [50] adopts the aforementioned streaming mechanisms, achieving state-of-the-art results on the Argoverse 2 dataset.

While RealMotion provides a promising framework for continuous motion forecasting, it has several limitations. First, its end-to-end training scheme is not well suited to variable sequence lengths, as the models are optimized on a fixed number of streaming passes, limiting flexibility across different streaming steps. Second, each inference pass uses a large 3 s observation window, which introduces delays for

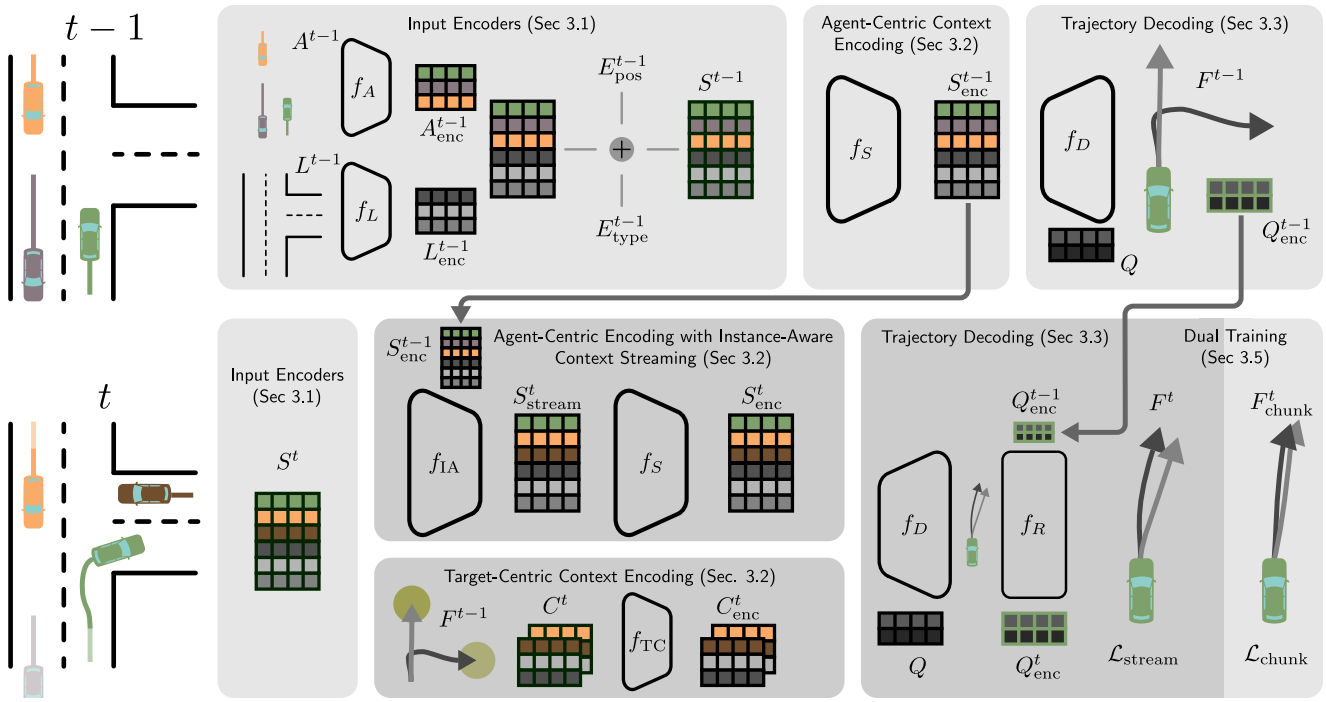



Figure 2. To jointly incorporate newly detected agents and long-term agent histories when forecasting motions in evolving scenes, we leverage a streaming-based motion forecasting model. The example at time step $t-1$ illustrates the standard model pass without streaming context, consisting of separate agent and lane encoders ($f_A$ and $f_L$), a scene encoder ($f_S$), and a trajectory decoder ($f_D$) with a streaming refinement module ($f_R$). At the next time step $t$, we integrate the previous scene context $S^{t-1}_{\text{enc}}$ via an instance-aware context streamer ($f_{\text{IA}}$) and generate auxiliary target-centric features $C^t$ by aggregating scene elements closely around the endpoints of the previous predictions $F^{t-1}$. To improve robustness, we also perform, only during training, a parallel model pass without streaming modules, producing $F^t_{\text{chunk}}$ which is used to compute the $\mathcal{L}_{\text{chunk}}$ objective.

newly detected agents. Finally, although the context relay mechanism is promising, it does not explicitly model instance correspondences during context streaming and therefore captures long-term relationships only implicitly. In contrast, our approach maintains consistent agent encodings across context updates, employs short input windows, and is specifically trained to handle varying observation lengths.

## 3. SHARP Motion Forecasting

We propose a novel trajectory prediction architecture and training scheme to address three key challenges: (1) achieving accurate predictions from short-horizon observation histories, (2) enabling streaming information propagation for operation in continuously evolving scenes, and (3) maintaining a lightweight architecture suitable for real-time inference. Our approach builds on an efficient transformer-based backbone and avoids extra bells and whistles, such as iterative refinement during decoding [34, 53, 55], which typically provide only marginal performance improvements while adding substantial latency. Fig. 2 provides an overview of our architecture. We first process historical agent data and map information to construct a scene representation (Sec. 3.1). As new agent observations arrive, our model incrementally updates this representation by fusing the incoming states with the previous scene context. To propagate information reliably over time, we introduce an instance-aware context streamer (Sec. 3.2) that explicitly models agent correspondences. We further enhance the context representation using the endpoints from previous predictions to extract target-centric auxiliary features. A DETR-like decoder (Sec. 3.3) is utilized to predict the trajectories, which can be easily extended for consistent predictions of all agents in the scene (Sec. 3.4). During training (Sec. 3.5), we jointly optimize the model to predict trajectories from both the streamed scene context and the single observation context, promoting more robust representations across varying observation lengths.

**Problem Definition.** In autonomous driving, motion forecasting aims to provide estimates $F$ for an agent's future trajectories $F$ alongside their associated probability scores $P$, conditioned on the historical states $A$ of all observed agents and the surrounding map information $L$. Following common practice [6, 30], we sample all agents and map elements within a fixed-radius region centered on the focal agent. The agent states are represented as a tensor $A \in \mathbb{R}^{N_a \times T_h \times D_a}$, where $N_a$ denotes the number of agents, $T_h$ the number of state observations and $D_a$ the feature dimension, *i.e.* histori-

cal positions and headings. Similarly, the map is represented as a tensor $L \in \mathbb{R}^{N_l \times P_l \times D_l}$, where $N_l$ is the number of lane segments, $P_l$ the number of sampled points per segment, and $D_l$ the feature dimension (typically the $xy$-coordinates). Segments are defined by uniformly sampling $P_l$ points along the lane centerline.

The trajectory prediction output consists of multi-modal future trajectories $F \in \mathbb{R}^{K \times T_f \times D_f}$ and their corresponding probability scores $P \in \mathbb{R}^K$, where $K$ is the number of motion modes (hypotheses), $T_f$ the prediction horizon (number of future time steps), and $D_f$ is the feature dimension (typically, $D_f = 2$, *i.e.* $xy$-coordinates). At a given time step $t$, the non-streaming (*snapshot-based*) trajectory prediction process can be expressed as $(F^t, P^t) = f(A^t, L^t)$, where $f$ denotes the trajectory prediction model including an encoder $f_E$ and a decoder $f_D$. In our approach, the encoder $f_E$ creates a standard agent-centric scene context $S^t_{\text{enc}}$ and auxiliary target-centric features $C^t_{\text{enc}}$ (details in Section 3.2). The decoder $f_D$ then predicts the output using both contexts: $(F^t, P^t) = f_D(S^t_{\text{enc}}, C^t_{\text{enc}})$. To address the evolving nature of real-world scenarios, we extend our model to a streaming formulation. The total available context length is denoted by $T_{\text{cl}}$, which the model processes through multiple streaming passes using an observation window of size $T_h$. We leverage the temporal continuity between consecutive passes by incorporating the previous scene context $S^{t-1}_{\text{enc}}$ and prior predictions $F^{t-1}$ into the current prediction step. This leads to the updated process $(F^t, P^t) = f(A^t, L^t, S^{t-1}_{\text{enc}}, F^{t-1})$, where $S^{t-1}_{\text{enc}}$ is integrated during encoding and $F^{t-1}$ is fused at the final stage during decoding.

### 3.1. Agent and Lane Encoding

We first encode all agent motions and lane geometries using element-wise local coordinates [6]. This improves learning efficiency, as motion patterns and lane shapes are modeled independently of their global positions. Agent states are normalized w.r.t. each agent's most recent pose, while lane segments are expressed relative to their polyline center pose. For agent encoding, we employ a compact self-attention-based encoder $f_A$ to process the historical agent states $A^t$ [21, 30]. The agent states are first projected into a $D$-dimensional feature space via a linear layer. Subsequently, multi-head self-attention blocks [38] capture agent motion dynamics, followed by a max-pooling operation across the temporal dimension. The resulting agent embeddings are given by $A^t_{\text{enc}} = f_A(A^t) \in \mathbb{R}^{N_a \times D}$. To encode the lane information $L^t$, we adopt a PointNet-based [32] lane encoder $f_L$, following [6, 36, 50]. This results in a lane context representation $L^t_{\text{enc}} = f_L(L^t) \in \mathbb{R}^{N_l \times D}$.

### 3.2. Streaming Scene Context Encoding

Following the individual encoding of scene elements in local coordinates, we concatenate the agent $A^t_{\text{enc}}$ and lane $L^t_{\text{enc}}$ tokens to form a scene representation $\mathbb{R}^{(N_a+N_l)\times D}$. We then establish a global scene representation relative to the focal agent's pose by augmenting each token with global positional embeddings derived from the current pose of each agent and the center pose of each lane segment [6]. Each pose is defined by its position $(x, y)$ and heading/rotation $(\gamma)$, which we encode as $(x, y, \sin\gamma, \cos\gamma)$. These pose features are processed by a two-layer multilayer perceptron (MLP) to produce positional embeddings $E^t_{\text{pos}} \in \mathbb{R}^{(N_a+N_l)\times D}$. Additionally, we include a type embedding $E^t_{\text{type}} \in \mathbb{R}^{(N_a+N_l)\times D}$ for each agent and lane token to capture categorical information such as the agent class or lane type. This yields a scene representation $S^t \in \mathbb{R}^{(N_a+N_l)\times D}$.

**Instance-Aware Context Streaming.** Our model is designed to operate in a streaming setting, where scene information evolves continuously over time. At each time step $t$, the motion of all agents and the nearby map information for the current observation window is encoded in the scene representation $S^t$. To enable temporal information propagation, we incorporate the past scene context $S^{t-1}_{\text{enc}}$ into the current representation $S^t$. Particularly, we introduce a cross–attention–based context streaming module $f_{\text{IA}}$ that explicitly leverages the temporal correspondences of an agent's encoding throughout its trajectory. Given the instance information inherently provided by the input trajectory, we implement instance-aware context streaming by leveraging an attention mask. This mask selectively modulates cross-attention weights, steering focus toward matching instances via a learned bias parameter. In contrast, prior context-streaming approaches [36, 50] rely solely on positional correspondences, modeling instance relations only implicitly within the streaming module despite their explicit use in constructing the agent encoder inputs $A^t$. Our attention mask-based design enables the model to incorporate these explicit correspondences while preserving spatial relational reasoning. Moreover, unlike conventional methods that assume perfect agent associations over extended context windows $T_{\text{cl}}$, our short-window design intrinsically enables recovery from tracking discontinuities. The context streaming mechanism generates an enhanced scene context $S^t_{\text{stream}}$, which effectively integrates and leverages information from preceding observation windows.

**Agent-Centric Context Encoding.** After integrating the previous context $S^{t-1}_{\text{enc}}$, we use a self-attention-based scene encoder $f_S$ [6, 21] to capture relationships between scene elements, including agent–agent interactions, map topology, and agent–map interactions. This yields a learned scene context $S^t_{\text{enc}} = f_S(S^t_{\text{stream}}) \in \mathbb{R}^{(N_a+N_l)\times D}$, which encodes not only the individual observations but also the relations learned among all elements.

**Target-Centric Context Encoding.** In addition to the standard agent-centric context, we incorporate a set of target-centric features [31, 40, 53] for enhanced performance in our

streaming setting. Specifically, we leverage the $K$ trajectory endpoints from the previous inference step, which represent map regions of potentially high relevance, as anchors to extract additional contextual features $C^t$ [31]. Each $C^t$ is obtained by aggregating all tokens within a compact region of interest centered at the corresponding endpoint. The target-centric context encoder $f_{\text{TC}}$ adopts the same architecture as the agent-centric encoder $f_S$, while the reduced token set ensures low latency overhead. We define the endpoints as the origin of their respective local coordinate systems to model the target-centric features effectively. To maintain global geometric relationships, we further incorporate a positional embedding that explicitly encodes the spatial relationship between the focal agent and each endpoint. The result is given as $C^t_{\text{enc}} \in \mathbb{R}^{K\times(N_a+N_l)'\times D}$ where $(N_a + N_l)'$ is the maximum number of tokens in the reduced sets.

### 3.3. Trajectory Decoding

We employ a DETR-like [3] decoder to generate $K$ multi-modal future trajectories $F^t$ along with their corresponding probability scores $P^t$. Our decoder $f_D$ follows the standard procedure, applying cross-attention between learnable mode queries $Q \in \mathbb{R}^{K\times D}$ and the scene features $S^t_{\text{enc}}$ and $C^t_{\text{enc}}$. To capture both global and target-specific context, we use separate cross-attention blocks for agent-centric and target-centric features. Finally, two shallow MLP heads decode the updated queries $Q^t_{\text{enc}} = f_{\text{D}}\left(S^t_{\text{enc}}, C^t_{\text{enc}}, Q\right)$ into the future trajectories and their associated probabilities.

**Trajectory Relay.** To enhance prediction consistency across streaming inference steps, we adopt a trajectory relay mechanism [36]. Previous forecasts are transformed into the current coordinate frame and used to refine the current predictions via a cross-attention module $f_R$. This design enables the model to refine its outputs based on past information without being constrained by previous predictions, for example, when new observations render earlier forecasts unlikely.

### 3.4. Extension to Scene-wide Joint Predictions

To obtain scene-consistent multi-agent predictions (joint predictions), we combine single-agent predictions (marginal predictions) by leveraging the encoded mode queries $Q^t_{\text{enc}}$. Initially, we execute marginal predictions for each relevant agent in the traffic scene to obtain the individual mode queries $Q^t_{\text{enc}}$. Subsequently, we augment these queries with positional embeddings of the agents relative to a scene-global coordinate system, *i.e.* relative to a reference agent or the ego vehicle. We then apply an interaction block, consisting of cross-attention across all agents within the scenario and cross-attention across all modes per agent. This explicitly models the spatial relationships and interactions between agents—considering multiple possible trajectories for each—and yields $K$ joint world predictions for the motion of all agents. The final trajectories are decoded using the same head structure as employed for the single-agent case.

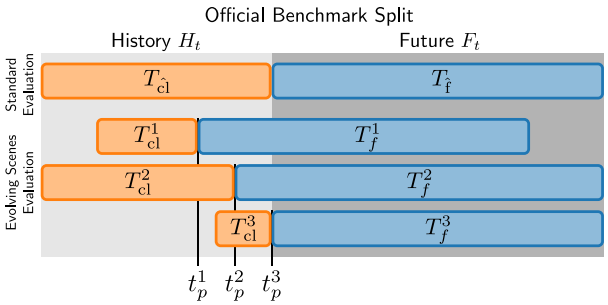



Figure 3. Comparison of different evaluation setups on a motion forecasting dataset with $H_t$ as historical context and $F_t$ as future for the prediction task. The standard benchmark evaluation only considers a single history/future split per scenario (first row). In our experiments, we test the models at different time steps $t^i_p$ into the scenario and using varying context lengths $T^i_{\text{cl}}$ as model input and also evaluating for different future horizons $T^i_f$.

### 3.5. Dual Training

We propose a dual training loss to enhance robustness across varying observation windows, rather than optimizing solely for a fixed context length. In contrast to prior streaming approaches [36], we reduce each streaming step to a short observation window (*e.g.* 1 s). This is aligned with the concept of gradually acquiring new observations over time. Additionally, this design enables more frequent gradient updates within each training scenario and supports longer prediction horizons, as only a short history is required for the initial training pass. During training, we employ non-overlapping windowing to balance input diversity and efficiency. For every new observation chunk, the model predicts future trajectories both with and without streaming context. We compute separate losses for each case: $\mathcal{L}_{\text{stream}}$ for predictions $F^t$ conditioned on the accumulated streaming history, and $\mathcal{L}_{\text{chunk}}$ for predictions $F^t_{\text{chunk}}$ based solely on the current observation chunk. Following prior work [6, 30, 36, 50], each term combines a cross-entropy classification loss to assign the highest probability to the best-fitting trajectory and a Smooth L1 regression loss [19] to fit the predicted trajectories to the ground truth, under a winner-takes-all strategy where only the trajectory with the smallest displacement error contributes to the gradient. The final objective is given by $\mathcal{L}_{\text{dual}} = \mathcal{L}_{\text{stream}} + \mathcal{L}_{\text{chunk}}$.

## 4. Experiments

We evaluate our approach using three large-scale autonomous driving datasets: Argoverse 1 (AV1) [4], Argoverse 2 (AV2) [44] (single- and multi-agent settings), and nuScenes [2]. Additional model ablations, robustness evaluations, as well as full implementation details and training protocol are provided in the supplementary material.

### 4.1. Experimental Settings

**Datasets.** Both Argoverse datasets [4, 44] were collected in the United States and recorded at 10 Hz. AV1 [4] includes 324k scenarios, each 5 s long, with $H_t$=2 s of historical context and a $F_t$=3 s prediction horizon. AV2 [44] comprises 250k scenarios with longer sequences of 11 s (5 s history, 6 s future). The nuScenes prediction dataset [2] is derived from 850 scenes of the nuScenes dataset and consists of 50k scenarios sampled at only 2 Hz, each featuring 2 s of history and 6 s of future motion. AV1 and nuScenes adopt a *single-agent evaluation* protocol, focusing on the prediction of *a designated focal agent per scenario*. In contrast, AV2 additionally supports a *multi-agent evaluation* setting, where predictions for *multiple agents within a scene are jointly assessed*, which closely reflects the requirements of real-world applications. Given its longer temporal span, higher frame rate, and multi-agent setting, AV2 serves as our primary benchmark for streaming-based evaluation. We further employ AV1 and nuScenes to ensure comparability with prior work and to conduct additional ablation studies.

**Metrics.** We evaluate our methods following the standard benchmark protocols using the miss rate ($\text{MR}_k$), minimum average displacement error ($\text{minADE}_k$), minimum final displacement error ($\text{minFDE}_k$), and brier-$\text{minFDE}_6$ as metrics. For each metric, the $k$ highest scoring trajectory hypotheses are considered and the minimum errors are reported based on the best-fitting trajectory. The brier-$\text{minFDE}_6$ also includes a penalty for assessing the probability score of the best fitting trajectory. For the AV2 multi-agent setting, we use the standard metrics which evaluate different *worlds* considering the predictions for each agent in a scene, namely $\text{actorMR}_k$, $\text{avgMinADE}_k$, $\text{avgMinFDE}_k$ and $\text{avgBrierMinFDE}_k$.

**Evolving Scenes Evaluation Setup.** In standard trajectory prediction benchmarks, models are evaluated using fixed input and output windows: a historical context of length $H_t$ and ground-truth future $F_t$. To obtain a more comprehensive and realistic assessment, we introduce an evaluation protocol that emulates continuously evolving scenes with heterogeneous observation lengths (see Fig. 3). Within this framework, streaming-based methods are evaluated on varying context lengths $T_{\text{cl}}$ at different inference time steps $t_p$ which also affect the available prediction horizons $T_f$. The conventional evaluation protocol is recovered when $T_{\text{cl}}$= $H_t$, $t_p$= $H_t$, and $T_f$= $F_t$.

### 4.2. Evolving Scene Evaluation

Table 1 provides a detailed evaluation for streaming trajectory prediction on the AV2 validation set. The results highlight that our model excels across different context lengths ($T_{\text{cl}}$) and prediction timesteps ($t_p$), providing accurate prediction on varying number of streaming steps. Despite achieving strong results on the standard benchmark, related work on streaming trajectory prediction [36, 50] does not generalize across different setups: Their standard setup utilizes three prediction passes covering an overall context length of $T_{\text{cl}}$ = 5 s. When run using different numbers of streaming steps, as in the continuous setting of a deployed automated vehicle, their forecasting performance degrades notably. These experiments demonstrate that our model design and training scheme lead to robust and accurate results, despite highly varying context lengths.

### 4.3. Argoverse 2 Results

Table 2 presents results on the AV2 multi-agent test set, comparing our method to the current state-of-the-art. Our SHARP achieves strong performance, substantially outperforming prior streaming- and snapshot-based methods. This demonstrates that our design—accurate marginal predictions combined with a lightweight fusion module for joint reasoning—is effective for multi-agent trajectory forecasting. Additionally, Table 3 demonstrates our approach on the AV2 single-agent test set. Here again, our method achieves highly accurate results. In particular, despite being able to handle flexible context lengths, our trajectory predictions are within only a few centimeters of the state-of-the-art. However, the correct predictions are more confident as demonstrated by our favorable Brier score.

### 4.4. Ablation Study

We conduct an ablation study in Table 4, where the baseline performance is obtained with standard information streaming [36]. Incorporating our target-centric features, refined via context from previously predicted trajectory endpoints, improves performance in both context settings. Our instance-aware streaming mechanism is especially favorable for longer historical contexts. However, using only the instance-aware streaming mechanism biases the model toward long-term dependencies, resulting in reduced performance for short inputs. Adding also our dual training scheme improves the performance for short contexts notably (*e.g.* $\text{mFDE}_6$ -1 m on AV2), as it encourages the model to learn a more robust temporal representation and to generalize across varying context lengths. This added robustness comes at a negligible accuracy decrease (*e.g.* $\text{mFDE}_6$ +0.01 m on AV2) when evaluated with long input contexts.

### 4.5. nuScenes Results

The short context length of nuScenes (2 sec at only 2 Hz) makes it a particularly challenging dataset, which is commonly not used for testing streaming-based motion forecasting at all. However, our SHARP is designed to handle varying context lengths and therefore performs well even under such constrained contexts, as shown in Tab. 5.

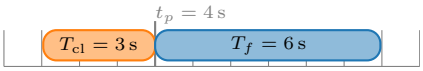
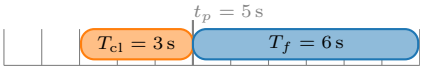
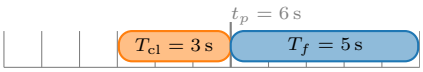
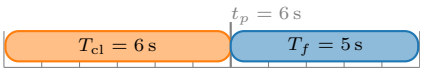
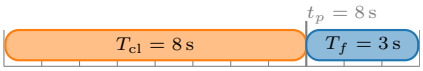
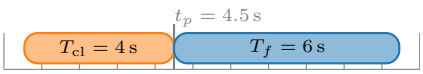
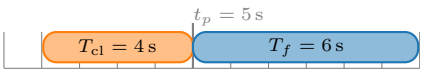
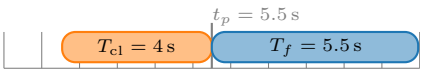
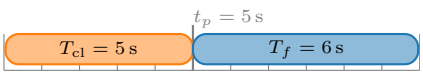
| History / Prediction Time Step / Future Horizon | Method | $\text{minADE}_1$ | $\text{minFDE}_1$ | $\text{MR}_6$ | $\text{minADE}_6$ | $\text{minFDE}_6$ | **brier-minFDE$_6$** |
|---|---|---|---|---|---|---|---|
|  | RealMotion [36] | 1.70 | 4.44 | 0.17 | 0.68 | 1.35 | 2.00 |
| | DeMo [50] | 1.58 | 4.21 | 0.16 | **0.62** | 1.26 | 1.95 |
| | SHARP (Ours) | **1.54** | **3.84** | **0.13** | 0.63 | **1.18** | **1.80** |
|  | RealMotion [36] | 2.04 | 5.33 | 0.23 | 0.79 | 1.65 | 2.32 |
| | DeMo [50] | 1.92 | 5.16 | 0.25 | 0.79 | 1.78 | 2.48 |
| | SHARP (Ours) | **1.84** | **4.66** | **0.19** | **0.73** | **1.44** | **2.09** |
|  | RealMotion [36] | 1.81 | 4.81 | 0.22 | 0.72 | 1.56 | 2.25 |
| | DeMo [50] | 1.70 | 4.66 | 0.25 | 0.74 | 1.75 | 2.45 |
| | SHARP (Ours) | **1.58** | **4.07** | **0.18** | **0.66** | **1.36** | **2.02** |
|  | RealMotion [36] | 1.37 | 3.46 | 0.10 | 0.56 | 1.05 | 1.70 |
| | DeMo [50] | 1.23 | 3.15 | 0.10 | 0.52 | 0.98 | 1.66 |
| | SHARP (Ours) | **1.21** | **2.93** | **0.09** | **0.51** | **0.94** | **1.56** |
|  | RealMotion [36] | 1.19 | 3.05 | 0.09 | 0.48 | 0.91 | 1.58 |
| | DeMo [50] | 1.07 | 2.77 | 0.11 | 0.48 | 0.93 | 1.62 |
| | SHARP (Ours) | **0.89** | **2.10** | **0.05** | **0.39** | **0.70** | **1.31** |
|  | RealMotion [36] | 0.94 | 2.36 | 0.07 | 0.39 | 0.70 | 1.39 |
| | DeMo [50] | 0.84 | 2.15 | 0.10 | 0.41 | 0.78 | 1.48 |
| | SHARP (Ours) | **0.60** | **1.36** | **0.02** | **0.28** | **0.48** | **1.08** |
|  | RealMotion [36] | 1.65 | 4.31 | 0.17 | 0.67 | 1.34 | 1.99 |
| | DeMo [50] | **1.54** | 4.11 | 0.15 | **0.61** | 1.27 | 1.95 |
| | SHARP (Ours) | 1.59 | **4.08** | **0.14** | 0.64 | **1.23** | **1.87** |
|  | RealMotion [36] | 1.77 | 4.51 | 0.18 | 0.71 | 1.39 | 2.04 |
| | DeMo [50] | **1.68** | 4.33 | 0.18 | 0.68 | 1.38 | 2.06 |
| | SHARP (Ours) | 1.69 | **4.16** | **0.15** | **0.67** | **1.25** | **1.89** |
|  | RealMotion [36] | 1.70 | 4.37 | 0.17 | 0.67 | 1.34 | 2.00 |
| | DeMo [50] | 1.65 | 4.29 | 0.19 | 0.68 | 1.44 | 2.13 |
| | SHARP (Ours) | **1.60** | **3.98** | **0.15** | **0.60** | **1.21** | **1.86** |
|  | RealMotion [36] | 1.65 | 4.10 | 0.16 | 0.67 | 1.30 | 1.94 |
| | DeMo [50] | **1.48** | **3.73** | **0.13** | **0.61** | **1.19** | 1.86 |
| | SHARP (Ours) | 1.57 | 3.85 | 0.14 | 0.64 | 1.20 | **1.82** |

Table 1. Evaluation of streaming-based methods on the AV2 validation set at **varying prediction time steps $t_p$ and context lengths $T_{\text{cl}}$**. To ensure a fair comparison, we test related work [36, 50] using their standard streaming configuration (3 s input windows with 1 s displacement) and only adjust the number of inference passes to vary $T_{\text{cl}}$. The top block evaluates short observation lengths, while the second block extends the context to achieve streaming processing beyond the standard 3 iterative steps of [36, 50]. The third block tests streaming with $T_{\text{cl}} = 4$ s at different prediction times. The bottom block corresponds to the standard AV2 evaluation. For predictions at $t_p > 5$ s, the evaluation is truncated as only 11 s are available. Further details on the experiment setup are provided in the supplementary material.

| Method | Streaming | $\text{avgMinADE}_1$ | $\text{avgMinFDE}_1$ | $\text{actorMR}_6$ | $\text{avgMinADE}_6$ | $\text{avgMinFDE}_6$ | **avgBrierMinFDE$_6$** |
|---|---|---|---|---|---|---|---|
| FJMP [33] | ✗ | 1.52 | 4.00 | 0.23 | 0.81 | 1.89 | 2.59 |
| Forecast-MAE [6] | ✗ | 1.30 | 3.33 | 0.19 | 0.69 | 1.55 | 2.24 |
| SRefiner [45] | ✗ | 1.48 | 3.85 | 0.19 | 0.68 | 1.52 | 2.21 |
| RealMotion [36] | ✓ | 1.14 | 2.87 | 0.18 | 0.62 | 1.32 | 2.01 |
| DeMo [50] | ✓ | 1.12 | 2.78 | 0.16 | 0.58 | 1.24 | 1.93 |
| SHARP (Ours) | ✓ | **1.03** | **2.53** | **0.15** | **0.56** | **1.15** | **1.80** |

Table 2. **Multi-agent benchmark** on the AV2 test set. For all metrics lower values indicate better performance, table sorted in descending order by $\text{avgBrierMinFDE}_6$. For all methods we report results without model ensembling.

### 4.6. Comparison to Varying-Length Methods

Table 6 presents a comparison with recent methods that explicitly target variable observation lengths: FLN [46], POP [42], and TSD [57]. Our approach achieves favorable results across all datasets and input durations, demonstrating better adaptability to diverse temporal contexts. These results highlight that explicitly modeling heterogeneous observation lengths through a progressively extending streaming process not only aligns with real-world dynamics but also delivers significant and consistent performance improvements over existing approaches.

### 4.7. Latency Comparison

Our approach achieves low latency by employing an efficient transformer-based backbone without introducing resource-intensive components, such as refinement steps during de-

| Method | Streaming | $MR_6$ | $mADE_6$ | $mFDE_6$ | **b-$mFDE_6$** |
|---|---|---|---|---|---|
| SIMPL [51] | ✗ | 0.19 | 0.72 | 1.43 | 2.05 |
| HPTR [52] | ✗ | 0.19 | 0.73 | 1.43 | 2.03 |
| Forecast-MAE [6] | ✗ | 0.17 | 0.71 | 1.39 | 2.03 |
| MTR [34] | ✗ | 0.15 | 0.73 | 1.44 | 1.98 |
| EMP-D [30] | ✗ | 0.17 | 0.71 | 1.37 | 1.98 |
| GANet [40] | ✗ | 0.17 | 0.72 | 1.34 | 1.96 |
| ProIn [10] | ✗ | 0.18 | 0.73 | 1.35 | 1.93 |
| QCNet [55] | ✗ | 0.16 | 0.65 | 1.29 | 1.91 |
| RealMotion [36] | ✓ | 0.15 | 0.66 | 1.24 | 1.89 |
| Tamba [18] | ✗ | 0.17 | 0.64 | 1.24 | 1.89 |
| HAMF [26] | ✗ | 0.14 | 0.64 | 1.23 | 1.89 |
| ProphNet [43] | ✗ | 0.18 | 0.68 | 1.33 | 1.88 |
| MTR++ [35] | ✗ | 0.14 | 0.71 | 1.37 | 1.88 |
| DyMap [12] | ✗ | - | 0.71 | 1.29 | 1.87 |
| ModeSeq [56] | ✗ | 0.14 | 0.63 | 1.26 | 1.87 |
| SmartRefine [53] | ✗ | 0.15 | 0.63 | 1.23 | 1.86 |
| MixForecast [39] | ✗ | 0.14 | 0.64 | 1.22 | 1.86 |
| DeMo [50] | ✓ | **0.13** | **0.61** | **1.17** | 1.84 |
| SHARP (Ours) | ✓ | 0.14 | 0.64 | 1.19 | **1.83** |

Table 3. **Single-agent benchmark** on the AV2 test set. For all metrics lower values indicate better, table sorted in descending order by brier-$minFDE_6$. We report results for published works on the official leaderboard without model ensembling.

| Dataset | $T_{cl}$ | TCF | IA | DT | $mADE_6$ | $mFDE_6$ | **b-$mFDE_6$** |
|---|---|---|---|---|---|---|---|
| AV2 Val | 5 s | ✗ | ✗ | ✗ | 0.74 | 1.28 | 1.91 |
| | | ✓ | ✗ | ✗ | 0.64 | 1.22 | 1.84 |
| | | ✓ | ✓ | ✗ | **0.63** | **1.19** | **1.81** |
| | | ✓ | ✓ | ✓ | 0.64 | 1.20 | 1.82 |
| | 1 s | ✗ | ✗ | ✗ | 1.18 | 2.57 | 3.28 |
| | | ✓ | ✗ | ✗ | 1.09 | 2.49 | 3.18 |
| | | ✓ | ✓ | ✗ | 1.11 | 2.55 | 3.25 |
| | | ✓ | ✓ | ✓ | **0.76** | **1.48** | **2.13** |
| AV1 Val | 2s | ✓ | ✗ | ✗ | 0.62 | 0.96 | 1.58 |
| | | ✓ | ✓ | ✗ | **0.59** | **0.91** | **1.52** |
| | | ✓ | ✓ | ✓ | **0.59** | **0.91** | 1.53 |
| | 0.5s | ✓ | ✗ | ✗ | 0.75 | 1.22 | 1.86 |
| | | ✓ | ✓ | ✗ | 0.71 | 1.13 | 1.78 |
| | | ✓ | ✓ | ✓ | **0.64** | **1.00** | **1.61** |

Table 4. **Ablation study** on the impact of our target-centric features (TCF), Dual Training (DT) and Instance-Aware (IA) Context Streaming using varying context lengths ($T_{cl}$). For each dataset, we compare full-history and partial-history settings. All predictions adhere to the standard evaluation protocol: $t_p = 2\,s$, $T_f = 3\,s$ for AV1 and $t_p = 5\,s$, $T_f = 6\,s$ for AV2.

coding. Since available codebases restrict comparison to the AV2 single-agent setting, we report results accordingly. On a single NVIDIA RTX 3090 GPU, our method achieves an online latency of 33 ms for predicting batches of 16 agents, compared to 35 ms for DeMo [50] and 44 ms for RealMotion [36]; 35 ms (*vs.* 49 ms and 79 ms) for 32 agents; and 57 ms (*vs.* 88 ms and 140 ms) for 64 agents. In the AV2 multi-agent setting, our approach achieves an average online latency of 30 ms per scenario on the same GPU.

| Method | Streaming | $mFDE_1$ | $MR_5$ | $mADE_5$ |
|---|---|---|---|---|
| THOMAS [16] | ✗ | 6.71 | 0.55 | 1.33 |
| PGP [9] | ✗ | 7.17 | 0.52 | 1.27 |
| DeMo [50] | ✗ | 6.60 | 0.43 | 1.22 |
| MacFormer [13] | ✗ | 7.50 | 0.57 | 1.21 |
| LAFormer [23] | ✗ | 6.95 | 0.48 | 1.19 |
| Q-EANet [5] | ✗ | 6.77 | 0.48 | 1.18 |
| FRM [29] | ✗ | 6.59 | 0.48 | 1.18 |
| CASPFormer [47] | ✗ | 6.70 | 0.48 | 1.15 |
| LMFormer [48] | ✗ | 6.85 | 0.47 | 1.14 |
| Goal-LBP [49] | ✗ | 9.20 | 0.32 | **1.02** |
| SHARP (Ours) | ✓ | **6.02** | **0.28** | 1.13 |

Table 5. Results on the **nuScenes prediction challenge** test set. Following the evaluation protocol, we set the model output to $K = 5$. DeMo [50] reports nuScenes results without streaming.

| Dataset | Input Steps | $T_{cl}$ | Method | $mADE_6$ | $mFDE_6$ |
|---|---|---|---|---|---|
| Argoverse 1 | 5 | 0.5 s | POP-H [42] | 0.74 | 1.10 |
| | | | TSD-H [57] | 0.72 | 1.04 |
| | | | SHARP (Ours) | **0.61** | **0.96** |
| | 10 | 1 s | HiVT-64-FLN [46] | 0.81 | 1.25 |
| | | | POP-H [42] | 0.70 | 1.06 |
| | | | TSD-H [57] | 0.67 | 0.98 |
| | | | SHARP (Ours) | **0.61** | **0.96** |
| | 20 | 2 s | HiVT-64-FLN [46] | 0.72 | 1.08 |
| | | | POP-H [42] | 0.69 | 1.04 |
| | | | TSD-H [57] | 0.65 | 0.94 |
| | | | SHARP (Ours) | **0.59** | **0.91** |
| Argoverse 2 | 10 | 1 s | POP-Q [42] | 2.24 | 4.16 |
| | | | SHARP (Ours) | **0.76** | **1.48** |
| | 30 | 3 s | POP-Q [42] | 0.92 | 1.70 |
| | | | SHARP (Ours) | **0.73** | **1.44** |
| | 50 | 5 s | POP-Q [42] | 0.79 | 1.45 |
| | | | SHARP (Ours) | **0.64** | **1.20** |
| Dataset | Input Steps | $T_{cl}$ | Method | $mADE_5$ | $mFDE_5$ |
| nuScenes | 2 | 0.5 s | AFormer-FLN [46] | 1.92 | 3.91 |
| | | | SHARP (Ours) | **1.18** | **2.01** |
| | 3 | 1.5 s | AFormer-FLN [46] | 1.88 | 3.89 |
| | | | SHARP (Ours) | **1.15** | **2.00** |
| | 4 | 2 s | AFormer-FLN [46] | 1.83 | 3.78 |
| | | | SHARP (Ours) | **1.13** | **1.92** |

Table 6. Comparison to the state-of-the-art **varying-length models** [42, 46, 57], following their evaluation protocol.

## 5. Conclusion

We presented a streaming-based motion forecasting approach that enables continuous and efficient prediction in dynamically evolving scenes. Our SHARP achieves competitive accuracies in the single-agent settings and sets a new state-of-the-art for streaming multi-agent prediction. Extensive experiments on AV2, AV1, and nuScenes demonstrate strong robustness across diverse temporal contexts and scene complexities. By explicitly modeling robust context propagation, our work advances real-time trajectory prediction and highlights the importance of streaming processing.

**Acknowledgments.** This work was partially funded by Addsafety (923936), a COMET Project funded by BMIMI, BMWET and the co-financing federal province of Styria. The COMET programme is managed by the Austrian Research Promotion Agency (FFG).

## References

[1] Görkay Aydemir, Adil Kaan Akan, and Fatma Güney. ADAPT: Efficient Multi-Agent Trajectory Prediction with Adaptation. In *ICCV*, 2023. 13

[2] 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. 1, 2, 5, 6, 14, 15, 16

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020. 2, 5

[4] 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. 1, 2, 5, 6, 12, 14, 15, 16

[5] Jiuyu Chen, Zhongli Wang, Jian Wang, and Baigen Cai. Q-EANet: Implicit Social Modeling for Trajectory Prediction via Experience-Anchored Queries. *IET Intell, Transp. Sys.*, 18(6), 2024. 8

[6] Jie Cheng, Xiaodong Mei, and Ming Liu. Forecast-MAE: Self-supervised Pre-training for Motion Forecasting with Masked Autoencoders. In *ICCV*, 2023. 1, 2, 3, 4, 5, 7, 8, 15, 16

[7] 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. 13

[8] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. GoRela: Go Relative for Viewpoint-Invariant Motion Forecasting. In *ICRA*, 2023. 2

[9] Nachiket Deo, Eric Wolff, and Oscar Beijbom. Multimodal Trajectory Prediction Conditioned on Lane-Graph Traversals. In *CoRL*, 2021. 8

[10] 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. *Neurocomputing*, 2025. 8

[11] 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. 1, 2

[12] 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. 8

[13] Chen Feng, Hangning Zhou, Huadong Lin, Zhigang Zhang, Ziyao Xu, Chi Zhang, Boyu Zhou, and Shaojie Shen. MacFormer: Map-Agent Coupled Transformer for Real-Time and Robust Trajectory Prediction. *RAL*, 2023. 8

[14] 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

[15] 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

[16] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. THOMAS: Trajectory Heatmap Output with learned Multi-Agent Sampling. In *ICLR*, 2022. 8

[17] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv*, 2016. 15

[18] Yizhou Huang, Yihua Cheng, and Kezhi Wang. Trajectory Mamba: Efficient Attention-Mamba Forecasting Model Based on Selective SSM. In *CVPR*, 2025. 2, 8

[19] Peter J Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 1964. 5, 16

[20] 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

[21] 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

[22] Yuhang Li, Changsheng Li, Ruilin Lv, Rongqing Li, Ye Yuan, and Guoren Wang. LaKD: Length-agnostic Knowledge Distillation for Trajectory Prediction with Any Length Observations. *NeurIPS*, 2024. 1, 2

[23] Mengmeng Liu, Hao Cheng, Lin Chen, Hellward Broszio, Jiangtao Li, Runjiang Zhao, Monika Sester, and Michael Ying Yang. LAformer: Trajectory Prediction for Autonomous Driving with Lane-Aware Scene Constraints. In *CVPR*, 2024. 8

[24] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal Motion Prediction with Stacked Transformers. *CVPR*, 2021. 2, 13

[25] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 16

[26] Xiaodong Mei, Sheng Wang, Jie Cheng, Yingbing Chen, and Dan Xu. HAMF: A Hybrid Attention-Mamba Framework for Joint Scene Context Understanding and Future Motion Representation Learning. In *IROS*, 2025. 8

[27] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion Forecasting via Simple & Efficient Attention Networks. In *ICRA*, 2023. 1, 2

[28] 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

[29] Daehee Park, Hobin Ryu, Yunseo Yang, Jegyeong Cho, Jiwon Kim, and Kuk-Jin Yoon. Leveraging Future Relationship Reasoning for Vehicle Trajectory Prediction. In *ICLR*, 2023. 8, 13

[30] 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, 5, 8, 15

[31] Alexander Prutsch, David Schinagl, and Horst Possegger. Streaming Real-Time Trajectory Prediction Using Endpoint-Aware Modeling. In *WACV*, 2026. 4, 5

[32] 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, 15

[33] 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

[34] 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, 8

[35] 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, 8

[36] Nan Song, Bozhou Zhang, Xiatian Zhu, and Li Zhang. Motion Forecasting in Continuous Driving. In *NeurIPS*, 2024. 2, 4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16

[37] 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, 12, 13

[38] 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

[39] Wenkang Wan, Nan Ouyang, Mingjin Zeng, Lei Ao, Qing Cai, Yuan Gao, and Kai Sheng. Multi-Agent Motion Forecasting via Mixed Supervision. *RAL*, 2025. 8

[40] 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. 4, 8

[41] 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. 2

[42] Sheng Wang, Yingbing Chen, Jie Cheng, Xiaodong Mei, Ren Xin, Yongkang Song, and Ming Liu. Improving Autonomous Driving Safety with POP: A Framework for Accurate Partially Observed Trajectory Predictions. In *ICRA*, 2024. 2, 7, 8

[43] Xishun Wang, Tong Su, Fang Da, and Xiaodong Yang. ProphNet: Efficient Agent-Centric Motion Forecasting with Anchor-Informed Proposals. In *CVPR*, 2023. 8

[44] 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, 14, 15, 16

[45] Liwen Xiao, Zhiyu Pan, Zhicheng Wang, Zhiguo Cao, and Wei Li. SRefiner: Soft-Braid Attention for Multi-Agent Trajectory Refinement. In *CVPR*, 2025. 7

[46] Yi Xu and Yun Fu. Adapting to Length Shift: FlexiLength Network for Trajectory Prediction. In *CVPR*, 2024. 1, 2, 7, 8

[47] Harsh Yadav, Maximilian Schaefer, Kun Zhao, and Tobias Meisen. CASPFormer: Trajectory Prediction from BEV Images with Deformable Attention. In *ICPR*, 2024. 8

[48] Harsh Yadav, Maximilian Schaefer, Kun Zhao, and Tobias Meisen. LMFormer: Lane based Motion Prediction Transformer. In *CVPRW*, 2025. 8

[49] Zhen Yao, Xin Li, Bo Lang, and Mooi Choo Chuah. Goal-LBP: Goal-Based Local Behavior Guided Trajectory Prediction for Autonomous Driving. *IEEE Trans. on Intell. Transp. Sys.*, 2023. 8

[50] Bozhou Zhang, Nan Song, and Li Zhang. DeMo: Decoupling Motion Forecasting into Directional Intentions and Dynamic States. In *NeurIPS*, 2024. 1, 2, 4, 5, 6, 7, 8, 12, 13, 14, 15, 16, 18, 19

[51] Lu Zhang, Peiliang Li, Sikang Liu, and Shaojie Shen. SIMPL: A Simple and Efficient Multi-agent Motion Prediction Baseline for Autonomous Driving. *RAL*, 2024. 8, 13

[52] 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, 8

[53] 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. 2, 3, 4, 8

[54] Zikang Zhou, Luyao Ye, Jianping Wang, Kui Wu, and Kejie Lu. HiVT: Hierarchical Vector Transformer for Multi-Agent Motion Prediction. In *CVPR*, 2022. 13

[55] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-Centric Trajectory Prediction. In *CVPR*, 2023. 1, 2, 3, 8, 14, 15

[56] Zikang Zhou, Hengjian Zhou, Haibo Hu, Zihao Wen, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. ModeSeq: Taming Sparse Multimodal Motion Prediction with Sequential Mode Modeling. In *CVPR*, 2025. 8

[57] Pengfei Zhu, Peng Shu, Mengshi Qi, Liang Liu, and Huadong Ma. Target-driven Self-Distillation for Partial Observed Trajectories Forecasting. *arXiv*, 2025. 7, 8

# SHARP: Short-Window Streaming for Accurate and Robust Prediction in Motion Forecasting

## Supplementary Material

In this supplementary, we first present additional insights to highlight the effectiveness and robustness of our approach, which were omitted from the original paper due to the page limit (Section A). To support reproducibility, we provide all implementation details of our architecture in Section B, including training hyperparameters and details on the multi-agent extension. We further detail the evaluation setup (Sec. C) and provide output visualizations (Sec. D). Finally, we provide an overview of the code framework (Sec. E), which is provided at https://github.com/a-pru/sharp.

## A. Additional Insights

### A.1. Observation Length Study

Table 7 provides an ablation study for predictions of our SHARP across different context lengths $T_{\text{cl}}$. The results indicate that our approach provides better prediction performance with increasing input context length, highlighting the advantage of a model that effectively exploits extended historical information – a key motivation for designing SHARP.

| Streaming Setup | mADE$_6$ | mFDE$_6$ | **b-mFDE$_6$** |
|---|---|---|---|
| $t_p = 5\,\text{s}$, $T_{\text{cl}}$, $T_f = 6\,\text{s}$ | 0.76 | 1.48 | 2.13 |
| $t_p = 5\,\text{s}$, $T_{\text{cl}} = 3\,\text{s}$, $T_f = 6\,\text{s}$ | 0.73 | 1.44 | 2.09 |
| $t_p = 5\,\text{s}$, $T_{\text{cl}} = 5\,\text{s}$, $T_f = 6\,\text{s}$ | **0.64** | **1.20** | **1.82** |
| $t_p = 7\,\text{s}$, $T_{\text{cl}} = 3\,\text{s}$, $T_f = 4\,\text{s}$ | 0.55 | 1.11 | 1.77 |
| $t_p = 7\,\text{s}$, $T_{\text{cl}} = 5\,\text{s}$, $T_f = 4\,\text{s}$ | 0.49 | 0.89 | 1.55 |
| $t_p = 7\,\text{s}$, $T_{\text{cl}} = 7\,\text{s}$, $T_f = 4\,\text{s}$ | **0.39** | **0.70** | **1.31** |
| $t_p = 8\,\text{s}$, $T_{\text{cl}}$, $T_f = 3\,\text{s}$ | 0.40 | 0.72 | 1.38 |
| $t_p = 8\,\text{s}$, $T_{\text{cl}} = 4\,\text{s}$, $T_f = 3\,\text{s}$ | 0.41 | 0.75 | 1.42 |
| $t_p = 8\,\text{s}$, $T_{\text{cl}} = 6\,\text{s}$, $T_f = 3\,\text{s}$ | 0.36 | 0.59 | 1.23 |
| $t_p = 8\,\text{s}$, $T_{\text{cl}} = 8\,\text{s}$, $T_f = 3\,\text{s}$ | **0.28** | **0.48** | **1.08** |

Table 7. Evaluation of SHARP on the AV2 validation set performed **across different context lengths** $T_{\text{cl}}$. Within each group, the prediction time step $t_p$ and forecasting horizon $T_f$ are held constant, while the observation length provided to the model is varied. We evaluate predictions at the standard $t_p$= 5 *s*, as well as at $t_p$= 7 *s* and $t_p$= 8 *s*, to study the impact of longer observation lengths on forecasting performance.

### A.2. Robustness to Noise

Table 8 evaluates our instance-aware context streaming under noisy instance masks. We apply two types of perturbations: (1) We randomize mask values, introducing false-positive associations between unrelated instances and removing correct correspondences; and (2) We remove existing associations, simulating ID switches during tracking. The results indicate that our instance-aware context streaming works well, even under perturbed instance masks. This means that our design can effectively leverage explicit instance correspondence information when available, while still falling back on implicit context fusion driven by global agent positions [36].

Additionally, Table 10 evaluates the effect of perturbing prediction endpoints during streaming. This experiment assesses the robustness of our target-centric features to errors in past predictions (*i.e.*, error propagation). We add Gaussian ($\mathcal{N}$) or uniform ($\mathcal{U}$) noise to the endpoint locations before using them to aggregate the target-centric features. While these perturbations cause a slight degradation, our approach effectively uses both agent- and target-centric features, enabling accurate predictions even when previous outputs are imperfect.

| Noise Type | Noise Ratio | mADE$_6$ | mFDE$_6$ | b-mFDE$_6$ |
|---|---|---|---|---|
| Randomized | 80% | 0.65 | 1.23 | 1.85 |
| | 40% | 0.65 | 1.22 | 1.84 |
| | 20% | 0.64 | 1.21 | 1.83 |
| Set to Zero (no match) | 80% | 0.65 | 1.22 | 1.84 |
| | 40% | 0.64 | 1.20 | 1.83 |
| | 20% | 0.64 | 1.20 | 1.82 |
| Clean Data | | 0.63 | 1.19 | 1.81 |

Table 8. We evaluate the **robustness of our instance-aware context streamer under perturbed instance masks**. In the first group, we partially randomize the masks, introducing incorrect correspondences and removing correct ones, resulting in both false positives and false negatives. In the second group, we remove associations by partially setting mask values to zero, producing false negatives while leaving other annotations intact.

### A.3. AV2 Multi-Agent Analysis

**Evolving Scenes.** Analogous to Table 1 in the main paper (*AV2 single-agent setting*), Table 9 analyzes the performance of SHARP across varying context lengths $T_{\text{cl}}$ and prediction time steps $t_p$ in the *AV2 multi-agent setting*. The results demonstrate that our approach remains effective under evolving multi-agent scene conditions, further highlighting the robustness and adaptability of our method.

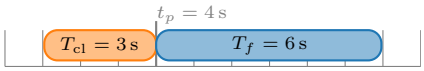
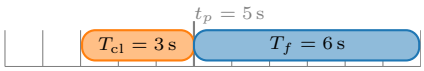
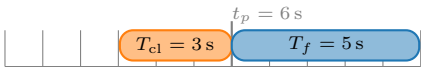
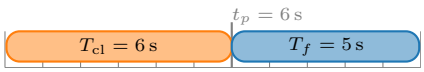
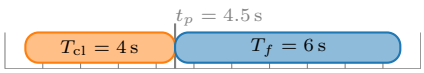
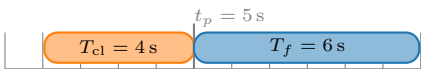
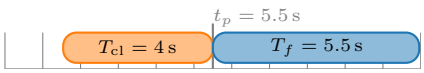
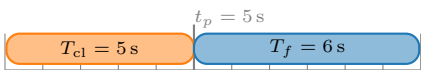
| History / Prediction Time Step / Future Horizon | avgMinADE$_6$ | avgMinFDE$_6$ | actorMR$_6$ | **avgBrierMinFDE$_6$** |
|---|---|---|---|---|
|  | 0.68 | 1.43 | 0.19 | 2.07 |
|  | 0.60 | 1.26 | 0.17 | 1.91 |
|  | 0.54 | 1.17 | 0.15 | 1.83 |
|  | 0.43 | 0.88 | 0.11 | 1.52 |
|  | 0.32 | 0.64 | 0.06 | 1.27 |
|  | 0.23 | 0.43 | 0.03 | 1.06 |
|  | 0.55 | 1.16 | 0.15 | 1.80 |
|  | 0.56 | 1.21 | 0.16 | 1.86 |
|  | 0.46 | 0.97 | 0.12 | 1.62 |
|  | 0.55 | 1.16 | 0.15 | 1.80 |

Table 9. Evaluation of SHARP in the **multi-agent setting** on the AV2 validation set at **varying prediction time steps $t_p$ and context lengths** $T_{\mathrm{cl}}$. The evaluation scope is limited to our method due to missing multi-agent implementation details and code bases for related work [36, 50].

| Endpoint Noise (meter) | mADE$_6$ | mFDE$_6$ | b-mFDE$_6$ |
|---|---|---|---|
| $\mathcal{U}(-5,5)$ | 0.65 | 1.23 | 1.85 |
| $\mathcal{N}(0,3)$ | 0.65 | 1.23 | 1.85 |
| $\mathcal{U}(-3,3)$ | 0.64 | 1.21 | 1.83 |
| $\mathcal{N}(0,1)$ | 0.61 | 1.20 | 1.82 |
| $\mathcal{U}(-1,1)$ | 0.64 | 1.20 | 1.83 |
| Clean Data | 0.63 | 1.19 | 1.81 |

Table 10. We evaluate the **robustness of our target-centric features under perturbations** of the predicted endpoints used for the token selection. Gaussian ($\mathcal{N}$) and uniform ($\mathcal{U}$) noise are applied to simulate degradation of predictions during streaming. This setup assesses how our model handles such errors and its ability to recover from inaccurate prior predictions.

**Scene Consistency.** To validate our approach for the extension to scene-wide joint predictions, we evaluate the benefit of our joint refinement module over a naive combination of single-agent marginals by measuring potential collision rates on the AV2 multi-agent validation set. We define a potential collision as an event where two predicted actors within the same predicted world are closer than 2 m to each other at the same time step. Utilizing our joint refinement module reduces the number of collision events from 5,533 (naive combination of marginals) to 973. For reference, the ground truth trajectories contain zero collisions.

## A.4. AV1 Validation Set

Table 11 compares our approach to the state-of-the-art on the AV1 [4] validation set. Overall, SHARP achieves top performance, closely matching the results of HPNet [37].

## A.5. Short Histories

Table 12 compares our method to DeMo [50] using a 1 s input length $T_{\mathrm{cl}}$ on the AV2 validation set. Our approach outperforms DeMo across all prediction horizons $t_p$, highlighting the effectiveness of our dual training and processing scheme.

| Method | Streaming | $MR_6$ | $mADE_6$ | **$mFDE_6$** |
|---|---|---|---|---|
| mmTransformer [24] | ✗ | 0.11 | 0.71 | 1.15 |
| FRM [29] | ✗ | - | 0.68 | 0.99 |
| ADAPT [1] | ✗ | 0.08 | 0.67 | 0.95 |
| SIMPL [51] | ✗ | 0.08 | 0.66 | 0.95 |
| HiVT [54] | ✗ | 0.09 | 0.66 | 0.96 |
| R-Pred [7] | ✗ | 0.09 | 0.66 | 0.95 |
| DeMo [50] | ✓ | **0.07** | 0.59 | 0.90 |
| SHARP (Ours) | ✓ | **0.07** | **0.58** | 0.90 |
| HPNet [37] | ✗ | **0.07** | 0.64 | **0.87** |

Table 11. Results on the **Argoverse 1 prediction challenge validation set**.

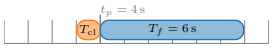
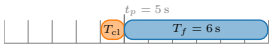
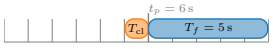
| Streaming Setup | Method | $mADE_6$ | $mFDE_6$ | **b-$mFDE_6$** |
|---|---|---|---|---|
|  | DeMo | 1.02 | 1.86 | 2.54 |
| | SHARP | **0.70** | **1.16** | **1.79** |
|  | DeMo | 0.80 | 1.46 | 2.15 |
| | SHARP | **0.70** | **1.34** | **1.99** |
|  | DeMo | 0.99 | 1.97 | 2.67 |
| | SHARP | **0.76** | **1.48** | **2.13** |
|  | DeMo | 0.95 | 1.96 | 2.67 |
| | SHARP | **0.65** | **1.27** | **1.92** |

Table 12. We compare our SHARP to DeMo [50] using **short context lengths $T_{cl}$ of 1 s** on the AV2 single-agent validation set.

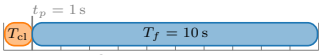
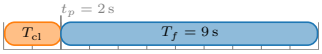
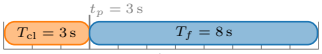
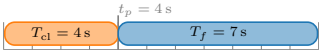
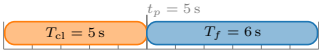
| Streaming Setup | $mADE_6$ | $mFDE_6$ | **b-$mFDE_6$** |
|---|---|---|---|
|  | 1.37 | 2.48 | 3.12 |
|  | 1.12 | 2.06 | 2.70 |
|  | 0.95 | 1.76 | 2.40 |
|  | 0.79 | 1.48 | 2.11 |
|  | **0.64** | **1.20** | **1.82** |

Table 13. Evaluation of SHARP on **extend prediction horizons** using the AV2 single-agent validation set. Predictions of the agent's final position at the end of each scenario are made at varying prediction times $t_p$. Even the initial predictions over long horizons provide reasonable estimates. As the available context $T_{cl}$ increases and the prediction horizon $T_f$ shortens, the forecasts naturally become more accurate.

## A.6. Extended Prediction Horizon

Our short-window processing scheme enables training with extended prediction horizons without requiring additional data. For instance, AV2 provides scenarios of 11 s duration. Within this setting, our model supports a 10 s prediction horizon, compared to the default 6 s horizon supported by baselines. Table 13 reports experiments evaluating our module on long prediction horizons $T_f$.

| $T_{cl}$ | Loss $\mathcal{L}$ Configuration | $mADE_6$ | $mFDE_6$ | **b-$mFDE_6$** |
|---|---|---|---|---|
| 5 s | $2 \cdot \mathcal{L}_{stream} + 1 \cdot \mathcal{L}_{chunk}$ | 0.66 | 1.23 | 1.85 |
| | $1 \cdot \mathcal{L}_{stream} + 2 \cdot \mathcal{L}_{chunk}$ | 0.66 | 1.24 | 1.87 |
| | $1 \cdot \mathcal{L}_{stream} + 1 \cdot \mathcal{L}_{chunk}$ | 0.66 | 1.24 | 1.86 |
| 1 s | $2 \cdot \mathcal{L}_{stream} + 1 \cdot \mathcal{L}_{chunk}$ | 0.97 | 2.14 | 2.82 |
| | $1 \cdot \mathcal{L}_{stream} + 2 \cdot \mathcal{L}_{chunk}$ | 0.87 | 1.84 | 2.52 |
| | $1 \cdot \mathcal{L}_{stream} + 1 \cdot \mathcal{L}_{chunk}$ | 0.94 | 2.03 | 2.70 |

Table 14. We evaluate training SHARP using a **weighted dual loss formulation**, where the weights allow us to emphasize either long-term or short-term context performance. In our final model, we omit these weights, resulting in equal contributions from both loss terms. Note that this analysis is based on a preliminary experiment conducted with a reduced training schedule.

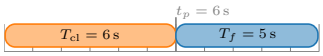
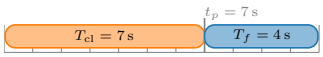
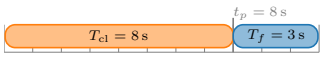
| Streaming Setup | $f_{IA}$ | $mADE_6$ | $mFDE_6$ | **b-$mFDE_6$** |
|---|---|---|---|---|
|  | ✗ | 0.54 | 0.99 | 1.63 |
| | ✓ | **0.51** | **0.94** | **1.56** |
|  | ✗ | 0.47 | 0.84 | 1.50 |
| | ✓ | **0.39** | **0.70** | **1.31** |
|  | ✗ | 0.39 | 0.66 | 1.33 |
| | ✓ | **0.28** | **0.48** | **1.08** |

Table 15. Ablation study on AV2 for long input horizons using our **instance-aware context streaming** $f_{IA}$ module compared to the standard context relay streaming module proposed by [36].

## A.7. Dual Loss Ablation

Table 14 presents results for training our model using weighting parameters to combine the two losses, $\mathcal{L}_{stream}$ and $\mathcal{L}_{chunk}$, in our dual-training formulation. The experiment shows that adjusting the loss weights enables the model to prioritize either short-term context or longer prediction horizons, and further highlights the effectiveness of our loss design. To obtain the best overall trade-off, we use equal weighting between the two losses in our final implementation.

## A.8. Context Streaming Ablation Study

Table 15 compares our instance-aware context streamer against the implicit approach of [36] across extended observation lengths $T_{cl}$. The results demonstrate that explicit instance modeling provides a significant advantage, particularly as the observation duration increases.

## A.9. Model Training Ablation Study

Table 16 analyzes the impact of our instance-aware context streaming across different model setups. All evaluations follow the standard AV2 protocol, using $T_{cl}$ =5 s of historical context and predictions at $t_p$ = 5 s for 6 s into the future. We vary the input observation window size ($T_h$) and the number of backpropagation passes during training. The first row corresponds to the default streaming setup of [36]. Reducing the input window size allows for creating more (non-overlapping) train samples per dataset scenario. However, the baseline architecture degrades due to limited temporal

| $T_h$ | Train Pass at $t_p$ | IA | mADE$_6$ | mFDE$_6$ | **b-mFDE$_6$** |
|---|---|---|---|---|---|
| 3 s | {3, 4, 5} s | ✗ | 0.66 | 1.23 | 1.84 |
| 2 s | 2 .. 5 s | ✗ | 0.66 | 1.24 | 1.85 |
| 1 s | 1 .. 5 s | ✗ | 0.64 | 1.22 | 1.85 |
| 1 s | 1 .. 8 s | ✗ | 0.65 | 1.22 | 1.84 |
| 2 s | 2 .. 5 s | ✓ | 0.66 | 1.22 | 1.83 |
| 1 s | 1 .. 5 s | ✓ | 0.65 | 1.21 | 1.83 |
| 1 s | 1 .. 8 s | ✓ | **0.63** | **1.19** | **1.81** |

Table 16. Ablation study on **training** our approach with varying input window sizes ($T_h$) and different numbers of back-propagation passes per scenario, with and without our instance-aware context streaming (IA). The displacement during consecutive train passes is 1 s for all experiments. The first row reports training results for our SHARP architecture using the baseline streaming setup of [36]. Reducing the input window size $T_h$ enables performing more back-propagation passes over each scenario. However, existing streaming mechanisms fail to fully exploit this potential, which is effectively unlocked by our instance-aware context streamer. All evaluations follow the standard protocol on the AV2 validation set. Experiments are conducted without dual training to neglect confounding influences and isolate the effect of instance-aware context streaming.

information per pass and restricted inter-frame propagation through the implicit agent-relation modeling. In contrast, our instance-aware context streaming significantly improves performance by preserving temporal coherence across streaming steps. The best results are achieved when training with eight model passes, which increases sample diversity and enables more effective learning of the streaming mechanism.

### A.10. Latency Analysis

We provide a detailed latency analysis of both streaming and standard methods on a single NVIDIA RTX 3090 GPU in Table 17. The results are reported for the *single-agent* setting, where each inference step produces predictions for a single agent. In real-world scenarios, however, we are interested in forecasting for multiple surrounding traffic agents. Thus, practically relevant latencies correspond to larger batch sizes $B$, since $B$ reflects the number of other agents for which predictions are produced. On the AV2 single-agent dataset, our approach requires under 10 GB of memory for inference with a batch size of 32 (<5 GB for batch size 16 and <15 GB for batch size 128), further highlighting the resource efficiency of our model.

We use the official implementations of QCNet[1] [55], RealMotion[2] [36], and DeMo[3] [50]. To ensure a fair comparison across different code bases, we measure only the model forward-pass latency to eliminate the influence of data preprocessing and loading.

[1] https://github.com/ZikangZhou/QCNet
[2] https://github.com/fudan-zvg/RealMotion
[3] https://github.com/fudan-zvg/DeMo

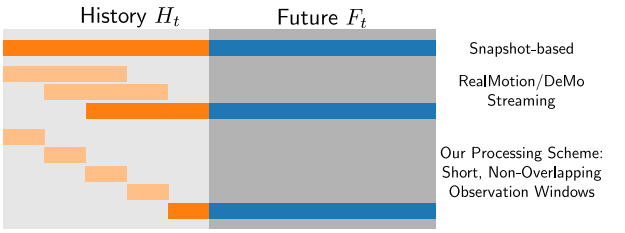



Figure 4. Comparison of motion forecasting processing schemes: standard snapshot-based forecasting, the streaming-based forecasting used by RealMotion [36] and Demo [50] (long, overlapping, observation windows) and our proposed approach (compact, subsequent, observation windows).

## B. Implementation Details

In the following, we outline all model details, including module designs, hyperparameters, and training setup. For all experiments, we use a model feature dimension of $D = 128$. Table 18 presents an overview of parameters used for loading agent and map data from the datasets (AV1 [4], nuScenes [2], and AV2 [44]). We also add a valid flag to the agent and lane features to pad lanes, which extend outside the region of interest. This leads to an agent feature dimension $D_a = 5$ for AV2 and $D_a = 3$ for AV1 and nuScenes. Across all datasets, the lane feature dimension is $D_l = 3$ ($xy$ coordinates of the center line and padding flag).

### B.1. Streaming Processing

Fig. 4 compares our proposed streaming processing scheme to the standard snapshot-based evaluation—which uses the whole history available in a benchmark dataset in one model pass—and to the streaming processing proposed by RealMotion [36]. Compared to RealMotion, we use a non-overlapping streaming mechanism with significantly shorter observation lengths per model pass.

### B.2. Target-Centric Features

A key challenge in trajectory prediction is identifying the map regions where detailed features should be extracted. Besides the immediate surroundings of each agent, as modeled by standard agent-centric features, the regions an agent is likely to move toward in the future are also highly important. To encode this information, we introduce a set of target-centric auxiliary features, constructed from the predicted endpoints of the previous inference pass $F^{t-1}$. For each endpoint, we establish a local coordinate frame with the endpoint as the origin, and we encode all contextual features within its vicinity into a target-centric feature set. Based on preliminary experiments, we use a 30 m radius for aggregating agent and map tokens. For each of the $K$ predicted endpoints, this yields an auxiliary feature tensor $C^t_{\text{enc}} \in \mathbb{R}^{K\times(N_a+N_l)'\times D}$, where $(N_a + N_l)'$ denotes the maximum number of features among all $K$ endpoint-centric sets; we pad smaller sets to ensure consistent batch processing. To incorporate

| Method | Streaming | $B = 1$ | $B = 16$ | $B = 32$ | $B = 64$ | $B = 128$ | Model Parameters | AV2 Test Set brier-minFDE$_6$ |
|---|---|---|---|---|---|---|---|---|
| QCNet [55] | ✗ | 141 ms | 303 ms | 565 ms | 1092 ms | - | 7.7M | 1.91 |
| RealMotion [36] | ✓ | **19 ms** | 44 ms | 79 ms | 140 ms | 273 ms | **2.9M** | 1.89 |
| DeMo [50] | ✓ | 25 ms | 35 ms | 49 ms | 88 ms | 180 ms | 5.9M | 1.84 |
| SHARP (Ours) | ✓ | 21 ms | **33 ms** | **35 ms** | **57 ms** | **105 ms** | 4.6M | **1.83** |

Table 17. **Latency analysis** for batches of size $B$ when performing *single-agent prediction*, evaluated on the Argoverse 2 validation set with an NVIDIA RTX 3090 GPU. Larger batch sizes $B$ correspond to more practically-relevant latency estimates, since real-world deployments must process multiple surrounding traffic agents, *i.e.* $B > 1$, simultaneously. For QCNet we run out of GPU memory when testing $B = 128$.

| Dataset | RoI Radius | Agents Features | Agents Types | Lanes Features | Lanes Types | $P_l$ | Input Window Length $T_h$ |
|---|---|---|---|---|---|---|---|
| AV1 [4] | 150 m | $xy$-pos | Vehicle | $xy$-pos | Standard Lane | 10 | 0.5 s |
| AV2 [44] | 150 m | $xy$-pos, $xy$-vel | Veh./Pedest./Cycl./Others | $xy$-pos | Veh./Bike/Bus | 20 | 1 s |
| nuScenes [2] | 100 m | $xy$-pos | Car/Pedest./Cycl./Other Veh./Motorcycl./Others | $xy$-pos | Standard/Intersection | 20 | 0.5 s |

Table 18. **Data preprocessing and sampling parameters** for loading data from Argoverse 1 (AV1), Argoverse 2 (AV2), and nuScenes. *RoI radius* denotes the region of interest radius used to sample surrounding agent and map data.

these features during decoding, we additionally encode the spatial transformation between the global coordinate frame and each endpoint-centric local frame through a positional embedding.

### B.3. Multi-Head Attention Blocks

We use a standard configuration [6, 30, 36] for our multi-head attention blocks used in the encoder and decoder modules:

- 8 Attention heads
- Dropout rate set to 0.2
- Feedforward path expansion factor set to $4 \cdot D$
- Norm executed before attention operation
- GELU [17] Activation function

### B.4. Encoder Modules

**Agent and Lane Encoder.** Initially, we use a single layer to project the input agent features $A \in \mathbb{R}^{N_a \times T_h \times D_a}$ to our model feature dimension $D$, leading to $\mathbb{R}^{N_a \times T_h \times D}$. Next, we apply self-attention across the historical time steps using four encoder blocks. To reduce the dimensionality, we apply a max-pooling layer, resulting in $A_{\text{enc}} \in \mathbb{R}^{N_a \times D}$ as output for $f_A$. For encoding the lanes $f_L$ we use the standard mini-PointNet-based [32] approach which is commonly used by related work [6, 30, 36, 50].

**Positional Embeddings.** We model all positional embeddings using the 2D coordinates $(x, y)$ and orientation $(\gamma)$. For agents, we use the agent's pose and heading; for lanes, we use the centerline's midpoint and rotation. To wrap the angle, we encode each pose as $(x, y, \sin\gamma, \cos\gamma)$. We implement the positional encodings using a shallow multilayer perceptron (MLP):

- Layer 1: $4 \times D$
  GELU [17] activation function
- Layer 2: $D \times D$

**Instance-aware Context Streamer.** Our instance-aware context streamer $f_{\text{IA}}$ follows the design principle proposed by [36] and applies cross-attention between the current scene $S^t$ and the previously encoded context $S^{t-1}_{\text{enc}}$. As agents enter or leave the scene and the map content within the region of interest changes due to ego-motion, the number of tokens $N_c$ generally differs between the two contexts. To explicitly model instance correspondences across time between these contexts, we introduce an attention mask $M \in \mathbb{R}^{N_c^t \times N_c^{t-1}}$, where $M_{ij} = \theta$ if token at position $i$ in $S^t$ corresponds to the token at position $j$ in $S^{t-1}_{\text{enc}}$ and 0 otherwise. The mask weight $\theta$ is a learnable parameter and is jointly optimized during training. After the context streaming, we obtain $S^t_{\text{stream}} \in \mathbb{R}^{N_c^t \times D}$.

**Scene Context Encoders.** After constructing the agent-centric scene context and augmenting it with the previous context, we encode $S^t_{\text{stream}}$ using the agent-centric scene context encoder $f_S$. This encoder consists of four attention blocks, where self-attention is applied across all agent and lane tokens to jointly capture their relationships. The target-centric feature encoder $f_T$ follows the same design principles but uses only two blocks because $C_t$ has fewer tokens.

### B.5. Decoder

Our decoder $f_D$ uses separate cross-attention blocks to attend to the agent-centric scene features $S^t_{\text{enc}}$ and the target-centric context $C^t_{\text{enc}}$. The decoder alternates between attending to $S^t_{\text{enc}}$ and $C^t_{\text{enc}}$. If no target-centric context is available (*e.g.*, for newly detected agents), we simply skip the blocks that would attend to $C^t_{\text{enc}}$. In our final architecture, we use three blocks per feature type, resulting in a total of six cross-attention blocks. After updating the mode queries $Q^t_{\text{enc}} \in \mathbb{R}^{K \times D}$, we apply two MLP heads to produce the trajectory forecasts $F$ and their corresponding prediction scores $P$. The forecast head has the following design:

- Layer 1: $D \times 2 \cdot D$

ReLU activation function
- Layer 2: $2 \cdot D \times 2 \cdot T_f$

Similarly, the prediction head is given as:
- Layer 1: $D \times 2 \cdot D$
  ReLU activation function
- Layer 2: $2 \cdot D \times 1$

### B.6. Optimization

We train our model using a single NVIDIA Quadro RTX 8000 with 48 GB VRAM using a batch size of 32. To accommodate the different dataset complexities, training is executed for 80 epochs on AV2, for 60 epochs on AV1 and for 25 epochs on nuScenes. For all trainings, we use warm-up epochs (13/10/4), where the learning rate is linearly increased to $1e-4$ before being decreased to $1e-5$ using a single cosine schedule. We apply gradient clipping and weight decay regularization. No augmentations are used, and we use no cross-dataset training.

We use AdamW [25] as our optimizer. Following common practice [6], only the best-matching trajectory hypothesis contributes to the loss via a winner-takes-all strategy. A Smooth L1 regression loss [19] is applied to fit this selected hypothesis to the ground truth, and a cross-entropy classification loss encourages the model to assign the highest probability to it. Additionally, we include an auxiliary regression head that predicts a single trajectory for other agents surrounding the focal agent [6]. For this auxiliary task, we apply a two-layer MLP to the agent features in $S^t_{\text{enc}}$ and supervise it using a Smooth L1 loss.

### B.7. Multi-Agent Extension

**Global Consistency Module.** We employ a global consistency module to combine the marginal predictions — represented by the per-agent mode queries after cross-attention, $Q^t_{\text{enc}} \in \mathbb{R}^{N_a \times K \times D}$ — into scene-level, jointly consistent forecasts. We first select a scene-wide reference frame, choosing either a specific agent or the self-driving vehicle; in our implementation, we simply use the first agent listed in each scenario. Subsequently, we add a positional embedding modeling the current agent position w.r.t. the scene global coordinate system. To capture both intra- and inter-agent relationships, we apply self-attention over all motion modes for each agent (across $K$) and across all agents for each hypothesized future world (across $N_a$), using two blocks for each. The resulting predictions are organized in $K$ worlds, where each world contains predictions for all agents and represents a collision-free, globally consistent future. Trajectories within each world are generated using a two-layer MLP following the single-agent design described above. World-level probability scores are obtained by fusing the agent queries associated with that world and feeding the fused representation into an MLP head, again following the single-agent setup. To avoid the influence of parked vehicles, we exclude them during the fusion step based on the trajectory head's output.

**Finetuning on Multi-Agent Data.** We finetune our model for 35 epochs on the multi-agent data using a single NVIDIA RTX PRO 6000 with 96 GB VRAM. We set the batch size to 32 scenarios and initialize our model for the marginal predictions with the weights trained on the single-agent dataset. During finetuning, we combine marginal prediction and joint prediction losses to train the global consistency module while also providing additional guidance to the early encoder layers. The single-agent losses are identical to those in the single-agent training. For the multi-agent output, we again employ a winner-takes-all strategy, only optimizing the *world* with the lowest overall displacement. Next, we compute a regression loss based on the prediction for each agent in this world, as well as a standard cross-entropy loss, so that this world receives the highest probability.

## C. Evaluation Details

### C.1. Metrics

We evaluate our approach using the standard benchmark metrics [2, 4, 44]. Each metric is computed over the top-$k$ highest-scoring trajectory forecasts. In the single-agent setting, we report the **miss rate** ($\text{MR}_k$), which evaluates if any predicted endpoint lies within a radius $r$ to the ground endpoints; **average displacement error** ($\text{minADE}_k$), which reports the average displacement between the ground truth and averaged across all future time steps; and the **final displacement error** ($\text{minFDE}_k$), which reports the smallest distance between the ground truth endpoint and a predicted endpoint. The **brier-minFDE**$_k$ adds a probability penalty $(1-p)^2$ for the $\text{minFDE}_k$ based on the score $p$ for the trajectory with the lowest displacement.

For the multi-agent setting, we use the joint extensions of the single-agent metrics [44]: $\text{avgMinADE}_k$, $\text{avgMinFDE}_k$, $\text{actorMR}_k$, and $\text{avgBrierMinFDE}_6$. Here, $k$ worlds are considered, where each world contains one prediction for every agent. Each metric is computed per world by averaging over all agent errors in this world.

### C.2. Evaluation Protocol for Main Paper Table 1

To evaluate related work on streaming trajectory prediction in the evolving scene setting (main paper Table 1), we benchmark all methods across different prediction time steps $t_p$ and varying numbers of input windows$T_{\text{cl}}$. To ensure fair comparison, we keep the observation window (3 s) and the window displacement (1 s) identical to the original setup [36, 50], effectively mimicking how these models behave in practical streaming deployment when provided with fewer or more observations. A visualization of the resulting streaming evaluation protocol is shown in Fig. 5.

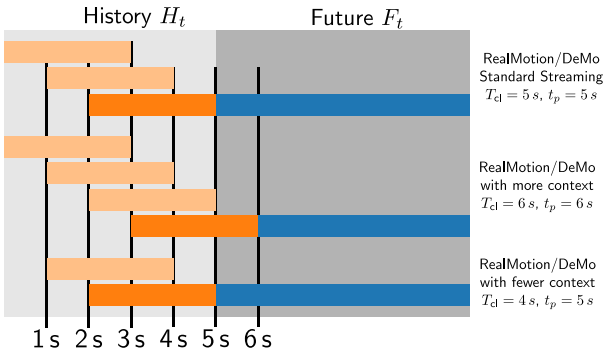



Figure 5. Details for evaluating related work on the evolving scene setting (main paper Table 1). The first example shows the standard execution on the AV2 benchmark. In the second example, we increase the input context by simply executing another streaming step. In the last example, we test the models with fewer context by evaluating after two streaming steps.

# D. Visualizations

## D.1. Evolving Scenes

We present qualitative results on scenarios from the Argoverse 2 validation set in Fig. 6. Additionally, we provide animated results as separate MP4 files on our project page.

## D.2. Failure Cases

We present failure cases in which our approach fails to correctly predict future trajectories in Fig. 7. Commonly, failures are introduced by agent movements which cannot be anticipated at the prediction time, often also due to inadequate map data, *e.g*. missing modeling of driveways.

# E. Code Implementation

To support reproducibility, we provide our code implementation, including data loaders for all three datasets, as well as the extension to the multi-agent setting.

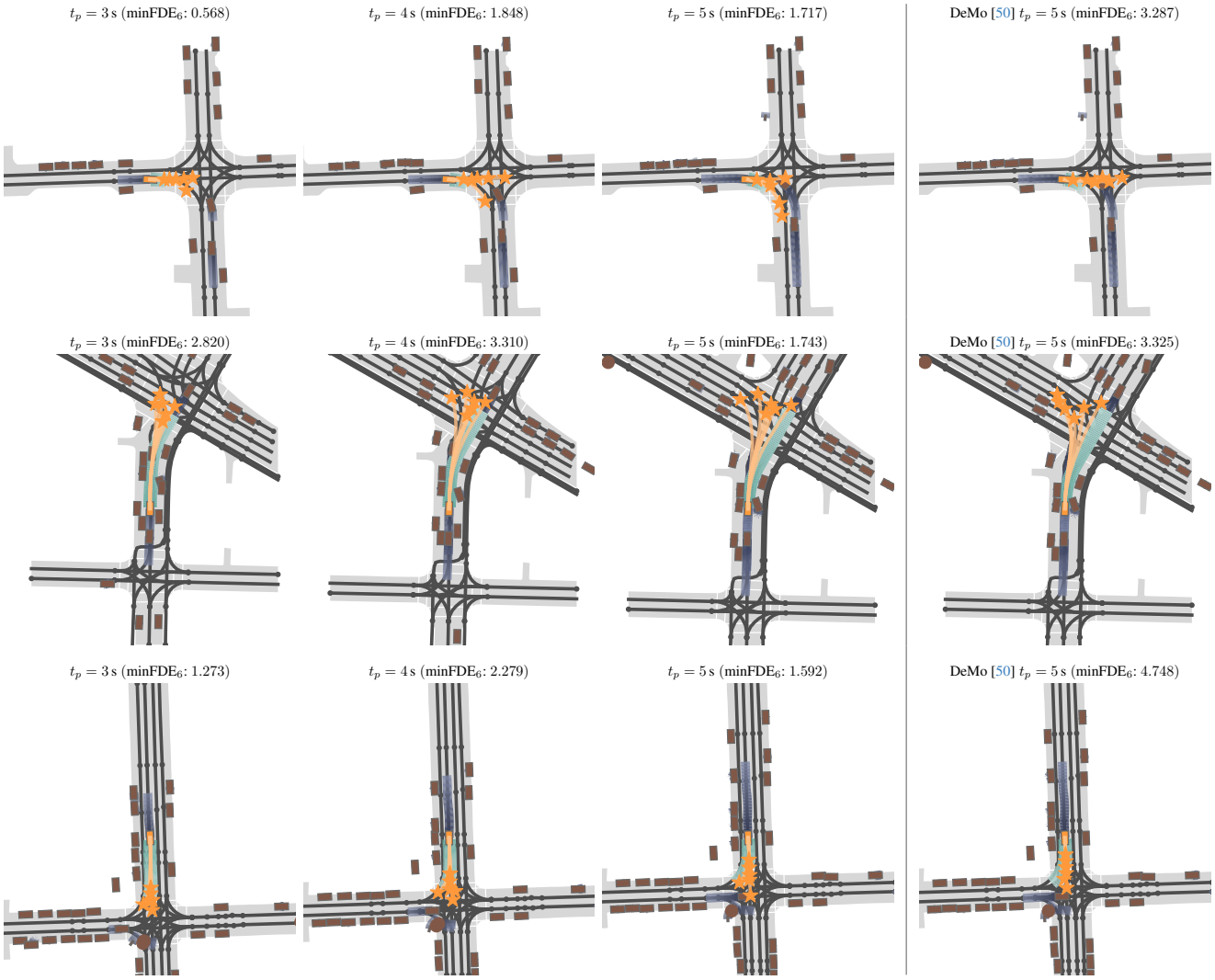



Figure 6. Qualitative single-agent prediction results of our SHARP on scenarios from the Argoverse 2 validation set. We visualize the **predictions** of our streaming-based method at $t_p \in \{3, 4, 5\}$s. The visualizations also show **ground truth future**, **historical agent observations**, and **surrounding agents**. For comparison, the right column shows the predictions of DeMo [50] at the AV2 standard prediction time step $t_p$=5 s.

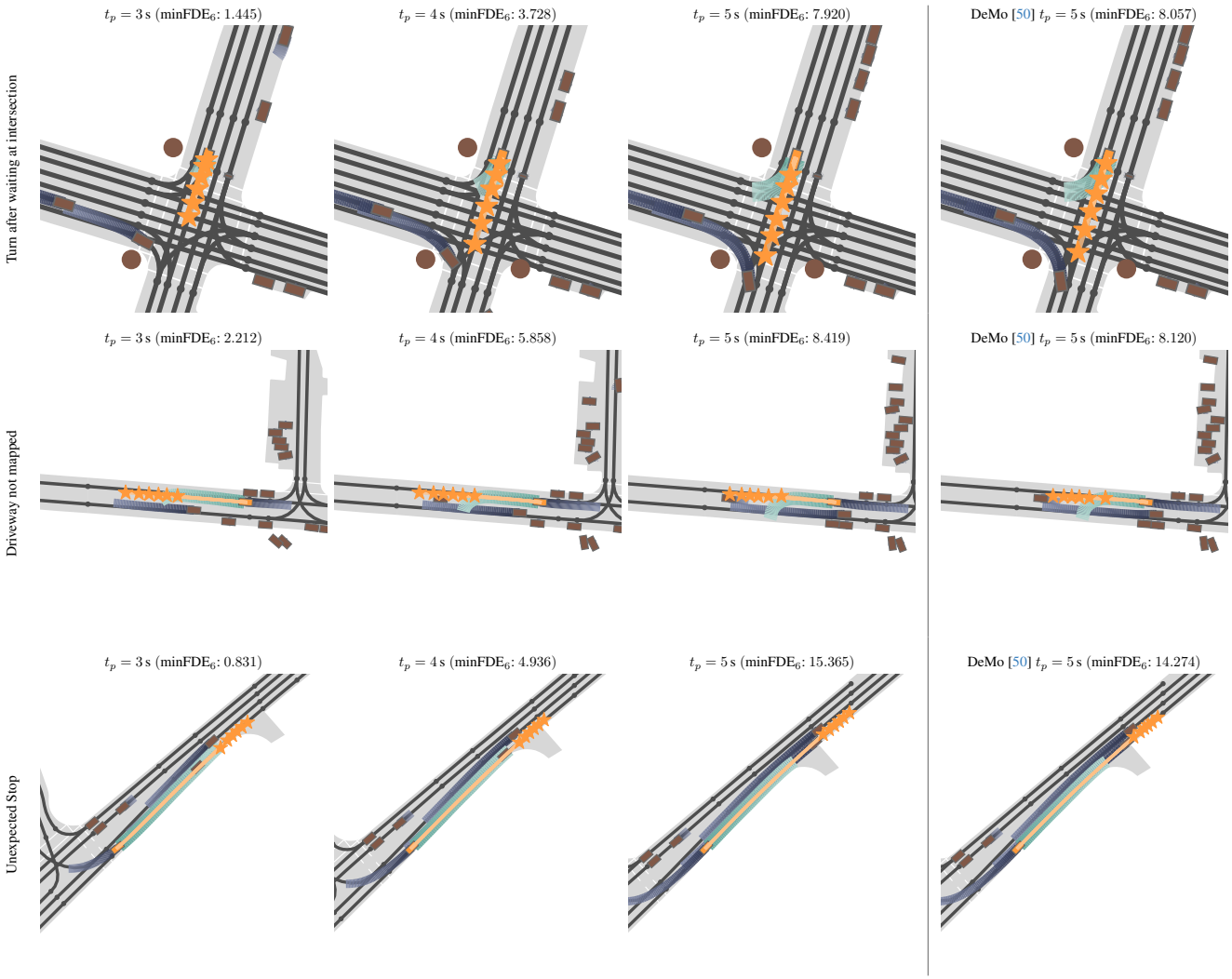



Figure 7. Failure cases of our SHARP on scenarios from the Argoverse 2 validation set. We visualize the **predictions** of our streaming-based method at $t_p \in \{3, 4, 5\}$s. The visualizations also show **ground truth future**, **historical agent observations**, and **surrounding agents**. For comparison, the right column shows the predictions of DeMo [50] at the AV2 standard prediction time step $t_p$=5 s. In the first scenario, a vehicle is waiting at an intersection but will initiate a right turn later – an action that is not anticipated from the current observations. In the second scenario, a vehicle turns into a driveway that is not represented in the map data. In the final scenario, a vehicle stops in the future for a reason that is not captured in the current perception data.