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 (Suppl. A). To support reproducibility, we provide all implementation details of our architecture in Suppl. B, including training hyperparameters and details on the multi-agent extension. We further detail the evaluation setup (Suppl. C) and provide output visualizations (Suppl. D). Finally, we provide an overview of the code framework (Suppl. E), which is provided at https://github.com/a-pru/sharp.

A. Additional Insights

A.1. Observation Length Study

Table 1 provides an ablation study for predictions of our SHARP across different context lengths T_{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_{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_{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 1. Evaluation of SHARP on the AV2 validation set performed **across different context lengths** T_{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 2 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 [13].

Additionally, Table 4 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 2. 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 3 analyzes the performance of SHARP across varying context lengths T_{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.

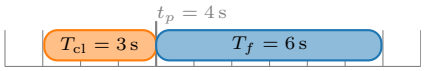
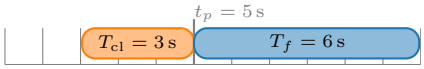
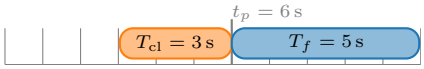
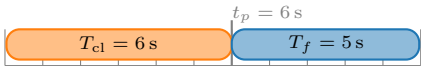
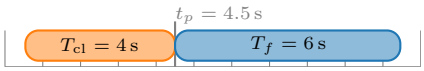
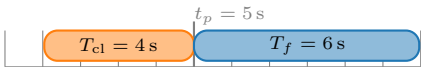
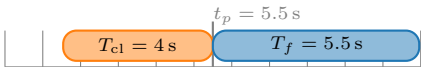
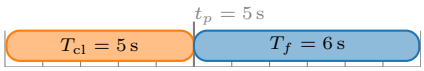
History / Prediction Time Step / Future Horizon	avgMinADE$_6$	avgMinFDE$_6$	actorMR$_6$	**avgBrierMinFDE$_6$**
	0.68	1.43	0.19	2.07
	0.60	1.26	0.17	1.91
	0.54	1.17	0.15	1.83
	0.43	0.88	0.11	1.52
	0.32	0.64	0.06	1.27
	0.23	0.43	0.03	1.06
	0.55	1.16	0.15	1.80
	0.56	1.21	0.16	1.86
	0.46	0.97	0.12	1.62
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

Table 3. Evaluation of SHARP in the **multi-agent setting** on the AV2 validation set at **varying prediction time steps t_p and context lengths** T_{cl}. The evaluation scope is limited to our method due to missing multi-agent implementation details and code bases for related work [13, 16].

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 4. 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 5 compares our approach to the state-of-the-art on the AV1 [3] validation set. Overall, SHARP achieves top performance, closely matching the results of HPNet [14].

A.5. Short Histories

Table 6 compares our method to DeMo [16] using a 1 s input length T_{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.

A.6. Extended Prediction Horizon

Our short-window processing scheme enables training with extended prediction horizons without requiring additional

Method	Streaming	MR_6	$mADE_6$	**$mFDE_6$**
mmTransformer [8]	✗	0.11	0.71	1.15
FRM [10]	✗	-	0.68	0.99
ADAPT [1]	✗	0.08	0.67	0.95
SIMPL [17]	✗	0.08	0.66	0.95
HiVT [18]	✗	0.09	0.66	0.96
R-Pred [5]	✗	0.09	0.66	0.95
DeMo [16]	✓	**0.07**	0.59	0.90
SHARP (Ours)	✓	**0.07**	**0.58**	0.90
HPNet [14]	✗	**0.07**	0.64	**0.87**

Table 5. Results on the **Argoverse 1 prediction challenge validation set**.

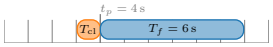
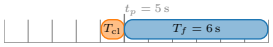
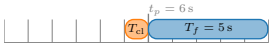
Streaming Setup	Method	$mADE_6$	$mFDE_6$	**b-$mFDE_6$**
	DeMo	1.02	1.86	2.54
	SHARP	**0.70**	**1.16**	**1.79**
	DeMo	0.80	1.46	2.15
	SHARP	**0.70**	**1.34**	**1.99**
	DeMo	0.99	1.97	2.67
	SHARP	**0.76**	**1.48**	**2.13**
	DeMo	0.95	1.96	2.67
	SHARP	**0.65**	**1.27**	**1.92**

Table 6. We compare our SHARP to DeMo [16] using **short context lengths T_{cl} of 1 s** on the AV2 single-agent validation set.

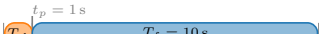
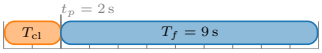
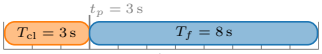
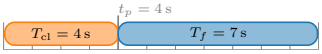
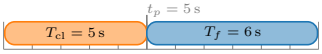
Streaming Setup	$mADE_6$	$mFDE_6$	**b-$mFDE_6$**
	1.37	2.48	3.12
	1.12	2.06	2.70
	0.95	1.76	2.40
	0.79	1.48	2.11
	**0.64**	**1.20**	**1.82**

Table 7. 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.

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 7 reports experiments evaluating our module on long prediction horizons T_f.

A.7. Dual Loss Ablation

Table 8 presents results for training our model using weighting parameters to combine the two losses, $\mathcal{L}_{\text{stream}}$ and $\mathcal{L}_{\text{chunk}}$, 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

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

Table 8. 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.

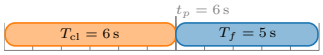
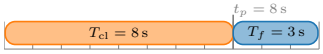
Streaming Setup	f_{IA}	$mADE_6$	$mFDE_6$	**b-$mFDE_6$**
	✗	0.54	0.99	1.63
	✓	**0.51**	**0.94**	**1.56**
	✗	0.47	0.84	1.50
	✓	**0.39**	**0.70**	**1.31**
	✗	0.39	0.66	1.33
	✓	**0.28**	**0.48**	**1.08**

Table 9. 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 [13].

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 9 compares our instance-aware context streamer against the implicit approach of [13] 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 10 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 [13]. 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 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.

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 10. 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 [13]. 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.

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 11. 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] [19], RealMotion[2] [13], and DeMo[3] [16]. 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.

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 12 presents an overview of parameters used for loading agent and map data from the datasets (AV1 [3], nuScenes [2], and AV2 [15]). 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).

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

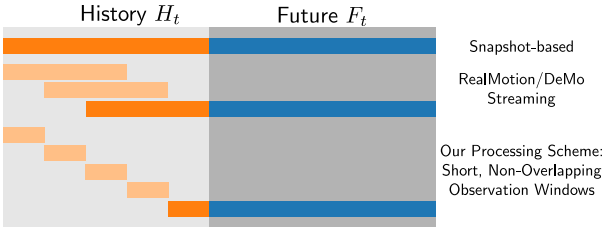


Figure 1. Comparison of motion forecasting processing schemes: standard snapshot-based forecasting, the streaming-based forecasting used by RealMotion [13] and Demo [16] (long, overlapping, observation windows) and our proposed approach (compact, subsequent, observation windows).

B.1. Streaming Processing

Fig. 1 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 [13]. 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 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 [4, 11, 13] for our multi-head attention blocks used in the encoder and decoder mod-

Method	Streaming	$B = 1$	$B = 16$	$B = 32$	$B = 64$	$B = 128$	Model Parameters	AV2 Test Set brier-minFDE$_6$
QCNet [19]	✗	141 ms	303 ms	565 ms	1092 ms	-	7.7M	1.91
RealMotion [13]	✓	**19 ms**	44 ms	79 ms	140 ms	273 ms	**2.9M**	1.89
DeMo [16]	✓	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 11. **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 [3]	150 m	xy-pos	Vehicle	xy-pos	Standard Lane	10	0.5 s
AV2 [15]	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 12. **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.

ules:

- 8 Attention heads
- Dropout rate set to 0.2
- Feedforward path expansion factor set to $4 \cdot D$
- Norm executed before attention operation
- GELU [6] 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 [12] approach which is commonly used by related work [4, 11, 13, 16].

Positional Embeddings. We model all positional embeddings using the 2D coordinates (x, y) and orientation (γ). 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 [6] activation function
- Layer 2: $D \times D$

Instance-aware Context Streamer. Our instance-aware context streamer f_{IA} follows the design principle proposed by [13] and applies cross-attention between the current scene S^t and the previously encoded context S^{t-1}_{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}_{enc} and 0 otherwise. The mask weight θ 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_{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_{enc} and the target-centric context C^t_{enc}. The decoder alternates between attending to S^t_{enc} and C^t_{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_{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 [9] as our optimizer. Following common practice [4], only the best-matching trajectory hypothesis contributes to the loss via a winner-takes-all strategy. A Smooth L1 regression loss [7] 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 [4]. For this auxiliary task, we apply a two-layer MLP to the agent features in S^t_{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.

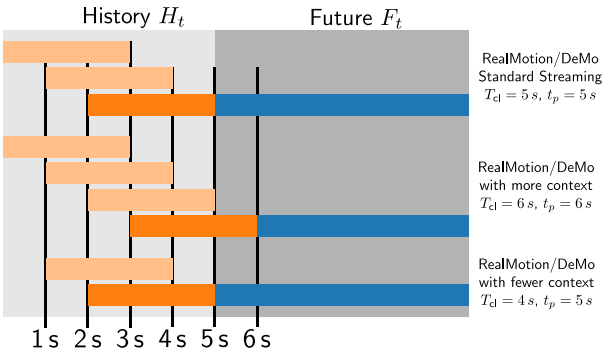


Figure 2. Details for evaluating related work on the evolving scene setting (main paper Table 1). The first example shows the standard execution on the AV2 benchmark. In the second example, we increase the input context by simply executing another streaming step. In the last example, we test the models with fewer context by evaluating after two streaming steps.

C. Evaluation Details

C.1. Metrics

We evaluate our approach using the standard benchmark metrics [2, 3, 15]. Each metric is computed over the top-k highest-scoring trajectory forecasts. In the single-agent setting, we report the **miss rate** (MR_k), which evaluates if any predicted endpoint lies within a radius r to the ground endpoints; **average displacement error** (minADE_k), which reports the average displacement between the ground truth and averaged across all future time steps; and the **final displacement error** (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 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 [15]: avgMinADE_k, avgMinFDE_k, actorMR_k, and 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 windowsT_{cl}. To ensure fair comparison, we keep the observation window (3 s) and the window displacement (1 s) identical to the original setup [13, 16], 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. 2.

D. Visualizations

D.1. Evolving Scenes

We present qualitative results on scenarios from the Argoverse 2 validation set in Fig. 3. 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. 4. 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.

References

[1] Görkay Aydemir, Adil Kaan Akan, and Fatma Güney. ADAPT: Efficient Multi-Agent Trajectory Prediction with Adaptation. In *ICCV*, 2023. 3

[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. 4, 5, 6

[3] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3D Tracking and Forecasting with Rich Maps. In *CVPR*, 2019. 2, 4, 5, 6

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

[5] Sehwan Choi, Jungho Kim, Junyong Yun, and Jun Won Choi. R-Pred: Two-Stage Motion Prediction Via Tube-Query Attention-Based Trajectory Refinement. In *ICCV*, 2023. 3

[6] Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *arXiv*, 2016. 5

[7] Peter J Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 1964. 6

[8] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal Motion Prediction with Stacked Transformers. *CVPR*, 2021. 3

[9] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 6

[10] 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. 3

[11] 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. 4, 5

[12] 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. 5

[13] Nan Song, Bozhou Zhang, Xiatian Zhu, and Li Zhang. Motion Forecasting in Continuous Driving. In *NeurIPS*, 2024. 1, 2, 3, 4, 5, 7

[14] Xiaolong Tang, Meina Kan, Shiguang Shan, Zhilong Ji, Jinfeng Bai, and Xilin Chen. HPNet: Dynamic Trajectory Forecasting with Historical Prediction Attention. In *CVPR*, 2024. 2, 3

[15] 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. 4, 5, 6

[16] Bozhou Zhang, Nan Song, and Li Zhang. DeMo: Decoupling Motion Forecasting into Directional Intentions and Dynamic States. In *NeurIPS*, 2024. 2, 3, 4, 5, 7, 8, 9

[17] Lu Zhang, Peiliang Li, Sikang Liu, and Shaojie Shen. SIMPL: A Simple and Efficient Multi-agent Motion Prediction Baseline for Autonomous Driving. *RAL*, 2024. 3

[18] Zikang Zhou, Luyao Ye, Jianping Wang, Kui Wu, and Kejie Lu. HiVT: Hierarchical Vector Transformer for Multi-Agent Motion Prediction. In *CVPR*, 2022. 3

[19] Zikang Zhou, Jianping Wang, Yung-Hui Li, and Yu-Kai Huang. Query-Centric Trajectory Prediction. In *CVPR*, 2023. 4, 5

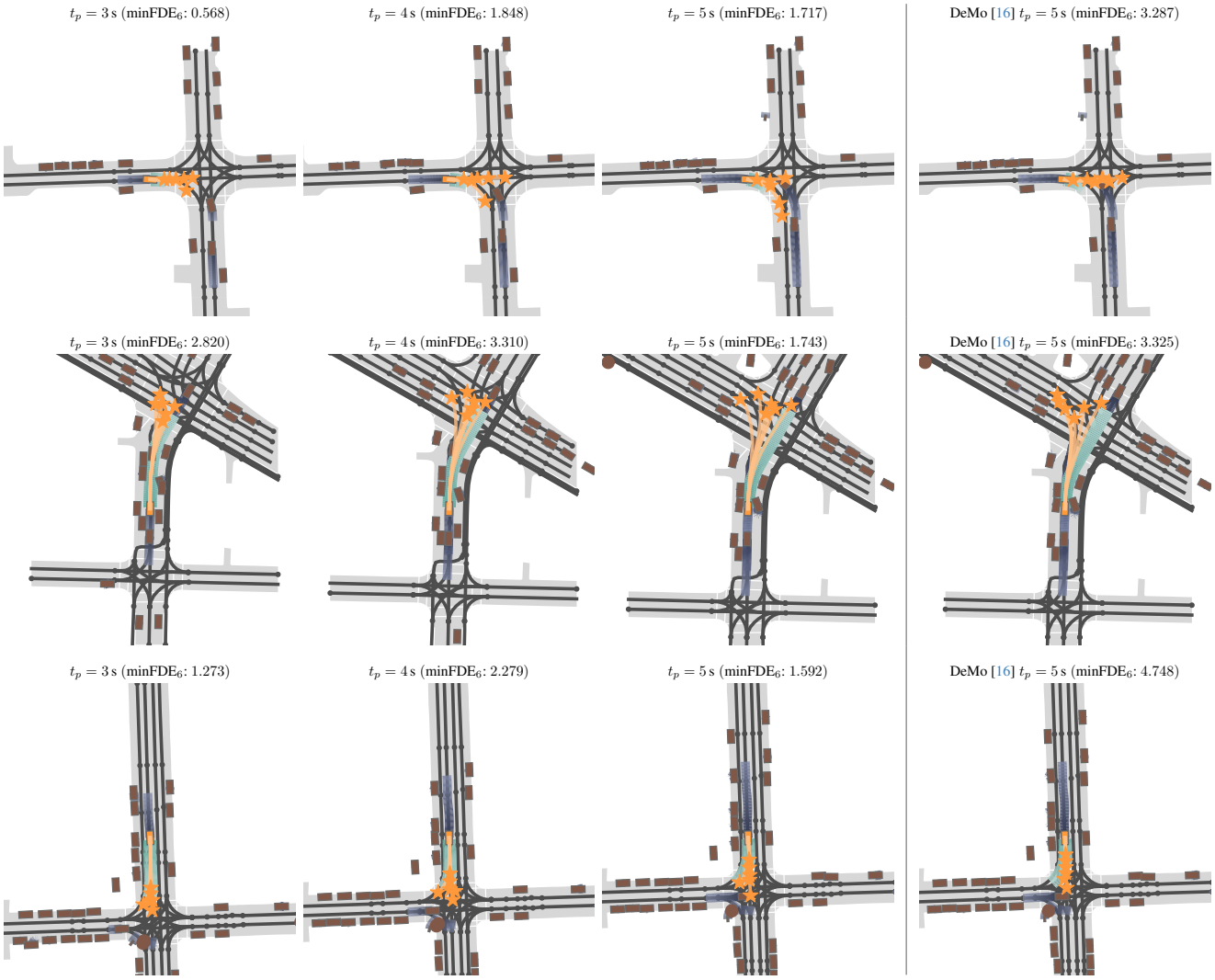


Figure 3. Qualitative single-agent prediction results of our SHARP on scenarios from the Argoverse 2 validation set. We visualize the **predictions** of our streaming-based method at $t_p \in \{3, 4, 5\}$s. The visualizations also show **ground truth future**, **historical agent observations**, and **surrounding agents**. For comparison, the right column shows the predictions of DeMo [16] at the AV2 standard prediction time step t_p=5 s.

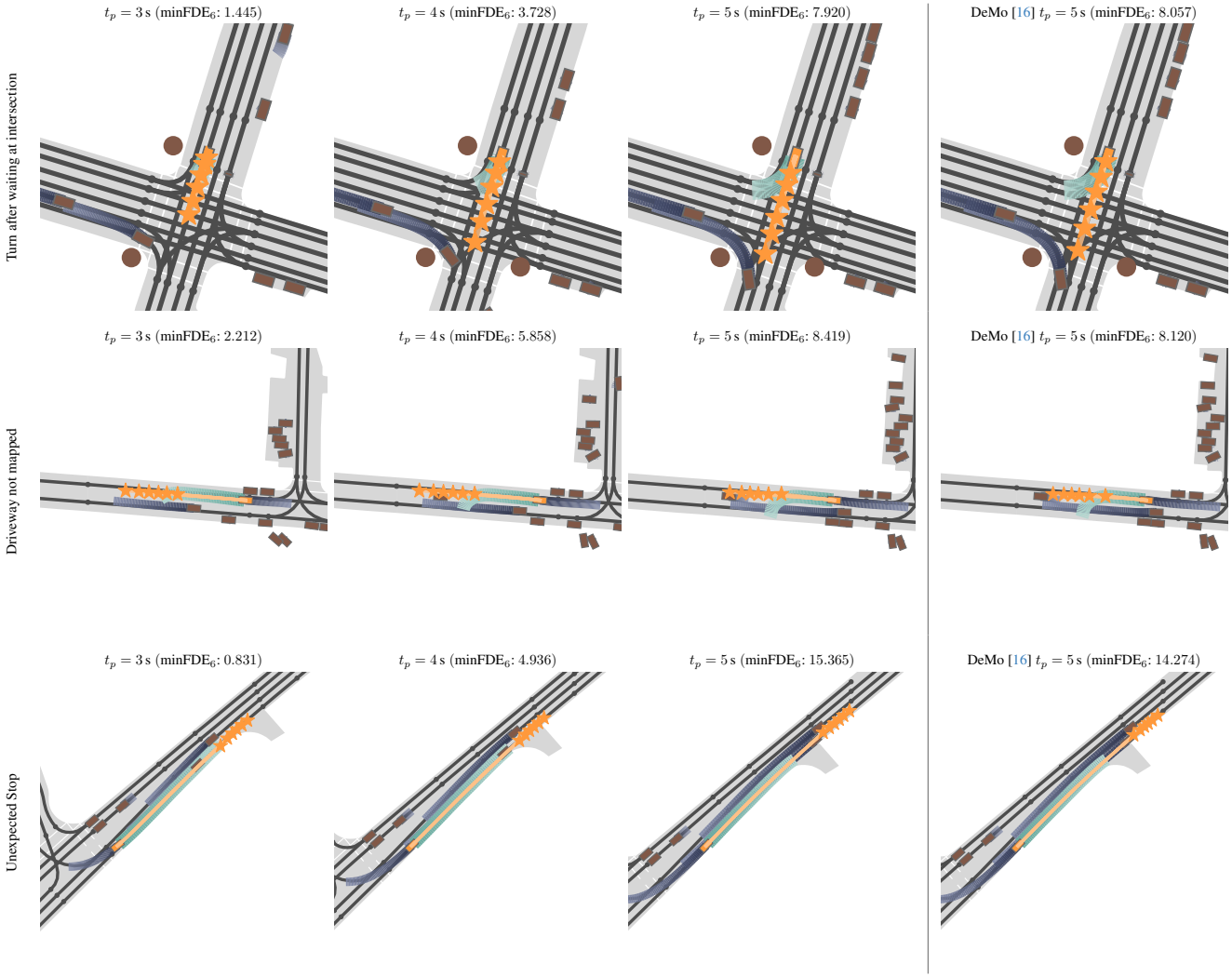


Figure 4. Failure cases of our SHARP on scenarios from the Argoverse 2 validation set. We visualize the **predictions** of our streaming-based method at $t_p \in \{3, 4, 5\}$s. The visualizations also show **ground truth future**, **historical agent observations**, and **surrounding agents**. For comparison, the right column shows the predictions of DeMo [16] 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.